МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ
РОССИЙСКОЙ ФЕДЕРАЦИИ
Белгородский государственный технологический университет
им. В.Г. Шухова

Кафедра информационных технологий

Утверждено
научно-методическим советом
университета

ИНФОКОММУНИКАЦИОННЫЕ СИСТЕМЫ И СЕТИ

Методические указания к выполнению лабораторных работ
для студентов направлений 09.03.02 – Информационные системы и технологии, 09.03.03 – Прикладная информатика.

Белгород
2017

УДК004.7 (07)
ББК 32.973.202я7
И74

Составитель ст. преподаватель *А.В. Глухоедов*

Рецензент канд. техн. наук, доц. *Д.Н. Старченко*

Инфокоммуникационные системы и сети: методические указа-
И74 ния к выполнению лабораторных работ / сост. А.В. Глухоедов. – Белгород: Изд-во БГТУ, 2017. – 72 с.

В методические указания включены требования и рекомендации к выполнению лабораторных работ по дисциплинам «Инфокоммуникационные системы и сети» и «Вычислительные системы, сети и технологии», а также задания для выполнения данных работ.

Методические указания предназначены для студентов направлений 09.03. 02 – Информационные системы и технологии и 09.03.03 – Прикладная информатика.

Данное издание публикуется в авторской редакции.

УДК 004.7 (07)
ББК 32.973.202я7

СОДЕРЖАНИЕ

ЛАБОРАТОРНАЯ РАБОТА №1. УПРАВЛЕНИЕ СЕТЯМИ И ОБЩИМ ДОСТУПОМ

Цель работы

Научиться использовать стандартные средства операционной системы Microsoft Windows 7 для настройки сети и общего доступа к файлам.

Краткие теоретические сведения

Операционная система Windows 7 выполняет за пользователя большую часть работы по установке и настройке сети. Все основные настройки, которые касаются работы с сетью, в Windows 7 собраны в окне «Центр управления сетями и общим доступом» (рис. 1.1).

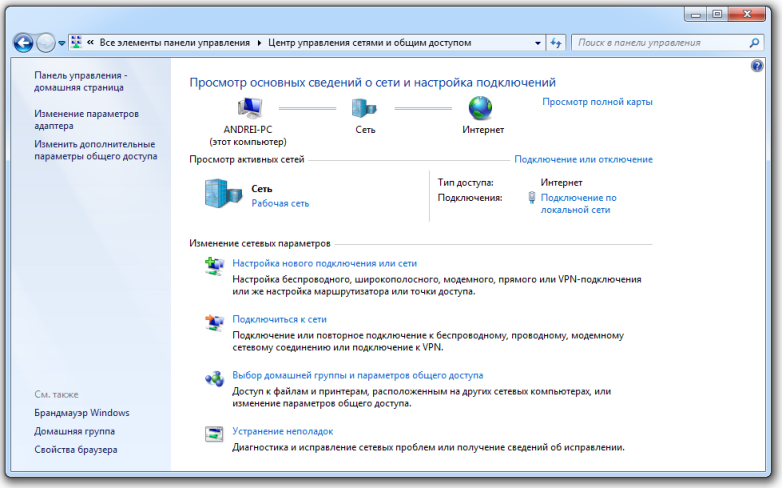


Рис. 1.1. Центр управления сетями и общим доступом

Следует отметить, большинство изменений настроек сети может выполнять только администратор или пользователь, который является членом группы «Администраторы» или группы «Операторы настройки сети».

Открытие центра управления сетями и общим доступом

Для того чтобы открыть окно «Центр управления сетями и общим доступом» откройте **Панель управления** и выполните одно из следующих действий:

- Если используется представление **Категории**, в разделе **Сеть и Интернет** выберите элемент **Просмотр состояния сети и задач**.
- Если используется представление **Крупные значки** или **Мелкие значки**, выберите элемент **Центр управления сетями и общим доступом**.

Также открыть «Центр управления сетями и общим доступом» можно щелкнув правой кнопкой мыши на значок **Сеть** в области уведомлений панели задач, а затем выдрав пункт **Центр управления сетями и общим доступом**.

Еще один способ открытия «Центра управления сетями и общим доступом» заключает в использования командной строки. Для этого введите в командной строке следующую команду (регистр не имеет значения):

```
control.exe /name Microsoft.NetworkAndSharingCenter
```

Настройка сетевого размещения

При первом подключении к сети необходимо выбрать сетевое размещение. При этом автоматически настраиваются параметры безопасности для типа сети, к которой производится подключение. Существую следующие типы сетевого размещения:

- **Домашняя сеть**. Компьютеры домашней сети могут принадлежать домашней группе. Для домашних сетей включается обнаружение сети, что обеспечивает использование остальных компьютеров и устройств, подключенных к сети, а также позволяет другим пользователям получать доступ к компьютеру из сети.
- **Сеть предприятия**. Обнаружение сети включено по умолчанию, но при этом нельзя создать домашнюю группу или присоединиться к ней.
- **Общественная сеть.** Данное сетевое размещение настроено таким образом, чтобы сделать компьютер «невидимым» для других пользователей. Домашняя группа недоступна в общественных сетях, а сетевое обнаружение отключено.

Для выбора сетевого размещения откройте «Цент управления сетями и общим доступом», а затем щелкните **Рабочая сеть**, **Сеть пред-**

приятия или **Общественная сеть**. В появившемся окне (рис. 1.2) выберите требуемое сетевое размещение.

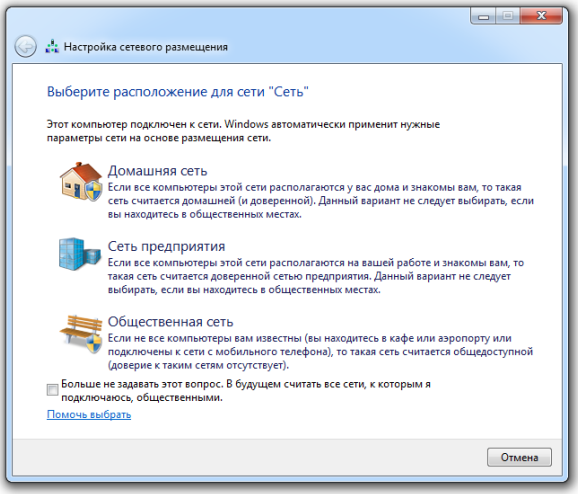


Рис. 1.2. Настройка сетевого размещения

Рабочие и домашние группы

Компьютеры под управлением Windows в сети должны быть частью рабочей группы. В составе одной рабочей группы обычно насчитывается не больше двадцати компьютеров. При этом все компьютеры должны находиться в одной локальной сети или подсети.

Компьютеры под управлением Windows 7 могут также быть частью домашней группы, но это не обязательно. Домашняя группа позволяет упростить совместное использование файлов. Кроме того в отличие от рабочей группы домашнюю группу можно защитить паролем, который можно изменить в любое время.

Присоединение к рабочей группе или создание рабочей группы

При настройке сети Windows автоматически создается рабочая группа, которой присваивается имя. Можно как присоединиться к уже существующей рабочей группе в сети, так и создать новую рабочую группу. Для этого нужно выполнить следующие действия:

1. Нажмите кнопку **Пуск**, щелкните правой кнопкой мыши на значке **Компьютер** и выберите пункт **Свойства**. Откроется окно «Система».

2. В группе **Имя компьютера, имя домена** и **параметры рабочей группы** нажмите кнопку **Изменить параметры**. Если отображается запрос на ввод пароля администратора или его подтверждения, укажите пароль или предоставьте подтверждение.
3. В диалоговом окне «Свойства системы» перейдите на вкладку **Имя компьютера** и затем нажмите кнопку **Изменить.**
4. В открывшемся диалоговом окне (рис. 1.3) введите в поле **Рабочая группа** название рабочей группы.
   - Чтобы присоединиться к существующей рабочей группе, введите имя рабочей группы, к которой будет присоединен компьютер, а затем нажмите **ОК**.
   - Чтобы создать новую рабочую группу, введите имя новой рабочей группы, а затем нажмите **ОК**.

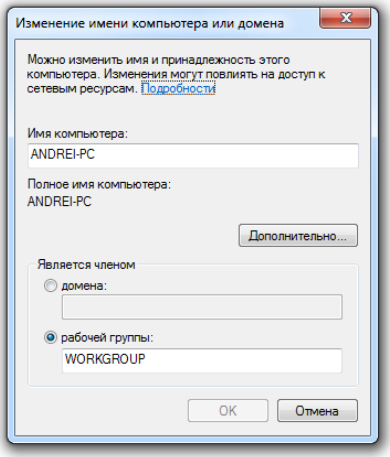


Рис. 1.3. Диалоговое окно «Изменение имени компьютера или домена»

Создание домашней группы

Для создания домашней группы необходимо открыть «Центр управления сетями и общим доступом». В разделе **Просмотр активных сетей** для соответствующей сети выполните одно из следующих действий:

- Если выбрано сетевое размещение **Домашняя сеть**, нажмите **Готовность к созданию**. В открывшемся окне нажмите кнопку **Создать домашнюю группу**.

– Если выбрано сетевое размещение **Сеть предприятия** или **Общественная сеть**, выберите **Домашняя сеть**.

В появившемся окне «Создание домашней группы» (рис. 1.4) для начала предлагается определиться с тем, к каким данным необходимо предоставить общий доступ. На этом этапе невозможно добавить в список общедоступных элементов отдельные папки, можно лишь указать, нужно ли открывать общий доступ к принтерам, а также к стандартным библиотекам. Выберите элементы, к которым нужно предоставить общий доступ, и нажмите кнопку **Далее**.

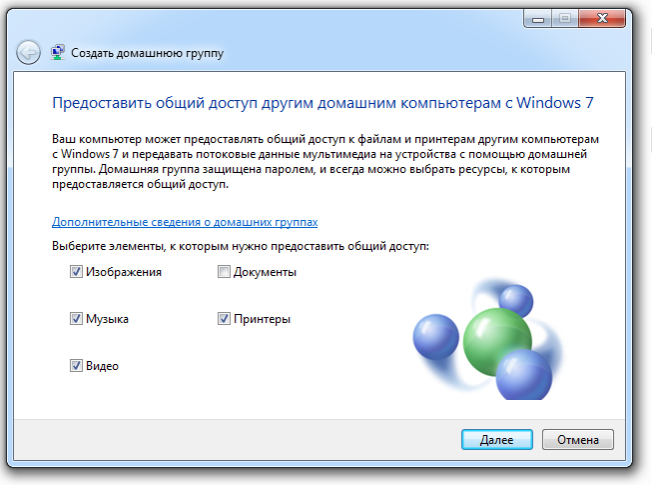


Рис. 1.4. Выбор элементов общего доступа в домашней группе

Создание домашней группы займет несколько секунд, после чего Windows сгенерирует пароль (рис. 1.5), который потребуется для добавления других компьютеров в домашнюю группу. Указать собственный пароль на данном этапе невозможно, однако его можно позже изменить в настройках домашней группы. После нажатия кнопки **Готово** откроется окно настроек домашней группы, в котором можно будет изменить параметры общего доступа к разным типам данных, просмотреть пароль или изменить его.

Присоединение к домашней группе

Для того чтобы присоединить компьютер к домашней группе откройте «Центр управления сетями и общим доступом». В разделе **Просмотр активных сетей** для соответствующей сети выполните одно из следующих действий:

- Если выбрано сетевое размещение **Домашняя сеть**, нажмите **Может присоединиться**. В открывшемся окне нажмите кнопку **Присоединиться**.
- Если выбрано сетевое размещение **Сеть предприятия** или **Общественная сеть**, выберите **Домашняя сеть**.

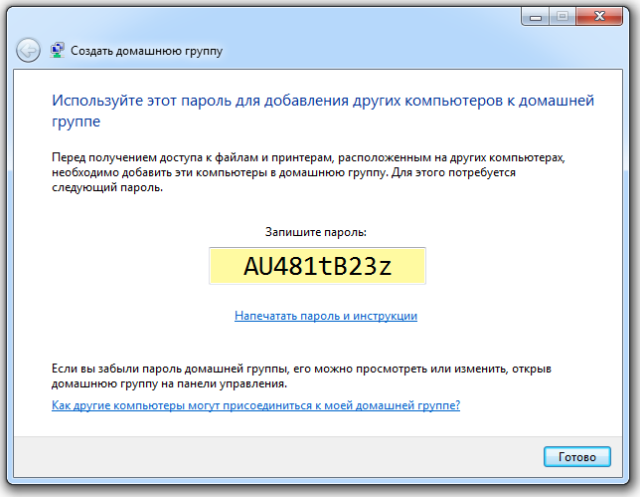


Рис. 1.5. Пароль для добавления компьютеров к домашней группе

В появившемся окне «Присоединиться к домашней группе» выберите с элементы, к которым нужно предоставить общий доступ: принтеры и стандартные библиотеки, и нажмите кнопку **Далее**. Затем введите пароль домашней группы и нажмите **Далее**. Присоединение к домашней группы займет несколько секунд, после чего нажмите кнопку **Готово**.

Настройка общего доступа

Для того чтобы предоставить общий доступ к какой-либо папке, нужно кликнуть по ней правой кнопкой мыши и выбрать пункт **Свойства**. В открывшемся диалоговом окне (рис. 1.6) на вкладке **Доступ** нажмите кнопку **Общий доступ**. Затем в диалоговом окне «Общий доступ к файлам» (рис. 1.7) нужно выбрать пользователей и предоставить им полный (чтение и запись) или ограничивающий уровень доступа (только чтение) и нажать кнопку **Общий доступ**. После этого откроется следующее окно с сообщением, что выбранная папка доступна для общего пользования. Нажмите кнопку **Готово**.

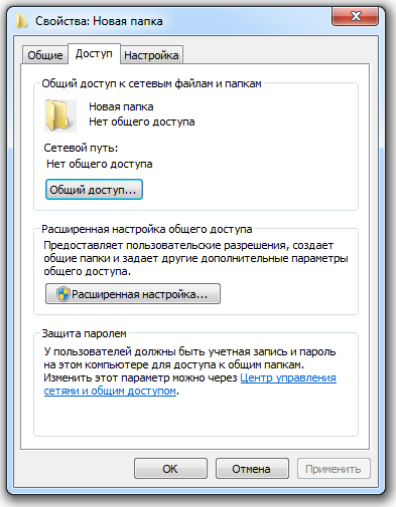


Рис. 1.6. Настройка общего доступа к папке

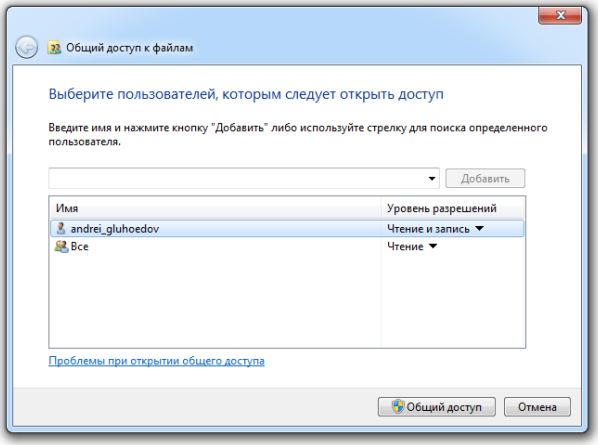


Рис. 1.7. Диалоговое окно «Общий доступ к файлам»

Также можно настроить общий доступ к папке через расширенные настройки. Для этого на вкладке **Доступ** (см. рис. 1.6) нажмите кнопку **расширенная настройка**. В открывшемся диалоговом окне (рис. 1.8) нужно выбрать **Открыть общий доступ к этой папке**, указать имя общего ресурса, задать разрешения и нажать кнопку **ОК**. При этом можно ограничить количество пользователей, которые могут одновре-

менно подключиться к общей папке (в Windows 7 предельное число пользователей не может превышать 20). Для того чтобы установить разрешения нажмите кнопку **Разрешения**. С помощью кнопок **Добавить** или **Удалить** (рис. 1.9) необходимо добавить или удалить группы или пользователей, а затем установить соответствующие флажки, которые позволяют разрешить или запретить доступ определенным пользователям или группам. Разрешения для выбранного пользователя или группы отображаются в поле **Разрешения для**.

