

Учреждение образования
«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ
ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Д. М. Романенко, Н. В. Пацей, М. Ф. Кудлацкая

КОМПЬЮТЕРНЫЕ СЕТИ

Минск 2016

УДК 004.7(076.5)

БКК 32.97я73

Р69

Рецензенты:

кафедра экономико-математических методов управления УО «Академия управления при Президенте Республики Беларусь» (кандидат физико-математических наук, доцент, заведующий кафедрой *Б. В. Новыши*);

кандидат технических наук, доцент, заведующий кафедрой программного обеспечения информационных технологий УО «Белорусский государственный университет информатики и радиоэлектроники» *Н. В. Лапицкая*;

Все права на данное издание защищены. Воспроизведение всей книги или ее части не может быть осуществлено без разрешения учреждения образования «Белорусский государственный технологический университет».

Романенко, Д. М.

Р69 Компьютерные сети : лабораторный практикум для студентов специальностей 1-40 05 01 «Информационные системы и технологии», 1-40 01 01 «Программное обеспечение информационных технологий», 1-98 01 03 «Программное обеспечение информационной безопасности мобильных систем», 1-47 01 02 «Дизайн электронных и веб-изданий» / Д. М. Романенко, Н. В. Пацей, М. Ф. Кудлацкая. – Минск : БГТУ, 2016. – 168 с.

ISBN

В лабораторном практикуме в простой и доступной форме даны общие принципы организации компьютерных локальных сетей. Приведены наиболее распространенные виды топологий, используемые для физического соединения компьютеров в сети, а также рассмотрено понятие архитектуры. Описаны основные правила адресации в TCP/IP сетях и типы адресов, определено понятие разрешения адресов, представлена настройка различных форм адресации в сетях с одноранговой и клиент-серверной архитектурой. Продемонстрированы принципы и правила настройки IIS-сервера. Даны рекомендации по разработке программ на языке C++ для определения и преобразования различных типов адресов удаленного хоста.

Пособие предназначено для выполнения заданий на занятиях по курсу «Компьютерные сети» для специальностей «Информационные системы и технологии», «Программное обеспечение информационных технологий», «Программное обеспечение информационной безопасности мобильных систем», а также может быть полезно магистрантам и аспирантам, изучающим данную предметную область.

УДК 004.7(076.5)

ББК 32.97я7

© Романенко Д. М., Пацей Н. В., Кудлацкая М. Ф., 2016

© УО «Белорусский государственный
технологический университет», 2016

ISBN

ПРЕДИСЛОВИЕ

Дисциплина «Компьютерные сети» представляет собой введение в сетевую тематику и дает базовые знания по организации и функционированию сетей.

В данном лабораторном практикуме представленные основные понятия, общие подходы в проектировании компьютерных сетей, особенности различных видов и топологий сетей, их практического использования авторы пытались изложить в контексте унифицированной структуры. При этом предполагалось, что читатель знаком с основами информационных технологий.

Основная задача лабораторного практикума – дать студентам общие систематизированные сведения об организации и структуре важной отрасли, которая затрагивает практически все сферы жизнедеятельности человека, динамично развивается. В целом, в издании в простой и доступной форме даны общие понятия компьютерных сетей, их архитектуры, приведены виды топологий, используемые для физического соединения компьютеров в сети, способы адресации и методы диагностики. Описываются правила и процедуры построения одноранговых и клиент-серверных сетей, а также программ, позволяющих выполнять стандартные операции по определению различных сетевых параметров удаленного компьютера. Рассматриваются методы тестирования компьютерных сетей.

В результате изучения дисциплины и выполнения заданий на лабораторных занятиях студент должен знать:

- методы моделирования компьютерных сетей с различными топологиями;
- правила и методы настройки статической и динамической адресации в сетях с одноранговой и клиент-серверной архитектурой;
- методы разделения ресурсов в компьютерных сетях;
- правила и методы настройки символьной адресации в компьютерных сетях;
- методы диагностики TCP/IP и DNS (NetBIOS);
- правила и методы разрешения адресов в IP-сетях. Принципы построения программ по определению сетевых параметров удаленного хоста, работе эхо-запроса, как средства диагностики сети.

Студент должен научиться применять рассматриваемые методы на практике.

1. МОДЕЛИРОВАНИЕ КОМПЬЮТЕРНЫХ СЕТЕЙ

1.1. Понятие и виды топологий

Понятие *топологии* широко используется при создании сетей. Одним из подходов к классификации топологий ЛВС является выделение двух основных классов топологий: широковещательные и последовательные.

В широковещательных топологиях ПК передает сигналы, которые могут быть восприняты остальными ПК. К таким топологиям относятся: общая шина, дерево, звезда.

В последовательных топологиях информация передается только одному ПК. Примерами таких топологий являются: произвольная (произвольное соединение ПК), кольцо, цепочка.

При выборе оптимальной топологии преследуются три основные цели:

- обеспечение альтернативной маршрутизации и максимальной надежности передачи данных;
- выбор оптимального маршрута передачи блоков данных;
- предоставление приемлемого времени ответа и нужной пропускной способности.

При выборе конкретного типа сети важно учитывать ее топологию. Например, в конфигурации сети ArcNet используется одновременно и линейная, и звездообразная топология. Сети Token Ring физически выглядят как звезда, но логически их пакеты передаются по кольцу. Передача данных в сети Ethernet происходит по линейнойшине, так что все станции видят сигнал одновременно.

Существуют пять основных топологий:

- общая шина (Bus);
- кольцо (Ring);
- звезда (Star);
- древовидная (Tree);
- ячеистая (Mesh).

Также возможны комбинации нескольких различных топологий.

1.1.1. Топология «общая шина»

Общая шина – это тип сетевой топологии, в которой рабочие станции расположены вдоль одного участка кабеля, называемого *сегментом* (рис. 1.1).

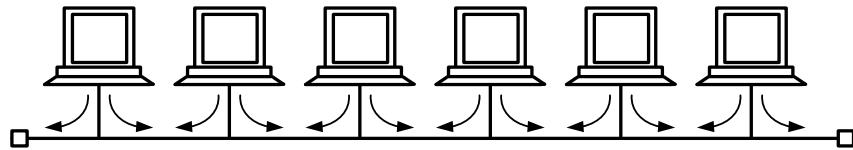


Рис. 1.1. Топология «общая шина»

Топология «*общая шина*» предполагает использование одного кабеля, к которому подключаются все компьютеры сети. Кабель используется всеми станциями по очереди. Для уменьшения зашумленности среды отраженными сигналами, мешающими передаче данных, используют так называемые «*терминаторы*» – специальные резисторы на концах кабеля, предотвращающие появление «отраженной волны».

Все сообщения, посылаемые отдельными компьютерами, принимаются и прослушиваются всеми остальными компьютерами, подключенными к сети. Рабочая станция отбирает адресованные ей сообщения, пользуясь адресной информацией. Надежность здесь выше, так как выход из строя отдельных компьютеров не нарушит работоспособность сети в целом. Поиск неисправности в сети затруднен. Кроме того, так как используется только один кабель, в случае обрыва нарушается работа всей сети.

Примерами использования топологии «общая шина» является сеть 10Base-5 (соединение ПК толстым коаксиальным кабелем) и 10Base-2 (соединение ПК тонким коаксиальным кабелем).

1.1.2. Кольцевая топология

«*Кольцо*» – это топология ЛВС, в которой каждая рабочая станция соединена с двумя другими рабочими станциями, образуя кольцо (рис. 1.2). Данные передаются от одной рабочей станции к другой в одном направлении (по кольцу).

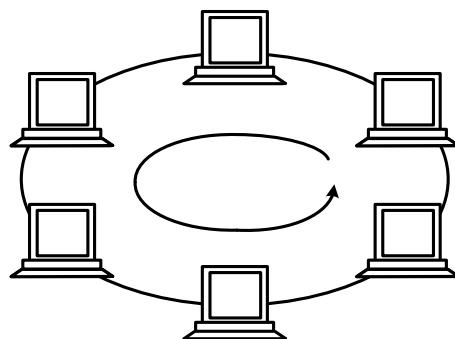


Рис. 1.2. Топология «кольцо»

Каждая рабочая станция выполняет роль повторителя, ретранслируя сообщения к следующей рабочей станции, т. е. данные передаются от одного компьютера к другому как по эстафете. Если компьютер получает данные, предназначенные для другого компьютера, он передает их дальше по кольцу, в ином случае они дальше не передаются.

Основная проблема при кольцевой топологии заключается в том, что каждая рабочая станция должна активно участвовать в пересылке информации, и в случае выхода из строя хотя бы одной из них, вся сеть парализуется. Подключение новой рабочей станции требует краткосрочного выключения сети, так как во время установки кольцо должно быть разомкнуто. Топология «кольцо» имеет хорошо предсказуемое время отклика, определяемое числом рабочих станций.

Чистая кольцевая топология используется редко. Вместо этого кольцевая топология играет транспортную роль в схеме метода доступа. Кольцо описывает логический маршрут, а пакет передается от одной станции к другой, совершая в итоге полный круг.

В сетях *Token Ring* кабельная ветвь из центрального концентратора называется MAU (Multiple Access Unit). MAU имеет внутреннее кольцо, соединяющее все подключенные к нему станции, и используется как альтернативный путь, когда оборван или отсоединен кабель одной рабочей станции. Когда кабель рабочей станции подсоединен к MAU, он просто образует расширение кольца: сигналы поступают к рабочей станции, а затем возвращаются обратно во внутреннее кольцо.

1.1.3. Топология типа «звезда»

«Звезда» – это топология ЛВС (рис. 1.3), в которой все рабочие станции присоединены к центральному узлу (например, к концентратору), который устанавливает, поддерживает и разрывает связи между рабочими станциями.

Преимуществом такой топологии является возможность простого исключения неисправного узла. Однако, если неисправен центральный узел, вся сеть выходит из строя. В этом случае каждый компьютер через специальный сетевой адаптер подключается отдельным кабелем к объединяющему устройству.

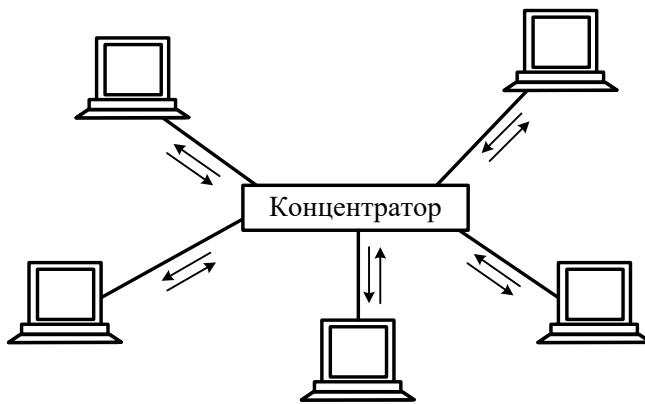


Рис. 1.3. Топология «звезды»

При необходимости можно объединять вместе несколько сетей с топологией «звезды», при этом получаются разветвленные конфигурации сети. В каждой точке ветвления необходимо использовать специальные соединители (распределители, повторители или устройства доступа).

Примером звездообразной топологии является топология *Ethernet* с кабелем типа *витая пара* 10BASE-T, 100BASE-T и т. д. Центром «звезды» обычно является *Hub* (хаб, концентратор).

Звездообразная топология обеспечивает защиту от разрыва кабеля. Если кабель рабочей станции будет поврежден, это не приведет к выходу из строя всего сегмента сети. Она позволяет также легко диагностировать проблемы подключения, так как каждая рабочая станция имеет свой собственный кабельный сегмент, подключенный к концентратору. Для диагностики достаточно найти разрыв кабеля, который ведет к неработающей станции. Остальная часть сети продолжает нормально работать.

Однако звездообразная топология имеет и недостатки. Во-первых, она требует для организации сети большое количество кабеля. Во-вторых, концентраторы довольно дороги. В-третьих, кабельные концентраторы при большом количестве кабеля трудно обслуживать. Однако в большинстве случаев в такой топологии используется недорогой кабель типа *витая пара*. В некоторых случаях можно даже использовать существующие телефонные кабели. Кроме того, для диагностики и тестирования выгодно собирать все кабельные концы в одном месте. По сравнению с концентраторами *ArcNet* концентраторы *Ethernet* и *MAU Token Ring* достаточно дороги. Новые подобные концентраторы включают в себя средства тестирования и диагностики, что делает их еще более дорогими.

1.1.4. Древовидные топологии

Кроме трех рассмотренных базовых топологий нередко применяется также сетевая топология «дерево» (*tree*), которую можно рассматривать как комбинацию нескольких звезд. Причем, как и в случае «звезды», «дерево» может быть активным или истинным (рис. 1.4).

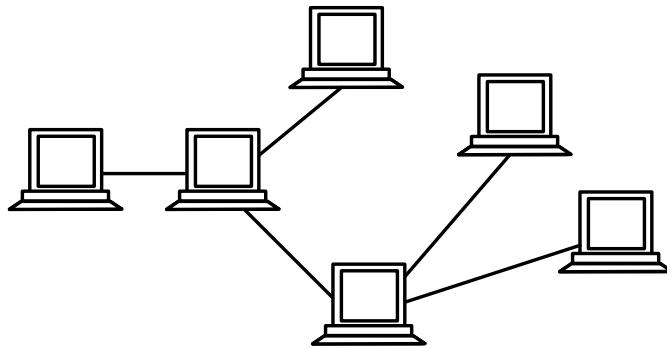


Рис. 1.4. Топология «активное дерево»

Также «дерево» может быть пассивным (рис. 1.5). При «активном дереве» в центрах объединения нескольких линий связи находятся центральные компьютеры, а при «пассивном» – концентраторы (хабы).

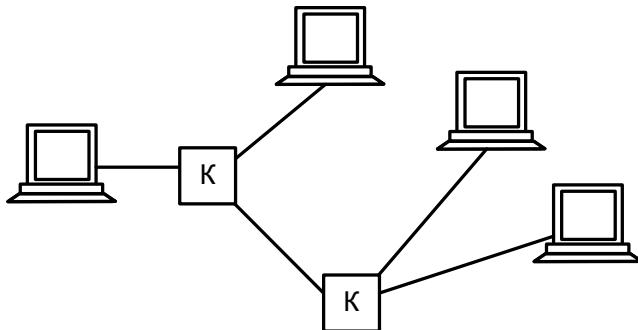


Рис. 1.5. Топология «пассивное дерево» (К – концентраторы)

Сетевая топология *«fat tree»* (утолщенное дерево) является дешевой и эффективной для суперкомпьютеров (рис. 1.6).

В отличие от классической топологии «дерево», в которой все связи между узлами одинаковы, связи в «утолщенном дереве» становятся более широкими (производительными по пропускной способности) с каждым уровнем по мере приближения к корню дерева. Часто используют удвоение пропускной способности на каждом уровне. Се-

ти с топологией «fat tree» являются предпочтительными для построения кластерных межсоединений.

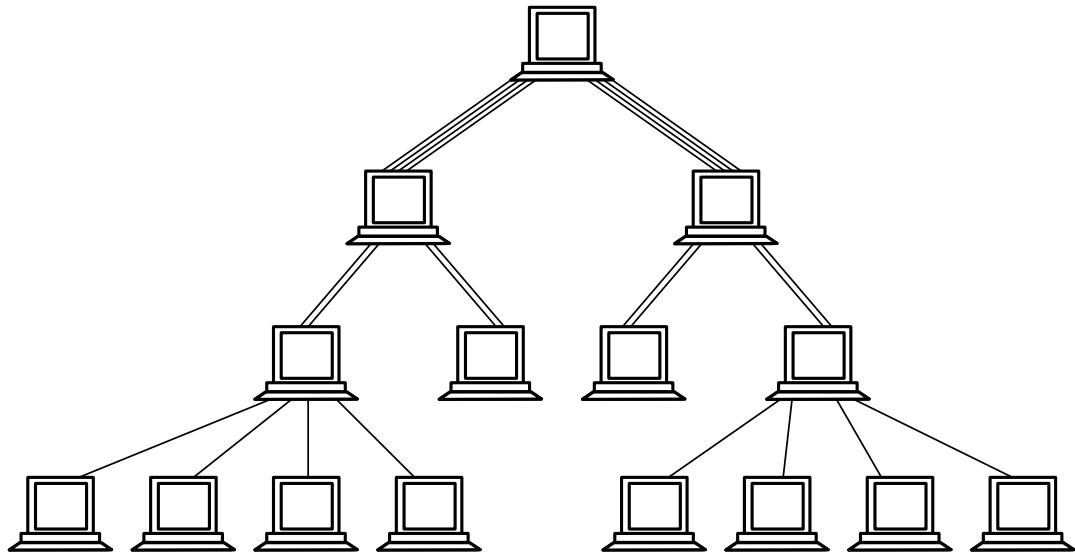


Рис. 1.6. Топология «fat tree»

1.1.5. Комбинированные топологии

Довольно часто применяются комбинированные топологии, среди которых наиболее распространены «звездно-шинная» (star – bus) (рис. 1.7) и «звездно-кольцевая» (star – ring) (рис. 1.8).

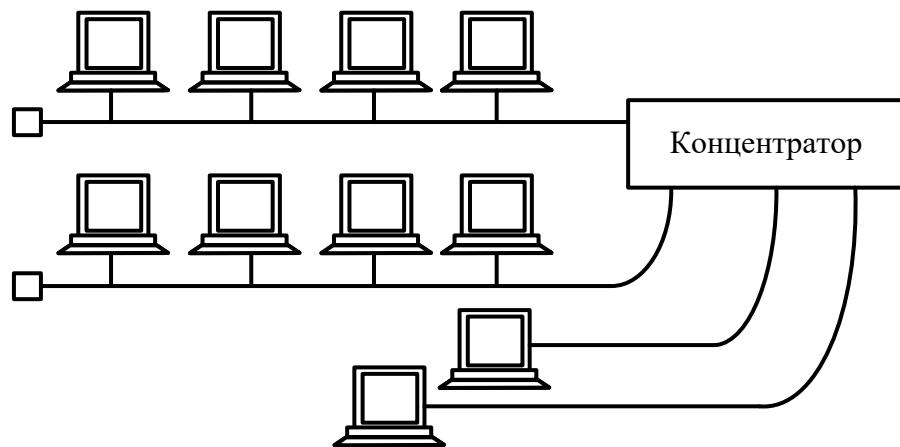


Рис. 1.7. Пример «звездно-шинной» топологии

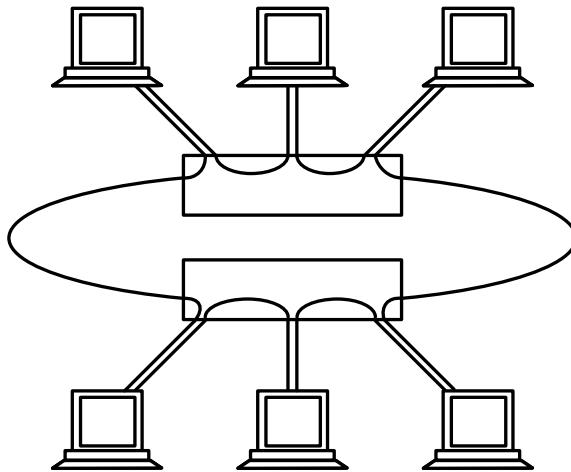


Рис. 1.8. Пример «звездно-кольцевой» топологии

В «звездно-шинной» топологии используется комбинация «шины» и «пассивной звезды». К концентратору подключаются как отдельные компьютеры, так и целые шинные сегменты. На самом деле реализуется физическая топология «шина», включающая все компьютеры сети. В данной топологии может использоваться и несколько концентраторов, соединенных между собой и образующих так называемую магистральную, опорную шину. К каждому из концентраторов при этом подключаются отдельные компьютеры или шинные сегменты. В результате получается «звездно-шинное дерево». Таким образом, пользователь может гибко комбинировать преимущества шинной и звездной топологий, а также легко изменять количество компьютеров, подключенных к сети. С точки зрения распространения информации данная топология равнозначна классической «шине».

В случае «звездно-кольцевой» топологии в кольцо объединяются не сами компьютеры, а специальные концентраторы (прямоугольник на рис. 1.8), к которым в свою очередь подключаются компьютеры с помощью звездообразных двойных линий связи. В действительности все компьютеры сети включаются в замкнутое кольцо, так как внутри концентраторов линии связи образуют замкнутый контур. Данная топология дает возможность комбинировать преимущества звездной и кольцевой топологий. Например, концентраторы позволяют собрать в одно место все точки подключения кабелей сети. Если говорить о распространении информации, данная топология равнозначна классическому «кольцу».

1.1.6. Ячеистые топологии

В *сеточной* (ячеистой) (mesh) топологии компьютеры связываются между собой не одной, а многими линиями связи, образующими сетку. Выделяют полную (рис. 1.9, *а*) и частичную (рис. 1.9, *б*) сеточную топологию.

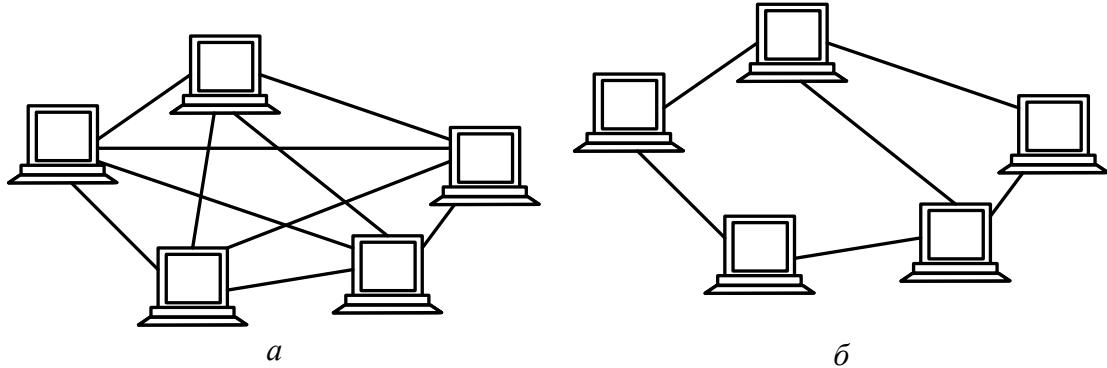


Рис. 1.9. Сеточная топология

В *полной сеточной топологии* каждый компьютер напрямую связан со всеми остальными компьютерами. В этом случае при увеличении числа компьютеров резко возрастает количество линий связи. Кроме того, любое изменение в конфигурации сети требует внесения изменений в сетевую аппаратуру всех компьютеров, поэтому полная сеточная топология не получила широкого распространения.

Частичная сеточная топология предполагает прямые связи только для самых активных компьютеров, передающих максимальные объемы информации. Остальные компьютеры соединяются через промежуточные узлы. Сеточная топология позволяет выбирать маршрут для доставки информации от абонента к абоненту, обходя неисправные участки. С одной стороны, это увеличивает надежность сети, с другой же – требует существенного усложнения сетевой аппаратуры, которая должна выбирать маршрут.

В заключение несколько слов о *решетчатой* топологии, в которой узлы образуют регулярную многомерную решетку. При этом каждое ребро решетки параллельно ее оси и соединяет два смежных узла вдоль этой оси.

Одномерная «решетка» – это цепь, соединяющая два внешних узла (имеющие лишь одного соседа) через некоторое количество внутренних (у которых по два соседа – слева и справа). При соединении обоих внешних узлов получается топология «кольцо». Двух- и трехмерные решетки используются в архитектуре суперкомпьютеров.

Многомерная решетка, соединенная циклически в более чем одном измерении, называется «тор».

Основным достоинством топологии «решетка» является высокая надежность, а недостатком – сложность реализации.

1.1.7. Многозначность понятия топологии

Топология сети указывает не только на физическое расположение компьютеров, как часто считают, но, что гораздо важнее, на характер связей между ними, особенности распространения информации, сигналов по сети. Именно характер связей определяет степень отказоустойчивости сети, требуемую сложность сетевой аппаратуры, наиболее подходящий метод управления обменом, возможные типы сред передачи (каналов связи), допустимый размер сети (длина линий связи и количество абонентов), необходимость электрического согласования и многое другое. Более того, физическое расположение компьютеров, соединяемых сетью, почти не влияет на выбор топологии. Как бы ни были расположены компьютеры, их можно соединить с помощью любой заранее выбранной топологии (рис. 1.10).

В том случае, если соединяемые компьютеры расположены по контуру круга, они могут соединяться, как «звезда» или «шина». Когда компьютеры расположены вокруг некоего центра, их допустимо соединить с помощью топологий «шина» или «кольцо».

Наконец, когда компьютеры расположены в одну линию, они могут соединяться «звездой» или «кольцом». Другое дело, какова будет требуемая длина кабеля.

Строго говоря, при упоминании о топологии сети, могут подразумеваться четыре совершенно разных понятия, относящихся к различным уровням сетевой архитектуры.

Физическая топология – географическая схема расположения компьютеров и прокладки кабелей. В этом смысле, например, «пассивная звезда» ничем не отличается от «активной», поэтому ее нередко называют просто «звездой».

Логическая топология – структура связей, характер распространения сигналов по сети. Это наиболее правильное определение топологии.

Топология управления обменом – принцип и последовательность передачи права на захват сети между отдельными компьютерами.

Информационная топология – направление потоков информации, передаваемой по сети.

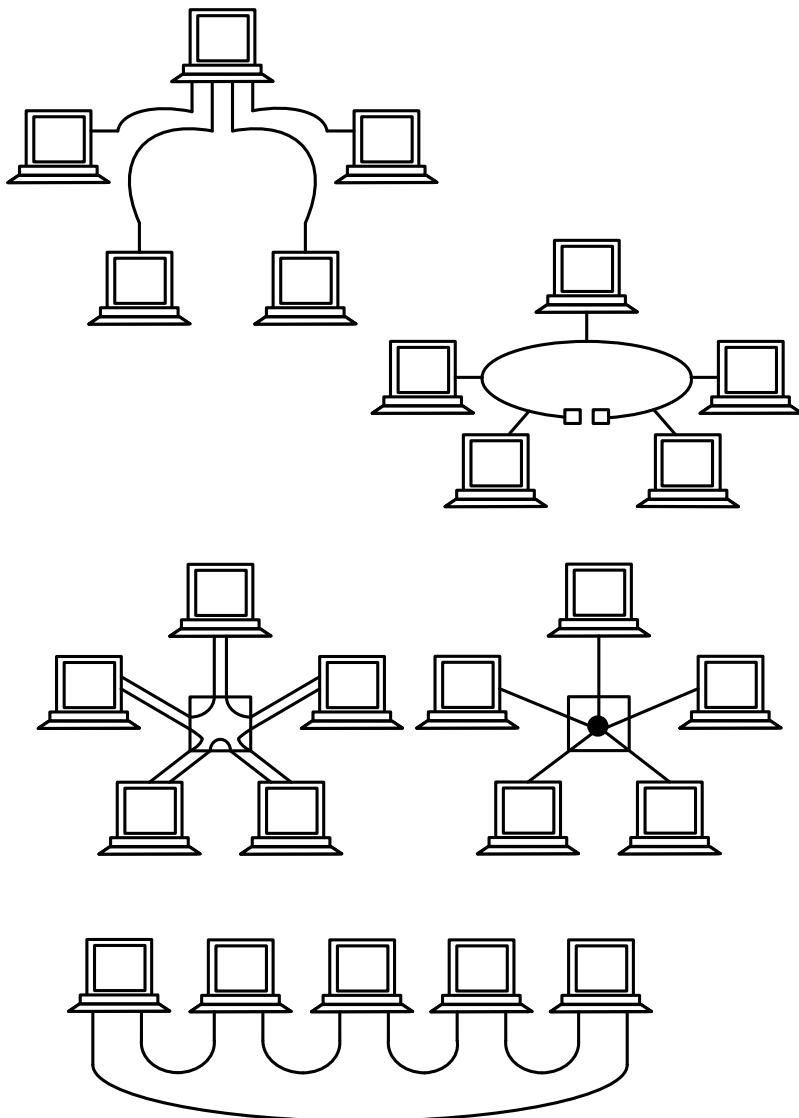


Рис. 1.10. Примеры использования разных топологий для соединения компьютеров

Например, сеть с физической и логической топологией «шина» может в качестве метода управления использовать эстафетную передачу права захвата сети (быть в этом смысле «кольцом») и одновременно передавать всю информацию через выделенный компьютер (быть в этом смысле «звездой»). Или сеть с логической топологией «шина» может иметь физическую топологию «звезда» (пассивная) или «дерево» (пассивное).

Сеть с любой физической топологией, логической топологией, топологией управления обменом может считаться «звездой» в смысле информационной топологии, если она построена на основе одного

сервера и нескольких клиентов, общающихся только с этим сервером. В данном случае справедливы все рассуждения о низкой *отказоустойчивости* сети к неполадкам центра (сервера). Точно так же любая сеть может быть названа «шиной» в информационном смысле, если она построена из компьютеров, являющихся одновременно как серверами, так и клиентами. Такая сеть будет малочувствительна к отказам отдельных компьютеров.

1.2. Использование пакета NetCracker для моделирования компьютерных сетей

Программа NetCracker предназначена для проектирования и моделирования компьютерных сетей. Для проектирования структуры сети программа предоставляет возможность выбора необходимого оборудования из встроенной базы данных, а также добавления в базу данных и конфигурирования нового оборудования различных типов. Пользователь размещает выбранные компоненты на наборном поле, задает структуру и тип связей между ними, определяет тип программного обеспечения и характер трафика между узлами сети. В дальнейшем имеется возможность указать перечень анализируемых характеристик и вид отображения статистической информации и выполнить имитационное моделирование спроектированной сети. Окно программы с топологией из восьми компьютеров, четырех коммутаторов и двух соединенных маршрутизаторов изображено на рис. 1.11.

Панель просмотра компонент, имеющихся в базе данных, располагается обычно в левой части окна и включается с помощью команды «View» – «Bars» – «Browser Pane». Панель содержит несколько закладок:

- 1) «Project Hierarchy» предназначена для отображения структуры документов создаваемого проекта сети;
- 2) «Devices» используется для отображения базы данных устройств. Список устройств имеет несколько видов отображения:
 - «Types» (Типы) – устройства в списке группируются по типам. Затем в каждой группе могут выделяться подтипы устройств по функциональным признакам. После этого устройства разделяются по изготовителям;

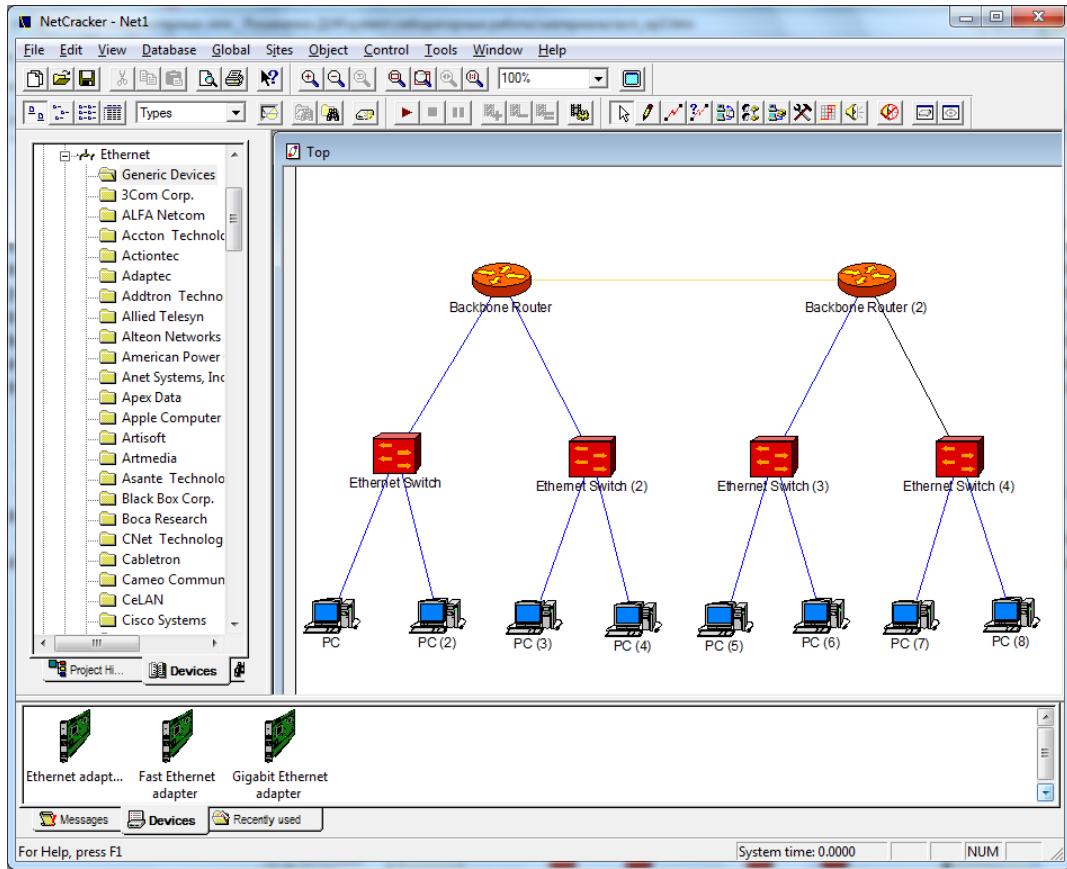


Рис. 1.11. Основное окно программы NetCracker

- «*Vendors*» (Изготовители) – устройства в списке группируются по изготовителям. Затем в каждой группе выделяются подгруппы, соответствующие типу устройств;
- «*User*» (Пользовательские) – устройства, определяемые пользователем. В свою очередь также могут группироваться по типам или изготовителям;
- Закладка «*Compatible Devices*» предназначена для отображения списка совместимых устройств.

В нижней части окна программы обычно располагается панель устройств, которая может быть отображена с помощью команды «View»–«Bars»–«Image Pane». Данная панель предназначена для отображения устройств из выбранной группы.

В правой верхней части главного окна программы располагается основное окно, представляющее собой наборное поле. В нем необходимо размещать используемые компоненты при проектировании структуры сети.

Методика построения компьютерной сети с заданной топологией в пакете NetCracker включает следующие шаги:

1. В окно проекта заносится сетевое оборудование, которое будет использоваться для построения сети. Если необходимо, то в рабочие станции и/или серверы добавляются сетевые адаптеры из списка. Возможно конфигурирование рабочих станций и серверов, которое выполняется при указании на них с нажатием правой кнопкой мыши.

2. В режиме «*Link devices*» соединяются сетевое оборудование и компьютеры.

3. Для того чтобы можно было задать трафик на серверы, обязательно устанавливается соответствующее общее программное обеспечение (ПО) (в списке оборудования выбирается опция «*Network and Enterprise Software*»). Поддержка по умолчанию общим ПО типов трафика приведена в табл. 1.1.

Если выбранное общее ПО не поддерживает конкретный тип трафика, то настройка осуществляется следующим образом:

- кликнуть правой клавишей по серверу в окне проекта;
- выбрать опцию «*Configuration*» в контекстном меню;
- выделить в окне конфигурации установленное на сервер общее ПО и нажать клавишу «*Plug-in Setup*»;
- выбрать вкладку «*Traffic*»;
- установить необходимые флаги типов трафика;
- нажать клавишу «*OK*»;
- закрыть окно конфигурации.

В этом же окне конфигурации, на вкладке «*Server*» можно задать параметры ответа сервера на поступающие запросы.

Таблица 1.1
Поддерживаемый трафик ПО в пакете NetCracker

| Общее ПО | Поддерживаемый трафик |
|------------------------------|------------------------------|
| E-mail server | SMTP; POP3 |
| File-server | Fileclient-server |
| SQL-server | SQL |
| FTP-server | FTP |
| Small office database server | Data base client-server; SQL |
| HTTP-server | HTTP |

4. После выбора типа трафика необходимо задать сам трафик между компьютерами. Для этого на панели инструментов надо нажать кнопку «*Set Traffic*», затем поочередно щелкнуть левой кнопкой мыши станцию-клиента и сервер, с которым клиент будет обмениваться данными. Трафик можно также задать и между клиентами. Направление трафика определяется от первого щелчка ко второму. Изменять свойства трафика можно с помощью пункта меню «*Global*» – «*Data Flow*», в том числе добавлять и удалять сетевой трафик.

5. При выборе компьютера или сегмента сети необходимо в соответствии с вариантом задания указать типы отображаемой статистики. Для этого следует выбрать в выпадающем меню пункт «*Statistics*», а в появившемся окне галочками отметить, в каком виде выводить статистику. Статистику можно выводить в виде диаграммы, числа, графика или голосом. Далее нажать «*OK*».

6. В случае, когда при построении сети показываются связи между различными сегментами (например, когда требуется показать связи между зданиями и строение сети внутри здания), следует нажать на правую кнопку мыши, и в выпадающем меню выбрать пункт «*Expand*». После этого можно продолжать рисовать сеть на новом листе.

7. Процесс имитации запускается с помощью кнопки «*Start*». После окончания процесса имитации отчеты выводятся следующим образом: в меню выбирается пункт «*Tools*» – «*Reports*» – «*Wizard*» – «*Statistical*» в зависимости от задания. Отчет можно также получить, не используя услуги мастера, а просто выбрав соответствующий пункт в подменю «*Reports*». Полученный отчет можно распечатать или сохранить в виде файла. Полученный рисунок сети можно вывести на печать, используя меню «*File*» – «*Print*».

1.3. Лабораторная работа № 1

Цель: изучение моделирования компьютерных сетей с заданной топологией для выявления их достоинств и недостатков, а также анализ различных типов трафика в моделируемых сетях, сбор статистики.

Задание: смоделировать в пакете NetCracker сеть с топологиями согласно варианту (табл. 1.2). Организовать передачу по сети различных пакетов. Проанализировать статистику передачи пакетов.

Таблица 1.2

Варианты заданий для лабораторной работы № 1

| № варианта | Моделируемые топологии | Количество хостов в сети | Типы трафика |
|------------|------------------------|--------------------------|--------------------------|
| 1 | 2 | 3 | 4 |
| 1 | кольцевая | 6 | FTP, HTTP, SMTP, POP3 |
| | полная сеточная | 6 | |
| | звездно-кольцевая | 10 | |
| | активное дерево | 12 | |
| 2 | звездообразная | 6 | FTP, HTTP, SMTP, POP3 |
| | полная сеточная | 6 | |
| | звездно-кольцевая | 12 | |
| | пассивное дерево | 8 | |
| 3 | звездообразная | 7 | FTP, HTTP, SMTP, POP3 |
| | частичная сеточная | 7 (4 активных хоста) | |
| | звездно-кольцевая | 10 | |
| | утолщенное дерево | 16 | |
| 4 | кольцевая | 8 | FTP, HTTP, SMTP, POP3 |
| | частичная сеточная | 8 (5 активных хостов) | |
| | звездно-кольцевая | 8 | |
| | пассивное дерево | 6 | |
| 5 | звездообразная | 5 | FTP, HTTP, SMTP, POP3 |
| | частичная сеточная | 7 (5 активных хостов) | |
| | звездно-кольцевая | 8 | |
| | активное дерево | 10 | |

2. СТАТИЧЕСКАЯ IP-АДРЕСАЦИЯ В СЕТЯХ С ОДНОРАНГОВОЙ АРХИТЕКТУРОЙ

Архитектура сети определяет основные элементы сети, характеризует ее общую логическую организацию, техническое обеспечение, программное обеспечение, описывает методы кодирования. Архитектура также определяет принципы функционирования и интерфейс пользователя.

В данном лабораторном практикуме будут рассмотрены два вида архитектур:

- одноранговая архитектура;
- архитектура *клиент – сервер*.

2.1. Понятие одноранговой архитектуры

Одноранговая архитектура (peer-to-peer architecture) – это концепция информационной сети, в которой ее ресурсы рассредоточены по всем взаимодействующим между собой системам (рис. 2.1). Данная архитектура характеризуется тем, что в ней все системы равноправны.

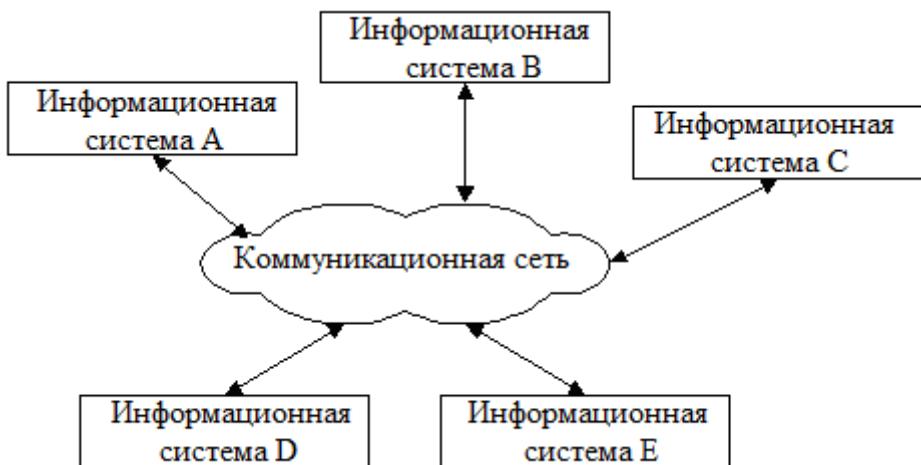


Рис. 2.1. Одноранговая архитектура

К одноранговым сетям относятся малые сети, где любая рабочая станция может выполнять одновременно функции файлового сервера и рабочей станции. В одноранговых ЛВС дисковое пространство и файлы на любом компьютере могут быть общими. Чтобы ресурс стал общим, его необходимо отдать в общее пользование, используя службы удаленного доступа сетевых одноранговых операционных систем.

В зависимости от того, как будет установлена защита данных, другие пользователи смогут пользоваться файлами сразу же после их создания. Одноранговые ЛВС достаточно хороши только для небольших рабочих групп.

Одноранговые ЛВС являются наиболее легким и дешевым типом сетей для установки. Они требуют на компьютере, кроме сетевой карты и сетевого носителя, наличие пользовательской операционной системы. При соединении компьютеров, пользователи могут предоставлять ресурсы и информацию в совместное пользование.

Одноранговые сети имеют следующие преимущества:

- они легки в установке и настройке;
- отдельные ПК не зависят от выделенного сервера;
- пользователи в состоянии контролировать свои ресурсы;
- малая стоимость и легкая эксплуатация;
- минимум оборудования и программного обеспечения;
- нет необходимости в администраторе;
- хорошо подходят для сетей с количеством пользователей, не превышающим десяти.

Проблемой одноранговой архитектуры является ситуация, когда компьютеры отключаются от сети. В этих случаях из сети исчезают виды сервиса, которые они предоставляли. Сетевую безопасность одновременно можно применить только к одному ресурсу, и пользователь должен помнить столько паролей, сколько сетевых ресурсов. При получении доступа к разделяемому ресурсу ощущается падение производительности компьютера. Существенным недостатком одноранговых сетей является отсутствие централизованного администрирования.

Для изучения основных приемов построения сетей с заданной архитектурой целесообразно использовать технологию виртуализации операционных систем.

2.2. Виртуализация операционных систем

Платформа Oracle VM VirtualBox представляет собой систему виртуализации для host-систем Windows, Linux и Mac OS и обеспечивает взаимодействие с гостевыми операционными системами Windows (2000/XP/2003/Vista/Seven), Linux (Ubuntu/Debian/ OpenSUSE/ Mandriva и пр.), OpenBSD, FreeBSD, OS/2 Warp.

Ключевые возможности VirtualBox заключаются в следующем:

- x86-виртуализация (при этом поддержка аппаратной реализации Intel VT и AMD-V необязательна);
- поддержка многопроцессорности и многоядерности;
- поддержка виртуализации сетевых устройств;
- поддержка виртуализации USB-host;
- высокая производительность и скромное потребление ресурсов персонального компьютера;
- поддержка различных видов сетевого взаимодействия (NAT, Host Network, Bridge, Internal);
- возможность сохранения снимков виртуальной машины (snapshots), к которым может быть произведен откат из любого состояния гостевой системы;
- настройка и управление приложением VirtualBox и виртуальной системой из командной строки.

Установка данного программного продукта на компьютер является стандартной, поэтому уделять внимание данной процедуре не будем. Рассмотрим далее процесс создания и первичной настройки виртуальной машины.

Запустим приложение Oracle VM VirtualBox (при установке платформы на рабочем столе создается ярлык, которым далее можно будет пользоваться). Для создания виртуальной машины необходимо щелкнуть кнопку «Создать / New» (рис. 2.2).

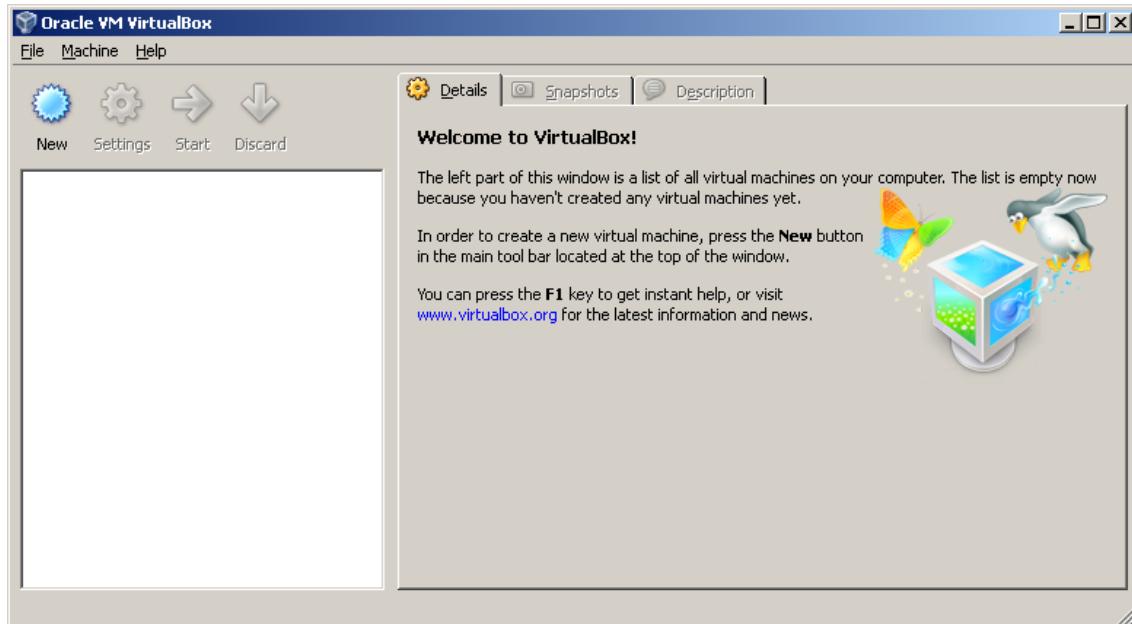


Рис. 2.2. Главное окно программы Oracle VM VirtualBox

После этого откроется новое окно, в котором будет сообщение о запуске мастера создания виртуальной машины. Далее необходимо нажать кнопку «*Next*» и появится новое окно, предлагающее выбрать имя операционной системы, ее семейство и версию (рис. 2.3).

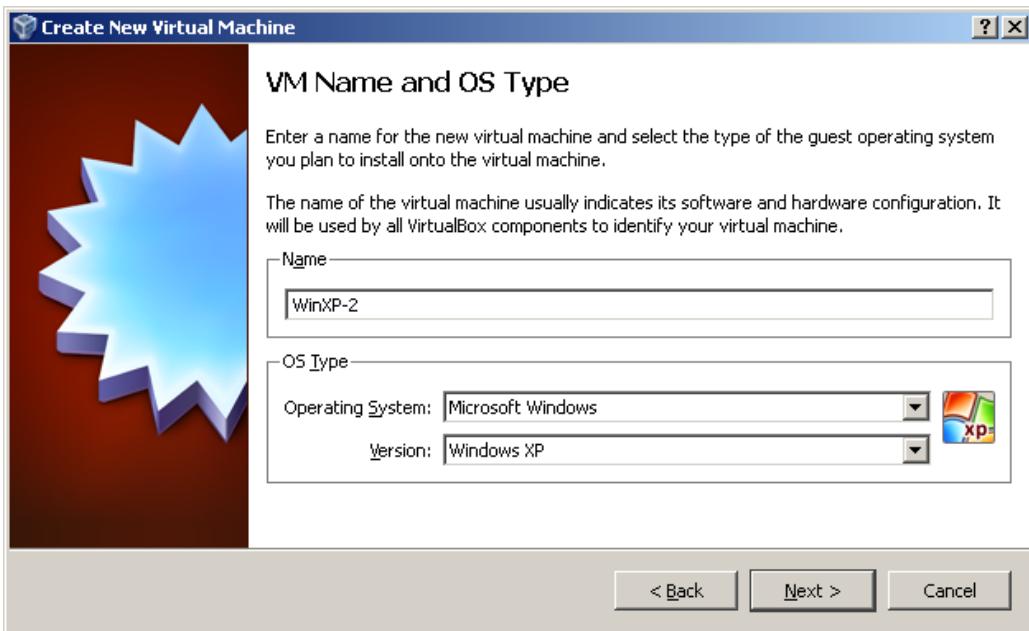


Рис. 2.3. Начальные параметры виртуальной машины

После нажатия кнопки «*Next*» будет предложено определить размер оперативной памяти, выделяемой виртуальной машине (рис. 2.4). Для стабильной работы с виртуальной системой Windows XP необходимо выделять не менее 256 Мбайт оперативной памяти.

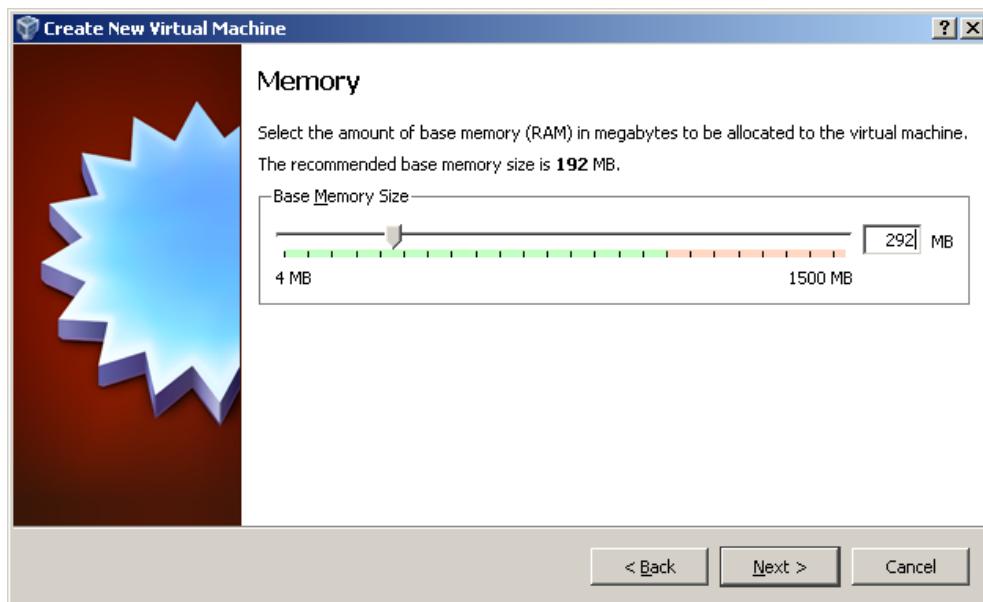


Рис. 2.4. Выделение оперативной памяти для виртуальной машины

Далее потребуется создать виртуальный жесткий диск (рис. 2.5). Если ранее были созданы виртуальные диски, можно использовать их, но в данном случае будет рассмотрен именно процесс создания нового диска. Для подтверждения, что создаваемый жесткий диск является загрузочным, необходимо поставить флажок «Создать новый жесткий диск / Create new hard disk» и нажать кнопку «Next».