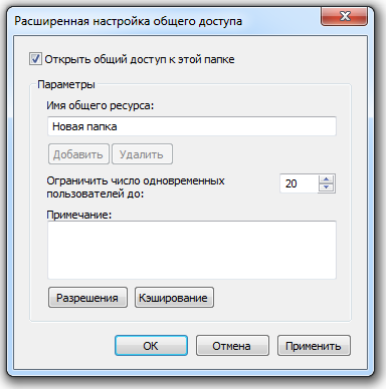


Рис. 1.8. Расширенная настройка общего доступа к папке

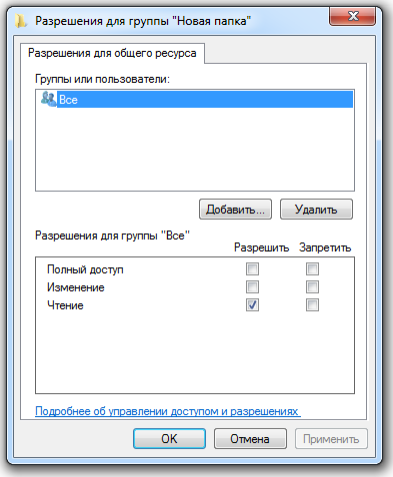


Рис. 1.9. Настройка разрешений для папки

Можно выбрать **Разрешить**, **Запретить** или ни одного из предложенных вариантов для каждой следующей записи по управлению доступом.

- **Полный доступ** позволяет пользователям создавать, читать, изменять, переименовывать и удалять файлы и папки.
- **Изменение** позволяет пользователям читать, изменять, переименовывать и удалять файлы и папки, однако создавать новые запрещено.
- **Чтение** позволяет пользователям читать файлы, однако записывать или удалять их запрещено.

Членство пользователей в группе позволяет им наследовать разрешения, назначенные группе. Пользователь может входить в одну или несколько групп.

Задания к работе

1. Создайте новую рабочую группу.
2. Создайте на компьютере две новые папки и предоставьте к ним общий доступ для всех:
   - для первой папки разрешите только чтение;
   - для второй папки разрешите полный доступ.
3. Создайте новую домашнюю группу.
4. Создайте на компьютере еще две новые папки и предоставьте к ним общий доступ для домашней группы:
   - для первой папки разрешите только чтение;
   - для второй папки разрешите чтение и запись.
5. Протестируйте с другого компьютера доступ по сети ко всем созданным папкам.

Содержание отчета

Отчет о выполнении работы должен включать в себя постановку задачи и описание проделанных операций с иллюстрациями.

Контрольные вопросы

1. Что такое информационная сеть?

2. Где применяется информационная сеть?
3. Как в операционных системах Windows осуществляется установка и настройка сети?
4. Что в операционных системах Windows называется рабочей группой?
5. Что в операционных системах Windows называется рабочей группой?
6. Как присоединить компьютер к рабочей группе?
7. Как создать новую рабочую группу?
8. Что в операционных системах Windows называется домашней группой?
9. Как присоединить компьютер к домашней группе?
10. Как создать новую домашнюю группу?
11. Как в операционных системах Windows осуществляется настройка общего доступа?
12. Какие существуют разрешения на доступ к ресурсам компьютера?

ЛАБОРАТОРНАЯ РАБОТА №2. ПРОЕКТИРОВАНИЕ СЕТЕЙ ETHERNET

Цель работы

Приобретение практических знаний и навыков в проектировании локальных сетей.

Краткие теоретические сведения

Расчетные методы оценки конфигурации сети

Для обеспечения соответствия требованиям спецификаций Ethernet в локальной сети должны одновременно выполняться два условия:

1. Задержка детектирования коллизий не должна превышать 575.
2. Сокращение межпакетного интервала не должно превышать 49.

Расчет времени задержки детектирования коллизий

Задержка детектирования коллизий (PDV) определяется продолжительностью передачи кадра по самому длинному пути. Если кадр передается через сеть слишком долго, станция может полностью завершить передачу своего кадра, не заметив того, что среда уже используется для передачи другой станцией (конфликт).

При расчете PDV нужно принимать во внимание задержки распространения сигналов в повторителях, приемопередатчиках и в различных физических средах. Для упрощения расчетов используются справочные данные из таблицы 2.1, содержащие значения этих задержек.

Таблица 2.1. Значения задержек, вносимых сегментами сети

Тип сегмента	Начальный сегмент	Промежуточный сегмент	Конечный сегмент	Задержка распространения на 1 м
10Base-5	11.8	46.5	169.5	0.0866
10Base-2	11.8	46.5	169.5	0.1026
10Base-T	15.3	42.0	165.0	0.113
AUI	0	0	0	0.1026

Начальным сегментом называется сегмент, в котором начинается путь сигнала от передатчика конечного узла. Затем сигнал проходит через промежуточные сегменты и доходит до приемника наиболее удаленного узла наиболее удаленного сегмента, который называется

конечным. С каждым сегментом связана постоянная задержка, которая зависит только от типа сегмента и от положения сегмента на пути сигнала (начальный, промежуточный или конечный). Кроме этого, с каждым сегментом связана задержка распространения сигнала вдоль кабеля сегмента, которая зависит от длины сегмента и вычисляется путем умножения времени распространения сигнала в одном метре кабеля на длину кабеля в метрах.

Значение PDV равно сумме задержек во всех сегментах сети:

```
PDV  = (Начальный сегмент + Задержка распространения * Длина)
     + (Промежуточный сегмент + Задержка распространения * Длина)
     + ...
     + (Промежуточный сегмент + Задержка распространения * Длина)
     + (Конечный сегмент + Задержка распространения * Длина)
```

Если крайние сегменты самого длинного пути при приеме и передаче различаются, нужно рассчитать PDV для обоих направлений и выбрать наибольшее значение.

Расчет сокращения межпакетного интервала

Этот расчет показывает насколько сократиться интервал между двумя последовательными кадрами, переданными по самому длинному пути. Сокращение межпакетного интервала (PVV) определяется изменением длины кадра в начальном и промежуточном сегментах. В конечном сегменте межпакетный интервал уже не меняется, так как кадр доходит по нему до принимающего узла без прохождения повторителей.

Значение PVV равно сумме величин из таблицы 2.2 для сегментов, входящих в путь максимальной длины:

```
PVV  = Начальный сегмент
     + Промежуточный сегмент
     + ...
     + Промежуточный сегмент
```

Таблица 2.2. Величины сокращения межпакетного интервала

Тип сегмента	**Начальный сегмент**	**Промежуточный сегмент**
10Base-5	16	11
10Base-2	16	11
10Base-T	10.5	8

Создание схемы сети с помощью Microsoft Office Visio

В Microsoft Office Visio для создания практически всех документов можно воспользоваться тремя основными действиями:

- выбор и открытие шаблона;
- перетаскивание и соединение фигур;
- добавление текста в фигуры.

Фигуры Visio представляют собой готовые изображения, которые перетаскиваются на страницу документа. К каждой фигуре можно добавить данные, введя их в окне «Данные фигуры» (**Вид → Области задач**). Также можно импортировать данные из внешнего источника.

При перетаскивании фигуры из набора элементов исходная фигура остается в наборе. Исходная фигура называется образцом. Фигура, которая помещается в документ, является копией – так называемым экземпляром фигуры-образца. В документ можно поместить сколько угодно экземпляров одной и той же фигуры.

Фигуры в каждом наборе элементов имеют схожие черты. В этих наборах могут содержаться коллекции фигур для создания определенного типа диаграммы или несколько различных видов одной и той же фигуры.

Например, набор элементов **Фигуры простой блок-схемы** содержит только основные фигуры блок-схем. Наборы элементов отображаются в окне «Фигуры». Чтобы просмотреть фигуры в конкретном наборе элементов, щелкните соответствующий заголовок.

Создание принципиальной и подробной схемы сети

Принципиальные схемы сети отображают главные составляющие сети и их соединение. Отличие шаблона принципиальной схемы сети от шаблона подробной схемы заключается в том, что в шаблоне «Подробная схема сети» находится больше фигур сети. На рис. 2.1 представлена схема сети.

Создание схемы сети в Microsoft Office Visio осуществляется с помощью следующей последовательности действий:

1. В меню **Файл** последовательно выберите команды **Создать**, **Сеть**, а затем – команду **Принципиальная схема сети** или **Подробная схема сети**.
2. Из набора элементов **Сетевые и периферийные устройства** перетащите на страницу документа фигуру **Кольцевая сеть** или **Ethernet**.

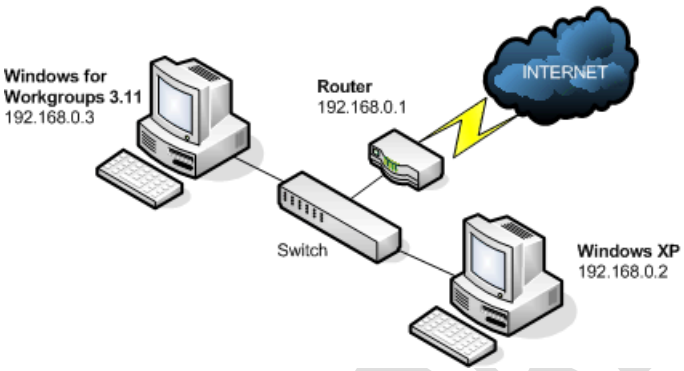


Рис. 2.1. Пример схемы сети

3. Из набора элементов **Компьютеры и мониторы** или **Сетевые и периферийные устройства** перетащите на страницу документа фигуры сетевых устройств.
4. Присоедините устройства к фигуре **Кольцевая сеть** или **Ethernet** с помощью встроенных соединительных линий фигуры:
   - Щелкните фигуру **Кольцевая сеть** или **Ethernet**.
   - Поместите указатель на управляющий маркер ◇. Когда указатель сменится на четырехстороннюю стрелку, перетащите его к точке соединения одной из фигур устройств. Точка соединения станет красной, обозначая, что фигура присоединена.
5. Чтобы соединять фигуры сетевых устройств между собой можно использовать инструмент **Соединительная линия** или фигуру **Динамическая соединительная линия**. Если применено соединение между точками, то при перемещении одной из фигур соединительная линия остается приклеенной к прежним точкам подключения.
6. Чтобы к фигуре сети добавить текст, щелкните эту фигуру и введите нужный текст. Чтобы переместить текст, перетащите управляющий маркер ◇.
7. Чтобы сохранить данные с фигурой, в меню **Вид** выберите команду **Области задач**, а затем в окне «Данные фигуры» введите значения данных, которые требуется сохранить.

Задание к работе

Согласно варианту задания спроектируйте локальную сеть, принимая во внимание возможность увеличения числа компьютеров. При проектировании необходимо решить следующие задачи:

1. Определить топологию сети и тип кабельной системы.
2. Подобрать необходимое сетевое оборудование.
3. Разработать подробную схему сети.
4. Рассчитать PDV и PVV.

Содержание отчета

Отчет о выполнении работы должен включать в себя:

1. Постановку задачи.
2. Описание выбранного типа кабеля, а также сетевого и монтажного оборудования с описанием и иллюстрациями. Обязательно необходимо указать суммарную длину кабеля и необходимое количество сетевого и монтажного оборудования.
3. Подробную схему сети.
4. Результаты расчетов PDV и PVV.

Варианты задания

№	Спецификация Ethernet	Количество комнат	Расстояние между соседними комнатами (м)	Число компьютеров в каждой комнате
1, 16	10Base-5	3	30	5
2, 17	10Base-2	4	40	3
3, 18	10Base-T	5	50	5
4, 19	10Base-5	3	30	4
5, 20	10Base-2	4	40	3
6, 21	10Base-T	5	50	6
7, 22	10Base-5	3	30	3
8, 23	10Base-2	4	40	4

№	Спецификация Ethernet	Количество комнат	Расстояние между соседними комнатами (м)	Число компьютеров в каждой комнате
9, 24	10Base-T	5	50	3
10, 25	10Base-5	3	30	4
11, 26	10Base-2	4	40	5
12, 27	10Base-T	5	50	6
13, 28	10Base-5	3	30	3
14, 29	10Base-2	4	40	5
15, 30	10Base-T	5	50	4

Контрольные вопросы

1. Как можно классифицировать сети по размеру сети?
2. Что называют топологией сети? Как можно классифицировать сети по типу топологии сети?
3. Как можно классифицировать сети по типу функционального взаимодействия?
4. Как можно классифицировать сети по типу технологии передачи?
5. Как можно классифицировать сети по типу среды передачи?
6. Как можно классифицировать сети по скорости передачи?
7. Что называют сетевыми устройствами?
8. Что такое сетевой адаптер? Для чего предназначены сетевые адаптеры?
9. Что называют пассивными сетевыми устройствами?
10. Что такое повторитель? Для чего предназначены повторители?
11. Что такое концентратор? Для чего предназначены концентраторы?
12. Что называют активными сетевыми устройствами?
13. Что такое коммутатор? Для чего предназначены коммутаторы?
14. Что такое мост? Для чего предназначены мосты?
15. Что такое маршрутизатор? Для чего предназначены маршрутизаторы?
16. Что такое линия и канал связи?
17. Что такое кабельные и беспроводные линии связи?
18. На какие типы делятся каналы связи в зависимости от того могут ли они передавать данные в обоих направлениях или нет?
19. Какие сети образуют каналы связи?

20. Что такое витая пара? Какие существуют разновидности и категории витой пары?
21. Как осуществляется обжим витой пары?
22. Что такое коаксиальный кабель? Какие существуют типы коаксиального кабеля?
23. Как осуществляется монтаж коаксиального кабеля?
24. Что такое оптоволоконный кабель? Какие существуют типы оптоволоконного кабеля?
25. Как осуществляется монтаж оптоволоконного кабеля?
26. Как строятся беспроводные линии связи? Какие типы антенн применяются в беспроводных линиях связи?
27. Какие выделяют основные частотные диапазоны радиоволн?
28. Что такое спутниковая связь?
29. Что называют сетевой технологией? Что определяет сетевая технология?
30. Какие существуют сетевые технологии?
31. Что такое метод доступа CSMA/CD? Где используется этот метод доступа?
32. Что такое маркерный метод доступа? Где используется этот метод доступа?
33. Что такое метод доступа CSMA/CA? Где используется этот метод доступа?
34. В чем заключается спецификация Ethernet 10Base-5?
35. В чем заключается спецификация Ethernet 10Base-2?
36. В чем заключается спецификация Ethernet 10Base-T?
37. Какие существуют расчетные методы оценки конфигурации локальной сети для обеспечения соответствия требованиям спецификаций Ethernet?

ЛАБОРАТОРНАЯ РАБОТА №3. АДРЕСАЦИЯ И МАРШРУТИЗАЦИЯ В IP-СЕТЯХ

Цель работы

Приобретение практических знаний и навыков в настройке локальных сетей средствами Microsoft Windows 7 и Windows Server 2008.

Краткие теоретические сведения

Хотя операционные системы Window 7 и Windows Server 2008 имеют немало общих компонентов, многие сетевые службы (по понятным причинам) доступны только в Windows Server 2008. Все основные сетевые службы в Windows Server 2008 собраны в окне «Диспетчер сервера» (рис. 3.1).

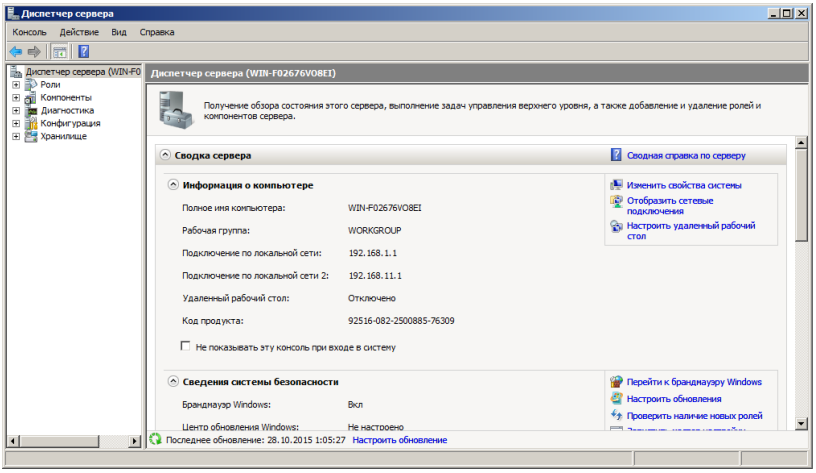


Рис. 3.1. Диспетчер сервера

Следует отметить, что начиная с Windows Server 2008 все основные сетевые службы устанавливаются в качестве ролей сервера, а не в качестве компонентов операционной системы, как это было в предыдущих серверных версиях Windows.

Открытие диспетчера сервера

Окно «Диспетчер сервера», как правило, открывается автоматически при входе администратора на компьютер с операционной системой

Windows Server 2008. Если «Диспетчер сервера» был закрыт, его можно открыть повторно следующими способами:

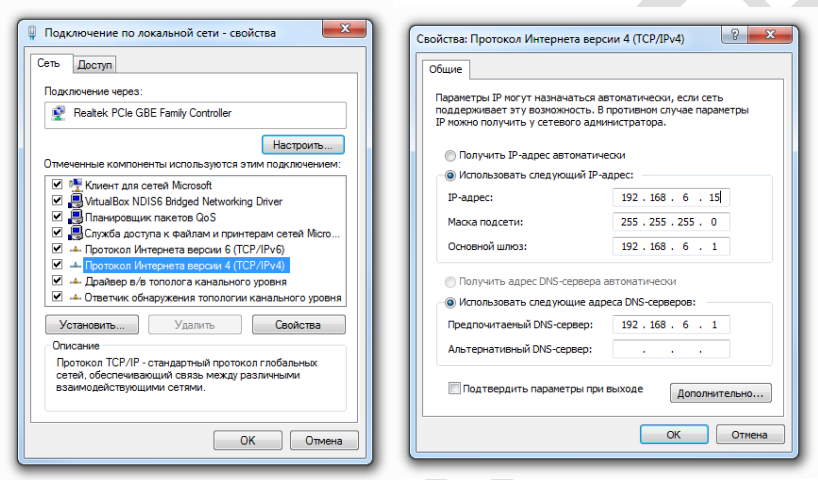
- Нажать кнопку **Пуск**, щелкнуть правой кнопкой мыши на значке **Компьютер** и выбрать пункт **Управление**.
- Нажать кнопку **Пуск**, выбрать пункт **Администрирование**, а затем – пункт **Диспетчер сервера**.
- Из панели быстрого запуска, расположенной в панели задач Windows.

Настройка TCP/IP в Windows

Для упрощения управления параметрами TCP/IP рекомендуется использовать протокол DHCP. Для включения DHCP или изменения других параметров TCP/IP на компьютере под управлением операционной системой Windows 7 выполните следующие действия.

1. Откройте «Панель управления» и щелкните левой кнопкой мыши на значке **Сетевое окружение**. Откроется системная папка «Сетевые подключения».
2. Щелкните правой кнопкой мыши изменяемое подключение и затем выберите **Свойства**. Откроется диалоговое окно, показанное на рис. 3.2, *а*.
3. На вкладке **Сеть** в разделе **Отмеченные компоненты, используемые этим подключением** выберите **Протокол Интернета версии 4 (TCP/IPv4)**, а затем нажмите кнопку **Свойства**. Откроется диалоговое окно «Свойства: Протокол Интернета версии 4 (TCP/IPv4)», показанное на рис. 3.2, *б*.
4. Чтобы настроить параметры IP-адреса, выполните одно из следующих действий:
   - Для автоматической настройки параметров IP с помощью DHCP выберите **Получить IP-адрес автоматически**.
   - Для установки IP-адреса вручную выберите **Использовать следующий IP-адрес**, а затем в поля **IP-адрес**, **Маска подсети**, и **Основной шлюз** введите соответствующие значения.
5. Чтобы указать адреса DNS-сервера, выполните одно из следующих действий:
   - Для автоматического получения адреса DNS-сервера с помощью DHCP выберите **Получить адрес DNS-сервера автоматически**.

– Для установки адреса DNS-сервера вручную щелкните **Использовать следующие адреса DNS-серверов**, а затем в поля **Предпочитаемый DNS-сервер** и **Альтернативный DNS-сервер** введите соответственно IP-адреса первичного и вторичного DNS-серверов.



а ***б***

Рис. 3.2. Диалоговое окно: *а* – свойства сетевого подключения; *б* – свойства протокола Интернета версии 4 (TCP/IPv4)

6. Чтобы изменить дополнительные параметры DNS, WINS и IP, нажмите кнопку **Дополнительно**.
7. Нажмите кнопку **ОК** для сохранения настроек и закройте окно «Свойства: Протокол Интернета версии 4 (TCP/IPv4)».
8. Нажмите кнопку **ОК**, чтобы закрыть окно «Свойства сетевого подключения».

Изменение параметров TCP/IP в Windows Server 2008 осуществляется аналогичным образом в окне «Сетевые подключения». Для того, чтобы открыть окно «Сетевые подключения» в Windows Server 2008 нужно выполнить одно из следующих действий:

– откройте «Центр управления сетями и общим доступом» и выберите команду **Управление сетевыми подключениями**;
– откройте «Диспетчер сервер», а затем в разделе **Диспетчер сервер** выберите команду **Отобразить сетевые подключения**.

Установка и настройка DNS-сервера

Процесс установки DNS-сервера в Windows Server 2008 довольно прост и не требует перезагрузки системы. Ниже перечислены шаги, необходимые для установки DNS-сервера на компьютере под управлением операционной системой Windows Server 2008.

1. Откройте «Диспетчер сервера», а затем в разделе **Роли** выберите команду **Добавить роли**.
2. Нажмите кнопку **Далее**, чтобы пропустить страницу **Перед началом работы**.

   *Если ранее выбрана команда **Пропустить эту страницу по умолчанию**, эта страница не отображается.*
3. На странице **Выбор ролей сервера** выберите роль **DNS-сервер** и нажмите кнопку **Далее**.
4. Нажмите кнопку **Далее**, чтобы пропустить страницу **DNS-сервер**.
5. На странице **Подтверждение** нажмите кнопку **Установить**, чтобы запустить процесс установки DNS-сервера.
6. По завершении процесса установки нажмите кнопку **Закрыть**, чтобы выйти из мастера добавления ролей.

После выполнения этих шагов DNS-сервер будет установлен, но не сконфигурирован. Далее описаны шаги, необходимые для его настройки.

1. Нажмите кнопку **Пуск**, выберите пункт **Администрирование**, а затем – пункт **DNS**.
2. В меню **Действие** выберите пункт **Конфигурация DNS-сервера...**.
3. Нажмите кнопку **Далее**, чтобы пропустить страницу приветствия мастера настройки сервера DNS.
4. Выберите пункт **Создать зоны прямого и обратного просмотра (рекомендуется для больших сетей)**, как показано на рис. 3.3, и нажмите кнопку **Далее**.
5. Выберите вариант **Да, создать зону прямого просмотра сейчас (рекомендуется)** и нажмите кнопку **Далее**.
6. Укажите, зону какого типа требуется создать, в данном случае выбрав вариант **Основная зона**, и нажмите кнопку **Далее**.
7. Введите полностью определенное доменное имя зоны в поле **Имя зоны** и нажмите кнопку **Далее**.

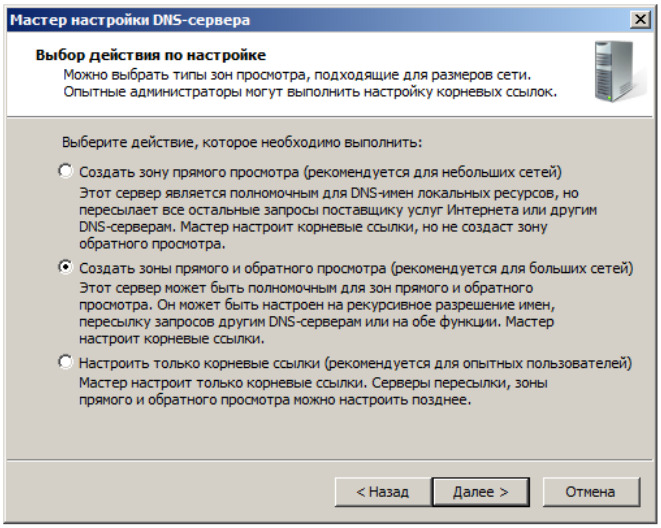


Рис. 3.3. Окно мастера настройки DNS-сервера

8. На этом этапе можно либо создать новый файл для зоны, либо импортировать уже существующий. В данном случае выберите вариант **Создать новый файл с таким именем** и оставьте предлагаемые по умолчанию параметры, после чего нажмите кнопку **Далее**.
9. На следующей странице будет предложено разрешить или запретить прием динамических обновлений сервером DNS. В рассматриваемом примере выберите пункт **Запретить динамические обновления** и нажмите кнопку **Далее**.

   Разрешая DNS-серверу принимать динамические обновления, нужно быть уверенным в надежности источников информации. В случае ненадежности этих источников существует риск повреждения или искажения данных из-за динамического обновления.
10. На следующе странице предлагается создать зону обратного просмотра. В данном случае выберите пункт **Да, создать зону обратного просмотра сейчас** и нажмите кнопку **Далее**.
11. Укажите, что зона обратного просмотра должна представлять собой основную зону, выбрав пункт **Основная зона**, и нажмите кнопку **Далее**.
12. Оставьте выбранным предлагаемый вариант **Зона обратного просмотра IPv4** и нажмите кнопку **Далее**.

13. Выделите идентификатор сети для зоны обратного просмотра, например как показано на рис. 3.4 и нажмите кнопку **Далее**.

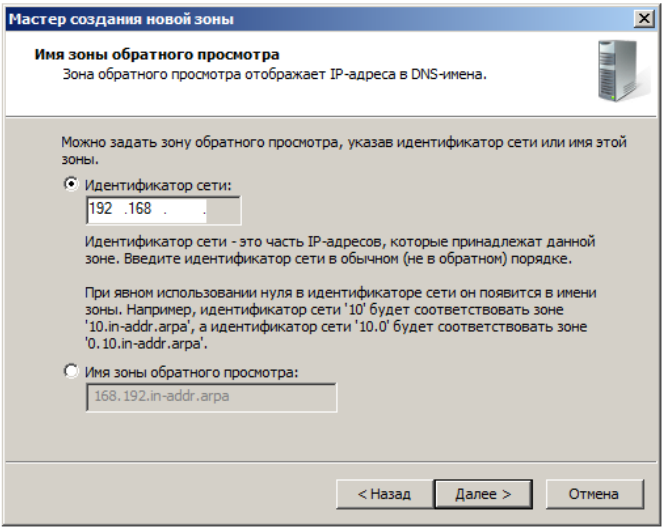


Рис. 3.4. Идентификатор сети для зоны обратного просмотра DNS

14. На этом этапе можно либо создать новый файл для зоны, либо импортировать уже существующий. В данном случае выберите вариант **Создать новый файл с таким именем** и нажмите кнопку **Далее**.
15. После этого будет предложено разрешить или запретить динамические обновления. В рассматриваемом примере выберите пункт **Запретить динамические обновления** и нажмите кнопку **Далее**.
16. На следующей странице будет предложено настроить параметры ретрансляторов. В рассматриваемом примере выберите пункт **Нет, не пересылать запросы** и нажмите кнопку **Далее**.
17. На последней странице будут представлены сводные сведения о выбранных для внесения и добавления в базу данных DNS изменениях и зонах. Нажмите кнопку Готово, чтобы внести все эти изменения и создать нужные зоны.

    *Если отсутствует подключение сети Интернет, отобразится диалоговое окно с сообщением об ошибке, связанное с поиском корневых ссылок. Но, несмотря на указанную ошибку, нажатие кнопки **ОК** в этом окне приводит к успешной настройке DNS.*

Установка и настройка DHCP-сервера

Установка DCHP-сервера в Windows Server 2008 очень проста. Ниже перечислены шаги, необходимые для установки DHCP-сервера на компьютере под управлением операционной системой Windows Server 2008.

1. Откройте «Диспетчер сервера», а затем в разделе **Роли** выберите команду **Добавить роли**.
2. Нажмите кнопку **Далее**, чтобы пропустить страницу **Перед началом работы**.

   *Если ранее выбрана команда **Пропустить эту страницу по умолчанию**, эта страница не отображается.*
3. На странице **Выбор ролей сервера** выберите роль **DHCP-сервер** и нажмите кнопку **Далее**.

   Если компьютеру не присвоен IP-адрес, вы получите предупреждение о том, что нельзя устанавливать DHCP-сервер с динамическим IP-адресом.
4. Нажмите кнопку **Далее**, чтобы пропустить страницу **DHCP-сервер**.
5. Выберите сетевые подключения, для которых будет использоваться DHCP-сервер, и нажмите кнопку **Далее**.
6. Заполните поля **Родительский домен**, **IPv4-адрес основного DNS-сервера** и **IPv4-адрес дополнительного DNS-сервера** (необязательно), а затем нажмите кнопку **Далее**.

   Значения этих полей будут назначаться соответствующим параметрам TCP/IP для DHCP-клиентов.
7. Выберите вариант **WINS не требуется для приложений в этой сети** и нажмите кнопку **Далее**.
8. На странице **Области DHCP** (рис. 3.5) предлагается добавить области DHCP. Необходима как минимум одна область. Чтобы создать новую область нажмите кнопку **Добавить** и выполните следующие действия:
   - В поле **Имя области** введите название новой DHCP-области, можно вводить любое.
   - В поля **Начальный IP-адрес** и **Конечный IP-адрес** введите диапазон IP-адресов, из которого будут назначаться адреса для DHCP-клиентов.

- В поля **Маска подсети** и **Основной шлюз** введите значения, которые будут назначаться соответствующим параметрам TCP/IP для DHCP-клиентов.
- В поле **Тип подсети** выберите тип, чтобы указать время аренды для назначаемых IP-адресов.
- Нажмите кнопку **ОК**, чтобы создать новую область DHCP.

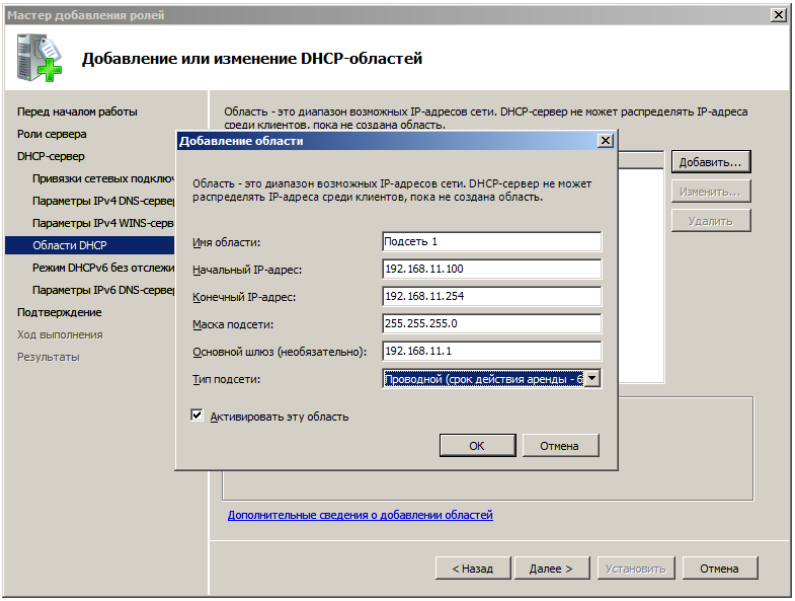


Рис. 3.5. Добавление новой DHCP-области

9. На следующей странице идет речь об использовании протокола DHCPv6. Поскольку в данный момент DHCP-сервер не будет использоваться для обслуживания клиентов IPv6, то выберите вариант **Отключить режим без отслеживания состояния DHCPv6 для этого сервера** и нажмите кнопку **Далее**.
10. На странице **Подтверждение** нажмите кнопку **Установить**, чтобы запустить процесс установки сервера DNS.
11. По завершении процесса установки нажмите кнопку **Закрыть**, чтобы выйти из мастера добавления ролей.