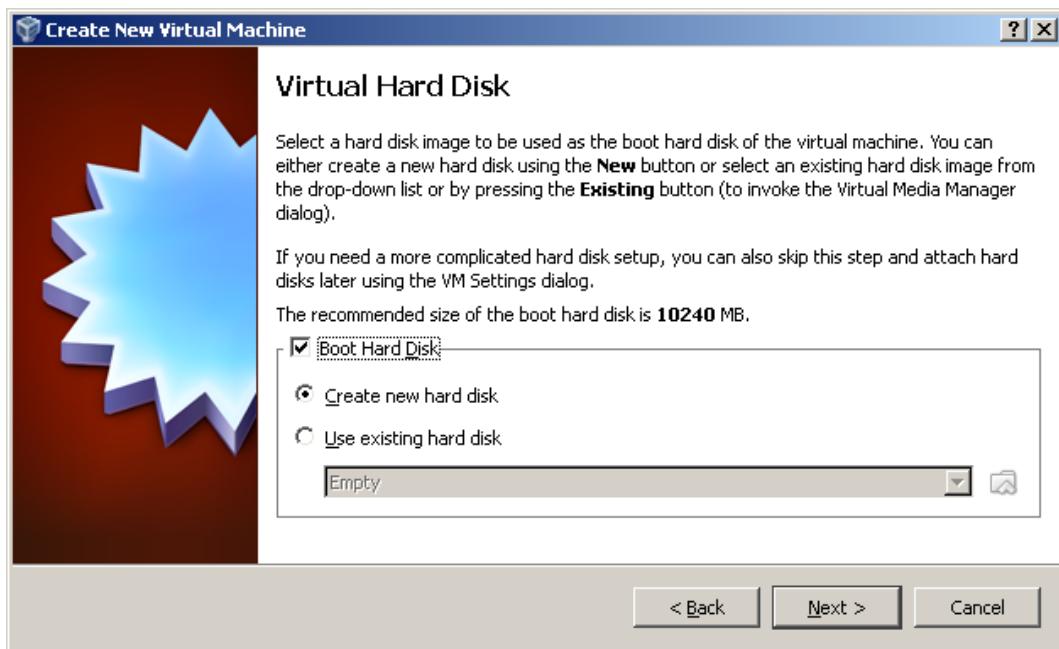


Рис. 2.5. Создание жесткого диска

Далее появится новое окно, которое сообщит, что запущенный мастер поможет в создании виртуального диска. Для продолжения работы необходимо нажать кнопку «Next». В новом окне (рис. 2.6) будет предложено выбрать тип создаваемого диска – «динамически расширяющийся образ» или «образ фиксированного размера». Разница объясняется в справке данного окна.

В следующем окне (рис. 2.7) потребуется выбрать расположение создаваемого виртуального жесткого диска и его размер. Для загрузочного жесткого диска с системой Windows XP достаточно размера, установленного по умолчанию (10 Гб), а вот расположить его лучше вне системного раздела.

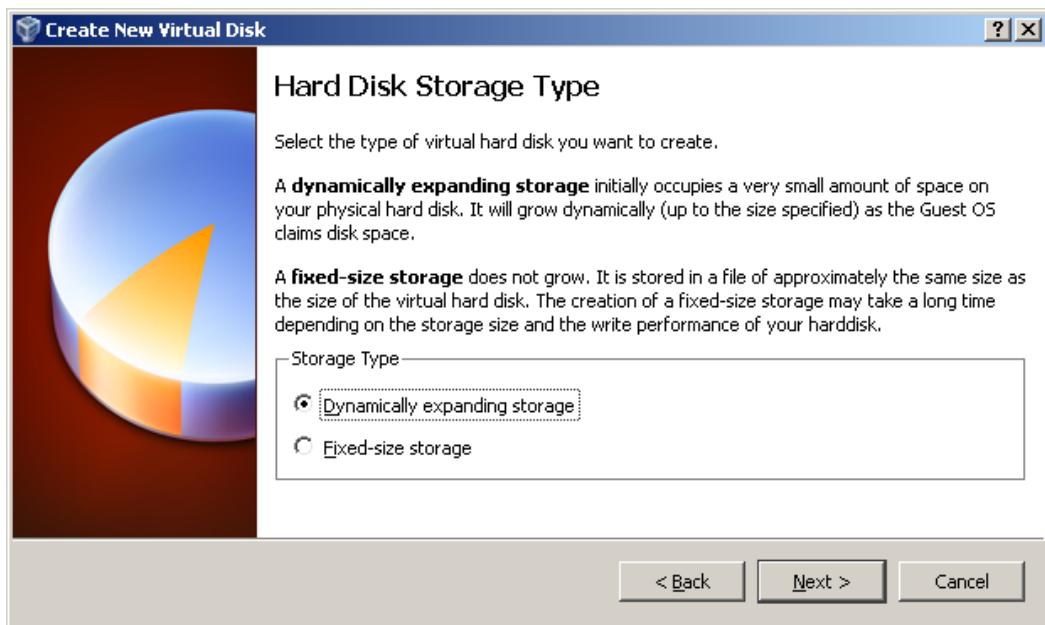


Рис. 2.6. Создание жесткого диска – выбор типа

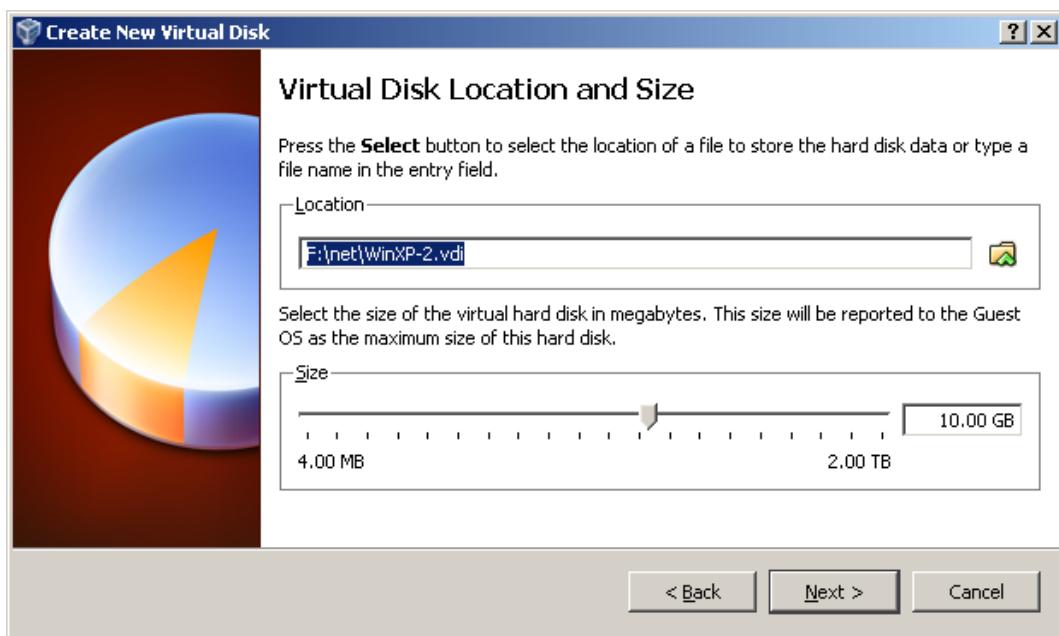


Рис. 2.7. Создание виртуального жесткого диска – выбор размера и расположения

После этого появится окно «*Итог / Summary*», в котором будет указан тип, расположение и размер создаваемого Вами жесткого диска. Для создания диска с такими параметрами, необходимо нажать «*Финиш / Finish*». Далее запустится процесс создания жесткого диска, по окончании которого появится новое окно «*Итог / Summary*»

(рис. 2.8), в котором будут указаны параметры создаваемой виртуальной машины. Далее необходимо нажать «Финиш / Finish» и перейти к настройке аппаратной части виртуальной машины.

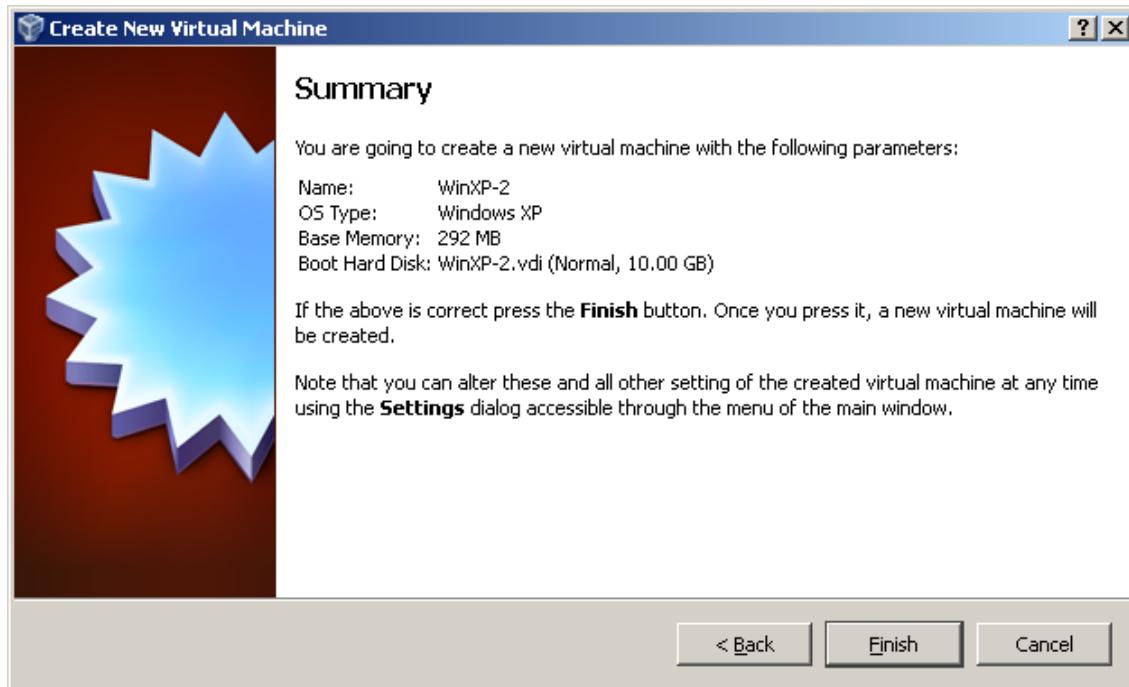


Рис. 2.8. Заключительная стадия создания виртуальной машины

2.2.1. Настройка аппаратной части виртуальной машины

После создания виртуального жесткого диска настала очередь собрать виртуальный компьютер полностью. Для этого снова вернемся к главному окну VirtualBox (рис. 2.9), в нем уже можно увидеть только что созданную виртуальную машину WinXP-2, а в поле с правой стороны представлено ее описание, которое еще не похоже на описание полноценного персонального компьютера.

В колонке слева далее необходимо выбрать WinXP-2 и открыть ее свойства («*Settings*») (рис. 2.10), где колонка с левой стороны напоминает диспетчер устройств. На первой вкладке раздела «*Общие / General*» представлены основные параметры виртуальной машины.

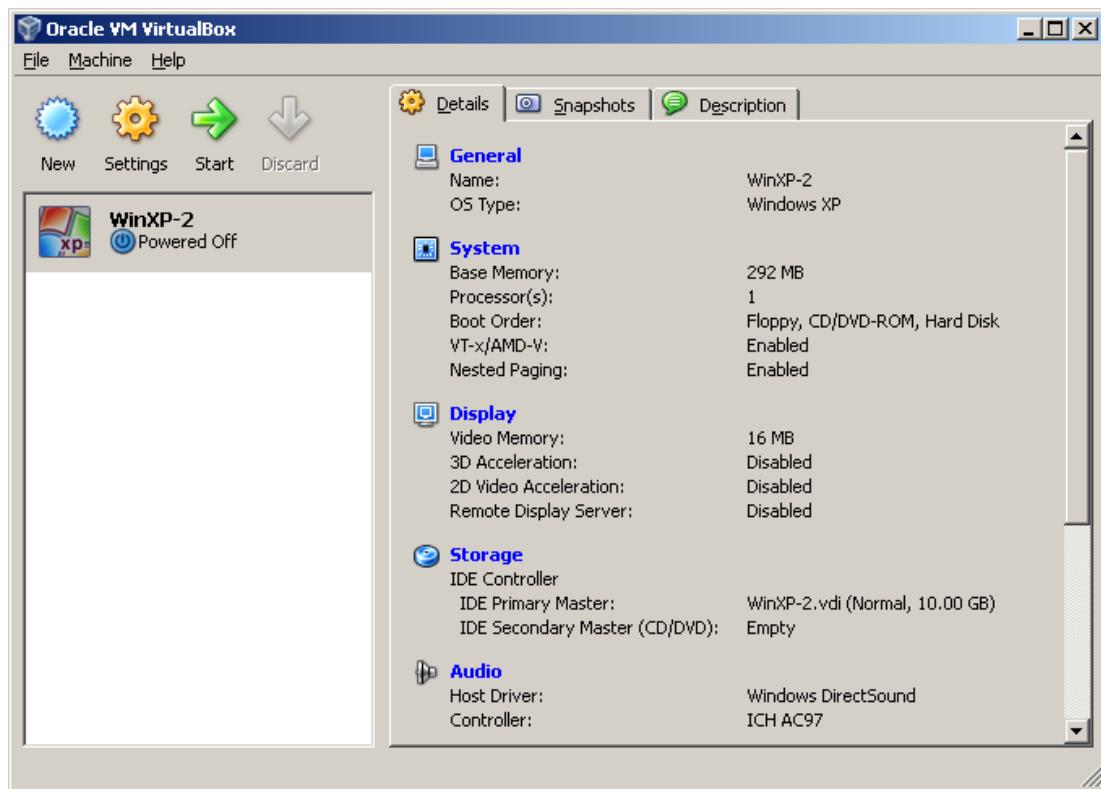


Рис. 2.9. Главное окно на этапе настройки аппаратной части

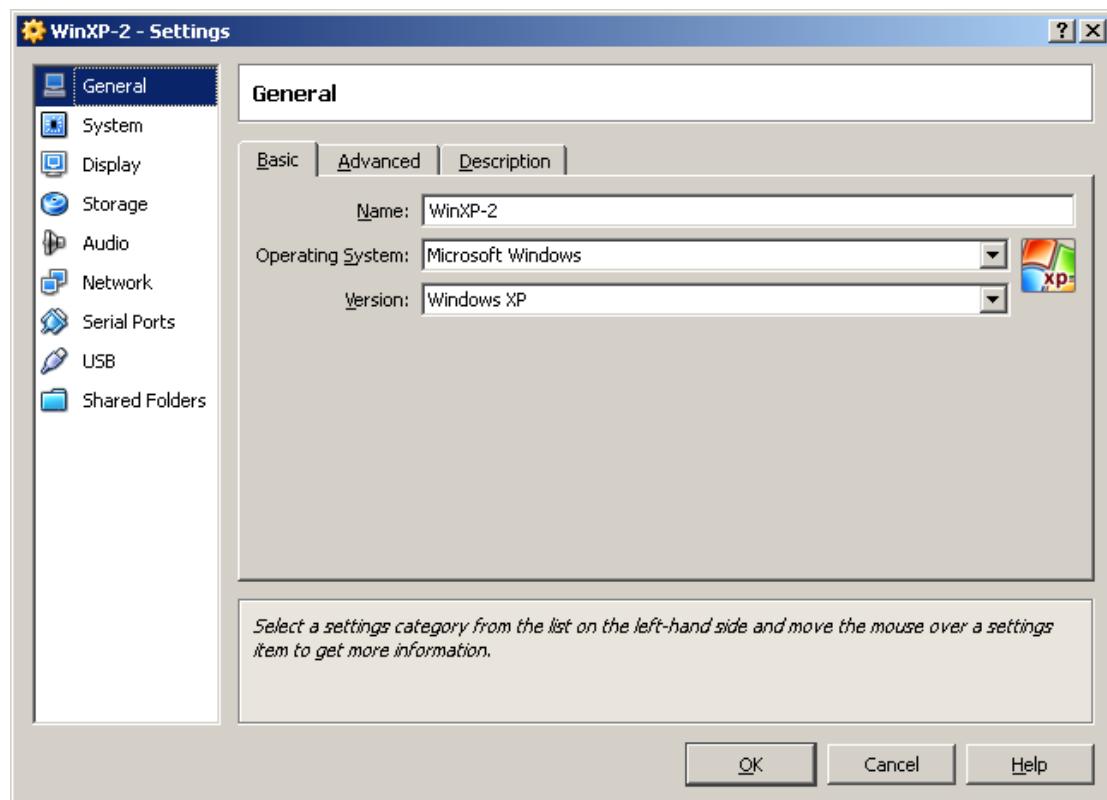


Рис. 2.10. Параметры виртуальной машины

На вкладке «Дополнительно / Advanced» (рис. 2.11) можно выбрать следующие параметры настройки системы:

- «Папка для снимков / Snapshot Folder». Если жесткий диск размещен в отдельной директории, то лучше и эту папку перенести туда же, т. к. снимки имеют большой вес;
- «Общий буфер обмена / Shared Clipboard» – определение того, как будет работать буфер обмена между host-системой и виртуальной машиной. Вариантов работы буфера несколько, но предпочтительно выбирать «дву направленный / Bidirectional», т. к. это обеспечивает максимальное удобство в работе;
- «Сменные носители информации / Removable Media» – лучше поставить флажок в поле «запоминать изменения в процессе работы / Remember Runtime Changes», т. к. данная опция позволит системе запомнить состояние CD/DVD-приводов;
- «Мини тулбар / Mini Toolbar» – это небольшая консоль, содержащая элементы управления виртуальной машиной. Ее лучше применять только в полноэкранном режиме, т. к. она полностью дублируется главным меню рабочего окна виртуальной машины.

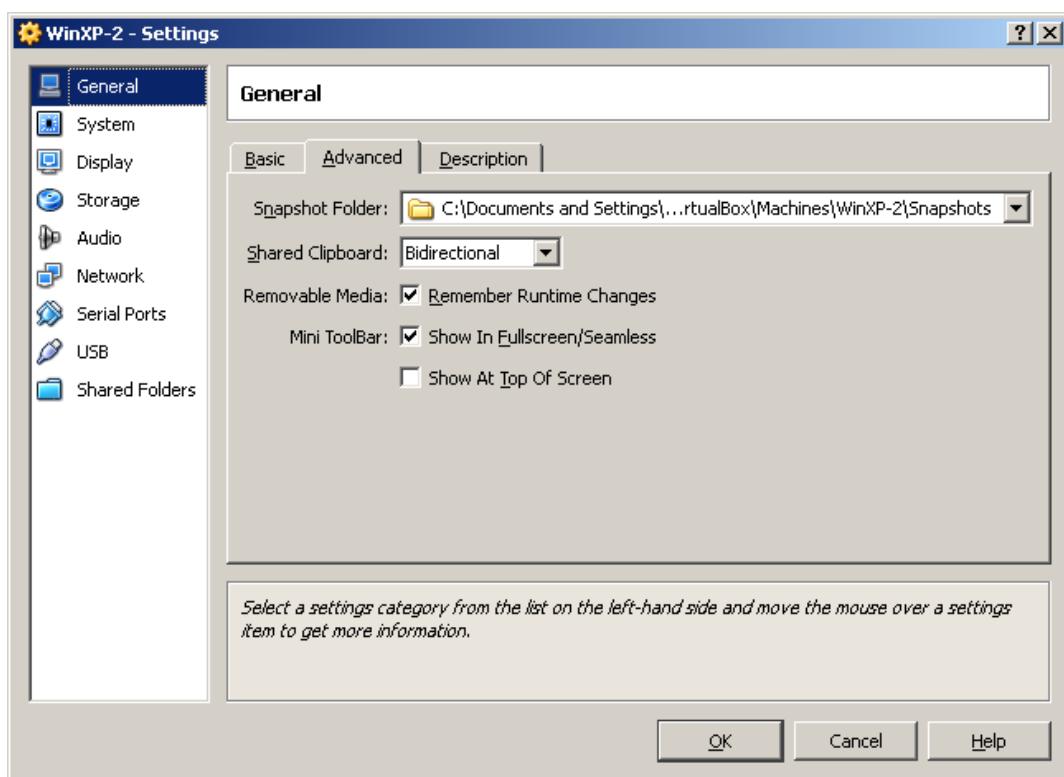


Рис. 2.11. Настройка дополнительных опций виртуальной машины

Располагать ее действительно лучше сверху просто потому, что можно случайно нажать на какой-нибудь элемент управления, пытаясь, например, развернуть окно из панели задач виртуальной машины.

Перейдем далее к разделу «система» и на первой вкладке «Материнская плата / Motherboard» (рис. 2.12) произведем следующие настройки.

1. Целесообразно откорректировать размер оперативной памяти виртуальной машины, хотя окончательно убедиться в правильности выбранного объема можно только после запуска виртуальной машины. Выбирать размер можно, исходя из объема доступной физической памяти, установленной на ПК. Например, при наличии 2ГБайт ОЗУ оптимальным будет выделение 512МБайт, т. е. одной четвертой части, что позволит виртуальной машине работать без малейших зависаний.

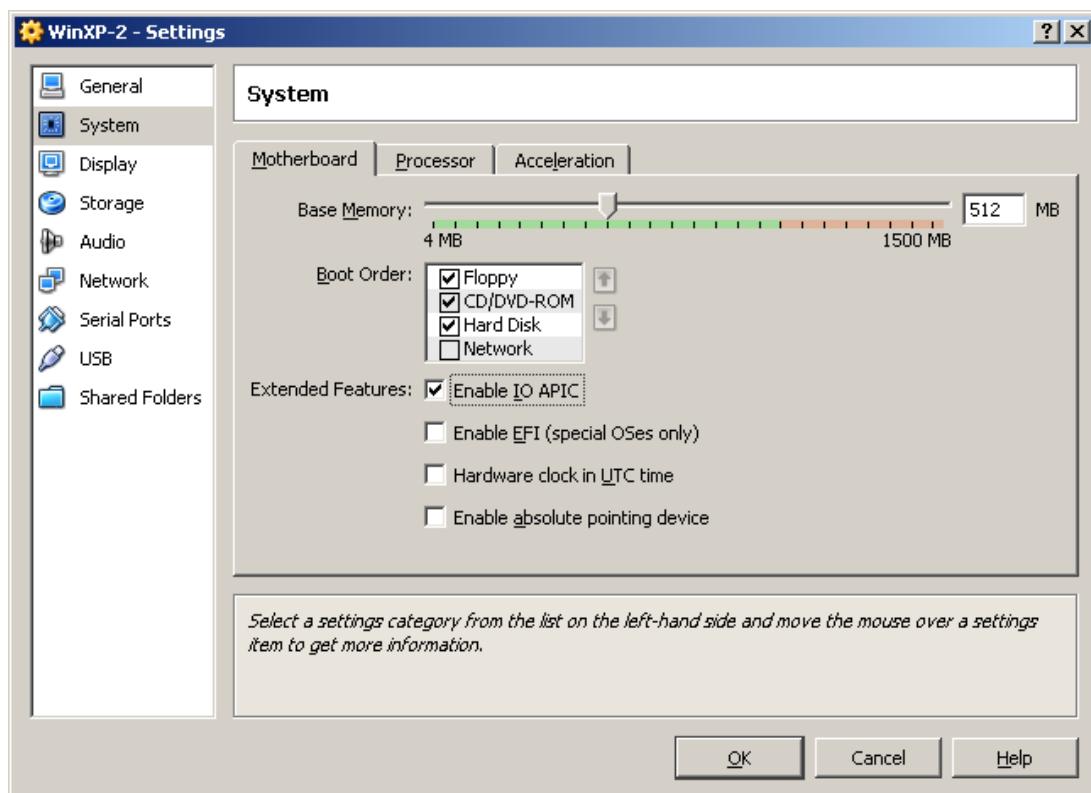


Рис. 2.12. Настройка параметров материнской платы

2. Рекомендуется изменить порядок загрузки – дисковод гибких дисков («дискета») можно вообще отключить, а первым обязательно поставить CD/DVD-ROM, чтобы обеспечить возможность установки ОС с загрузочного диска. При этом в роли загрузочного диска может выступать как компакт-диск, так и образ ISO.

3. Все остальные настройки описаны в динамической справке снизу, и их применение зависит от аппаратной части вашего реального ПК, причем если выставить настройки, неприменимые к используемому ПК, система с виртуальной машиной просто не запустится.

На вкладке «Процессор / Processor» (рис. 2.13) можно выбрать количество процессоров, установленных на виртуальную материнскую плату. Необходимо обратить внимание, что данная опция будет доступна только при условии поддержки аппаратной виртуализации AMD-V или VT-x (рис. 2.14), а также включенной опции IO APIC на предыдущей вкладке.