Чтобы настроить DHCP-сервер и посмотреть, какие IP-адреса были назначены DHCP-клиентам, нажмите кнопку **Пуск**, выберите пункт

Администрирование, а затем – пункт **DHCP**. Откроется окно, показанное на рис. 3.6, в котором предоставлена информация об областях DHCP, включая пулы адресов, владельцев, зарезервированные адреса, а также параметры DHCP-сервера.

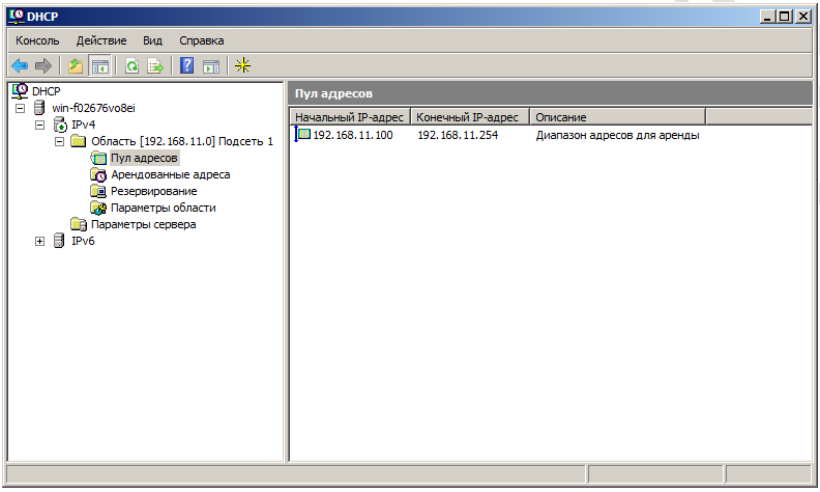


Рис. 3.6. Управление DHCP-сервером

Установка и настройка службы RRAS

Для маршрутизации в IP-сетях наряду с *аппаратными* маршрутизаторами могут использоваться *программные* маршрутизаторы. В операционной системе Window Server 2008 имеется *служба маршрутизации и удаленного доступа* (Routing and Remote Access Service, RRAS), которая представляет собой программный маршрутизатор.

Ниже перечислены шаги, необходимые для установки службы RRAS на компьютере под управлением операционной системой Windows Server 2008.

1. Откройте «Диспетчер сервера», а затем в разделе **Роли** выберите команду **Добавить роли**.
2. Нажмите кнопку **Далее**, чтобы пропустить страницу **Перед началом работы**.

   *Если ранее выбрана команда **Пропустить эту страницу по умолчанию**, эта страница не отображается.*
3. На странице **Выбор ролей сервера** выберите роль **Службы политики сети и доступа** и нажмите кнопку **Далее**.

4. Нажмите кнопку **Далее**, чтобы пропустить страницу **Службы политики сети и доступа**.
5. На странице **Выбор служб ролей** выберите пункт **Службы маршрутизации и удаленного доступа**, как показано на рис. 3.7, и нажмите кнопку **Далее**.

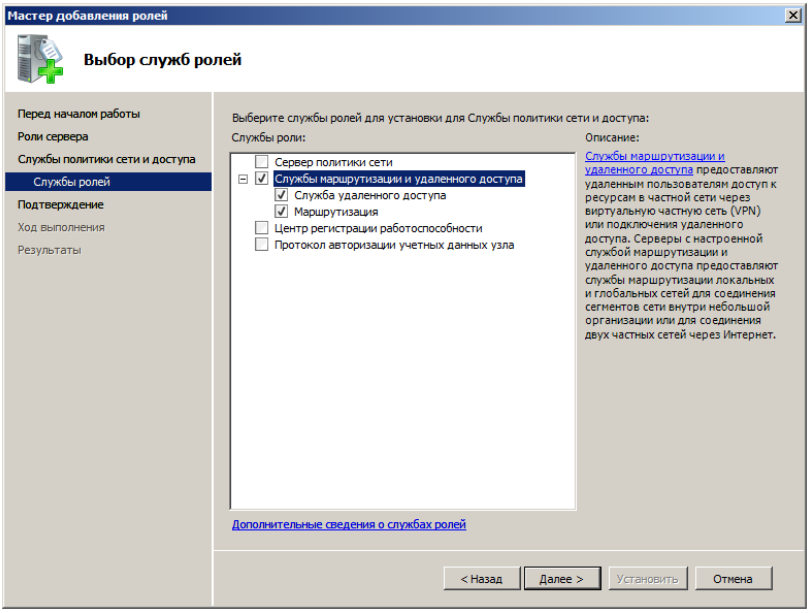


Рис. 3.7. Выбор службы маршрутизации и удаленного доступа

6. На странице **Подтвердите выбранные элементы** нажмите кнопку **Установить**, чтобы запустить процесс установки.
7. По завершении процесса установки просмотрите состояние на странице **Результаты установки**, а затем нажмите кнопку **Закрыть**, чтобы выйти из мастера добавления ролей.

После установки службы RRAS необходимо ее настроить. Далее описаны шаги, необходимые для настройки службы RRAS.

1. Нажмите кнопку **Пуск**, выберите пункт **Администрирование**, а затем – пункт **Маршрутизация и удаленный доступ**.
2. Щелкните правой кнопкой мыши имя сервера, для которого требуется включить маршрутизацию и выберите пункт **Настроить и**

включить маршрутизацию и удаленный доступ, как показано на рис. 3.8.

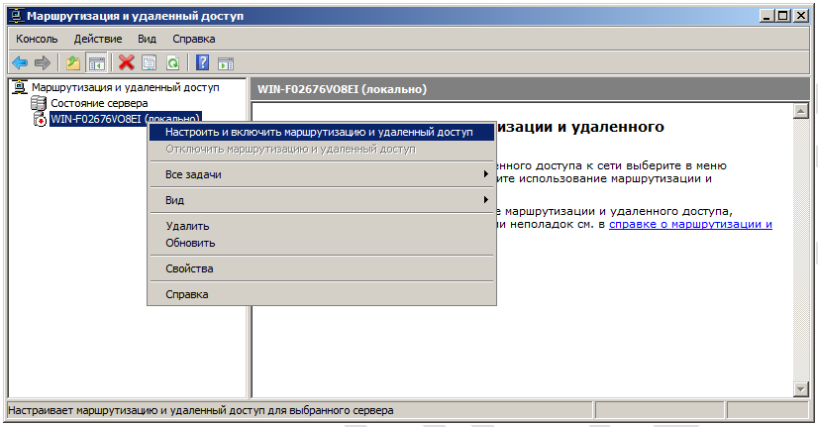


Рис. 3.8. Настройка службы RRAS

3. На странице **Мастер установки сервера маршрутизации и удаленного доступа** нажмите кнопку **Далее**.
4. На странице **Конфигурация** выберите вариант **Особая конфигурация** и нажмите кнопку **Далее**.
5. На странице **Настраиваемая конфигурация** выберите пункт **Маршрутизация локальной сети** и нажмите кнопку **Далее**.
6. На последней странице нажмите кнопку **Готово**, чтобы внести изменения и запустить службу.
7. При выводе запроса на запуск службы RRAS нажмите кнопку **Запустить службу**.

Статическая маршрутизация

Программные маршрутизаторы Windows Server 2008 имеют свою таблицу маршрутизации. Ниже перечислены шаги, необходимые для изменения и просмотра таблицы маршрутизации.

1. Нажмите кнопку **Пуск**, выберите пункт **Администрирование**, а затем – пункт **Маршрутизация и удаленный доступ**.
2. В разделе **IPv4**, щелкните правой кнопкой мыши **Статические маршруты**, а затем выберите пункт **Новый статический маршрут...**, как показано на рис. 3.9.

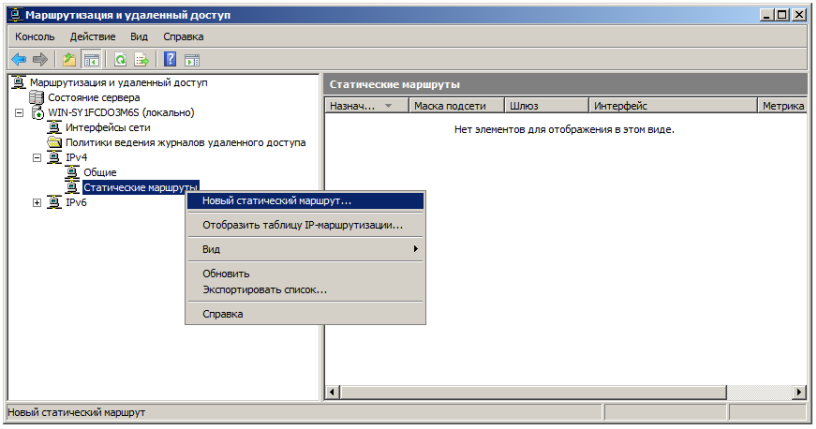


Рис. 3.9. Создание статического маршрута IPv4 в RRAS

3. В появившемся окне «Статический маршрут IPv4» заполните все необходимые поля и нажмите **ОК**, чтобы создать новый статический маршрут.
4. Для просмотра таблицы маршрутизации (рис. 3.10) щелкните правой кнопкой мыши **Статические маршруты**, а затем выберите пункт **Отобразить таблицу IP-маршрутизации...**.

Destination	Network mask	Gateway	Interface	Metric	Protocol
127.0.0.0	255.0.0.0	127.0.0.1	Loopback	51	Local
127.0.0.1	255.255.255.255	127.0.0.1	Loopback	306	Local
192.168.1.0	255.255.255.0	0.0.0.0	Local Area C...	276	Network ma...
192.168.1.1	255.255.255.255	0.0.0.0	Local Area C...	276	Network ma...
192.168.1.255	255.255.255.255	0.0.0.0	Local Area C...	276	Network ma...
192.168.2.0	255.255.255.0	0.0.0.0	Local Area C...	276	Network ma...
192.168.2.1	255.255.255.255	0.0.0.0	Local Area C...	276	Network ma...
192.168.2.255	255.255.255.255	0.0.0.0	Local Area C...	276	Network ma...
224.0.0.0	240.0.0.0	0.0.0.0	Local Area C...	276	Network ma...
255.255.255.255	255.255.255.255	0.0.0.0	Local Area C...	276	Network ma...

Рис. 3.10. Таблица маршрутизации в RRAS

Динамическая маршрутизация

В операционной системе Windows Server 2008 динамическая маршрутизация осуществляется посредством протокола RIP. По умолчанию протокол RIP отключен. Далее перечислены шаги, необходимые для включения и настройки протокола RIP.

1. Нажмите кнопку **Пуск**, выберите пункт **Администрирование**, а затем – пункт **Маршрутизация и удаленный доступ**.

2. В разделе **IPv4**, щелкните правой кнопкой мыши **Общие**, а затем выберите пункт **Новый протокол маршрутизации...**, как показано на рис. 3.11.

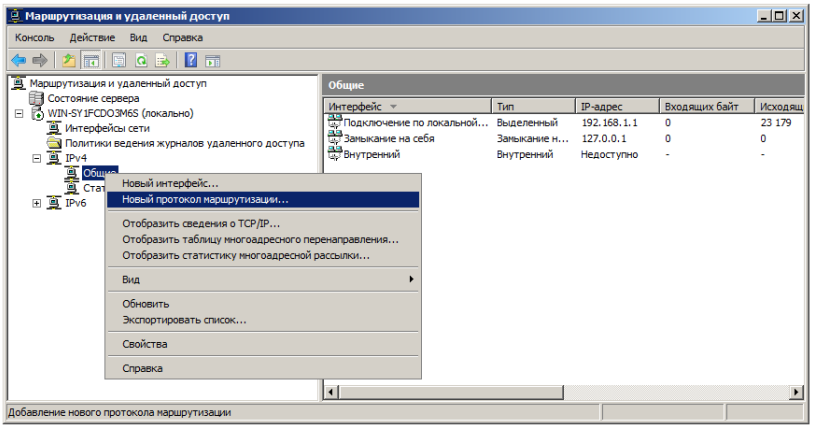


Рис. 3.11. Общие протоколы маршрутизации в RRAS

3. В появившемся окне выберите **RIP версии 2 для IP** и нажмите **ОК**. В результате в разделе **IPv4** появится подраздел **RIP**.
4. Щелкните правой кнопкой мыши **RIP**, а затем выберите пункт **Новый интерфейс...**.
5. В появившемся окне выберите интерфейс, который собираетесь использовать для динамической маршрутизации и нажмите **ОК**.

   *Будет предложено настроить параметры выбранного интерфейса. Оставьте предложенные по умолчанию параметры и нажмите кнопку **ОК**.*

Настройка агента DHCP-ретрансляции

Поскольку протокол DHCP основан на широковещательной рассылке, пакеты этого протокола по умолчанию не проходят через маршрутизаторы. Агент DHCP-ретрансляции обеспечивает обмен сообщениями DHCP между DHCP-клиентами и DHCP-серверами в различных подсетях.

Для настройки агента DHCP-ретрансляции необходимо выполнить следующие действия:

1. Нажмите кнопку **Пуск**, выберите пункт **Администрирование**, а затем – пункт **Маршрутизация и удаленный доступ**.

2. В разделе **IPv4**, щелкните правой кнопкой мыши **Общие**, а затем выберите пункт **Новый протокол маршрутизации...**.
3. В появившемся окне выберите **Агент DHCP-ретрансляции** и нажмите **ОК**. В результате в разделе **IPv4** появиться подраздел **Агент DHCP-ретрансляции**.
4. Щелкните правой кнопкой мыши **Агент DHCP-ретрансляции**, а затем выберите пункт **Новый интерфейс...**.
5. В появившемся окне выберите соответствующий сетевой интерфейс и нажмите **ОК**.
6. В диалоговом окне «Свойства DHCP-ретрансляции» выберите **Ретрансляция DHCP-пакетов** и нажмите **ОК**.
7. Щелкните правой кнопкой мыши **Агент DHCP-ретрансляции** и выберите пункт **Свойства**.
8. На вкладке **Общие** введите IP-адреса DHCP-серверов, на которые необходимо обеспечить ретрансляцию сообщений DHCP, нажмите **Добавить** и затем **ОК**.

Следует отметить, что агент DHCP-ретрансляции нельзя использовать на компьютере, на котором запущен DHCP-сервер.

Служебные программы

В состав операционных систем Windows входит целый ряд служебных программ для работы с сетью, которые можно применять для тестирования подключений и устранения неполадок. Некоторые из них будут рассмотрены в данной лабораторной работе.

Чтобы воспользоваться этими служебными программами в Windows 7 или Windows Server 2008, нужно в меню Пуск в поле поиска ввести команду `cmd`, а затем в открывшемся окне командной строки ввести название соответствующей служебной программы, указать ее параметры и нажать клавишу «`Enter`».

Команда ping

Ping представляет собой аббревиатуру от Packet InterNet Groper. Действие этой команды заключается в том, что она отправляет ICMP-пакет удаленному компьютеру и ожидает его возвращения. Если пакет возвращается, подключение между двумя компьютерами функционирует нормально. Отсутствие ответа может свидетельствовать о недоступности удаленного компьютера.

Версия `ping`, входящая в состав Windows, по умолчанию отправляет четыре пакета и ожидает ответ на каждый из них в течение 1 с.

При необходимости можно указать другое число отправляемых пакетов и другое время ожидания. Пример использования команды `ping` представлен на рис. 3.12.