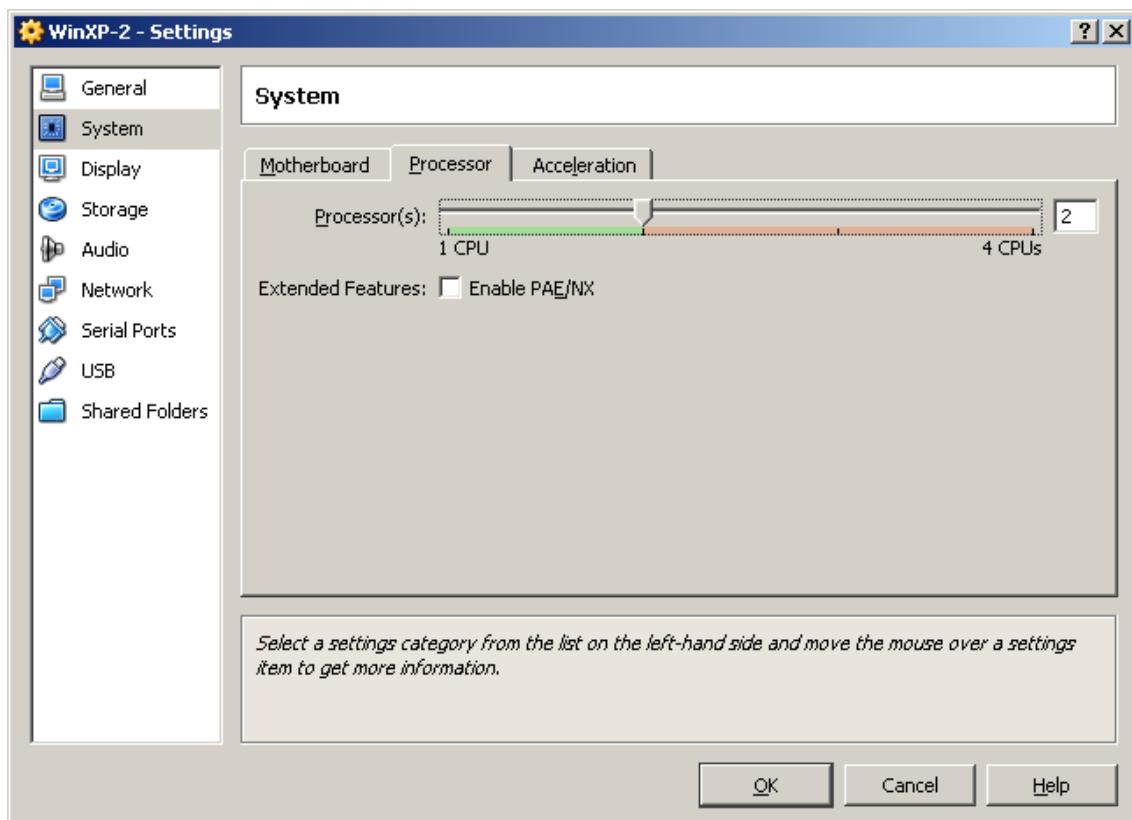


Рис. 2.13. Настройка параметров процессора

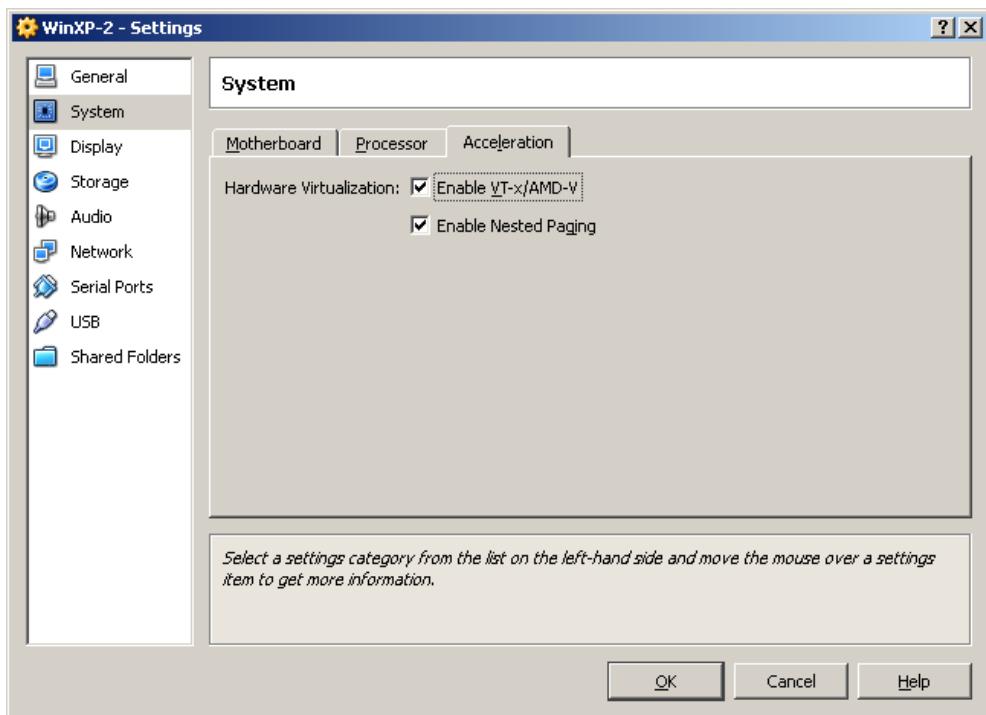


Рис. 2.14. Настройка аппаратной виртуализации

В разделе «Дисплей / Display» (рис. 2.15) на вкладке «Video / Video» можно установить размер памяти виртуальной видеокарты, а также включить 2D и 3D ускорение, причем включение 2D ускорения желательно, а 3D необязательно.

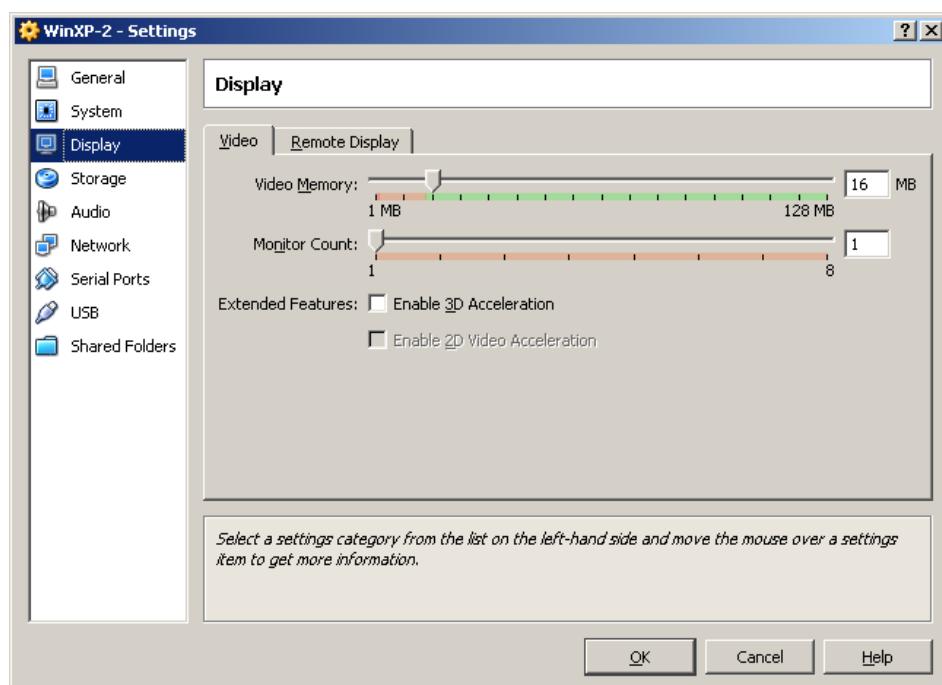


Рис. 2.15. Настройка параметров виртуального видеоадаптера

На вкладке «Удаленный дисплей / Remote Display» можно включить опцию, при которой виртуальная машина будет работать как сервер удаленного рабочего стола (RDP).

В разделе «Носители / Storage» (рис. 2.16) отображен созданный ранее виртуальный жесткий диск и позиция с надписью «Пусто / Empty». Далее следует выделить данную позицию и осуществить ее настройку. Для этого необходимо щелкнуть по пиктограмме папки и в открывшемся окне (рис. 2.17) добавить ISO-образ загрузочного диска операционной системы Windows. На рис. 2.18 представлена процедура добавления ISO-образов в менеджер виртуальных носителей. В него можно внести любое количество образов различного назначения, например, игры, дистрибутивы приложений, базы данных и т. д.

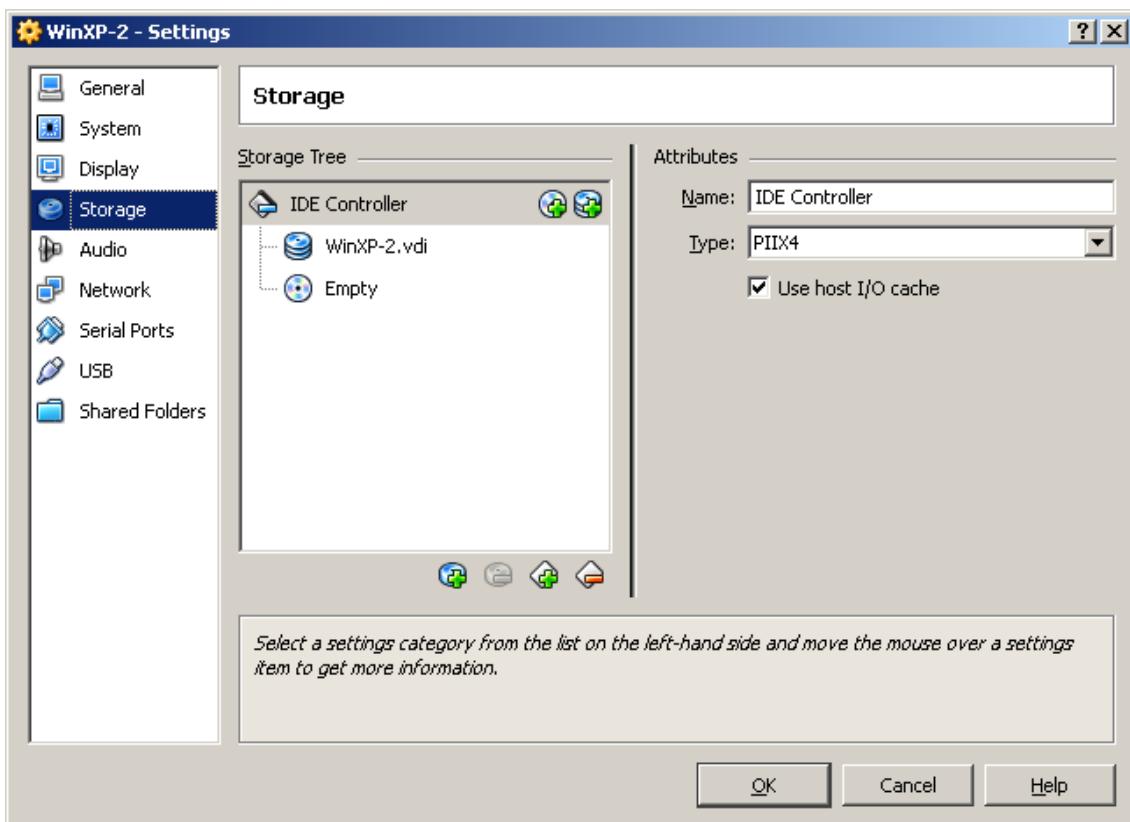


Рис. 2.16. Настройка виртуального CD-ROM

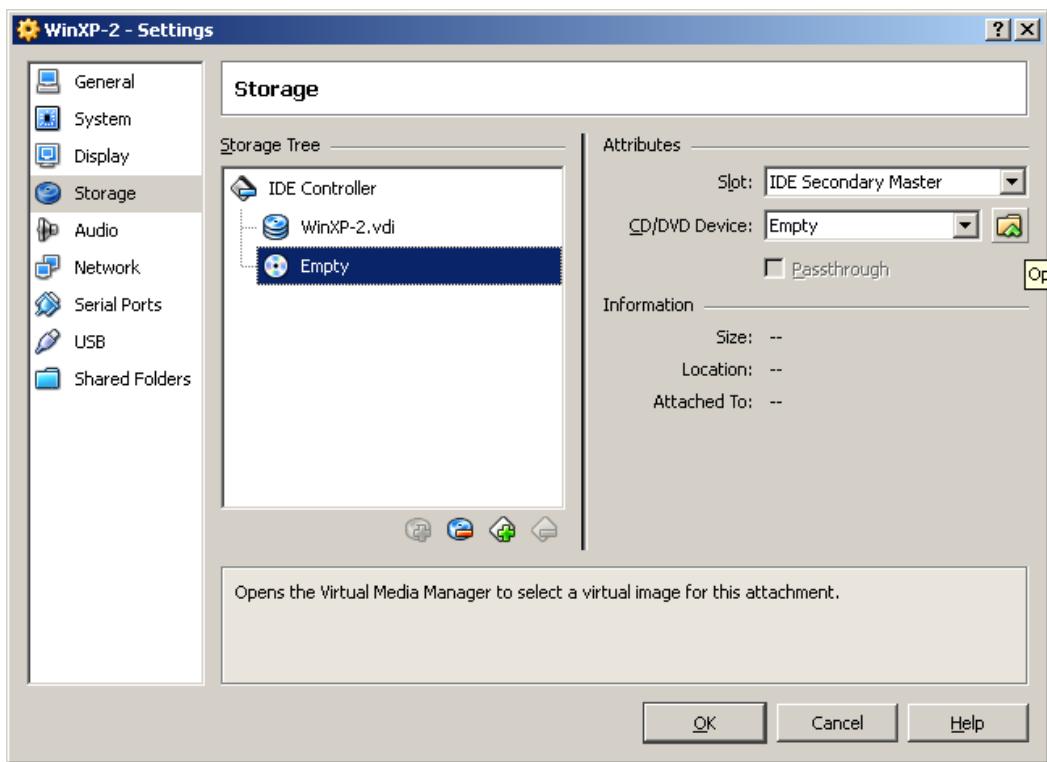


Рис. 2.17. Менеджер виртуальных носителей

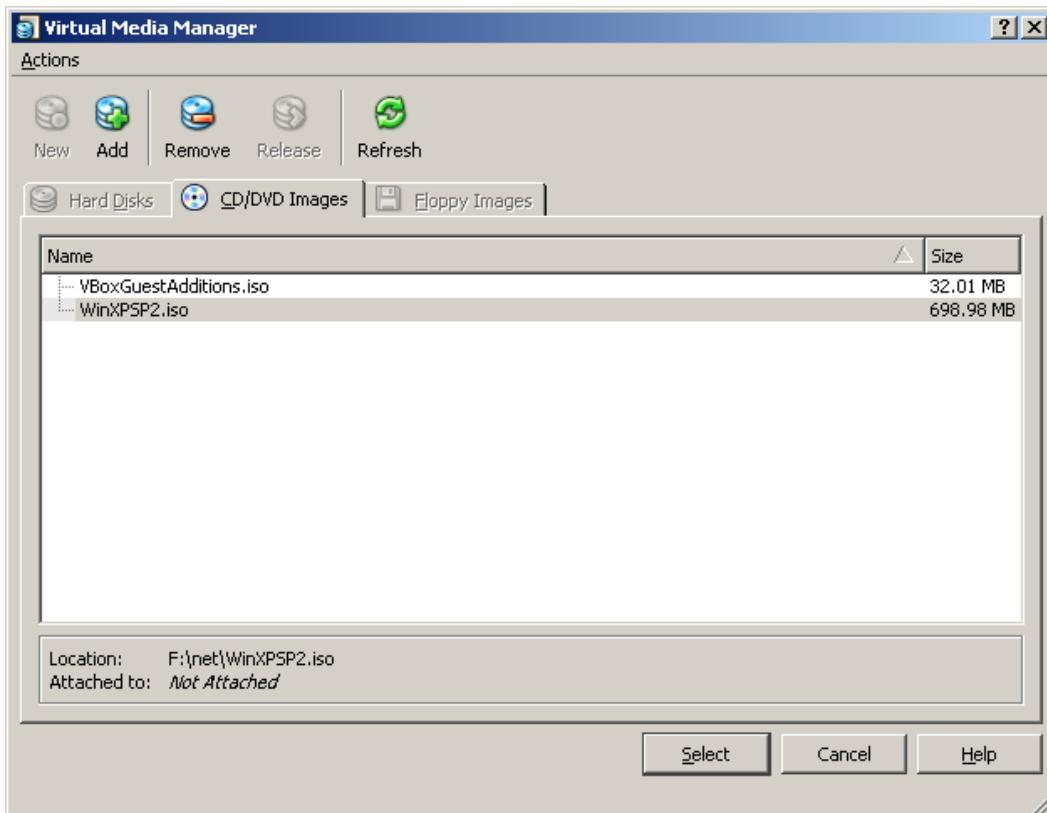


Рис. 2.18. Добавление виртуальных носителей

Далее (рис. 2.19 и 2.20) необходимо настроить слоты подключения накопителей.

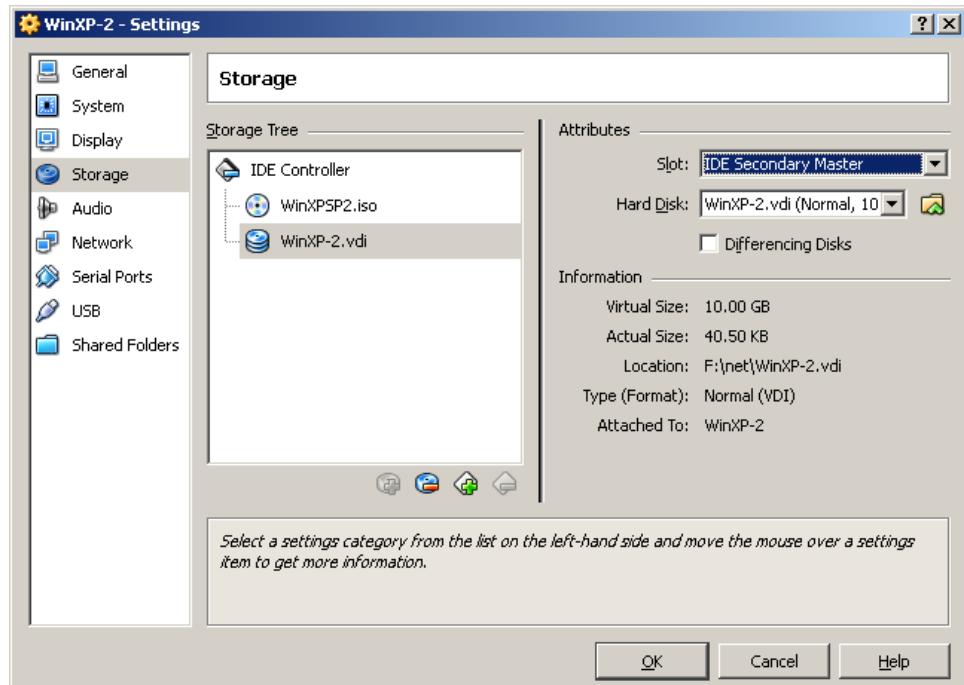


Рис. 2.19. Настройка слота подключения виртуального жесткого диска

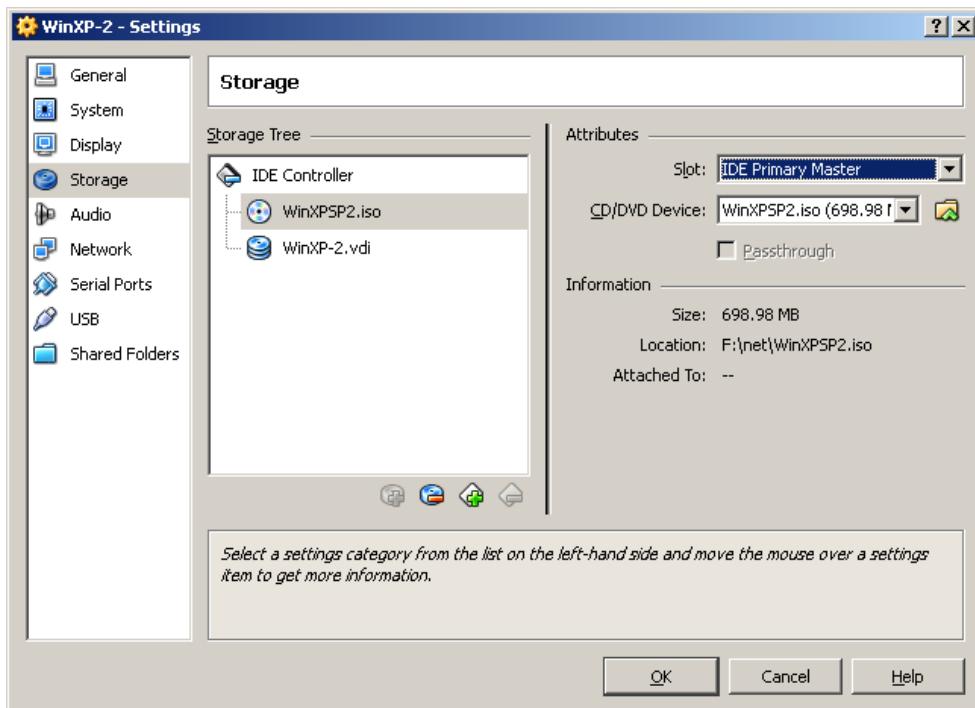


Рис. 2.20. Настройка слота подключения виртуального привода оптических дисков

При этом устанавливается привод компакт-дисков как «Первичный мастер IDE / IDE Primary Master», жесткий диск, содержащий загрузочный раздел, как «Вторичный мастер IDE / IDE Secondary Master», а дополнительный виртуальный жесткий диск – «Первичный слейв IDE / IDE Primary Slave».

При настройке сети в качестве типа подключения необходимо использовать «сетевой мост», как показано на рис. 2.21.

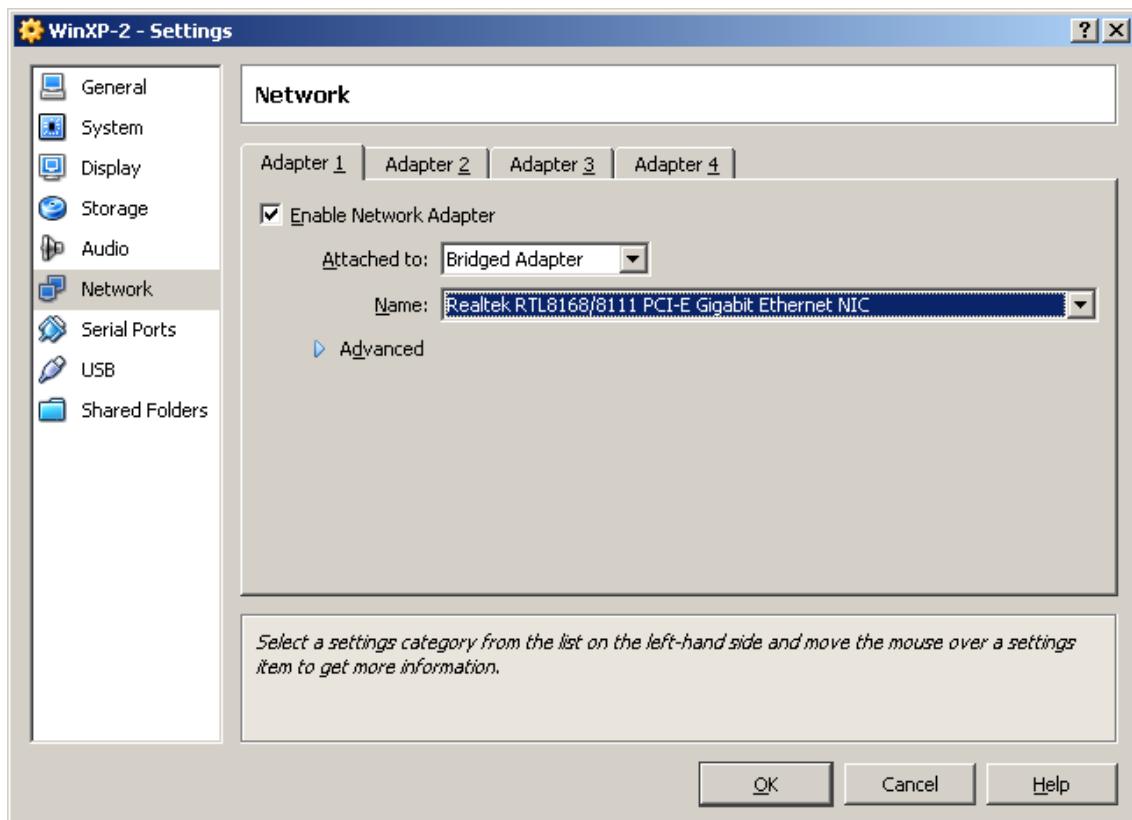


Рис. 2.21. Настройка сетевого адаптера виртуальной машины

Отметим, что в случае использования нескольких виртуальных машин с операционными системами на одном компьютере, целесообразно использовать не «Сетевой мост», а «Внутренняя сеть».

Далее переходим к разделу USB (рис. 2.22), где необходимо поставить оба доступных флажка, а затем, используя кнопку с изображением «вилки» USB и «плюса», добавить все доступные контроллеры.

В разделе «Общие папки / Shared Folders» (рис. 2.23) можно выбрать папки, которые необходимо сделать доступными для виртуальной машины.