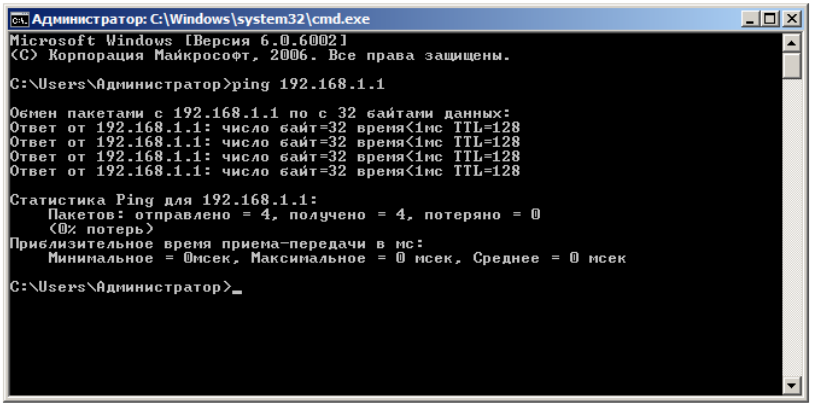


Рис. 3.12. Результат работы команды `ping`

Ниже приведен синтаксис команды `ping`:

```
ping [-t] [-a] [-n <число>] [-l <размер>] [-f]
     [-i <TTL>] [-v <TOS>] [-r <число>][-s <число>]
     [[-j <список узлов>] | [-k <список узлов>]]
     [-w <тайм-аут>][-R][-S <адрес источника>] [-4] [-6]
     <конечный узел>
```

Здесь [ ] – это опционный, т.е. необязательный, параметр, а < > обозначают сущность. Назначение параметров команды `ping` поясняется в таблице 3.1.

Таблица 3.1. Параметры команды `ping`

Параметр	**Назначение**
`-t`	Применяется для отправки пакетов по указанному узлу до тех пор, пока не будет выполнено прерывание «Ctrl + C». Для вывода статистики необходимо нажать сочетание клавиш «Ctrl + Break»
`-a`	Применяется для определения имен узлов по адресам
`-n <число>`	Применяется для задания числа отправляемых пакетов

Окончание табл. 3.1

Параметр	**Назначение**
`-l <размер>`	Применяется для задания размера отправляемых пакетов (в байтах)
`-f`	Применяется для установки флага запрещающего фрагментации отправляемых пакетов (только для IPv4)
`-i <TTL>`	Применяется для задания времени жизни пакета
`-v <TOS>`	Применяется для задания типа службы (только для IPv4 и этот параметр не доступен)
`-r <число>`	Применяется для записи маршрута для указанного числа переходов (только для IPv4)
`-s <число>`	Применяется для установки временной метки для указанного числа переходов (только для IPv4)
`-j <список узлов>`	Используется для направления пакетов по заданному маршруту; узлы могут разделяться промежуточными шлюзами (свободная маршрутизация с набором промежуточных узлов назначения)
`-k <список узлов>`	Используется для направления пакетов по заданному маршруту; узлы не могут разделяться промежуточными шлюзами (строгая маршрутизация с набором промежуточных узлов назначения)
`-w <тайм-аут>`	Применяется для задания интервала ожидания ответа (в миллисекундах)
`-R`	Применяется для отслеживания пути приема-передачи пакетов (только для IPv6)
`-S <адрес источника>`	Используется для указания адреса источника пакетов
`-4`	Применяется для указания на необходимость использования протокола IPv4
`-6`	Применяется для указания на необходимость использования протокола IPv6

Команда ipconfig

Команда `ipconfig` используется для отображения свойств всех имеющихся адаптеров, а также установки определенных параметров TCP/IP. Команда `ipconfig` применяется для определения свойств адаптеров в любой системе, однако наиболее полезна в системах, получающих параметры TCP/IP от DHCP-сервера. Пример использования команды `ipconfig` приведен на рис. 3.13.

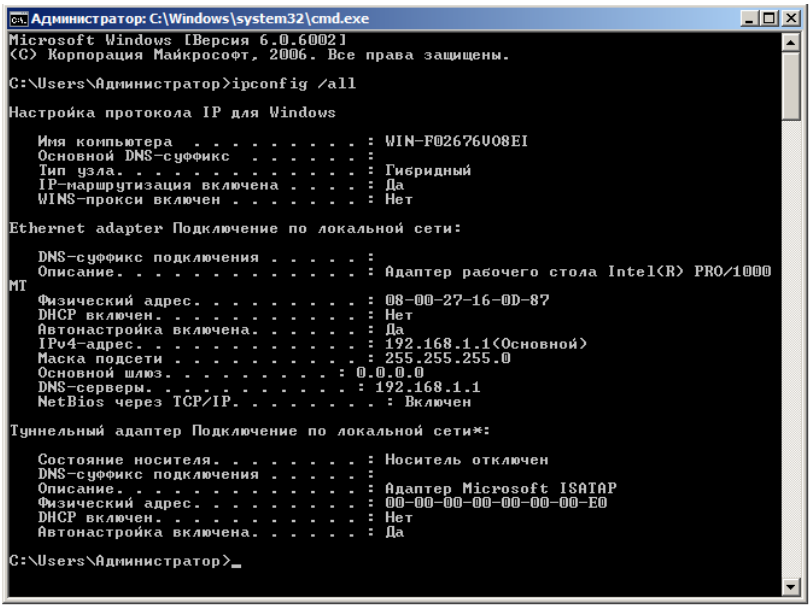


Рис. 3.13. Результат работы команды `ipconfig`

Ниже приведен синтаксис команды `ipconfig`:

```
ipconfig [/all | /renew [<адаптер>] |
         /release [<адаптер>] | /flushdns | /displaydns |
         /registerdns | /showclassid <адаптер> |
         /setclassid <адаптер> [<идентификатор класса>]]
```

Здесь `адаптер` – имя сетевого подключения (можно использовать знаки подстановки * и ?). Назначение параметров команды `ipconfig` поясняется в таблице 3.2.

Таблица 3.2. Параметры команды `ipconfig`

Параметр	**Назначение**
`/all`	Применяется для вывода подробных сведений обо всех сетевых подключениях
`/renew [<адаптер>]`	Применяется для обновления IP-адреса для указанного сетевого подключения с помощью DHCP
`/release [<адаптер>]`	Применяется для освобождения IP-адреса для указанного сетевого подключения с помощью DHCP

Окончание табл. 3.2

Параметр	**Назначение**
`/flushdns`	Применяется для очистки кэша DNS
`/displaydns`	Применяется для отображения содержимого кэша DNS
`/registerdns`	Применяется для обновления всех DHCP-аренд и перерегистрация имени в DNS
`/showclassid <`*`адаптер`*`>`	Применяется для отображения идентификатора класса DHCP для указанного сетевого подключения
`/setclassid <`*`адаптер`*`> [<`*`идентификатор класса`*`>]`	Применяется для задания идентификатора класса DHCP для указанного сетевого подключения

Команда route

Команда `route` позволяет просматривать и изменять статическую таблицу маршрутизации компьютера. Синтаксис команды `route` приведен ниже:

```
route  [-f] [-p] [print | add | delete | change]
       [<адресуемый узел>] [mask <маска подсети>]
       [<адрес шлюза>] [metric <метрика>]
       [if <номер интерфейса>]
```

Назначение параметров команды `ping` поясняется в таблице 3.3. С параметрами `print` и `delete` допускается использование символов подстановки * и `?`.

Таблица 3.3. Параметры команды `route`

Параметр	**Назначение**
`-f`	Применяется для удаления всех записей о маршрутизаторах из таблицы
`-p`	Применяется для создания постоянных маршрутов в таблице маршрутизации. По умолчанию добавленные маршруты не сохраняются при перезапуске компьютера
`print`	Применяется для вывода маршрутов
`add`	Применяется для добавления маршрута
`delete`	Применяется для удаления маршрута

Окончание табл. 3.3

Параметр	**Назначение**
`change`	Применяется для изменения существующего маршрута
`<`*адресуемый узел*`>`	Применяется для указания IP-адреса назначения маршрута
`mask <`*маска подсети*`>`	Применяется для указания маски подсети, соответствующей заданному IP-адресу назначения.. Если этот параметр не задан, по умолчанию используется значение `255.255.255.255`
`<`*адрес шлюза*`>`	Применяется для указания адреса шлюза. Для локально подключенных маршрутов, адрес шлюза – это IP-адрес сетевого интерфейса. Для удаленных маршрутов, которые доступны через один или несколько маршрутизаторов, адрес шлюза – IP-адрес ближайшего маршрутизатора
`metric <`*метрика*`>`	Применяется для задания метрики (в пределах от 1 до 9999) маршрута, которая используется при выборе в таблице маршрутизации одного из нескольких маршрутов, наиболее близко соответствующего адресуемому узлу. Метрика отражает количество переходов, скорость прохождения пути, надежность пути и пропускную способность пути
`if <`*номер интерфейса*`>`	Применяется для задания номера интерфейса, соответствующего указанному маршруту. В случае, когда параметр пропущен, номер интерфейса определяется из адреса шлюза

Просмотреть таблицу маршрутизации очень просто, достаточно вызвать команду `route print`, как показано на рис. 3.14, на котором представлен пример результата выполнения этой команды.

При выводе таблицы маршрутизации первое, на что нужно обратить внимание, это *список интерфейсов* (Interface List). Сетевые интерфейсы Windows обозначены номером интерфейса. Эти номера интерфейсов используются всегда, когда необходимо добавить или удалить маршрут.

Чтобы вывести маршруты из таблицы маршрутизации, например, которые начинаются с `127.`, как показано на рис. 3.15, необходимо вызвать команду:

```
route print 127.*
```

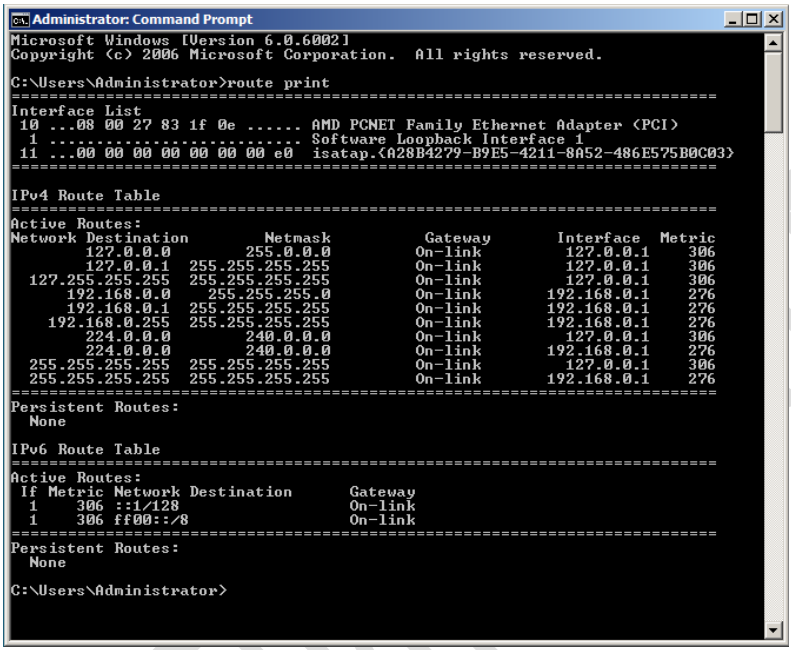


Рис. 3.14. Просмотр таблицы маршрутизации командой `route`

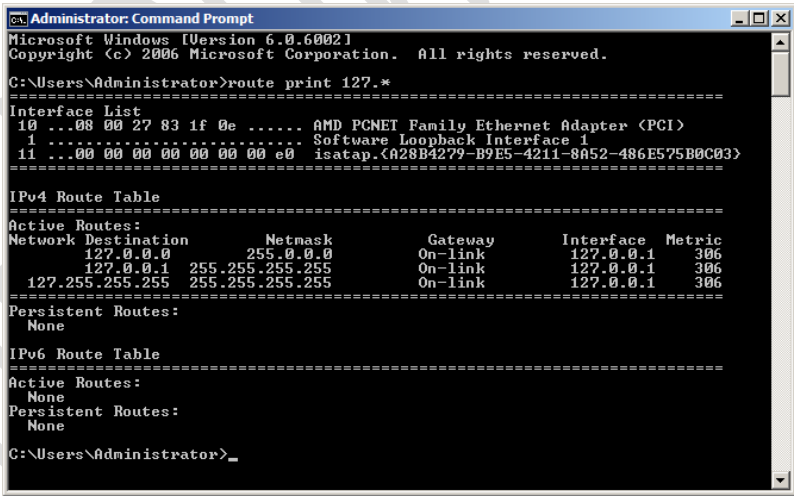


Рис. 3.15. Просмотр маршрутов `127.*` командой `route`

Для добавления нового маршрута необходимо использовать команду `route` следующим образом:

```
route add 192.168.1.0 mask 255.255.255.0 192.168.1.1 if 10
```

Здесь адресуемый узел принимает значение `192.168.1.0`, маской подсети – `255.255.255.0`, а адрес шлюза – `192.168.1.1`. Параметр `if` указывает, какой сетевой интерфейс необходимо использовать.

Для удаления маршрута необходимо вызвать команду `route delete` и указать маршрут, который необходимо удалить, следующим образом:

```
route delete 192.168.1.0
```

Задания к работе

Необходимо организовать объединенную сеть, как показано на рис. 3.16. В роли маршрутизаторов должны выступать компьютеры под управлением Windows Server 2008.

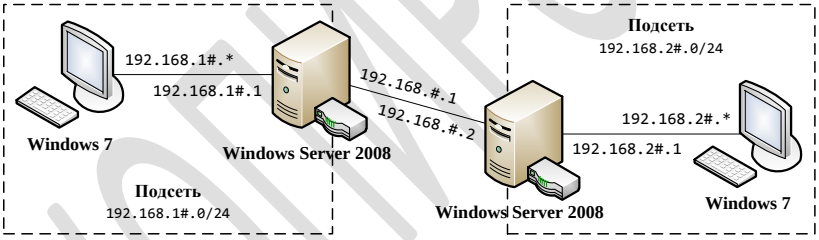


Рис. 3.16. Схема объединенной сети: # – номер варианта

1. Настройте на одном из двух компьютеров под управлением Windows Server 2008 параметры TCP/IP:

   Адаптер 1

   - IP-адрес: `192.168.#.1`
   - Маска подсети: `255.255.255.0`
   - Предпочитаемый DNS-сервер: `192.168.#.1`

   Адаптер 2

   - IP-адрес: `192.168.1#.1`
   - Маска подсети: `255.255.255.0`
   - Предпочитаемый DNS-сервер: `192.168.#.1`

2. На этом же компьютере установите и настройте DNS-сервер и DHCP-сервер:

   DNS-сервер

   - Зона прямого просмотра: *первые три буквы фамилии*
   - Зона обратного просмотра: `168.192.in-addr.arpa`

   DHCP-сервер

   - Родительский домен: *первые три буквы фамилии*
   - IPv4-адрес основного DNS-сервера: `192.168.#.1`
   - Диапазоны IP-адресов для аренды (основной шлюз):

     от `192.168.1#.100` до `192.168.1#.254` (`192.168.1#.1`)
     от `192.168.2#.100` до `192.168.2#.254` (`192.168.2#.1`)

   - Маска подсети: `255.255.255.0`

3. Настройте на втором компьютере под управлением Windows Server 2008 параметры TCP/IP:

   Адаптер 1

   - IP-адрес: `192.168.#.2`
   - Маска подсети: `255.255.255.0`
   - Предпочитаемый DNS-сервер: `192.168.#.1`

   Адаптер 2

   - IP-адрес: `192.168.2#.1`
   - Маска подсети: `255.255.255.0`
   - Предпочитаемый DNS-сервер: `192.168.#.1`

4. На двух компьютерах под управлением Windows Server 2008 установите службу RRAS и настройте статическую маршрутизацию.

   Настройку статической маршрутизации в первом случае необходимо выполнять с помощью команды `route`*, а во втором – посредством настройки службы RRAS.*

5. Настройте агента DHCP-ретрансляции на втором компьютере под управлением Windows Server 2008.

6. Сконфигурируйте все компьютеры под управлением Windows 7 на автоматическую настройку параметров TCP/IP.

7. Протестируйте работу DHCP-сервера:

   - Убедитесь в том, что параметры TCP/IP на компьютерах под управлением Window 7 были настроены автоматически;

- Зарезервируйте IP-адрес для любого компьютера под управлением Window 7 и убедитесь в том, что он будет назначен.

Для выполнения этого задания необходимо использовать команду `ipconfig`.

8. Создайте записи A и PTR в базе данных DNS для всех компьютеров объединенной сети.
9. Протестируйте работу DNS-сервера:
   - Убедитесь в том, что все компьютеры доступны по своим доменным именам;
   - Определите доменное имя любого компьютера по IP-адресу.

   Для выполнения этого задания необходимо использовать команду `ping`.
10. Удалите статические маршруты, созданные в п. 4, и настройте динамическую маршрутизацию между подсетями.

    Статические маршруты необходимо удалять соответственно с помощью команды `route` *или посредством настройки службы RRAS.*
11. Проверьте, чтобы компьютеры под управлением Windows 7 из разных подсетей могли установить соединение друг с другом.

    Для выполнения этого задания необходимо использовать команду `ping`.
12. Создайте запись CNAME в базе данных DNS для любого компьютера объединенной сети и убедитесь в том, что он доступен по созданному доменному псевдониму.
13. На любом компьютере под управлением Windows 7 с помощью файла hosts заблокируйте доменное имя любого другого компьютера объединенной сети и убедитесь в том, что он не будет доступен по данному имени.

    В операционных системах Windows файл hosts находится в каталоге %WINDIR%\system32\drivers\etc\.

Содержание отчета

Отчет о выполнении работы должен включать в себя постановку задачи и описание проделанных операций с иллюстрациями.

Контрольные вопросы

1. Что называют физическим адресом?
2. Что называют сетевым адресом?
3. Что называют доменным именем?
4. Что такое MAC-адрес? Где используется MAC-адрес?
5. Что такое IP-адрес? Где используется IP-адрес?
6. Для чего используются классовая и бесклассовая адресация?
7. Что называют классовой адресацией?
8. Что называют бесклассовой адресацией?
9. Какие существуют специальные IP-адреса?
10. Что называют системой доменных имен (DNS)?
11. Какова структура и принцип работы DNS?
12. Что такое база данных DNS? Какие существуют записи ресурсов в базе данных DNS?
13. Для чего предназначен протокол DHCP?
14. Как происходит аренда IP-адреса с помощью протокола DHCP?
15. Для чего предназначен протокол ARP?
16. Как определить MAC-адрес для заданного IP-адреса с помощью протокола ARP?
17. Для чего предназначен протокол ICMP?
18. Как определить MAC-адрес для заданного адреса IPv6 с помощью протокола ICMPv6?
19. Что называют объединенной сетью?
20. Как осуществляется объединение сетей с помощью мостов?
21. Для чего предназначен протокол STP?
22. Как осуществляется объединение сетей с помощью маршрутизаторов?
23. Что называют маршрутизацией? Как выполняется маршрутизация?
24. Для чего применяются алгоритмы маршрутизации?
25. В чем заключается дистанционно-векторный алгоритм маршрутизации?
26. В чем заключается алгоритм маршрутизации по состоянию канала?
27. Что такое сходимость маршрутов?
28. Что называют внутренним и внешним протоколом маршрутизации?
29. Какие существуют протоколы маршрутизации?
30. Для чего применяется технология NAT?
31. Как осуществляется преобразование внутренних адресов?

32. Как осуществляется перегрузка глобальных адресов?
33. Как осуществляется преобразование при перекрытии адресов?
34. Как в операционных системах Windows осуществляется управление параметрами TCP/IP?
35. Как в операционных системах Windows выполняется установка и настройка DNS-сервера?
36. Как в операционных системах Windows выполняется установка и настройка DHCP-сервера?
37. Как в операционных системах Windows выполняется установка и настройка службы RRAS?
38. Как и для чего используется команда `ping`?
39. Как и для чего используется команда `ipconfig`?
40. Как и для чего используется команда `route`?

ЛАБОРАТОРНАЯ РАБОТА №4. РАЗРАБОТКА СЕТЕВЫХ ПРИЛОЖЕНИЙ

Цель работы

Получение навыков в проектировании прикладных протоколов для передачи данных и реализации этих протоколов в приложениях Windows на языке C/C++ с применением Windows Sockets.

Краткие теоретические сведения

Возможность обмена данными между сетевыми приложениями обеспечивается операционными системами с помощью сокетов. ***Сокетом*** (*socket*) называется абстрактный объект, представляющий конечную точку соединения.

В операционной системе Windows за поддержку сокетов отвечает Windows Sockets. Windows Sockets (WinSock) – это часть Windows API, которая представляет собой библиотеку функций, позволяющих создавать сетевые приложения Windows. Библиотека WinSock полностью реализована в DLL-библиотеке Ws2_32.dll, для получения доступа к которой следует подключить к программе библиотеку импорта Ws2_32.lib и заголовочный файл Winsock2.h.

Документация по всем функциям WinSock имеется в Platform Software Development Kit (Platform SDK), которая доступна по адресу: http://msdn.microsoft.com/.

Инициализация и завершение работы WinSock

Перед тем как использовать функции WinSock следует инициализировать эту библиотеку с помощью функции `WSAStartup`, которая должна быть первой из функций WinSock, вызываемых программой.

```
int WSAStartup(WORD wVersionRequested,
    LPWSADATA lpWSAData);
```

Первый параметр, *`wVersionRequested`*, указывает версию библиотеки WinSock, которая должна использоваться. Младший байт параметра *`wVersionRequested`* указывает основной номер версии, а старший байт – дополнительный. В современных системах Windows доступна версия WinSock 2.2. Для задания версии можно использовать макрос `MAKEWORD`, например, `MAKEWORD(2,2)`.

Второй параметр, *`lpWSAData`*, указывает на структуру `WSADATA`, которая возвращает информацию о конфигурации библиотеки WinSock,

включая номер доступной версии. Структура WSADATA имеет следующее определение:

```
typedef struct WSAData
{
  WORD wVersion;
  WORD wHighVersion;
  char szDescription[WSADESCRIPTION_LEN+1];
  char szSystemStatus[WSASYS_STATUS_LEN+1];
  unsigned short iMaxSockets;
  unsigned short iMaxUdpDg;
  char *lpVendorInfo;
} WSADATA, *LPWSADATA;
```

Подробное описание структуры WSADATA можно найти в документации Platform SDK.

Функция WSAStartup в случае успеха возвращает нулевое значение, в ином случае – код ошибки.

В листинге 4.1 представлен фрагмент программного кода приложения, в котором выполняется инициализация WinSock версии 2.2.

Листинг 4.1. Инициализация библиотеки WinSock

```
1  WSADATA wsaData;
2  int err = WSAStartup(MAKEWORD(2, 2), &wsaData);
3
4  if (0 == err)
5  {
6     /* Инициализация WinSock выполнена успешно */
7  } // if
```

Когда больше нет необходимости в использовании функций WinSock, следует вызвать функцию WSACleanup:

```
int WSACleanup();
```

В случае успеха функция WSACleanup возвращает нулевое значение, в ином случае – значение SOCKET_ERROR. Узнать какая именно это ошибка можно, вызвав функцию WSAGetLastError, которая вернет код последней ошибки в текущем потоке:

```
int WSAGetLastError();
```

Большинство функций WinSock в случае возникновения ошибки возвращают значение SOCKET_ERROR, которое свидетельствует о том, что произошла ошибка.

Открытие и закрытие сокета

Для работы с сокетом его нужно открыть вызовом функции `socket`, которая имеет следующий прототип:

```
SOCKET socket(int af, int type, int protocol);
```

Первый параметр, *af*, указывает семейство адресов (address family), используемое для сетевой адресации; для указания протокола IPv4 следует использовать значение `AF_INET`, а для IPv6 – `AF_INET6`.

Второй параметр, *type*, указывает тип сокета. Чаще встречаются следующие значения:

- `SOCK_STREAM` – обеспечивает надежный дуплексный протокол на основе установления логического соединения. В стеке протоколов TCP/IP, это протокол TCP;
- `SOCK_DGRAM` – обеспечивает ненадежный протокол доставки дейтаграмм без установления логического соединения. В стеке протоколов TCP/IP, это протокол UDP;
- `OCK_RAW` – обеспечивает доступ к некоторым пакетам сетевого уровня. Данный тип используется в особых случаях, например для просмотра всех ICMP-сообщений.

Последний параметр, *protocol*, указывает, какой протокол следует использовать для открываемого сокета. В контексте TCP/IP он неявно определяется типом сокета, поэтому в качестве значения этого параметра задают нулевое значение.

При успешном открытии сокета функция `socket` возвращает значение типа `SOCKET`, используемое в функциях WinSock для работы с сокетом; в противном случае – значение `INVALID_SOCKET`. Используемый в WinSock тип данных `SOCKET` аналогичен типу данных `HANDLE` в Win32 API.

Когда работа с сокетом закончена, его следует закрыть, вызвав функцию `closesocket`:

```
int closesocket(SOCKET s);
```

В случае успеха функция `closesocket` возвращает нулевое значение, в ином случае – значение `SOCKET_ERROR`.

Ассоциирование сокета

Для того чтобы через сокет можно было получать данные его необходимо предварительно ассоциировать с локальным адресом и номером порта. Для этого предназначена функция `bind`:

```
int bind(SOCKET s, const sockaddr *name, int namelen);
```

Первый параметр, *s*, – открытый сокет, который необходимо ассоциировать с локальным адресом и номером порта.

Второй параметр, *name*, указывает на структуру sockaddr, которая содержит локальный адрес и номер порта, а третий параметр, *namelen*, – размер (в байтах) этой структуры.

В случае успешного выполнения функция bind возвращает нулевое значение, а в случае ошибки – SOCKET_ERROR.

Структура sockaddr имеет следующее определение:

```
struct sockaddr
{
    u_short sa_family;
    char sa_data[14];
};
```

Первое поле, *sa_family*, задает семейство адресов, используемое для сетевой адресации; для указания протокола IPv4 следует использовать значение AF_INET, а для IPv6 – AF_INET6. Второе поле, *sa_data*, зависит от заданного семейства адресов.

Для IPv4 вместо структуры sockaddr используется структура sockaddr_in, которая имеет следующее описание:

```
struct sockaddr_in
{
    short sin_family; // семейство адресов
    u_short sin_port; // номер порта
    struct in_addr sin_addr; // IP-адрес
    char sin_zero[8]; // зарезервировано
};
```

Первое поле, *sin_family*, задает семейство адресов и должно принимать значение AF_INET.

Второе поле, *sin_port*, задает номер порта. Если значение этого поля равно нулю, операционная система сама задаст номер порта для сокета.

Третье поле, *sin_addr*, указывает на структуру *s_addr*, где в поле *s_addr* содержится IPv4-адрес, с которым будет ассоциирован сокет.

Четвертое поле, *sin_zero*, зарезервировано для будущего применения и должно содержать нулевое значение.

Для преобразования текстовой строки с IPv4-адресом к числовому типу данных можно использовать функцию inet_addr, которой в качестве аргумента передается строка, содержащая IP-адрес.

```
unsigned long inet_addr(const char *cp);
```

В следующем небольшом примере продемонстрированно использование функции `inet_addr` для задания IPv4-адреса:

```
1   sockaddr_in sin;
2   sin.sin_addr.s_addr = inet_addr("192.168.1.1");
```

Для задания IPv4-адреса можно, также, воспользоваться одной из следующих констант:

- `INADDR_ANY` – неопределенный адрес (`0.0.0.0`), который также может использоваться для связывания сокета со всеми сетевыми интерфейсами локального компьютера;
- `INADDR_LOOPBACK` – адрес обратной петли (`127.0.0.1`);
- `INADDR_BROADCAST` – ограниченный широковещательный адрес (`255.255.255.255`).

При присвоении значений номеру порта и IP-адресу следует учитывать, что порядок следования байтов в разных архитектурах различен. При передаче данных по сети общепринятым является представление чисел в формате *от старшего к младшему* (*big-endian*), в котором запись начинается со старшего байта и заканчивается младшим. Компьютеры, построенные на архитектуре Intel x86, используют формат представления чисел *от младшего к старшему* (*little-endian*), в котором запись начинается с младшего байта и заканчивается старшим. Для преобразования числа из формата, который используется на компьютере (*локальный порядок байтов* (*host byte order*)), к формату, который используется в сети (*сетевой порядок байтов* (*network byte order*)), и наоборот, применяются функции `htonl`, `htons`, `ntohl` и `ntohs`:

```
// преобразует 32-битную беззнаковую целую величину
// из локального порядка байтов в сетевой
u_long htonl(u_long hostlong);

// преобразует 16-битную беззнаковую целую величину
// из локального порядка байтов в сетевой
u_short htons(u_short hostshort);

// преобразует 32-битную беззнаковую целую величину
// из сетевого порядка байтов в локальный
u_long ntohl(u_long netlong);

// преобразует 16-битную беззнаковую целую величину
// из сетевого порядка байтов в локальный
u_short ntohl(u_short netshort);
```

В листинге 4.2 показан пример, в котором выполняет ассоциирование сокета со всеми сетевыми интерфейсами локального компьютера, а также к порту 5150.

Листинг 4.2. Ассоциирование сокета

```
1   sockaddr_in sin = { 0 };
2
3   sin.sin_family = AF_INET; // семейство адресов = IPv4
4   sin.sin_port = htons(5150); // номер порта = 5150
5   sin.sin_addr.s_addr = htonl(INADDR_ANY); // 0.0.0.0
6
7   // ассоциируем сокет с адресом 0.0.0.0:5150
8   int err = bind(s, (struct sockaddr *)&sin, sizeof(sin));
9
10  if (SOCKET_ERROR != err)
11  {
12      /* Сокет успешно ассоциирован */
13  } // if
```

Получение IP-адресов и имени сетевого узла

Для того, что бы узнать IP-адрес сетевого узла зная его имя, в WinSock имеется функция `gethostbyname`:

```
hostent *gethostbyname(const char *name);
```

Параметр *name* указывает на буфер, где содержится имя сетевого узла, IP-адрес которого необходимо узнать.

В случае успеха функция `gethostbyname` возвращает указатель на структуру `hostent`, в ином случае – значение `NULL`.

Структура `hostent` имеет следующее описание:

```
typedef struct hostent
{
  char  *h_name; // имя сетевого узла
  char  **h_aliases; // список псевдонимов
  short h_addrtype; // тип адреса
  short h_length; // размер (в байтах) адреса
  char  **h_addr_list;
} HOSTENT, *PHOSTENT, FAR *LPHOSTENT;
```

Первый поле, *h_name*, указывает на буфер, в котором содержится имя сетевого узла.

Второе поле, *h_aliases*, указывает на буфер, в котором содержится оканчивающийся нулем список псевдонимов сетевого узла.

Третье поле, *h_addrtype*, содержит тип сетевого адреса. Для протокола IPv4 это значение AF_INET. Четвертое поле, *h_length*, содержит размер (в байтах) сетевого адреса.

Пятое поле, *h_addr_list*, указывает на буфер, в котором содержится оканчивающийся нулем список сетевых адресов (в сетевом порядке байтов).

В листинге 4.3 показано, как следует использовать функцию gethostbyname для получения списка IP-адресов для указанного узла.

Листинг 4.3. Функция, возвращающая IP-адреса для указанного узла

```
1   int gethostaddr(const char *host, u_long addr[], u_long max)
2   {
3       hostent *h = gethostbyname(host);
4       if (NULL == h || AF_INET != h->h_addrtype) return -1;
5
6       u_long i;
7       in_addr **addr_list = (in_addr **)h->h_addr_list;
8
9       for (i = 0; (i < max) && (NULL != addr_list[i]); ++i)
10      {
11          addr[i] = ntohl(addr_list[i]->s_addr);
12      } // for
13
14      return (int)i;
15  } // gethostaddr
```

Отметим, что представленная в этом примере функция gethostaddr в случае успеха возвращает число полученных IP-адресов, а в случае ошибки – значение -1.