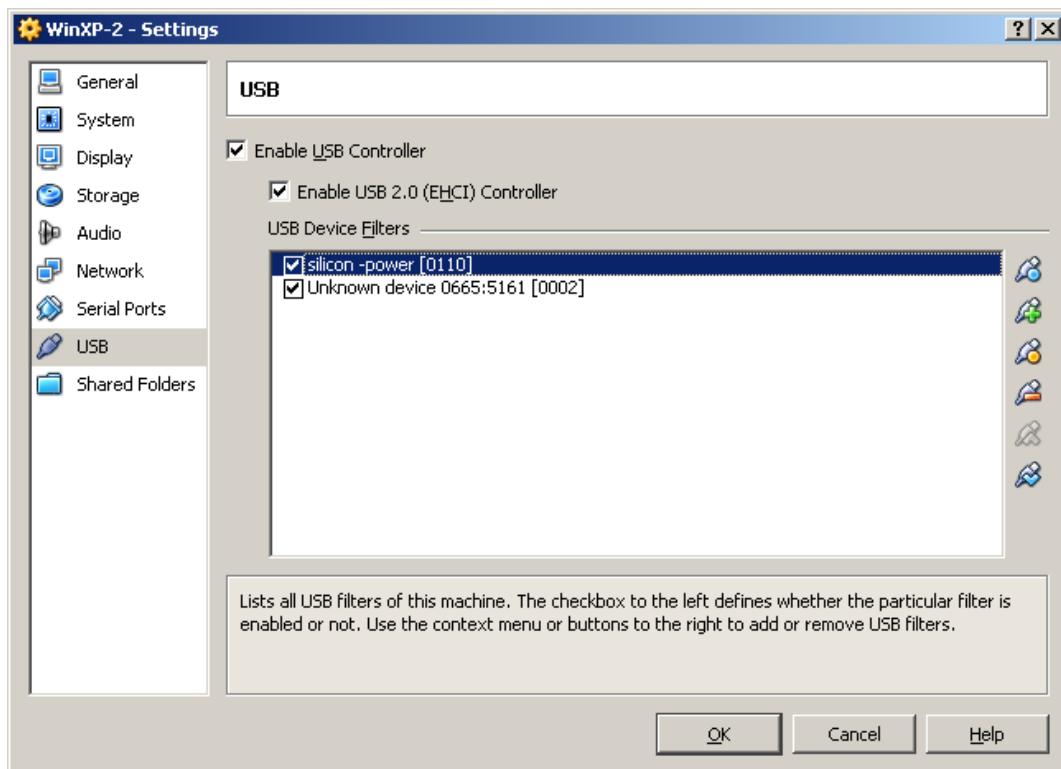


Рис. 2.22. Настройка USB контроллера виртуальной машины

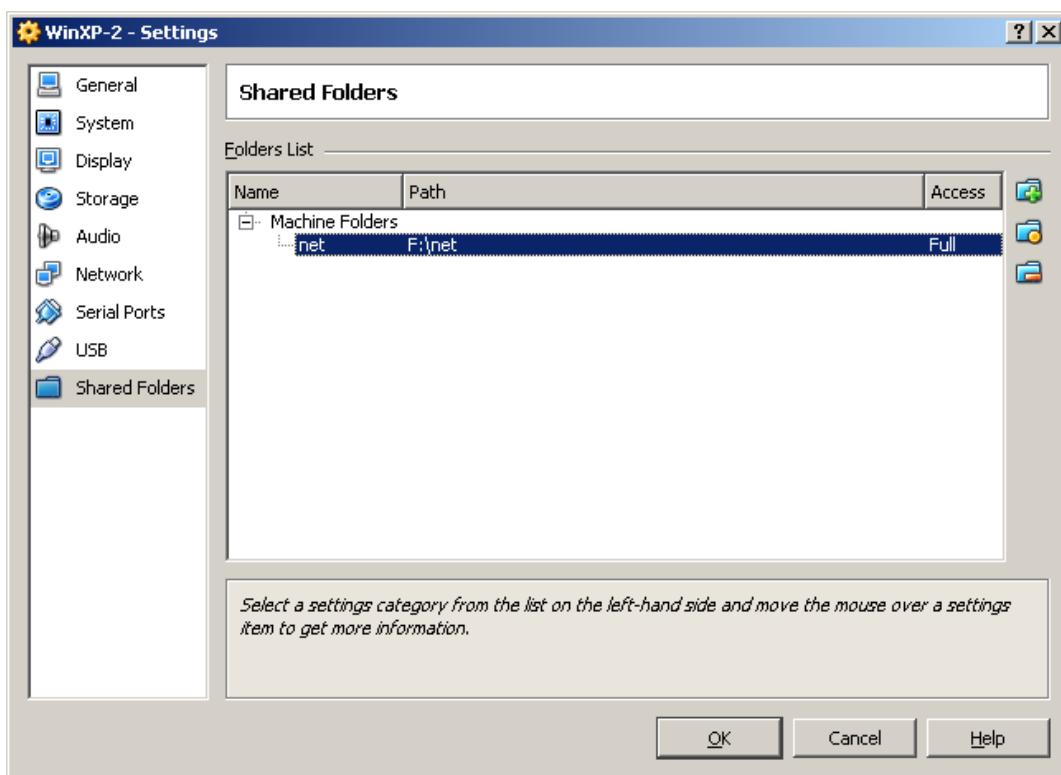


Рис. 2.23. Настройка общих папок

На этом настройка аппаратной части виртуальной машины может считаться законченной, и можно переходить к установке операционной системы.

2.2.2. Настройка операционной системы виртуальной машины

Описание установки операционной системы рассматриваться не будет, так как данная процедура является стандартной. Запуск установки операционной системы осуществляется из главного окна VirtualBox путем выбора соответствующей виртуальной машины и нажатия кнопки «*Старт / Start*» (рис. 2.24).

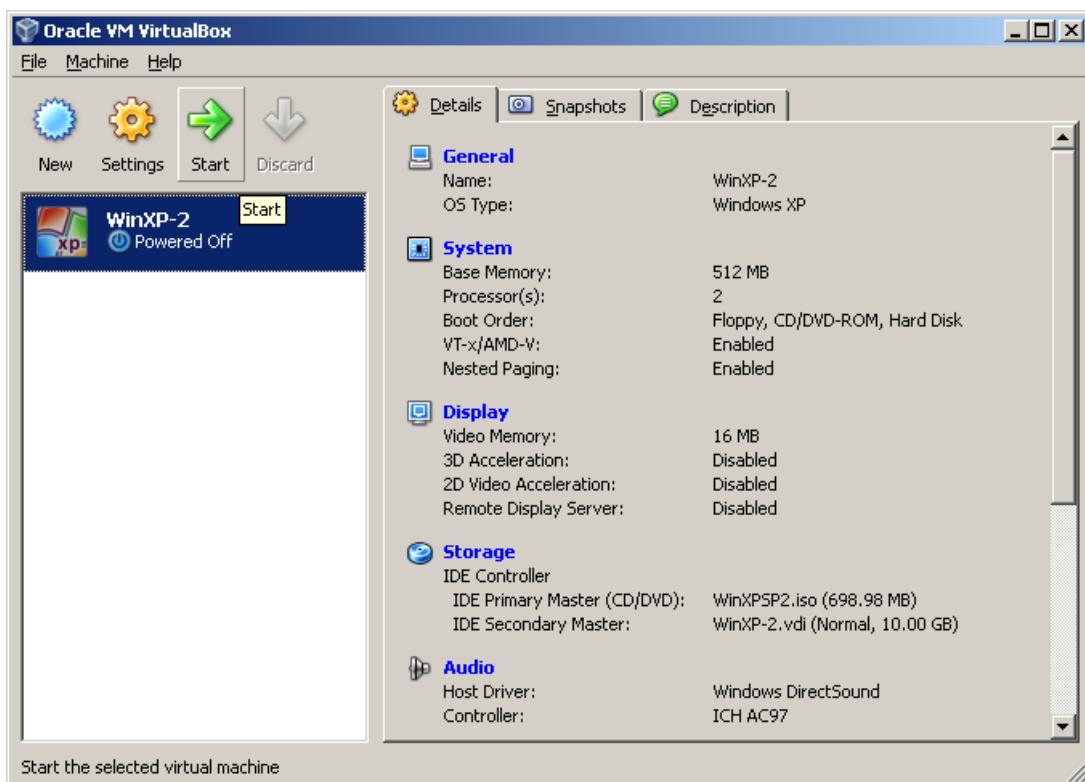


Рис. 2.24. Запуск установки операционной системы

После проведения действий, описанных выше, появится окно с установкой операционной системы. Это означает, что все настройки выполнены правильно, и остается установить и настроить операционную систему. После того, как система будет установлена и загружена (рис. 2.25), можно приступать к настройке операционной системы виртуальной машины.

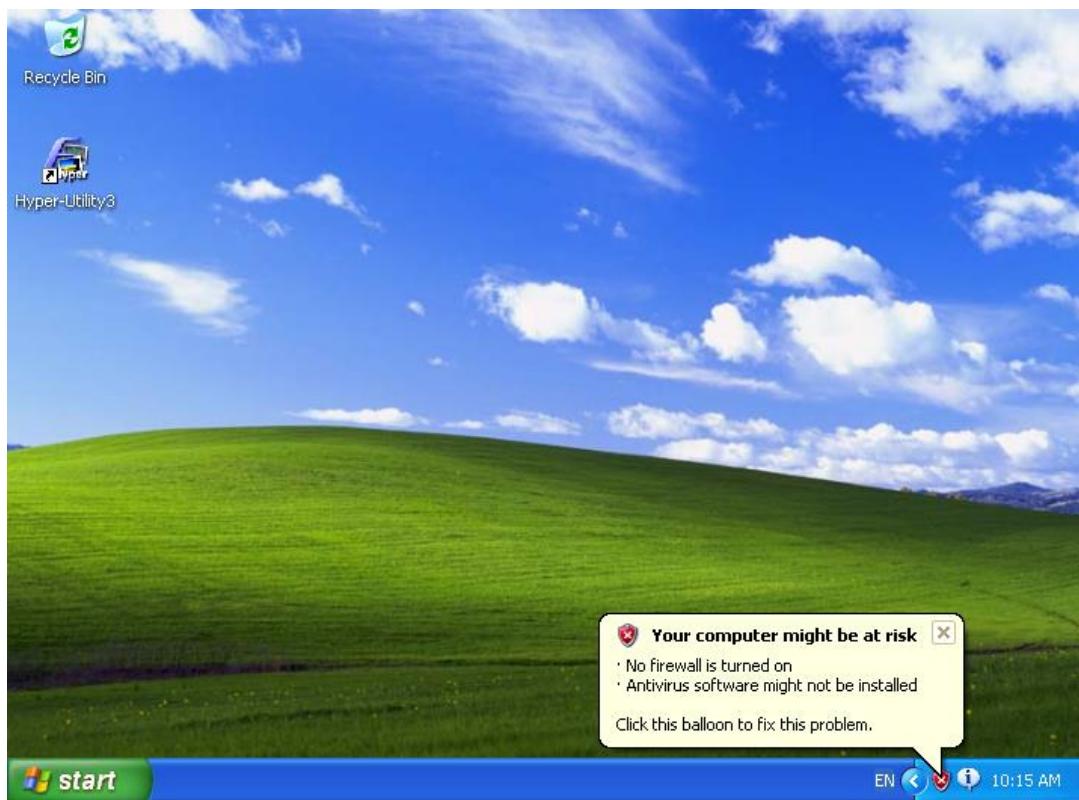


Рис. 2.25. Настройка операционной системы

Для начала необходимо установить драйверы для всех виртуальных аппаратных компонентов виртуального ПК. Для этого в главном меню (рис. 2.26) нужно выбрать пункт «Устройства» – «Приводы оптических дисков» – «VboxGuestAdditions.iso». Впоследствии таким же образом можно подключить к виртуальной машине физический CD-ROM или загрузить ISO-образ.

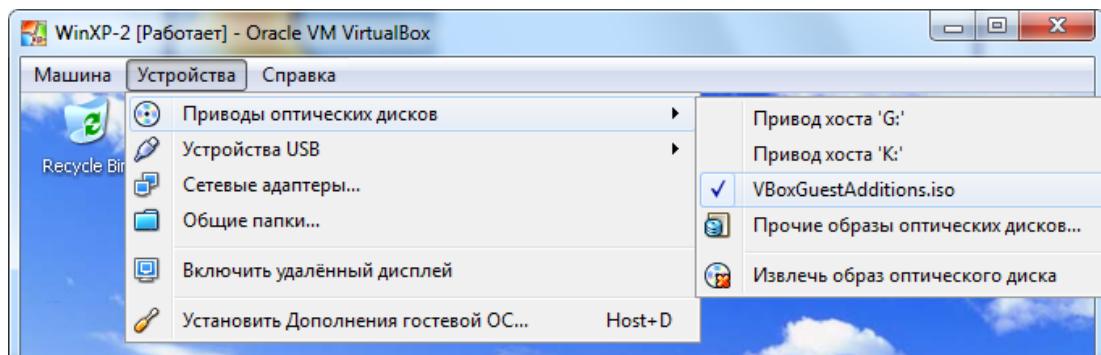


Рис. 2.26. Загрузка VboxGuestAdditions.iso при настройке операционной системы

После подключения образа *VboxGuestAdditions.iso* в папке «Мой компьютер» в привод компакт-дисков загрузится данный виртуальный диск – далее необходимо запустить двойным щелчком левой кнопки мыши (рис. 2.27).

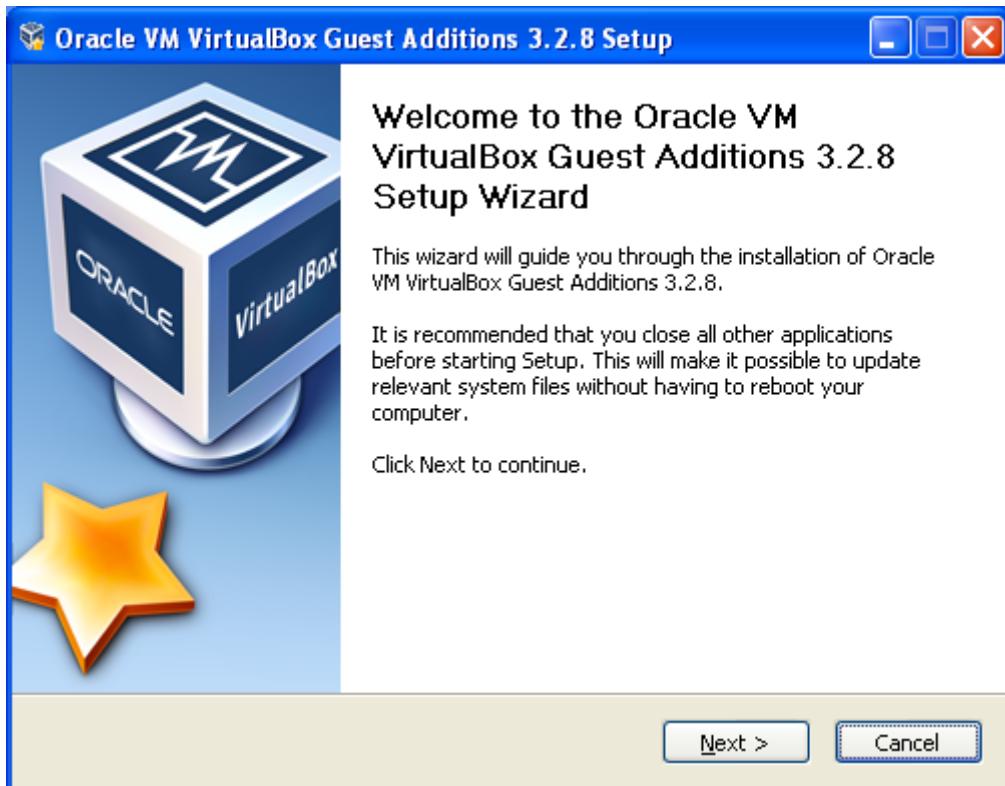


Рис. 2.27. Установка *VboxGuestAdditions.iso* при настройке операционной системы

Сам процесс установки происходит практически без участия пользователя и только в случае, если ранее было включено 3D-ускорение, то следует выбрать соответствующий компонент (рис. 2.28) для дополнительной установки.

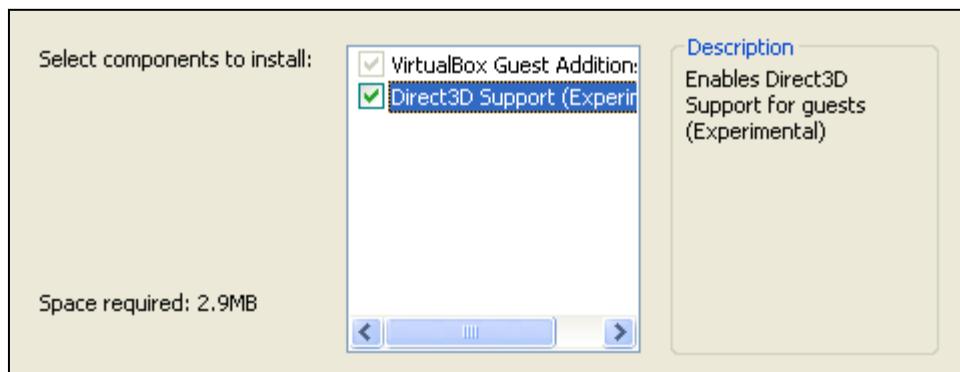


Рис. 2.28. Выбор дополнений при настройке операционной системы

Далее подключим общие папки, чтобы получить возможность переносить в созданную виртуальную машину нужные для работы файлы и устанавливать приложения. Это можно сделать с помощью командной строки, следуя справке Virtua lBox, но лучше использовать следующий способ: необходимо открыть папку «*Мой компьютер*», в главном меню выбрать «*Сервис*» – «*Подключить сетевой диск*» и в открывшемся окне в поле «*Папка*» ввести `\vboxsrv\имя_общей_папки`, например как показано ниже.

```
\vboxsrv\WinXP-2-Share
```

После этих действий в папке «*Мой компьютер*» появится общая папка, доступная в качестве сетевого диска.

2.3. Организация одноранговой сети

Каждый компьютер в сетях TCP/IP имеет адреса трех уровней: физический (MAC-адрес), сетевой (IP-адрес) и символьный (DNS-имя).

Ключевую роль в организации любой компьютерной сети играет сетевой адрес (IP-адрес), который представляет собой 32-разрядное двоичное число, разделенное на группы по 8 бит, называемые *октетами*. Например, 00010001 11101111 00101111 01011110.

Обычно IP-адреса записываются в виде четырех десятичных октетов и разделяются точками. Таким образом, приведенный выше IP-адрес можно записать в следующей форме: 17.239.47.94.

Следует заметить, что максимальное значение октета равно 11111111_2 (двоичная система счисления), что соответствует в десятичной системе 255_{10} . Поэтому IP-адреса, в которых хотя бы один октет превышает это число, являются недействительными. Пример: 172.16.123.1 – действительный адрес, а 172.16.123.256 – несуществующий адрес, поскольку 256 выходит за пределы допустимого диапазона: от 0 до 255.

IP-адрес состоит из двух логических частей – *номера подсети* (ID подсети) и *номера узла* (ID хоста) в этой подсети. При передаче пакета из одной подсети в другую используется ID подсети. Когда пакет попал в подсеть назначения, ID хоста указывает на конкретный узел в рамках этой подсети.

Чтобы записать ID подсети, в поле номера узла в IP-адресе ставят нули. Чтобы записать ID хоста, в поле номера подсети ставят нули. Например, если в IP-адресе 172.16.123.1 первые два байта отводятся под номер подсети, остальные два байта – под номер узла, то номера записываются следующим образом: ID подсети 172.16.0.0; ID хоста 0.0.123.1.

По числу разрядов, отводимых для представления номера узла (или номера подсети), можно определить общее количество узлов (или подсетей) по простому правилу: если число разрядов для представления номера узла равно N , то общее количество узлов равно $2^N - 2$. Два узла вычтены вследствие того, что адреса со всеми разрядами, равными нулям или единицам, являются особыми и используются в специальных целях.

Например, если под номер узла в некоторой подсети отводится два байта (16 бит), то общее количество узлов в такой подсети равно $2^{16} - 2 = 65534$ узла.

Общее правило: под ID подсети отводятся *первые* несколько бит IP-адреса, оставшиеся биты обозначают ID хоста.

Служба распределения номеров IANA (Internet Assigned Numbers Authority) зарезервировала для частных (локальных) сетей три блока адресов:

- 10.0.0.0 – 10.255.255.255 (префикс 10/8);
- 172.16.0.0 – 172.31.255.255 (префикс 172.16/12);
- 192.168.0.0 – 192.168.255.255 (префикс 192.168/16).

Будем называть первый блок 24-битовым, второй – 20-битовым, а третий – 16-битовым. Отметим, что первый блок представляет собой не что иное, как одну сеть класса *A*, второй блок – 16 последовательных сетей класса *B*, а третий блок – 256 последовательных сетей класса *C*.

Любая организация может использовать IP-адреса из этих блоков без согласования с IANA или Internet-регистраторами. В результате эти адреса используются во множестве организаций. Таким образом, уникальность адресов сохраняется только в масштабе одной или нескольких организаций, согласованно использующих общий блок адресов. В такой сети каждая рабочая станция может обмениваться информацией с любой другой рабочей станцией частной сети.

Если организации требуются уникальные адреса для связи с внешними сетями, такие адреса следует получать обычным путем через регистраторов Internet. Такие адреса никогда не будут входить ни в один из указанных выше блоков частных адресов.

Рассмотрим конфигурирование IP-адресации (v4) в операционных системах типа *Windows*.

Пример 1. Рассмотрим настройку протокола TCP/IP v4.

1. Запустите папку «Сетевые подключения». Для этого в операционных системах типа Windows Seven необходимо нажать кнопку «Пуск», ввести в строке поиска начальные буквы слова «Центр». Из списка выберите пункт «Центр управления сетями и общим доступом» (рис. 2.29).

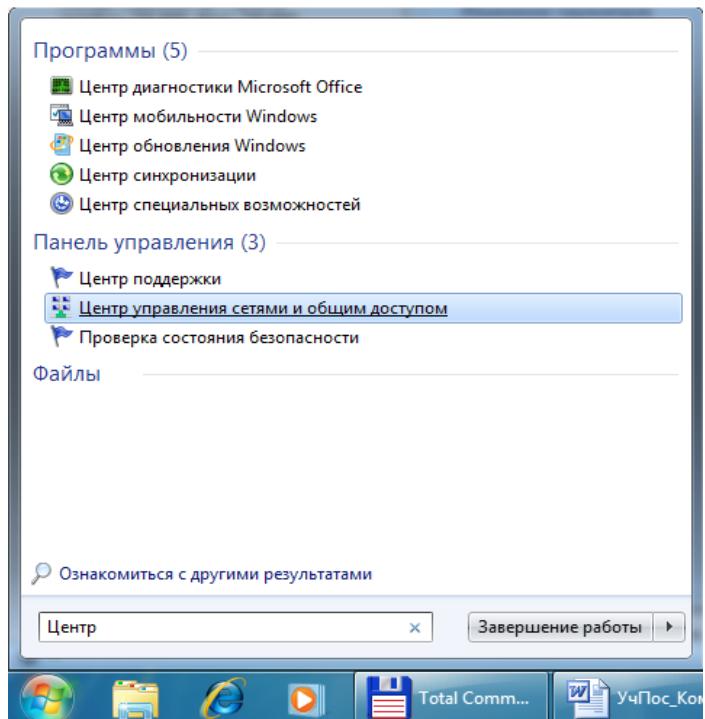


Рис. 2.29. Пример вызова Центра управления сетями и общим доступом

2. В окне Центра управления сетями и общим доступом щелкните по изменению параметров адаптера (рис. 2.30). Далее откроется окно с сетевыми подключениями (рис. 2.31).

3. Щелкните правой кнопкой мыши по подключению, которое требуется настроить, а затем выберите команду «Свойства». Если появится диалоговое окно «Управление учетной записью пользователя», убедитесь, что действие, указанное в окне, совпадает с тем, которое вы хотите выполнить, и нажмите «Продолжить».

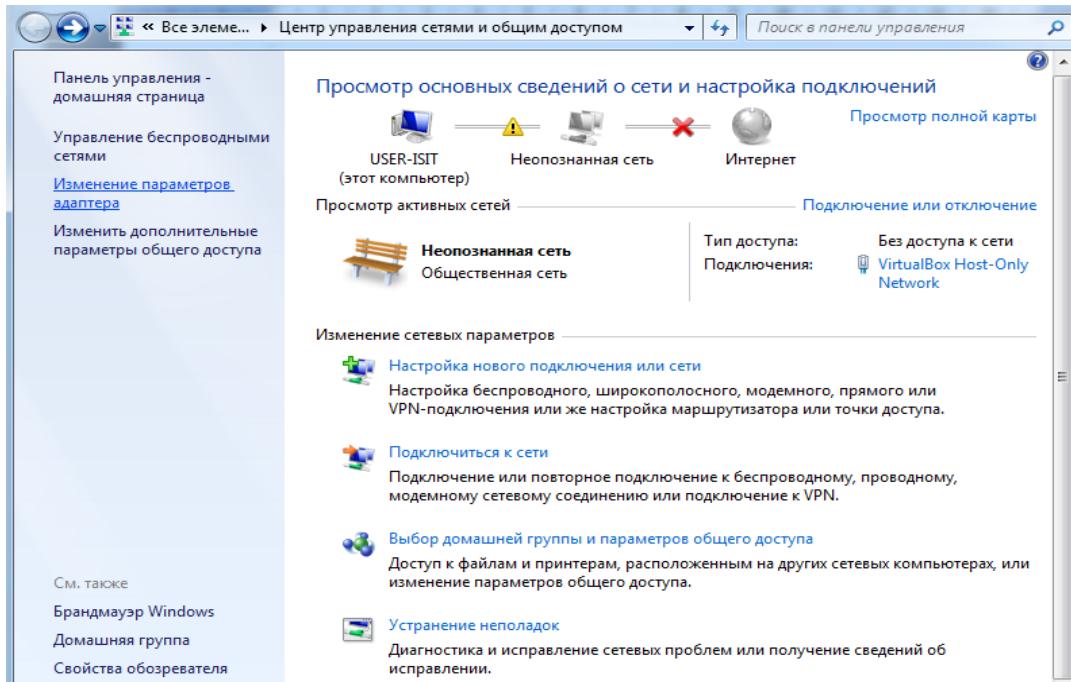


Рис. 2.30. Общий вид центра управления сетями и общим доступом

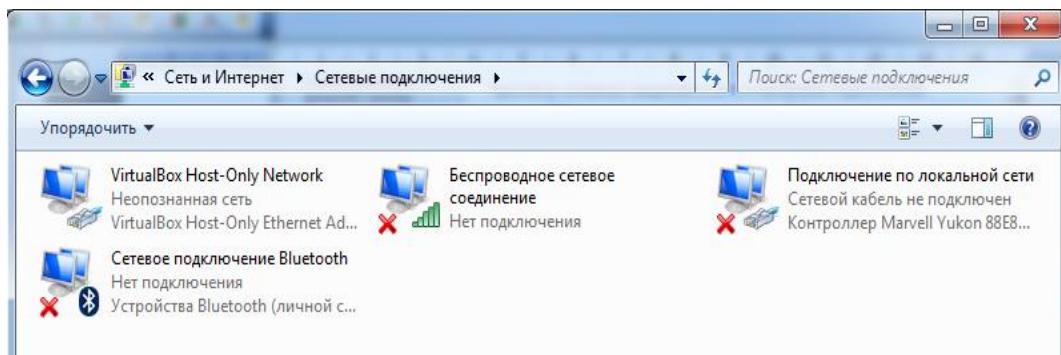


Рис. 2.31. Общий вид папки «Сетевые подключения»

4. Далее выполните одно из указанных ниже действий:

- в случае подключения по локальной сети на вкладке «Общие» в списке «Компоненты», используемые этим подключением, выберите пункт «Протокол Интернета версии 4» (TCP/IPv4) и нажмите кнопку «Свойства»;
- в случае подключения удаленного доступа, VPN-подключения или высокоскоростного подключения на вкладке «Сеть» в списке «Компоненты», используемые этим подключением, выберите пункт «Протокол Интернета версии 4» (TCP/IPv4) и нажмите кнопку «Свойства». В результате откроется окно с настройками протокола TCP/IP (рис. 2.32).