Для получения имени сетевого узла по его IP-адресу может использоваться функция gethostbyaddr:

```
hostent *gethostbyaddr(const char *addr, int len,
    int type);
```

Первый параметр, *addr*, указывает на буфер, в котором содержится адрес сетевого узла, имя которого необходимо узнать, а второй параметр, *len*, – размер адреса (в байтах).

Последний параметр, *type*, определяет тип указанного адреса. Для IPv4 следует использовать значение AF_INET.

В случае успешного выполнения функции gethostbyaddr возвращают указатель на структуру hostent, а в случае ошибки – NULL.

Отправка и получение данных без установления соединения

Для отправки данных без установления соединения в WinSock имеется функция `sendto`, а для получения этих данных – `recvfrom`:

```
int sendto(SOCKET s, const char *buf, int len, int flags,
    const struct sockaddr *to, int tolen);

int recvfrom(SOCKET s, char *buf, int len, int flags,
    struct sockaddr *from, int *fromlen);
```

Первый параметр, *s*, – открытый сокет, который используется для отправки/получения данные.

Второй параметр, *buf*, указывает на буфер, из которого будут браться данные для их отправки, либо в котором будут сохраняться полученные данные. Третий параметр, *len*, задает размер буфера, на который указывает параметр *buf*.

Четвертый параметр, *flags*, определяет режимы использования функций `sendto` и `recvfrom`. Этот параметр может принимать нулевое значение.

Параметр *to* для функции `sendto` указывает на структуру, содержащую адрес получателя, а параметр *tolen* – размер (в байтах) этой структуры.

Параметр *from* для функции `recvfrom` указывает на структуру, в которую помещается адрес отправителя, а параметр *fromlen* – размер (в байтах) этой структуры.

В случае успешного выполнения функции `sendto` и `recvfrom` возвращают число отправленных/полученных байтов; в случае ошибки – `SOCKET_ERROR`.

В листинге 4.4 показан пример, в котором отправка и получение данных без установления соединения.

Листинг 4.4. Отправка и получения данных без установления соединения

```
1   sockaddr_in sin = { 0 };
2
3   sin.sin_family = AF_INET; // семейство адресов = IPv4
4   sin.sin_port = htons(7581); // номер порта = 7581
5   sin.sin_addr.s_addr = inet_addr("192.168.1.1"); // 192.168.1.1
6
7   msg_t msg; // буфер для отправки текстового сообщения
8
9   msg.ver = 1;
10  msg.type = 0;
11
12  StringCchCopyA(msg.text, _countof(msg.text), "Привет мир!");
```

```
13
14  // отправка текстового сообщения
15  int n = sendto(s, (const char *)&msg, sizeof(msg), (struct
    sockaddr *)&sin, sizeof(sin));
16
17  if (SOCKET_ERROR != n) // сообщение успешно отправлено
18  {
19      int len = sizeof(sin);
20
21      // получаем текстовое сообщение
22      n = recvfrom(s, (char *)&msg, sizeof(msg), (struct sockaddr
    *)&sin, &len);
23
24      if (SOCKET_ERROR != n) // сообщение успешно получено
25      {
26          /* получено n байт */
27      } // if
28  } // if
```

Следует отметить, что в этом примере для отправки текстового сообщения используется структура `msg_t`, которая может быть определена следующим образом:

```
#pragma pack(1)

struct msg_t
{
    short ver; // версия
    short type; // тип сообщения
    char text[255]; // текст сообщения
};

#pragma pack()
```

Директива `#pragma pack(1)` при определении структуры `msg_t` позволяет разместить ее члены непосредственно друг за другом в памяти. Если не использовать эту директиву структура размером 259 байт будет выровнена до 260 байт.

Управление флагами сокета

Для получения значений флагов сокета в WinSock есть функция `getsockopt`, а для изменения значений – `setsockopt`.

```
int getsockopt(SOCKET s, int level, int optname,
        char *optval, int *optlen);
```

```
int setsockopt(SOCKET s, int level, int optname,
        const char *optval, int optlen);
```

Первый параметр, *s*, – открытый сокет, для которого будут выполняться манипуляции флагами.

Второй параметр, *level*, указывает уровень, на котором находится флаг. Для манипуляции флагами на уровне сокета этот параметр должен принимать значение SOL_SOCKET.

Третий параметр, *optname*, указывает флаг. Полный список флагов, а также их описание см. в документации Platform SDK.

Параметр *optval* для функции getsockopt указывает на буфер, в который будет сохранено значение флага, а параметр *optlen* – указывает на переменную, содержащую размер буфера и в которую будет сохранен необходимый размер буфера. Для функции setsockopt параметр *optval* указывает на буфер, в который будет, в котором содержится значение флага, а параметр *optlen* – задает размер буфера.

В случае успешного выполнения функции getsockopt и setsockopt возвращают нулевое значение; в случае ошибки – SOCKET_ERROR.

В листинге 4.5 показан пример, в котором определяется разрешено ли сокету выполнять широковещательную передачу, если нет эта ситуация исправляется.

Листинг 4.5. Включение широковещательной передачи

```
1   BOOL optval;
2   int optlen = sizeof(optval);
3
4   // определяем значение флага SO_BROADCAST
5   int err = getsockopt(s, SOL_SOCKET, SO_BROADCAST,
    (char *)&optval, &optlen);
6
7   if ((SOCKET_ERROR != err) && (TRUE != optval))
8   {
9       optval = TRUE;
10      optlen = sizeof(optval);
11
12      // изменяем значение флага SO_BROADCAST
13      err = setsockopt(s, SOL_SOCKET, SO_BROADCAST,
    (char *)&optval, optlen);
14  } // if
15
16  if (SOCKET_ERROR == err)
17  {
18      /* Не удалось установить флаг SO_BROADCAST */
19  } // if
```

Установление и завершение соединения

На клиенте и на сервере установление соединения осуществляется по-разному. Для установления соединения клиент отправляет серверу запрос на соединение тогда, как сервер ожидает клиентских запросов на соединение.

Прием входящих запросов соединения

Для того чтобы сокет можно было использовать для приема клиентских запросов на соединение, нужно перевести сокет в режим прослушивания вызовом функции `listen`:

```
int listen(SOCKET s, int backlog);
```

Первый параметр, *s*, – открытый сокет, который нужно перевести в режим прослушивания.

Второй параметр, *backlog*, – максимальное число ожидающих, но еще не принятых соединений. Следует отметить, что это *не* максимальное число одновременных соединений, а лишь число частично установленных соединений, ожидающих в очереди, пока прикладной процесс их примет.

В случае успеха функция `listen` возвращает нулевой значение; в случае ошибки – `SOCKET_ERROR`.

После этого сокет уже больше нельзя использовать для отправки или получения данных. Но зато его можно использовать для открытия соединений, ожидающих в очереди, используя функцию `accept`:

```
SOCKET accept(SOCKET s, sockaddr *addr, int *addrlen);
```

Первый параметр, *s*, – открытый сокет в режиме прослушивания, который используется для установления соединений.

Параметр *addr* указывает на буфер, в который будет сохранен адрес сетевого узла, желающего установить соединение, параметр *addrlen* указывает на переменную, содержащую размер буфера и в которую будет сохранен необходимый размер буфера. Параметры *addr* и *addrlen* могут быть установлены в `NULL`.

В случае успеха функция `accept` возвращает сокет, который затем можно будет использовать для отправки и получения данных; в случае ошибки – `INVALID_SOCKET`.

В листинге 4.6 показан пример цикла ожидания входящих запросов соединения. Заметим, что в этом примере выход из цикла происходит при возникновении ошибки `WSAEINTR`, которая может возникнуть, например, при закрытии «слушающего» сокета.

Листинг 4.6. Прием входящих запросов соединения

```
// переводим сокет в режим прослушивания
int err = listen(ss, 5);

if (SOCKET_ERROR != err)
{
    for (;;) // цикл ожидания входящих соединений
    {
        // ожидание входящих соединений
        SOCKET s = accept(ss, NULL, NULL);

        if (INVALID_SOCKET == s)
        {
            if (WSAEINTR == WSAGetLastError()) break; // ошибка : операция блокирования была прервана
        } // if
        else
        {
            /* новое соединение открыто */
        } // else
    } // for
} // if
```

Установление соединения

Клиент, который желает установить соединение с сервером, может это сделать при помощи функции `connect`:

```
int connect(SOCKET s, const struct sockaddr *name,
    int namelen);
```

Первый параметр, *s*, – открытый сокет, который используется для установления соединений.

Второй параметр, *name*, указывает на структуру `sockaddr`, в которой хранится адрес сервера и номер порта, а параметр *namelen* определяет размер этой структуры в байтах.

В случае успешного выполнения функция возвращает нулевое значение, а случае ошибки – `SOCKET_ERROR`.

В листинге 4.7 показан пример, в котором устанавливается соединение по адресу `192.168.1.1:7581`.

Листинг 4.7. Установление соединения

```
sockaddr_in sin = { 0 };

sin.sin_family = AF_INET; // семейство адресов = IPv4
sin.sin_port = htons(7581); // номер порта = 7581
```

```
5   sin.sin_addr.s_addr = inet_addr("192.168.1.1"); // 192.168.1.1
1
2   int err = connect(s, (sockaddr *)&sin, sizeof(sin));
3
4   if (SOCKET_ERROR != err)
5   {
6       /* новое соединение открыто */
7   } // if
```

Аккуратное завершение соединений

Нельзя завершить соединение, просто закрыв сокет, поскольку у другой стороны могут быть еще не отправленные данные. В WinSock имеется функция `shutdown`, которая позволяет закрыть определенное направление передачи.

```
int shutdown(SOCKET s, int how);
```

Первый параметр, *s*, – открытый сокет, для которого необходимо закрыть направление передачи.

Второй параметр, *how*, определяет какое направление передачи необходимо закрыть. Этот параметр должен принимать следующие значения:

- `SD_RECEIVE` – закрывается принимающая сторона;
- `SD_SEND` – закрывается передающая сторона.
- `SD_BOTH` – закрываются оба направления передачи.

Если закрывается принимающая сторона, в сокете делается пометка, что он больше не может принимать данные; все последующие попытки выполнить для него операцию получения данных оканчиваются ошибкой.

Если закрывается передающая сторона, в сокете делается пометка о том, что данные отправляться больше не будут; все последующие попытки отправить данные оканчиваются ошибкой `WSAESHUTDOWN`.

Важно отметить, что вызов функции `shutdown` не закрывает сокет по-настоящему, даже если он вызван с параметром `SD_BOTH`. Если работа с сокетом закончена, его следует закрыть, вызвав функцию `closesocket`.

Отправка и получение данных по установленному соединению

После установления соединения можно обмениваться данными с помощью функций `send` и `recv`:

```
int send(SOCKET s, const char *buf, int len,
    int flags);
```

```
int recv(SOCKET s, void *buf, int len, int flags);
```

Первый параметр, *s*, – открытый сокет, который используется для отправки/получения данные.

Второй параметр, *buf*, указывает на буфер, из которого будут браться данные для их отправки, либо в котором будут сохраняться полученные данные. Третий параметр, *len*, задает размер буфера, на который указывает параметр *buf*.

Четвертый параметр, *flags*, определяет режимы использования функций `send` и `recv`. Этот параметр может принимать нулевое значение.

В случае успешного выполнения функции `send` и `recv` возвращают число отправленных/полученных байтов; в случае ошибки – `SOCKET_ERROR`.

Нужно отметить, что данные доставляются получателю в виде потока байтов, в котором нет понятий «сообщения» или «границы сообщения». Представим, например, что имеется TCP-соединение между двумя приложениями. Допустим одно из приложений должно отправить два сообщения, для чего в нем дважды вызывается функция `send` – по разу для каждого сообщения. Можно предположить, что эти сообщения передаются в виде разделенных блоков, каждое в своем TCP-сегменте. Однако реальная передача данных вероятнее всего будет происходить не так. На самом деле функция `send` обычно не передает данные, а просто копирует их в буфер TCP. Модуль TCP самостоятельно определяет, сколько данных нужно передать. В частности, он вообще может отложить передачу до более подходящего момента. Принятие такого решения зависит от многих факторов, например: объема данных, которые готов принять получатель (окно приема), загруженности сети, максимальный размер TCP-сегмента и т.п. Возможны следующие способы разбиения двух сообщения по TCP-сегментам:

- оба сообщения передаются каждый в своем TCP-сегменте;
- оба сообщения передаются в одном TCP-сегменте;
- часть одного сообщения передается в одном TCP-сегменте, а оставшаяся его часть передается в другом TCP-сегменте (возможно вместе с другим сообщением или его частью).

Важно отметить, что если при вызове функции `send` в буфере TCP не окажется свободного места, приложение либо блокируется, пока данные не будут скопированы, либо функция `send` возвращает нулевое значение. Это зависит от того, был ли сокет помечен как блокирующий. Если же в буфере TCP есть свободное место, но его не достаточ-

но для копирования всех отправляемых данных, то будет скопирована только их часть, а функция `send` вернет число скопированных байт. Приложение должно проверять возвращаемое значение для отслеживания того, сколько байт было отправлено и повторно отправлять оставшуюся часть данных. Стандартный способ решения этой проблемы показан в листинге 4.8.

Листинг 4.8. Функция для отправки данных полностью

```
1   int sendn(SOCKET s, const char *buf, int len)
2   {
3       int size = len;
4
5       while (size > 0)
6       {
7           int n = send(s, buf, size, 0);
8
9           if (n <= 0) // ошибка
10          {
11              if (WSAGetLastError() == WSAEINTR) // операция была
    прервана
12              {
13                  continue;
14              } // if
15
16              return SOCKET_ERROR;
17          } // if
18
19          buf  += n;
20          size -= n;
21      } // while
22
23      return len;
24  } // sendn
```

Функция `sendn` используется точно также как `send`, только она не возвращает управление, пока не будут отправлены все данные или не возникнет ошибка.

В общем случае принимающее приложение не имеет информации относительно реального количества полученных данных. Например, когда вызывается функция `recv`, возможны следующие варианты:

– данных еще нет, и приложение либо блокируется, пока данные не будут получены, либо функция `recv` возвращает нулевое значение. Это зависит от того, был ли сокет помечен как блокирующий.

- приложение получает лишь часть данных, если сообщение передается по частям в разных TCP-сегментах;
- приложение получает все сообщение, если оно передается в одном TCP-сегменте;
- приложение получает все сообщение и часть или все следующее сообщение.

Еще раз следует отметить, что хотя сообщения передаются в TCP-сегментах, размер сегмента напрямую не связан с количеством данных, переданных при вызове функции `send`. У принимающего приложения нет надежного способа определить, как именно данные распределены по сегментам, поскольку между соседними вызовами функции `recv` может прийти несколько TCP-сегментов. Это не возможно, даже если принимающее приложение реагирует очень быстро. Например, если один TCP-сегмент был потерян (вполне обычная ситуация в крупных сетях), то модуль TCP будет «придерживать» последующие полученные данные, пока не будет повторно передан и получен пропавший TCP-сегмент. В этот момент приложение получает данных из всех принятых TCP-сегментов.

В некоторых приложениях границы сообщений важны, тогда необходимо реализовывать их сохранение на прикладном уровне. Самый простой случай – это сообщения фиксированной длины. В этом случае заранее известно число байтов, которые нужно получить. При этом *недостаточно* выполнить однократный вызов функции `recv` поскольку при этом можно получить меньше байт, чем требуется. Стандартный способ решения этой проблемы показан в листинге 4.9.