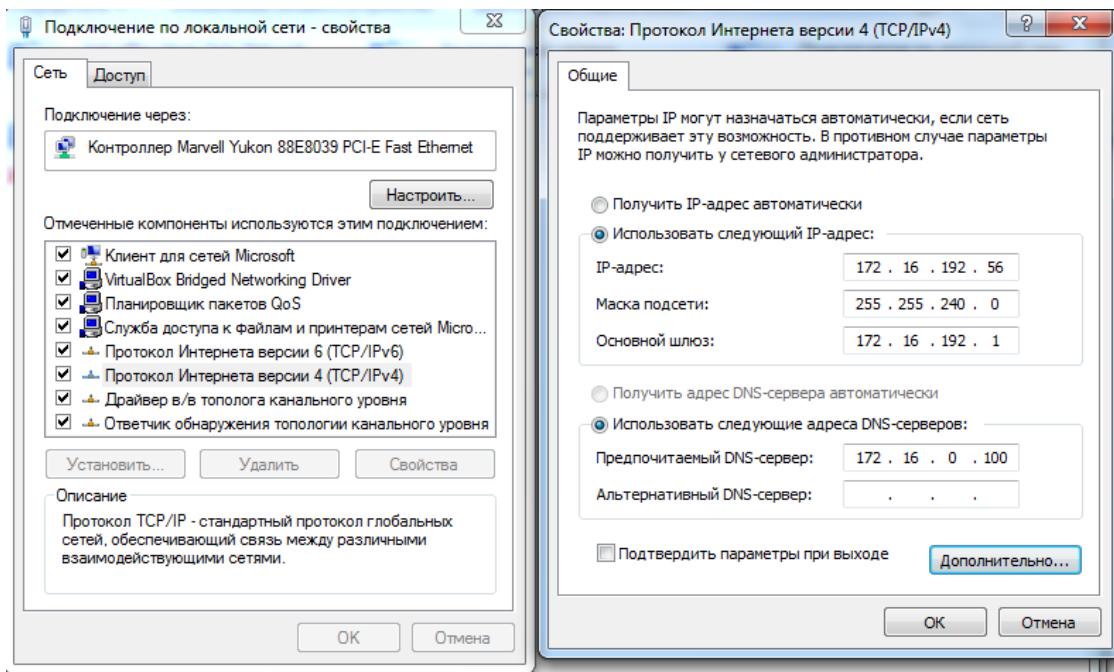


Рис. 2.32. Свойства Протокола Интернета версии 4 (TCP/IPv4)

5. Выполните далее одно из указанных ниже действий:

- если необходимо, чтобы параметры IP-адреса назначались автоматически, выберите пункт «*Получить IP-адрес автоматически*» и нажмите кнопку *OK*;
- если необходимо указать IP-адрес IPv4 или адрес DNS-сервера, выполните следующие действия:
 - a) выберите пункт «*Использовать следующий IP-адрес*» и в поле «*IP-адрес*» введите IP-адрес, соответствующую маску подсети и адрес шлюза по умолчанию (в примере на рис. 2.33 IP адрес: 172.16.192.56; маска подсети: 255.255.240.0; основной шлюз: 172.16.192.1);
 - b) выберите пункт «*Использовать следующие адреса DNS-серверов*» и в полях «*Предпочитаемый DNS-сервер*» и «*Альтернативный DNS-сервер*» введите адреса основного и дополнительного DNS-сервера (в примере на рис. 2.32 IP-адрес предпочтаемого DNS сервера: 172.16.0.100);
 - c) для настройки параметров DNS, WINS и IP нажмите кнопку «*Дополнительно*» (рис. 2.33).

6. В подключении по локальной сети при выборе параметра «*Получить IP-адрес автоматически*» включается вкладка «*Альтернативная конфигурация*». Если компьютер используется более чем в одной сети, используйте эту вкладку для ввода альтернативных параметров IP-адреса. Для настройки параметров DNS, WINS и IP открой-

те вкладку «Настраиваемый пользователем» или «Альтернативная конфигурация».

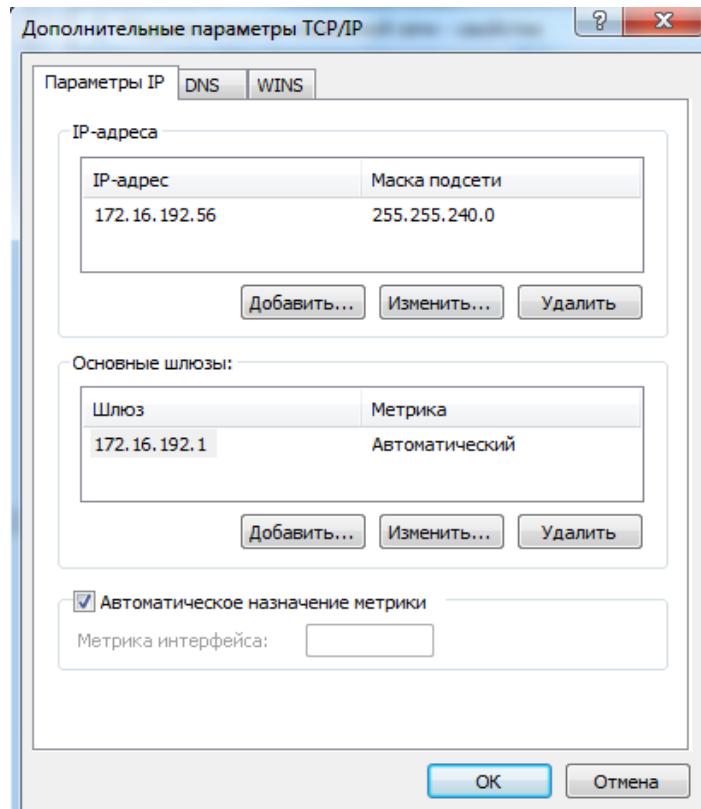


Рис. 2.33. Окно с дополнительными параметрами TCP/IP

Дополнительные рекомендации. Если это возможно, используйте автоматическую настройку параметров протокола IP (DHCP), поскольку при этом устраняется необходимость настройки таких параметров, как IP-адрес, адрес DNS-сервера и адрес WINS-сервера.

Параметры «Альтернативная конфигурация» определяют второй набор параметров протокола IP, который используется при недоступности DHCP-сервера. Это весьма полезно для пользователей портативных компьютеров, которые часто перемещаются между двумя различными сетевыми средами (например, между средой со службой DHCP и средой со статическими IP-адресами).

Отметим, что аналогично осуществляется конфигурирование TCP/IPv6 (используются свойства «Протокол Интернета версии 6» (TCP/IPv6)).

С помощью масок администратор может структурировать свою сеть, не требуя от поставщика услуг дополнительных номеров сетей.

2.4. Распределение ресурсов по сети

Распределение ресурсов в сетях TCP/IP при поддержке операционных систем Windows осуществляется в два этапа:

1. На компьютере, к которому будут клиенты получать в дальнейшем доступ к разделяемым ресурсам, создаются папки либо структура папок каким-либо стандартным способом (используемая файловая система – NTFS). Будем их называть далее сетевыми папками или сетевыми ресурсами. Данные ресурсы открываются в сеть вызовом контекстного меню и выполнением из него команды «*Общий доступ и безопасность*» (рис. 2.34).

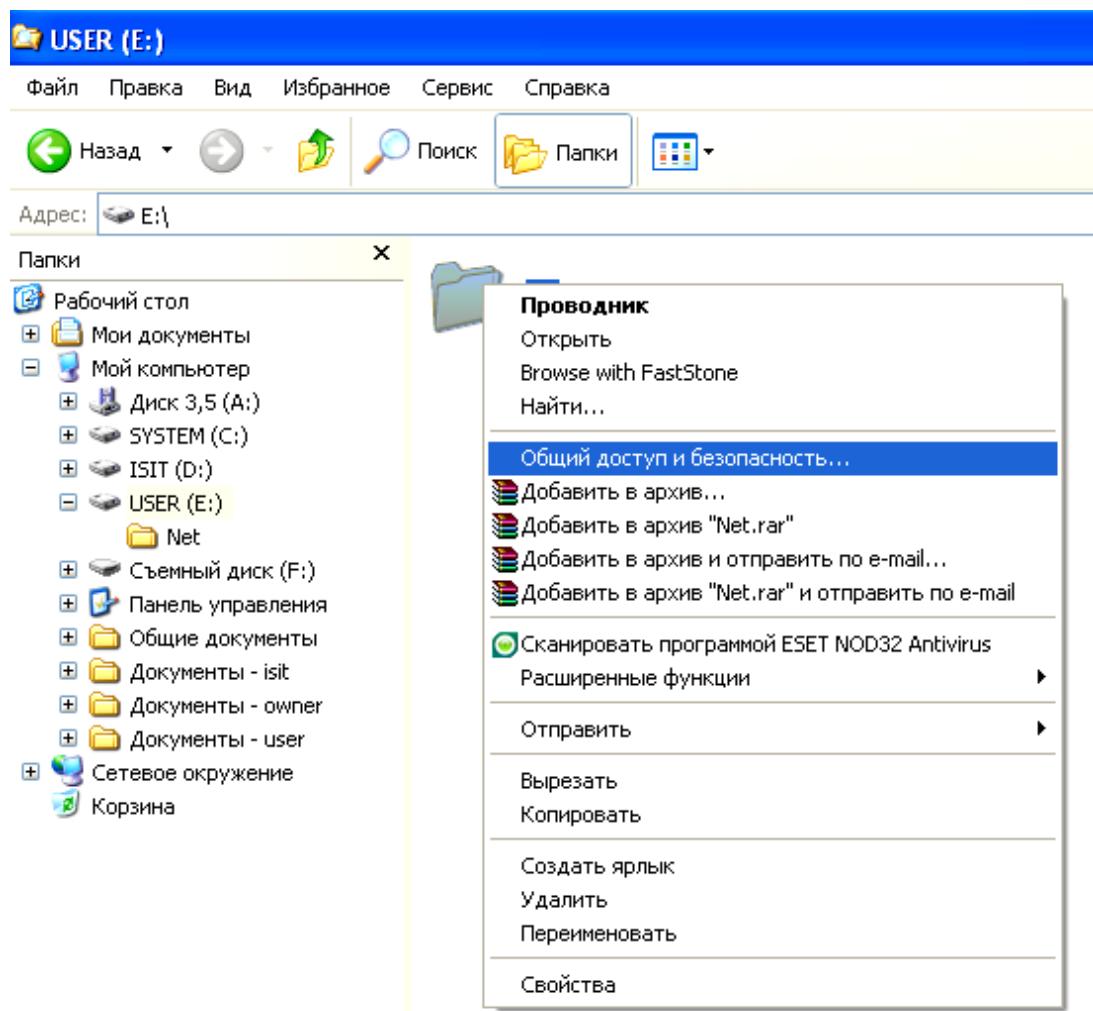


Рис. 2.34. Открытие доступа к папке по сети

В результате появится окно следующего вида (рис. 2.35), в котором необходимо выбрать «*Открыть общий доступ к этой папке*».

Целесообразно также установить предельное число пользователей, например, как показано на рис. 2.35.

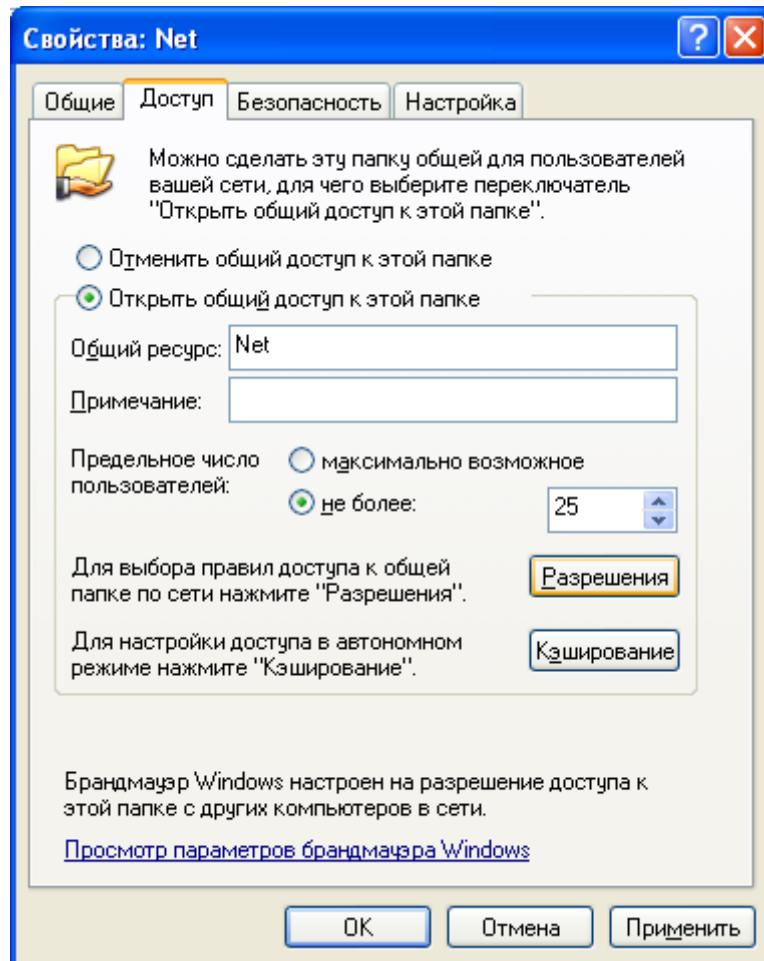
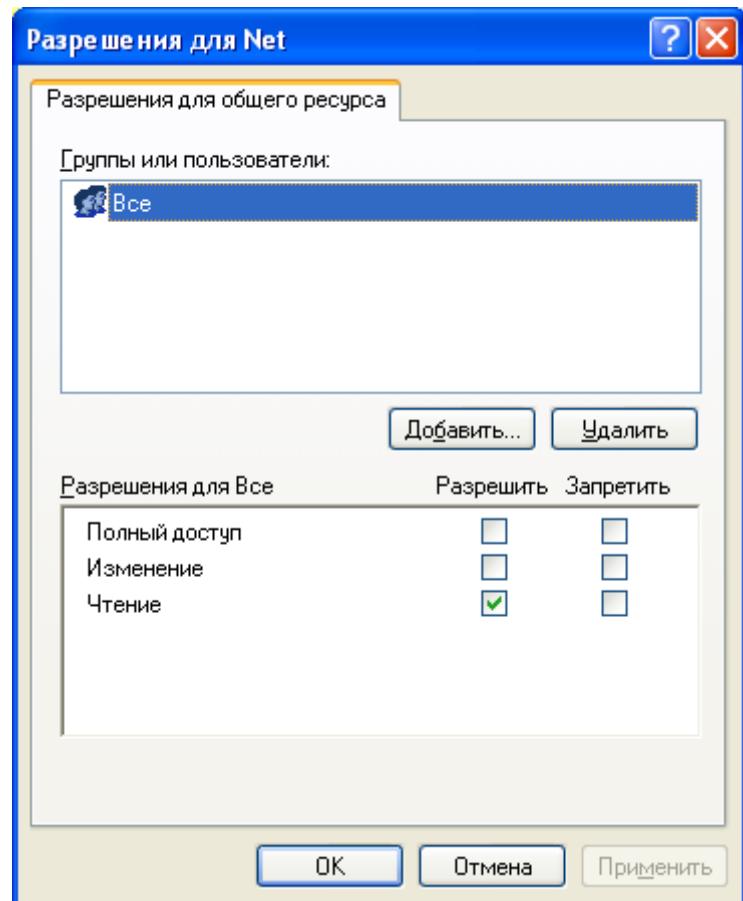


Рис. 2.35. Окно настройки общего доступа и безопасности

Далее для выбора и настройки правил доступа к папке по сети необходимо щелкнуть по кнопке «Разрешения». Откроется стандартное окно, показанное на рис. 2.36, из которого видно, что папка будет открыта по сети для всех пользователей с правами «Только чтение». Если необходимо просто предоставить всем пользователям сети просмотр информации, хранящейся в сетевой папке, то данный стандартный вариант можно считать приемлемым.



Однако чаще всего требуется более тонкая настройка правил общего доступа. Для этого необходимо удалить группу пользователей «*Все*», а добавить отдельных пользователей. Так, согласно рис. 2.37, для пользователя *user* предоставлен доступ типа «Чтение», а для пользователя *user1* – «Полный доступ» (рис. 2.38).

Во всех операционных системах Windows существует важное правило: *запрещающие правила всегда имеют более высокий приоритет, чем разрешающие правила*. Из чего следует важный момент: если на этапе открытия доступа к папке будут использованы запрещающие правила, то их действие на всех последующих этапах настройки сетевого ресурса отменить будет невозможно.

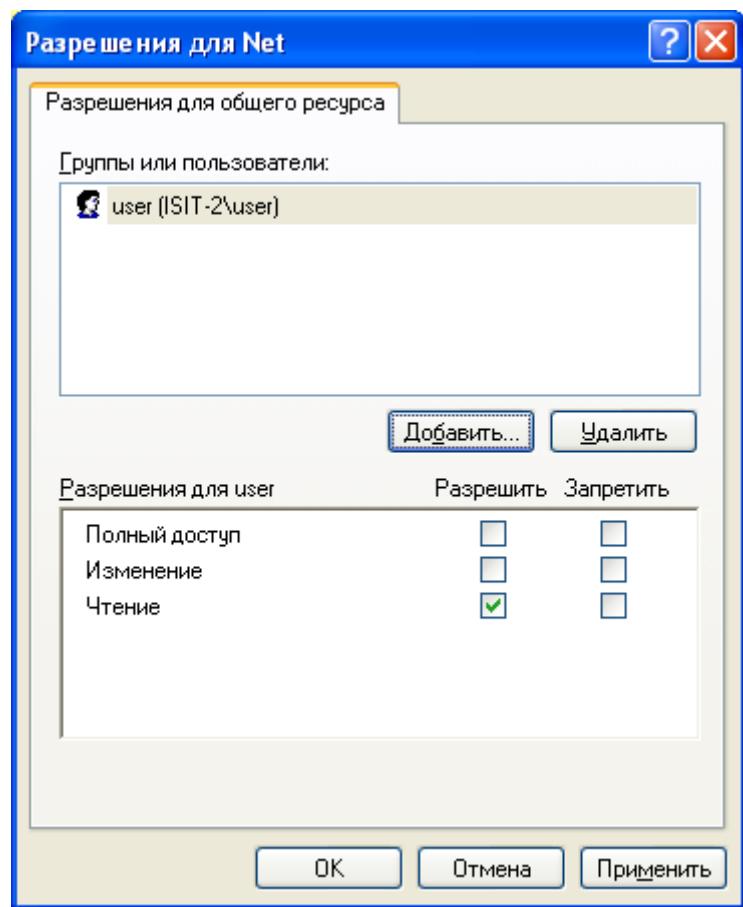


Рис. 2.37. Предоставление ограниченного доступа для пользователя *user*

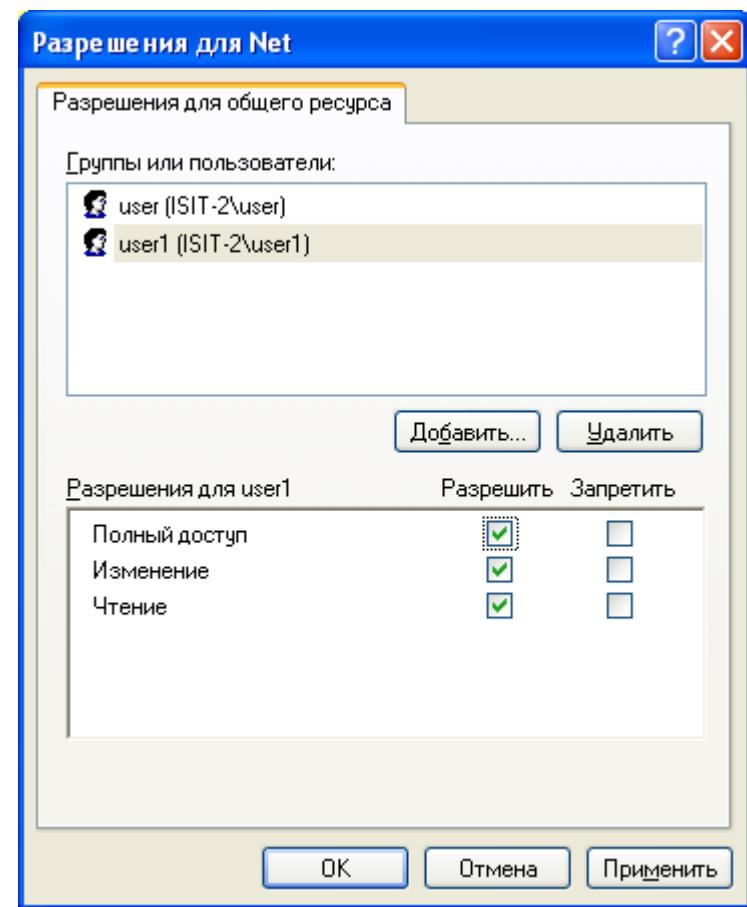


Рис. 2.38. Предоставление полного доступа для пользователя *user1*

2. Далее перейдем к настройке вкладки «Безопасность», общий вид которой представлен на рис. 2.39.

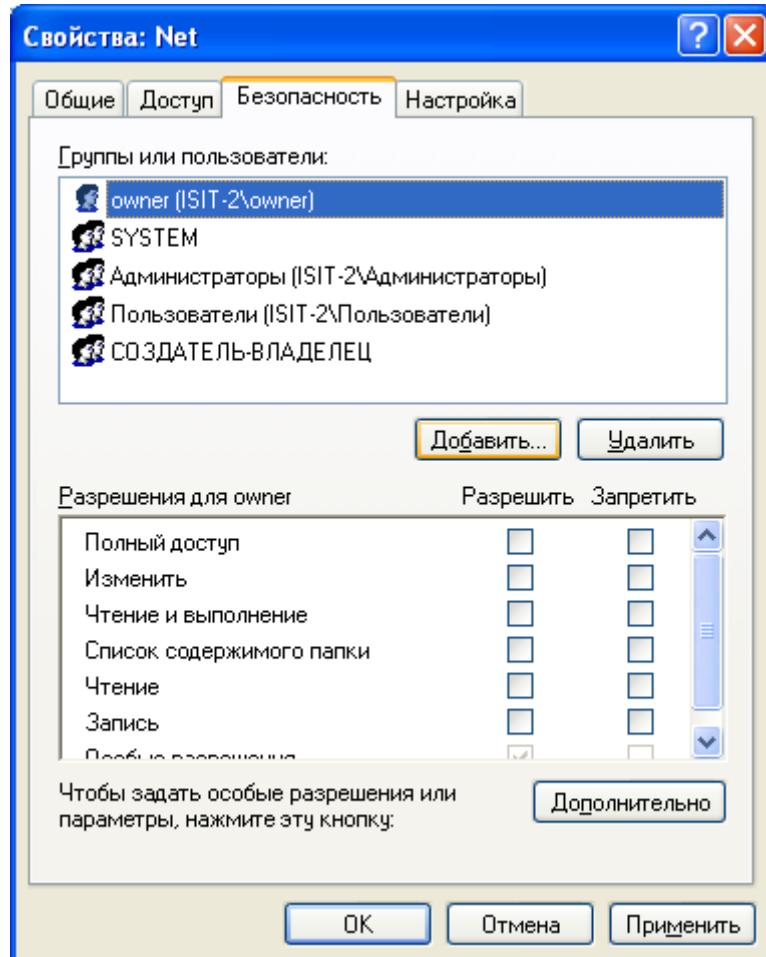


Рис. 2.39. Вид окна для настройки безопасности сетевой папки

Для тонкой настройки безопасности сетевой папки рекомендуется удалить из перечня «групп или пользователей» группу пользователей «Пользователи». Для этого необходимо сначала отменить в дополнительных параметрах наследование от родительского объекта применимых к дочерним объектам разрешений (рис. 2.40), а лишь затем выполнить операцию удаления группы «Пользователи».