Листинг 4.9. Функция для получения данных фиксированной длины

```
1   int recvn(SOCKET s, char *buf, int len)
2   {
3       int size = len;
4
5       while (size > 0)
6       {
7           int n = recv(s, buf, size, 0);
8
9           if (n <= 0) // ошибка
10          {
11              if (WSAGetLastError() == WSAEINTR) // операция была
    прервана
12              {
13                  continue;
14              } // if
```

```
15
16              return SOCKET_ERROR;
17          } // if
18
19          buf  += n;
20          size -= n;
21      } // while
22
23      return len;
24  } // recvn
```

Функция `recvn` используется точно также как `recv`, только она не возвращает управление, пока не будет получено нужно число байт или не возникнет ошибка.

Если приложение должно работать с сообщениями переменной длины, то существует два метода передачи таких сообщений. Во-первых, можно разделять сообщения специальными маркерами. Принимающей стороне нужно посмотреть все полученные данные и найти маркеры, разделяющие сообщения. Такой метод лучше применять только при наличии «естественного» разделителя, например, последовательности символов `CR LF` (`CR` – символ возврата каретки, а `LF` – символ перевода строки), разделяющей строки текста. Например, такие разделительные маркеры используются в прикладных протоколах FTP, HTTP, SMTP, POP3 и т.п.

Другой метод передачи сообщений переменной длины предусматривает снабжение каждого сообщения заголовком, содержащим (как минимум) длину следующего за ним тела сообщения. Принимающее приложение должно получить сначала заголовок фиксированной длины и извлечь из него длину тела сообщения, а затем – само тело. В листинге 4.10 представлен пример для простого случая, когда в заголовке хранится только длина сообщения.

Листинг 4.10. Получения данных переменной длины

```
1   int len = 0; // длина сообщения
2
3   // получим длину сообщения
4   int err = recvn(s, (char *)&len, sizeof(int));
5
6   if ((SOCKET_ERROR != err) && (len > 0))
7   {
8       char *buf = new char[len]; // выделяем память для сообщения
9       err = recvn(s, buf, len); // получим само сообщение
10
11      if (SOCKET_ERROR != err)
```

```
12      {
13          /* сообщение успешно получено */
14      } // if
15
16      delete[] buf; // освобождаем память
17  } // if
```

Определение состояния сокета

Определить текущее состояние сокета (например, есть ли полученные данные или есть ли место в буфере TCP) можно с помощью функции `select`.

```
int select(int nfds, fd_set *readfds, fd_set *writefds,
    fd_set *exceptfds, const struct timeval *timeout);
```

Первый параметр, *`nfds`*, игнорируется и включен только для совместимости с сокетами Беркли.

Параметр *`readfds`* – указывает на множество сокетов, которые должны быть проверены на возможность получения данных. Параметр *`writefds`* – указывает на множество сокетов, которые должны быть проверены на возможность записи данных в буфер. Параметр *`exceptfds`* – указывает на множество сокетов, которые должны быть проверены на наличие ошибок.

Последний параметр, *`timeout`*, определяет максимальное время ожидания, в течение которого сокеты будут проверяться на готовность к различным операциям. Собственно параметр *`timeout`* указывает на структуру `timeval`:

```
struct timeval
{
    long tv_sec;  // интервал времени в секундах
    long tv_usec; // интервал времени в микросекундах
};
```

Каждый из параметров *`readfds`*, *`writefds`*, *`exceptfds`* и *`timeout`* является необязательным, и может быть установлен в `NULL`. Если параметры *`readfds`*, *`writefds`* и *`exceptfds`* установлены в `NULL`, проверка производиться не будет. В случае, когда параметр *`timeout`* установлен в `NULL`, функция `select` будет ожидать первого готового сокета.

Функция `select` возвращает общее количество сокетов (во всех заданных множествах *`readfds`*, *`writefds`* и *`exceptfds`*), готовых к чтению или отправке данных, а в случае ошибки – `SOCKET_ERROR`.

Для работы с типом данных `fd_set`, определяющим множество сокетов, существуют специальные макросы:

```
FD_CLR(s, set)    // удаляет сокет s из множества set

FD_ISSET(s, set) // возвращает ненулевое значение,
                 // если сокет s присутствует во множестве
                 // set, иначе, возвращает ноль

FD_SET(s, set)    // добавляет сокет s во множество set

FD_ZERO(set)      // очищает множество set
```

В следующем примере показано как с помощью функции `select`, определить готов ли сокет `s` для получения данных.

Листинг 4.11. Определение готовности сокета к получению данных

```
1   fd_set readfds; // множество сокетов
2   timeval timeout; // время ожидания
3
4   // очистим множество readfds
5   FD_ZERO(&readfds);
6   // добавим сокет s во множество readfds
7   FD_SET(s, &readfds);
8
9   // зададим время ожидания 5 сек.
10  timeout.tv_sec  = 5;
11  timeout.tv_usec = 0;
12
13  // ждем готовности сокета к получению данных
14  int err = select(0, &readfds, NULL, NULL, &timeout);
15
16  if (SOCKET_ERROR == err)
17  {
18      /* возникла ошибка */
19  } // if
20  else if (0 == err)
21  {
22      /* время ожидания истекло */
23  } // if
24  else if (FD_ISSET(s, &readfds)) // FD_ISSET используется
25                                  // только для примера
26  {
27      /* сокет готов к получению данных */
28  } // if
```

Блокирующие и неблокирующие сокеты

Сокеты могут использоваться в одном из двух режимов: *блокирующем* (*blocking*) и *неблокирующем* (*nonblocking*). При использовании блокирующего сокета такие функции, как `accept`, `connect`, `send`, `recv` и т.п. будут ожидать завершения выполняемых ими операций. Это может привести к тому, что программа может «зависнуть» из-за того что соответствующая операция с сокетом может никогда не завершится.

После создания сокет находится в блокирующем режиме. Изменить режим блокирования сокета можно с помощью функции `ioctlsocket`, которая позволяет управлять режимом ввода/вывода сокета.

```
int ioctlsocket(SOCKET s, long cmd, u_long *argp);
```

Первый параметр, *s*, – открытый сокет, для которого нужно изменить режим ввода/вывода.

Второй параметр, *cmd*, определяет параметр, значение которого необходимо определить или задать. Список всех таких параметров, а также их описание можно найти в документации Platform SDK.

Третий параметр, *argp*, указывает на переменную, которая устанавливает или возвращает значение параметра.

В случае успешного выполнения функция `ioctlsocket` возвращает нулевое значение; в случае ошибки – `SOCKET_ERROR`.

Для перевода сокета `s` в неблокирующий режим используется параметр `FIONBIO`, а параметр *argp* должен указывать на ненулевое значение:

Листинг 4.12. Перевод сокета в неблокирующий режим

```
1  u_long argp = 1;
2  int err = ioctlsocket(s, FIONBIO, &argp);
3
4  if (SOCKET_ERROR != err)
5  {
6      /* сокет переведен в неблокирующий режим */
7  } // if
```

После перевода сокета в неблокирующий режим функции, выполняющие операции для этого сокета, не дожидаются окончания операции, что в свою очередь не вызывает нежелательных пауз в работе программы. Однако теперь необходимо определять, что операция выполнила полностью. Если функция `WSAGetLastError` вернула код ошибки `WSAEWOULDBLOCK`, т.е. текущая операция не завершена, и ее придется повторить позже.

Организация тайм-аута при установлении соединения

При использовании блокирующего сокета функция `connect` не возвращает управления, пока не будет установлено соединение или не произойдет ошибка. В случае недоступности сервера на это может потребоваться достаточно много времени.

Для организации тайм-аута при установлении соединения можно перевести сокет в неблокирующий режим, а затем ожидать его с помощью вызова функции `select`. В листинге 4.13 показано как это делается.

Листинг 4.13. Функция для установления соединения с тайм-аутом

```
8   int connect(SOCKET s, const struct sockaddr *name, int namelen,
    const struct timeval *timeout)
9   {
10      // переводим сокет в неблокирующий режим
11      u_long argp = 1;
12      int err = ioctlsocket(s, FIONBIO, &argp);
13
14      if (SOCKET_ERROR == err)
15      {
16          // не удалось перевести сокет в неблокирующий режим
17          return SOCKET_ERROR;
18      } // if
19
20      err = connect(s, name, namelen); // пытаемся установить
    соединение
21
22      if (SOCKET_ERROR == err) // соединение не установлено
23      {
24          fd_set writefds, exceptfds;
25
26          // очистим множество writefds
27          FD_ZERO(&writefds);
28          // добавим сокет s во множество writefds
29          FD_SET(s, &writefds);
30          // очистим множество exceptfds
31          FD_ZERO(&exceptfds);
32          // добавим сокет s во множество exceptfds
33          FD_SET(s, &exceptfds);
34
35          // ждем соединения
36          err = select(0, NULL, &writefds, &exceptfds, &timeout);
37
38          if (SOCKET_ERROR == err)
39          {
```

```
40              /* возникла ошибка */
41          } // if
42          else if (0 == err)
43          {
44              /* время ожидания истекло */
45              err = SOCKET_ERROR;
46          } // if
47          else if (FD_ISSET(s, &writefds) && !FD_ISSET(s,
    &exceptfds))
48          {
49              /* соединение установлено */
50              err = 0;
51          } // if
52          else
53          {
54              /* соединение не установлено */
55              err = SOCKET_ERROR;
56          } // else
57      } // if
58
59      // переводим сокет обратно в блокирующий режим
60      argp = 0;
61      ioctlsocket (s, FIONBIO, &argp);
62
63      return err;
64  } // connect
```

Задания к работе

1. Разработать прикладной протокол для передачи текстовых сообщений с помощью транспортного протокола UDP.

   Предусмотреть возможность разбиения больших сообщений на отдельные фрагменты.

2. Разработать в Visual C++ приложение Windows для обмена сообщениями по протоколу, разработанному в п. 1.

   Отправка сообщений должна производиться как широковещательно, так и с указанием адреса назначения (имя или IP-адрес).

3. Разработать прикладной протокол для передачи файлов с помощью транспортного протокола TCP.

4. Разработать в Visual C++ два приложения Windows для копирования файлов по протоколу, разработанному в п. 3:

- серверное приложение, которое принимает файлы от клиентских приложений и сохраняет их на диске;
- клиентское приложение, которое должно подключаться к серверному приложению для отправки файлов.

Предусмотреть возможность одновременного копирования нескольких файлов.

5. Протестируйте работу приложений, разработанных в п. 2 и 4, на нескольких компьютерах под управлением Windows.

Содержание отчета

Отчет о выполнении работы должен включать в себя:

1. Постановку задачи.
2. Описание разработанных протоколов.
3. Блок-схемы *основных* алгоритмов.
4. Исходный программный код.
5. Результаты тестирования.

Контрольные вопросы

1. Для чего предназначены протоколы транспортного уровня?
2. Для чего предназначены номера портов? На какие диапазоны делятся номера портов?
3. Для чего предназначен протокол UDP? Какими преимуществами и недостатками обладает UDP?
4. Для чего предназначен протокол TCP?
5. Как осуществляется установление и завершение TCP-соединения?
6. Какие состояния определены для TCP-соединения?
7. Как осуществляется передача данных в TCP?
8. Для чего предназначены прикладные протоколы?
9. Какие протоколы из стека протоколов TCP/IP являются прикладными протоколами?
10. Какие прикладные протоколы используют в качестве транспортного протокола протокол UDP, а какие – протокол TCP?
11. Что называют сокетом? Для чего используются сокеты?
12. Что такое WinSock?
13. Как выполняется инициализация и завершение работы WinSock?
14. Как в WinSock открыть и закрыть сокет?
15. Как в WinSock ассоциировать сокет с локальным адресом и номером порта?

16. Как в WinSock осуществляется отправка и получение данных без установления соединения?
17. Как в WinSock узнать IP-адрес сетевого узла зная его имя?
18. Как в WinSock получить имя сетевого узла по его IP-адресу?
19. Как в WinSock осуществляется управление флагами сокета?
20. Как в WinSock осуществляется прием входящих запросов соединение?
21. Как в WinSock выполнить установление соединения?
22. Как в WinSock выполнить аккуратное завершение соединения?
23. Как в WinSock осуществляется отправка и получение данных по установленному соединению?
24. Как в WinSock осуществляется определение состояние сокета?
25. Что такое блокирующий и неблокирующий режим использования сокета в WinSock?

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. *Бройдо, В.Л.* Вычислительные системы, сети и телекоммуникации: Учебник для вузов, 4-е издание / В.Л. Бройдо, О.П. Ильина. – СПб.: Питер, 2010. – 560 с.
2. *Ватаманюк А.И.* Создание, обслуживание и администрирование сетей на 100% / А.И. Ватаманюк. – СПб.: Питер, 2010. – 288 с.
3. *Глухоедов, А.В.* Инфокоммуникационные системы и сети: конспект лекций: учебное пособие / А.В. Глухоедов. – Белгород: Изд-во БГТУ, 2015. – 160 с.
4. *Голдовский, Я.М.* Маршрутизация в IP-сетях: Учебное пособие / Я.М. Голдовский, Б.В. Желенков. – М: МИИТ, 2007. – 151 с.
5. *Лавров, Д.Н.* Сети и системы телекоммуникаций: учебное пособие / Д.Н. Лавров. – Омск: Изд-во ОмГУ, 2006. – 183 с.
6. *Моримото, Р.* Microsoft Windows Server 2008 R2. Полное руководство / Р. Маримото, М. Ноэл, О. Драуби и др.; пер. с англ. Я.П. Волкова [и др.]. – М.: ООО «И.Д. Вильямс», 2011. – 1456 с.: ил.
7. *Олифер, В.Г.* Компьютерные сети. Принципы, технологии, протоколы: Учебник для вузов, 3-е издание / В.Г. Олифер, Н.А. Олифер. – СПб.: Питер, 2009. – 958 с.: ил.
8. *Олифер, В.Г.* Основы компьютерных сетей / В.Г. Олифер, Н.А. Олифер. – СПб.: Питер, 2009. – 352 с.: ил.
9. *Паркер, Т.* TCP/IP. Для профессионалов, 3-е издание / Т. Паркер, К. Сиян; пер. с англ. Е. Матвеев. – СПб.: Питер, 2004. – 859 с.: ил.
10. *Семенов, А.Б.* Структурированные кабельные системы, 5-е издание / А.Б. Семенов, С.К. Стрижаков, И.Р. Сунчелей. – М.: Компания АйТи; ДМК Пресс, 2006. – 640 с.
11. *Снейдер, Й.* Эффективное программирование TCP/IP. Библиотека программиста / Й. Снейдер; пер. с англ. А Слинкин. – СПб.: Питер, 2002. – 320 с.: ил.
12. *Таненбаум, Э.* Компьютерные сети. 5-е издание / Э. Таненбаум, Д. Уэзеролл; пер. с англ. А. Гребеньков. – СПб.: Питер, 2012. – 960 с.: ил.
13. *Чеппел, Л.* TCP/IP. Учебный курс / Л. Чеппел, Э. Титтел; пер. с англ. Ю. Гороховский. – СПб.: БХВ-Петербург, 2003. – 976 с.: ил.

14. *Шапиро, Д.* Windows Server 2003. Библия пользователя / Д. Шапиро, Д. Бойс; пер. с англ. А.В. Журавлев [и др.]. – М.: Издательский дом «Вильямс», 2004. – 1216 с.: ил.

15. *Яремчук С.*, Системное администрирование Windows 7 и Windows Server 2008 R2 на 100 % / С. Яремчук, А. Матвеев. – СПБ.: Питер, 2016. – 384 с.: ил.

16. The IEEE Standards Association [Электронный ресурс]. – URL: http://standards.ieee.org/

Учебное издание

ИНФОКОММУНИКАЦИОННЫЕ СИСТЕМЫ И СЕТИ

Методические указания к выполнению лабораторных работ для студентов направлений 09.03.02 – Информационные системы и технологии, 09.03.03 – Прикладная информатика.

Составитель **Глухоедов** Андрей Владимирович

Подписано в печать 02.10.17. Формат 60х84/16. Усл. печ. л. 4,1. Уч.-изд. л. 4,5.
Тираж 50 экз. Заказ Цена
Отпечатано в Белгородском государственном технологическом университете им. В.Г. Шухова
308012, г. Белгород, ул. Костюкова, 46