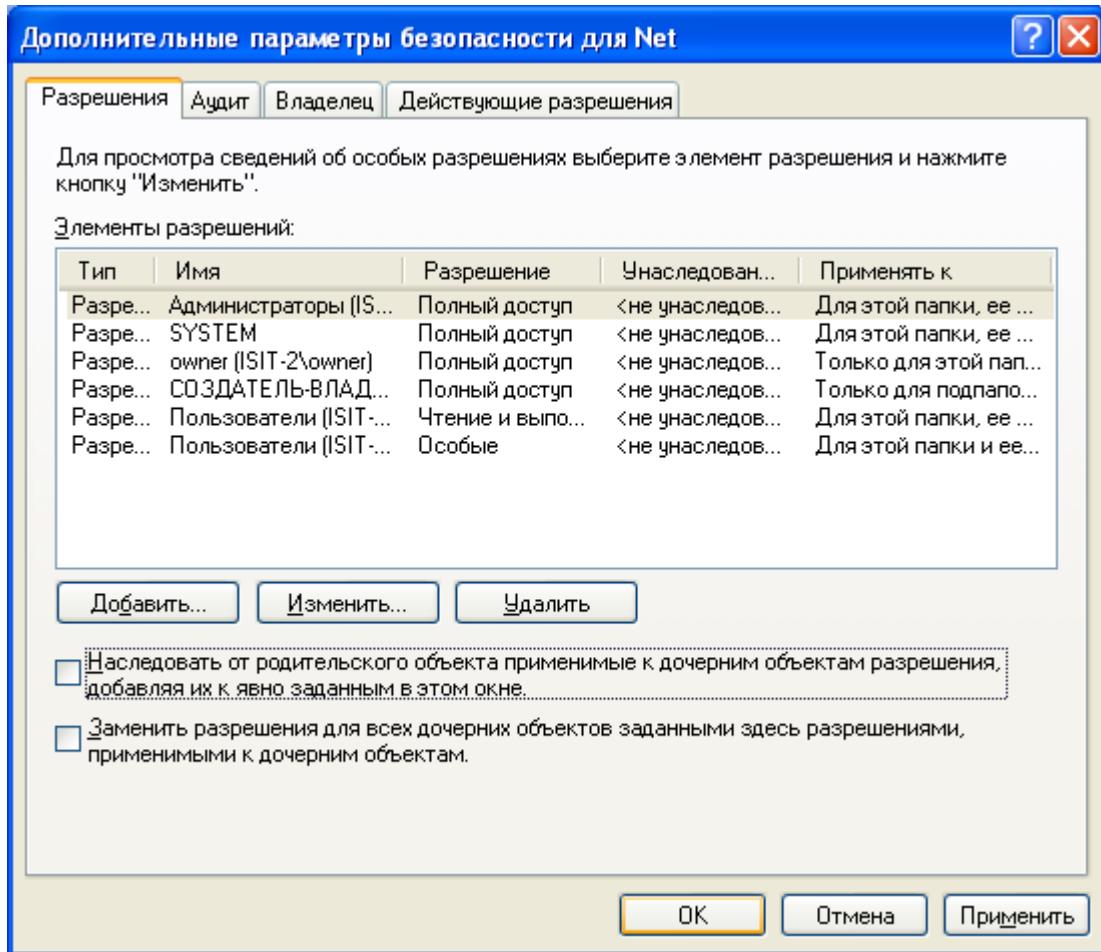


Рис. 2.40. Вид окна для настройки дополнительных параметров безопасности сетевой папки

Далее можно приступить к добавлению отдельных пользователей (согласно рис. 2.41, добавлен пользователь *user1*). Детальную настройку разрешений и запретов для определенного пользователя можно выполнить в окне дополнительных параметров безопасности сетевой папки, щелкнув по кнопке «Изменить» (рис. 2.42). Отметим, что настройка может производиться на нескольких уровнях – возможна настройка разрешений и запретов «для этой папки, ее подпапок и файлов» (приведено в примере), «только для этой папки» и т. д. (рис. 2.43).

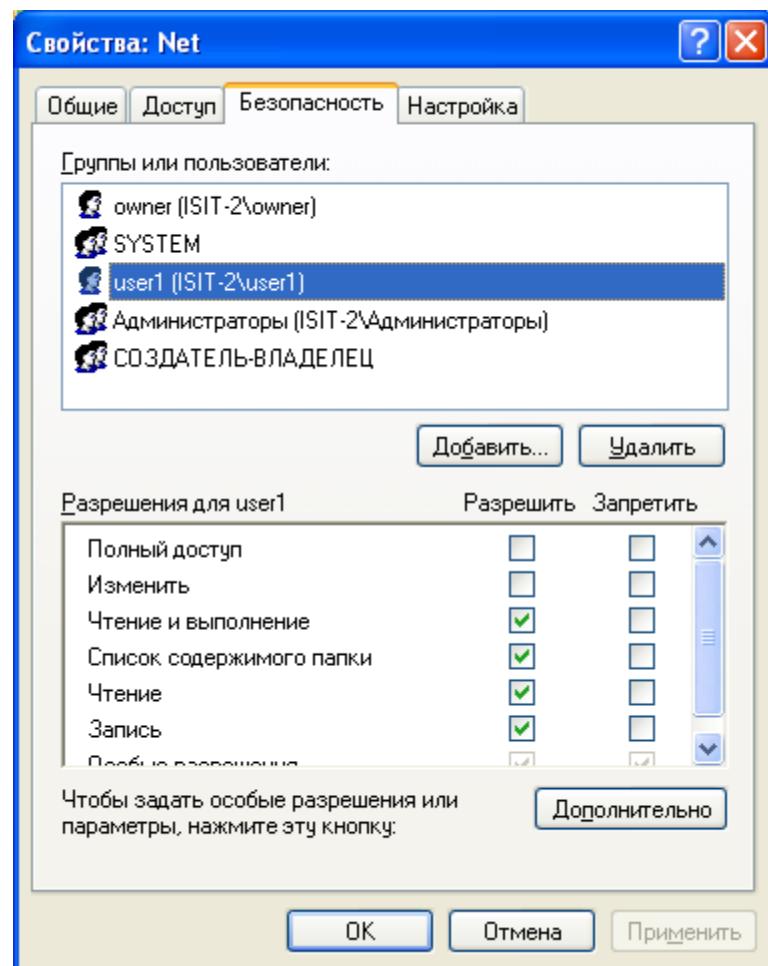


Рис. 2.41. Вид окна для настройки безопасности сетевой папки с добавленным пользователем *user1*

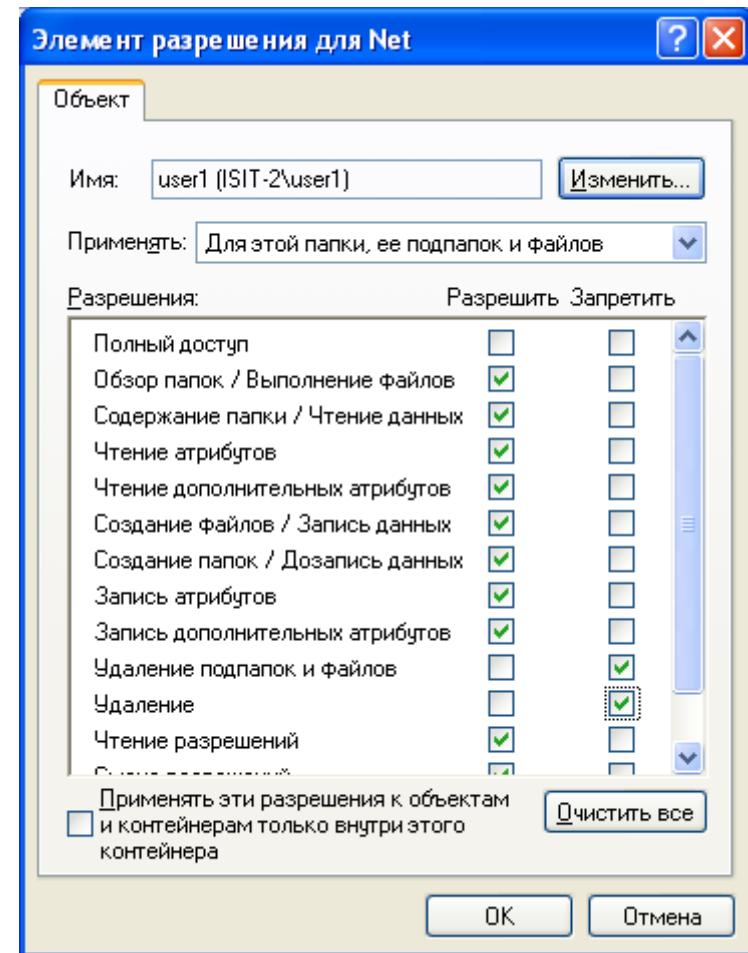


Рис. 2.42. Вид окна для тонкой настройки безопасности сетевой папки для пользователя *user1*

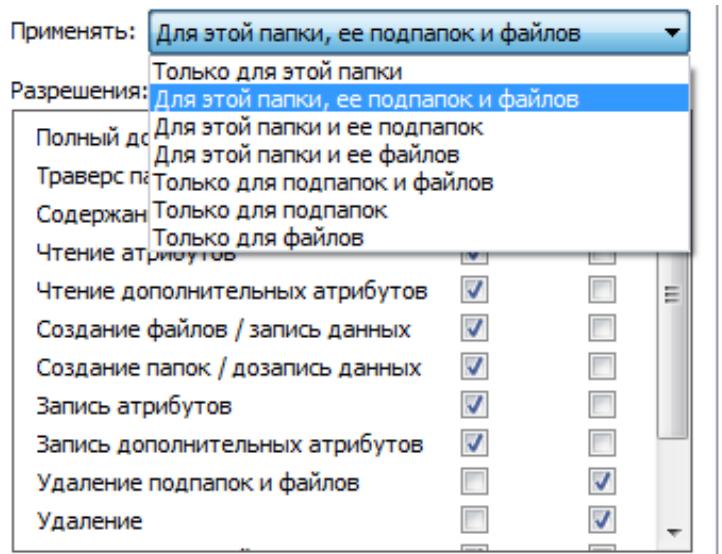


Рис. 2.43. Вид окна для тонкой настройки безопасности сетевой папки (уровни действия настроек)

Просмотреть открытые ресурсы для доступа по сети, а также подключенных к ним пользователей можно, открыв «Управление компьютером» (рис. 2.44) (вызывается при помощи контекстного меню для объекта «Мой компьютер» с дальнейшим выполнением команды «Управление» или через «Панель управления»).

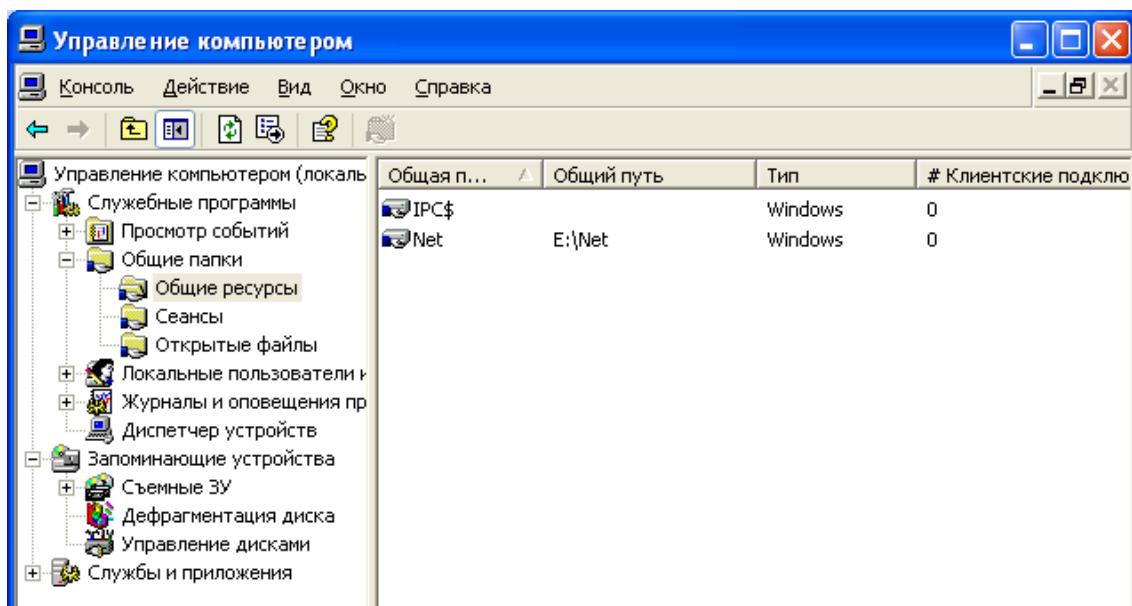


Рис. 2.44. Вид окна консоли «Управление компьютером»

2.5. Лабораторная работа № 2–3

Цель: изучение методов организации одноранговой сети на базе операционных систем Windows.

Задание: организация одноранговой сети со статической адресацией между двумя хостами с разделением ресурсов между пользователями. Лабораторная работа может выполняться в парах (на каждой стороне студент настраивает один хост и связь организуется через реальную компьютерную сеть), либо индивидуально (на одном компьютере настраиваются два хоста с организацией между ними виртуальной сети). В качестве хостов должны выступать виртуальные операционные системы типа Windows (XP, Vista, Seven и т. д.) со статически заданными сетевыми адресами согласно варианту (таблица). Адрес основного компьютера выбрать самостоятельно из диапазона 172.16.193.233/20 – 172.16.193.250/20.

Варианты заданий для лабораторной работы № 2–3

| № варианта | Имя хоста | IP-адрес хоста | Пользователи |
|------------|-----------|----------------|----------------|
| 1 | Host1_1 | 172.16.193.201 | User2 User3 |
| | Host1_2 | 172.16.193.202 | User1 User4 |
| 2 | Host2_1 | 172.16.193.203 | User4 User5 |
| | Host2_2 | 172.16.193.204 | User1 User2 |
| 3 | Host3_1 | 172.16.193.205 | User2 User4 |
| | Host3_2 | 172.16.193.206 | User3 User5 |
| 4 | Host4_1 | 172.16.193.207 | User2 User5 |
| | Host4_2 | 172.16.193.208 | User1 User3 |
| 5 | Host5_1 | 172.16.193.211 | User1 User5 |
| | Host5_2 | 172.16.193.212 | User3 User4 |
| 6 | Host6_1 | 172.16.193.213 | User2 User3 |
| | Host6_2 | 172.16.193.214 | User1 User4 |

Окончание таблицы

| № варианта | Имя хоста | IP-адрес хоста | Пользователи |
|------------|-----------|----------------|----------------|
| 7 | Host7_1 | 172.16.193.215 | User4 User5 |
| | Host7_2 | 172.16.193.216 | User1 User2 |
| 8 | Host8_1 | 172.16.193.217 | User2 User4 |
| | Host8_2 | 172.16.193.218 | User3 User5 |
| 9 | Host9_1 | 172.16.193.219 | User2 User5 |
| | Host9_2 | 172.16.193.220 | User1 User3 |
| 10 | Host10_1 | 172.16.193.221 | User1 User5 |
| | Host10_2 | 172.16.193.222 | User3 User4 |
| 11 | Host11_1 | 172.16.193.223 | User2 User3 |
| | Host11_2 | 172.16.193.224 | User1 User4 |
| 12 | Host12_1 | 172.16.193.225 | User4 User5 |
| | Host12_2 | 172.16.193.226 | User1 User2 |
| 13 | Host13_1 | 172.16.193.227 | User2 User4 |
| | Host13_2 | 172.16.193.228 | User3 User5 |
| 14 | Host14_1 | 172.16.193.229 | User2 User5 |
| | Host14_2 | 172.16.193.230 | User1 User3 |
| 15 | Host15_1 | 172.16.193.231 | User1 User5 |
| | Host15_2 | 172.16.193.232 | User3 User4 |

Схема соединения компьютеров в сети представлена на рис. 2.45.

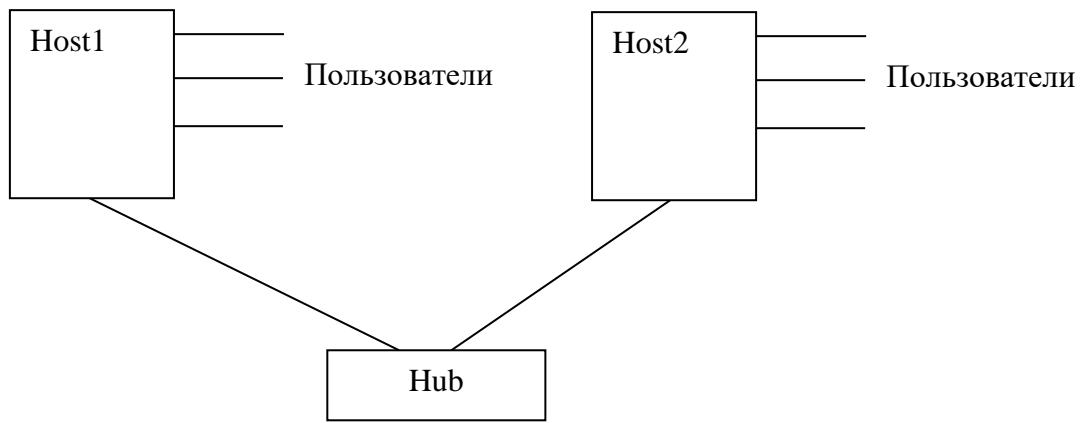


Рис. 2.45. Схема соединения компьютеров в сети

Для каждого пользователя (в соответствии с таблицей) на хосте создаются две папки: одна – для «полного доступа», а вторая – с правами «только чтение». При этом папки, предназначенные для одного пользователя, не должны быть доступны другим пользователям.

3. ДИНАМИЧЕСКАЯ IP-АДРЕСАЦИЯ В СЕТЯХ С КЛИЕНТ-СЕРВЕРНОЙ АРХИТЕКТУРОЙ

3.1. Понятие архитектуры клиент – сервер

Архитектура клиент – сервер (client – server architecture) – это концепция информационной сети, в которой основная часть ее ресурсов сосредоточена в серверах, обслуживающих своих клиентов (рис. 3.1). Рассматриваемая архитектура определяет два типа компонентов: серверы и клиенты.

Сервер – это объект, предоставляющий сервис другим объектам сети по их запросам. *Сервис* – это процесс обслуживания клиентов.

Сервер работает по заданиям клиентов и управляет выполнением их заданий. После выполнения каждого задания сервер посылает полученные результаты клиенту, пославшему это задание.

Сервисная функция в архитектуре клиент – сервер описывается комплексом прикладных программ, в соответствии с которым выполняются разнообразные прикладные процессы.

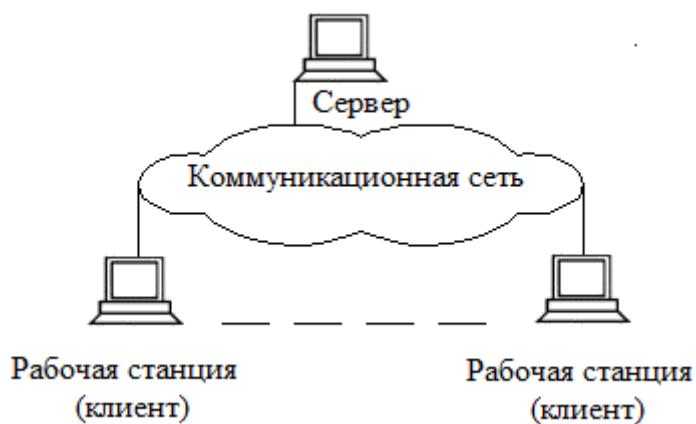


Рис. 3.1. Архитектура клиент – сервер

Процесс, который вызывает сервисную функцию с помощью определенных операций, называется клиентом. Им может быть программа или пользователь. На рис. 3.2 приведен перечень сервисов в архитектуре клиент – сервер.

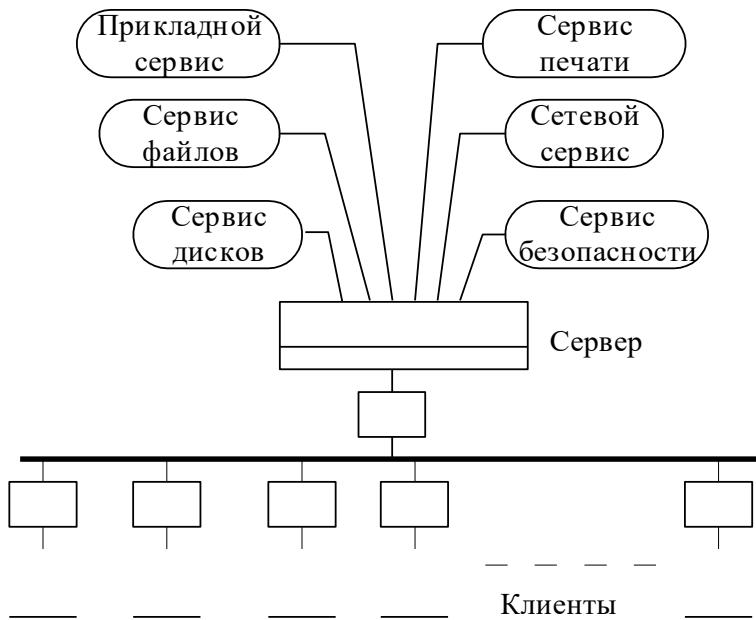


Рис. 3.2. Модель клиент – сервер

Клиенты – это рабочие станции, которые используют ресурсы сервера и предоставляют удобные интерфейсы пользователя. *Интерфейсы пользователя* – это процедуры взаимодействия пользователя с системой или сетью.

Клиент является инициатором и использует электронную почту или другие сервисы сервера. В этом процессе клиент запрашивает вид обслуживания, устанавливает сеанс, получает нужные ему результаты и сообщает об окончании работы.

В сетях с выделенным файловым сервером на выделенном автономном ПК (персональном компьютере) устанавливается серверная сетевая операционная система. Этот ПК становится сервером. Программное обеспечение (ПО), установленное на рабочей станции, позволяет ей обмениваться данными с сервером. Наиболее распространенные сетевые операционные системы:

- NetWare фирмы Novell;
- Windows фирмы Microsoft;
- UNIX фирмы AT&T;
- Linux.

Помимо сетевой операционной системы необходимы сетевые прикладные программы, реализующие преимущества, предоставляемые сетью.

Сети на базе серверов имеют лучшие характеристики и повышенную надежность. Сервер владеет главными ресурсами сети, к которым обращаются остальные рабочие станции.

В современной клиент-серверной архитектуре выделяется четыре группы объектов: клиенты, серверы, данные и сетевые службы. Клиенты располагаются в системах на рабочих местах пользователей. Данные в основном хранятся на серверах. Сетевые службы являются совместно используемыми серверами и данными. Кроме того службы управляют процедурами обработки данных.

Сети клиент-серверной архитектуры имеют следующие преимущества:

- позволяют организовывать сети с большим количеством рабочих станций;
- обеспечивают централизованное управление учетными записями пользователей, безопасностью и доступом, что упрощает сетевое администрирование;
- эффективный доступ к сетевым ресурсам;
- пользователю нужен один пароль для входа в сеть и для получения доступа ко всем ресурсам, на которые распространяются права пользователя.

Наряду с преимуществами сети клиент-серверной архитектуры имеют и ряд недостатков:

- неисправность сервера может сделать сеть неработоспособной, либо, как минимум, приведет к потере сетевых ресурсов;
- требуют квалифицированного персонала для администрирования;
- имеют более высокую стоимость сетей и сетевого оборудования.

3.2. Настройка динамической адресации в клиент-серверных компьютерных сетях

Как уже было сказано, IP-адреса могут назначаться администратором сети вручную. Это представляет для администратора утомительную процедуру. Ситуация усложняется еще тем, что многие пользователи не обладают достаточными знаниями для того, чтобы конфигурировать свои компьютеры для работы в интернете, и должны поэтому полагаться на администраторов.

Протокол Dynamic Host Configuration Protocol (DHCP) был разработан для того, чтобы освободить администратора от этих проблем. Основным назначением DHCP является динамическое назначение IP-

адресов. Однако, кроме динамического, DHCP может поддерживать и более простые способы ручного и автоматического статического назначения адресов.

В ручной процедуре назначения адресов активное участие принимает администратор, который предоставляет DHCP-серверу информацию о соответствии IP-адресов физическим адресам или другим идентификаторам клиентов. Эти адреса сообщаются клиентам в ответ на их запросы к DHCP-серверу.

При автоматическом статическом способе DHCP-сервер присваивает IP-адрес (и, возможно, другие параметры конфигурации клиента) из пула наличных IP-адресов без вмешательства оператора. Границы пула назначаемых адресов задает администратор при конфигурировании DHCP-сервера. Между идентификатором клиента и его IP-адресом по-прежнему, как и при ручном назначении, существует постоянное соответствие. Оно устанавливается в момент первичного назначения сервером DHCP IP-адреса клиенту. При всех последующих запросах сервер возвращает тот же самый IP-адрес.

При динамическом распределении адресов DHCP-сервер выдает адрес клиенту на ограниченное время, что дает возможность впоследствии повторно использовать IP-адреса другими компьютерами. Служба DHCP обеспечивает надежный и простой способ конфигурации сети TCP/IP, гарантируя отсутствие конфликтов адресов за счет централизованного управления их распределением. Администратор управляет процессом назначения адресов с помощью параметра «продолжительности аренды», который определяет, как долго компьютер может использовать назначенный IP-адрес, перед тем как снова запросить его от сервера DHCP в аренду.

Примером работы протокола DHCP может служить ситуация, когда компьютер, являющийся клиентом DHCP, удаляется из подсети. При этом назначенный ему IP-адрес автоматически освобождается. Когда компьютер подключается к другой подсети, то ему автоматически назначается новый адрес. Ни пользователь, ни сетевой администратор не вмешиваются в этот процесс. Это свойство очень важно для мобильных пользователей.

Рассмотрим настройку DHCP-сервера на примере ОС Windows Server 2012.

1. Установка и авторизация сервера DHCP

Установка службы DHCP выполняется так же, как и установка любой другой компоненты Windows Server: *Пуск – Панель управления – Установка и удаление программ – Установка компонентов*

Windows – Сетевые службы – кнопка *Состав* – выбрать пункт *DHCP* – кнопки *OK*, *Далее* и *Готово* (если потребуется, то указать путь к дистрибутиву системы). Также можно установить DHCP-сервер, используя *Server Manager* (*Диспетчер серверов*), а именно *Start* (*Пуск*) – *Server Manager* (*Диспетчер серверов*), общий вид которого показан на рис 3.3.



Рис. 3.3. Общий вид *Server Manager*

Далее нажимаем *Add roles and features* (*Добавить роль сервера*), можно непосредственно через быстрый запуск, а можно через меню *Управление*, и на странице приветствия жмем *Next* (*Далее*) (рис. 3.4).

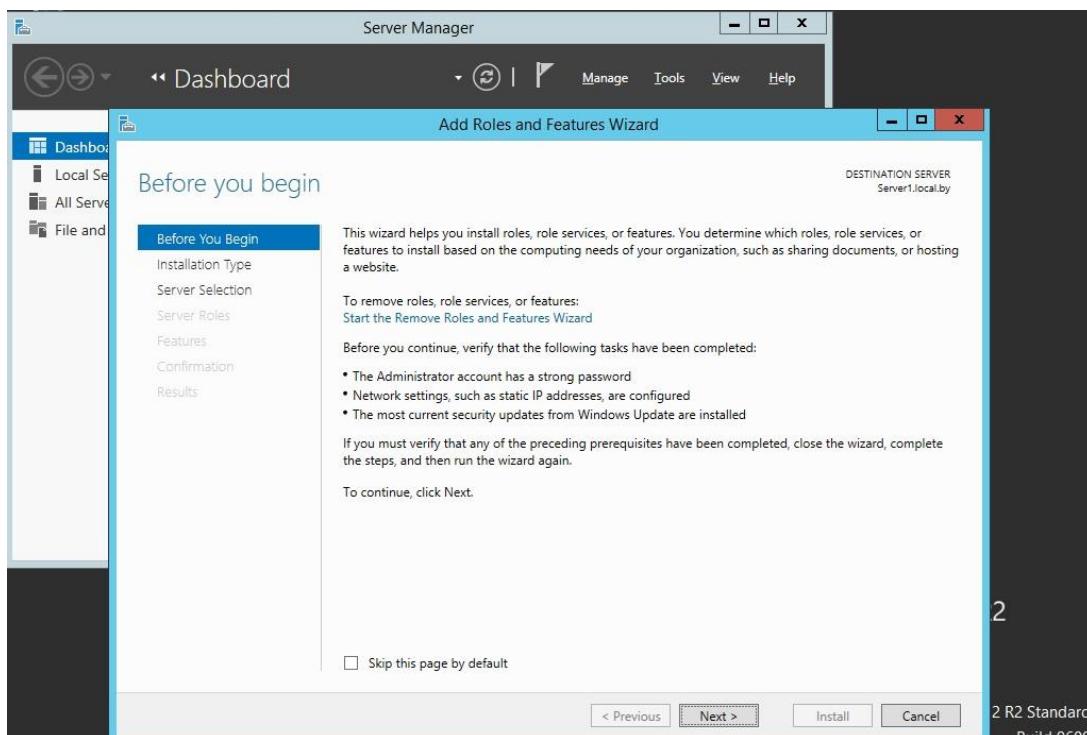


Рис. 3.4. Старт работы мастера установки роли сервера

Далее уже по умолчанию выбран необходимый пункт, т. е. *Role-based or feature-based installation* (*Установка ролей или компонентов*), и поэтому жмем *Далее* (рис. 3.5).

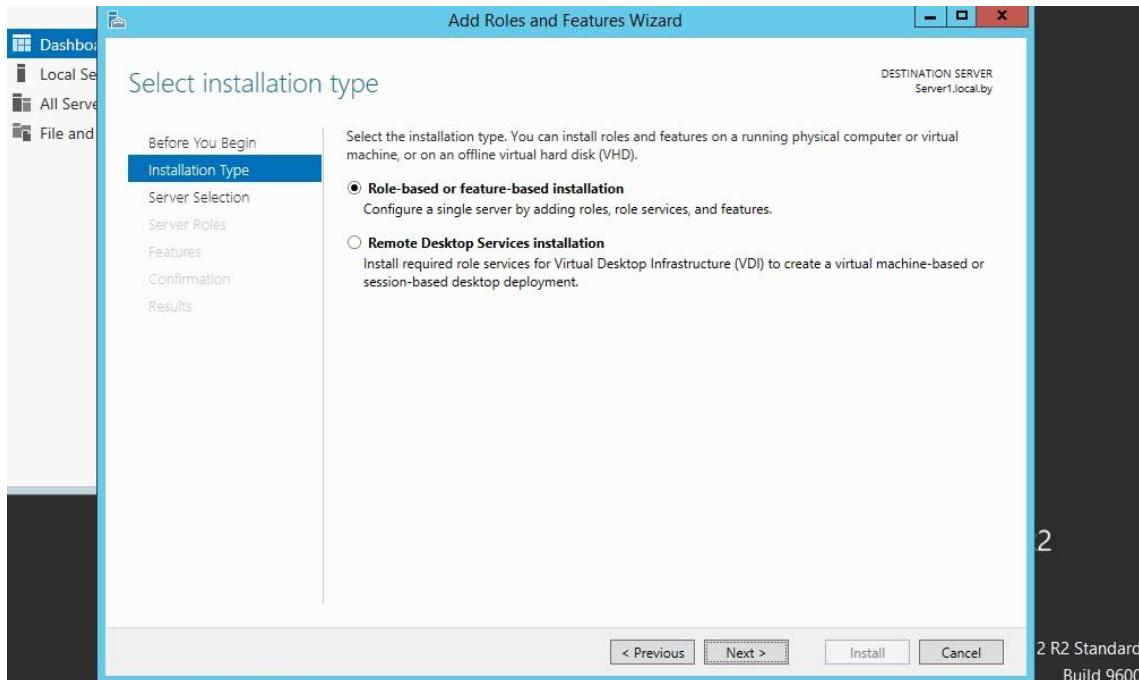


Рис. 3.5. Выбор опции установки ролей и компонент

Затем необходимо выбрать, на какой сервер или виртуальный жесткий диск будет устанавливаться DHCP-сервер (в нашем случае локально, т. е. этот же самый сервер). Далее следует выбрать, какую роль собираемся устанавливать, и соответственно выбираем DHCP-сервер (рис. 3.6).

После нажатия откроется окно, в котором сразу предложат выбрать для установки средства администрирования DHCP-сервера. Необходимо согласиться, иначе далее все равно придется это выбирать, так как администрировать DHCP будем с данного компьютера, и далее жмем *Add Features* (*Добавить компоненты*) (рис. 3.7).

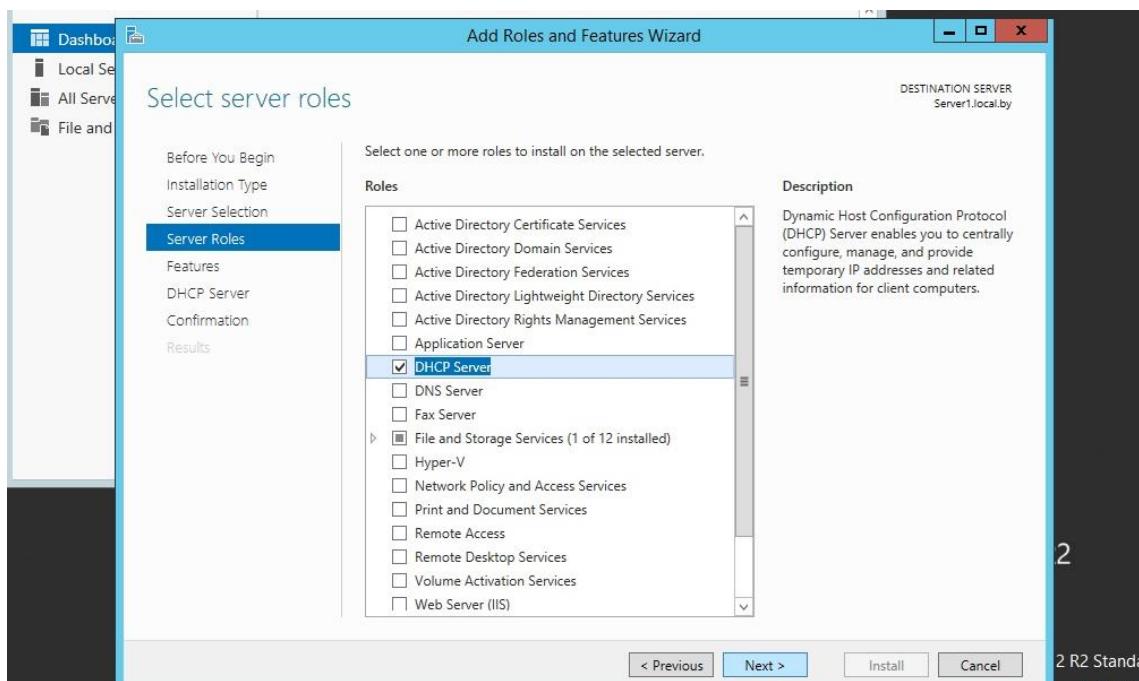


Рис. 3.6. Выбор устанавливаемой роли

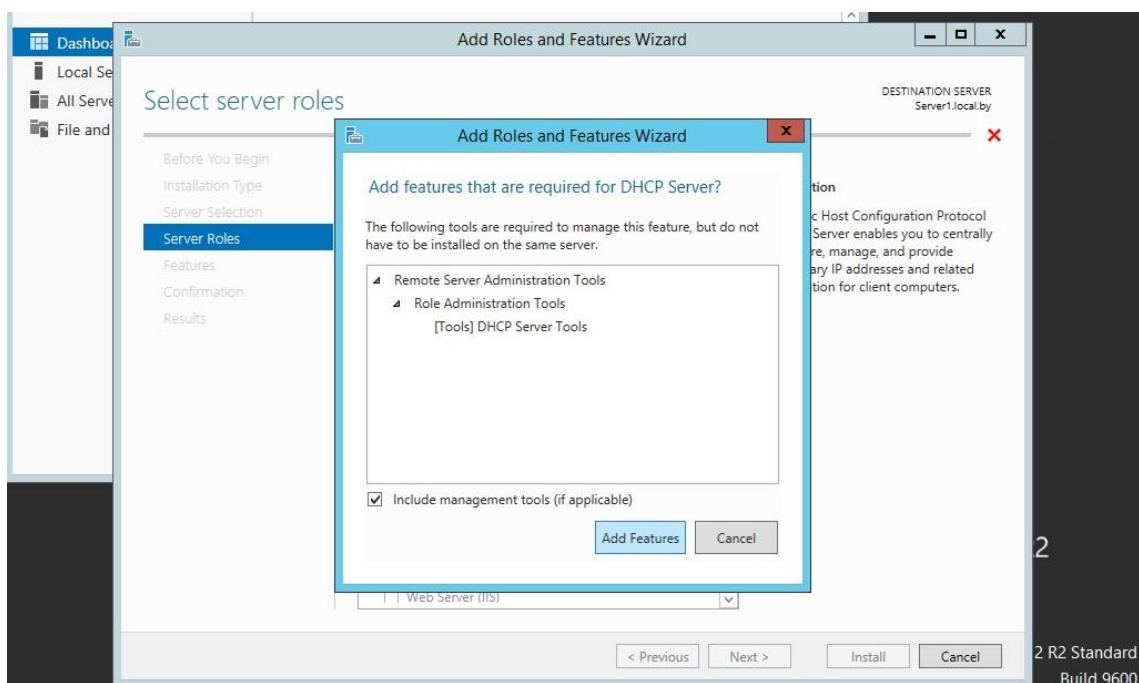


Рис. 3.7. Выбор средств администрирования

Далее будет предложено выбрать необходимые компоненты. Если на прошлом шаге были выбраны *Добавить компоненты*, то необходимые компоненты уже будут выбраны, а соответственно жмем *Next (Далее)* (рис. 3.8).

Еще на нескольких последующих этапах также жмем *Next* (*Далее*), и затем начнется установка DHCP-сервера (рис. 3.9).

После завершения установки будет предложено выполнить предварительную настройку. Рассмотрим настройку DHCP-сервера далее отдельно.

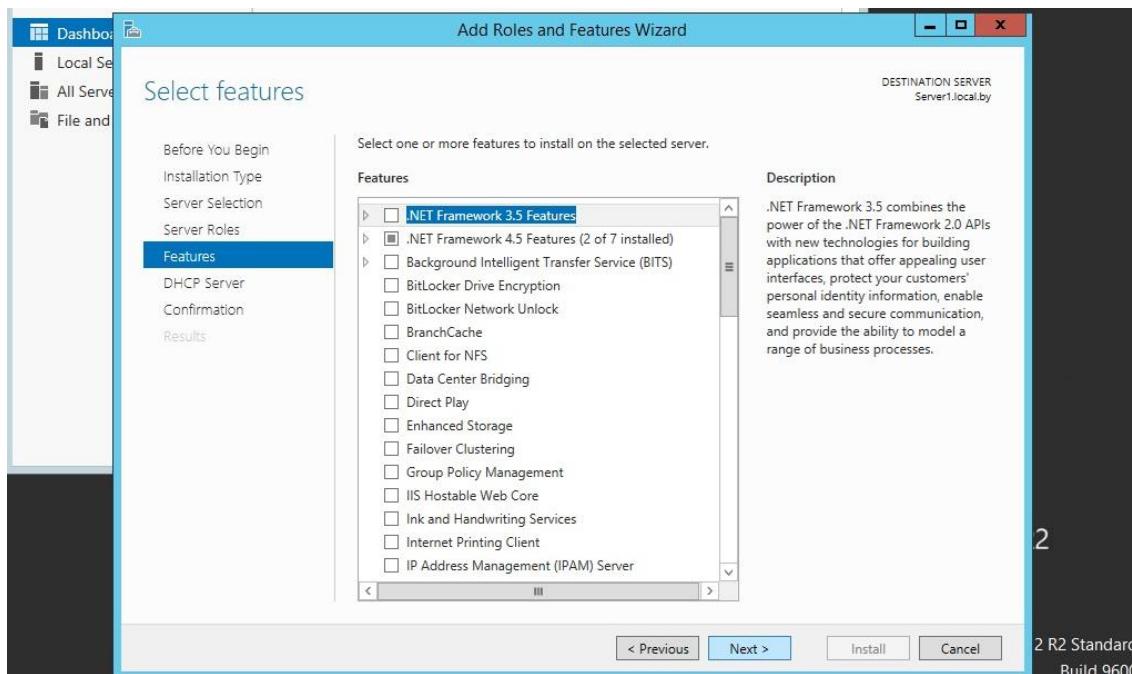


Рис. 3.8. Выбор компонент устанавливаемой роли

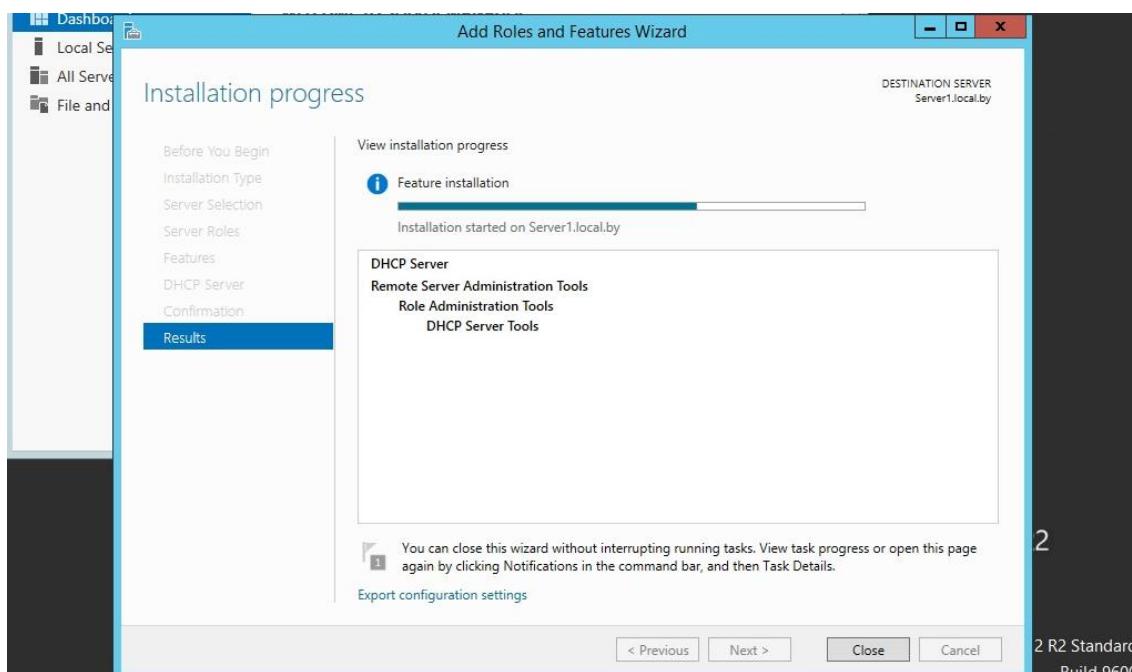


Рис. 3.9. Установка DHCP-сервера

2. Настройка параметров DHCP-сервера

После установки DHCP-сервер и средства его администрирования необходимо настроить. Для этого запускаем оснастку управления DHCP-сервером. Это можно сделать через *Server Manager* (*Диспетчер серверов*), меню *Tools* (*Средства*) (рис. 3.10).

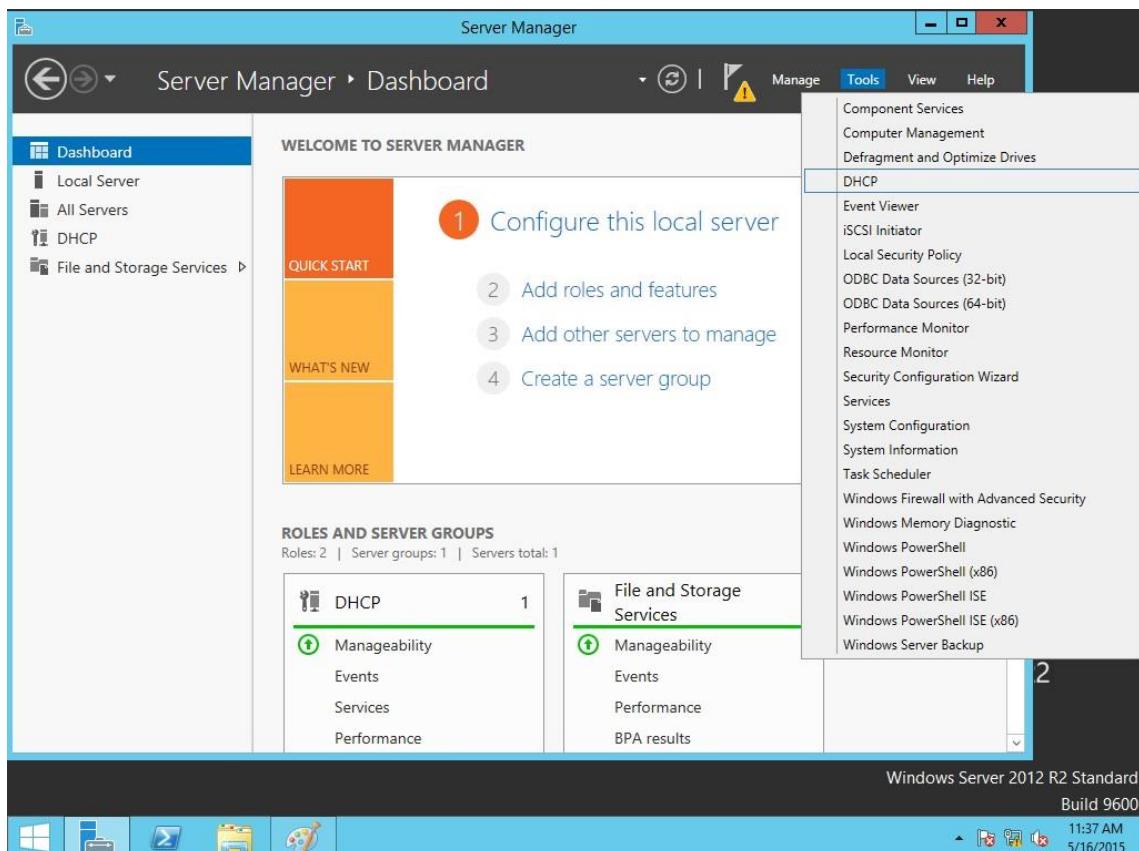


Рис. 3.10. Запуск DHCP-сервера

Создать область можно, щелкнув правой кнопкой мыши на имени сервера и выбрав пункт меню *New Scope* (*Создать область*) (или аналогичный пункт в меню *Действие* консоли DHCP) (рис. 3.11). Консоль запустит *Мастер создания области*, который позволяет по шагам определить все необходимые параметры.

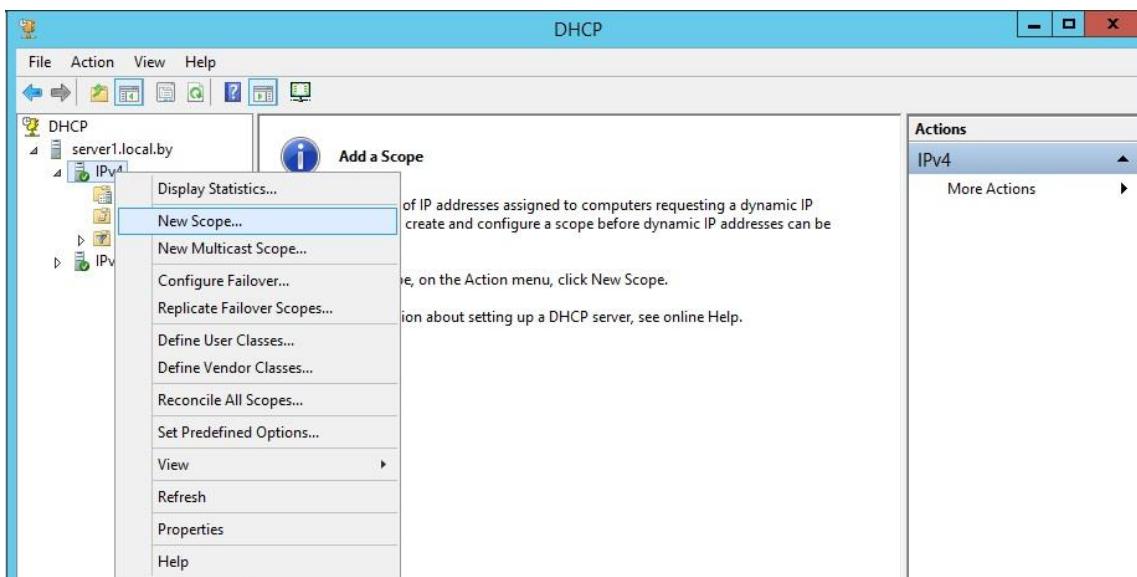


Рис. 3.11. Создание новой области DHCP-сервера

Имя и описание области. В больших сетях именование областей и задание их краткого описания облегчает работу администратора за счет более наглядного отображения в консоли всех созданных областей (рис. 3.12).

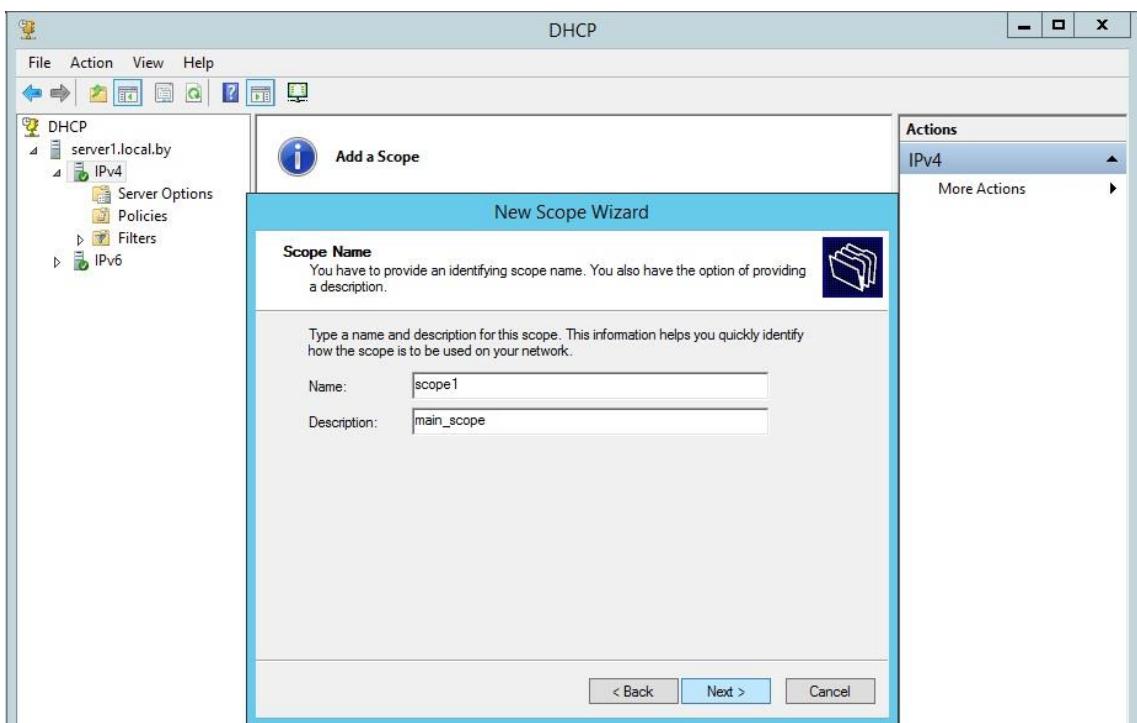


Рис. 3.12. Создание области DHCP-сервера

Дальнейший процесс создания и настройки области в Windows Server 2012 практически ничем не отличается от настройки Windows Server 2003, которая была рассмотрена и изучена в курсе «Компьютерные сети». Фактически необходимо определить диапазон IP-адресов и маски подсети (в данном примере используется подсеть с Network ID 192.168.1.0 и маской 24 бита) (рис. 3.13). Отметим, что при настройке каждый должен использовать диапазон IP-адресов и другие параметры исходя из выбранного варианта задания.

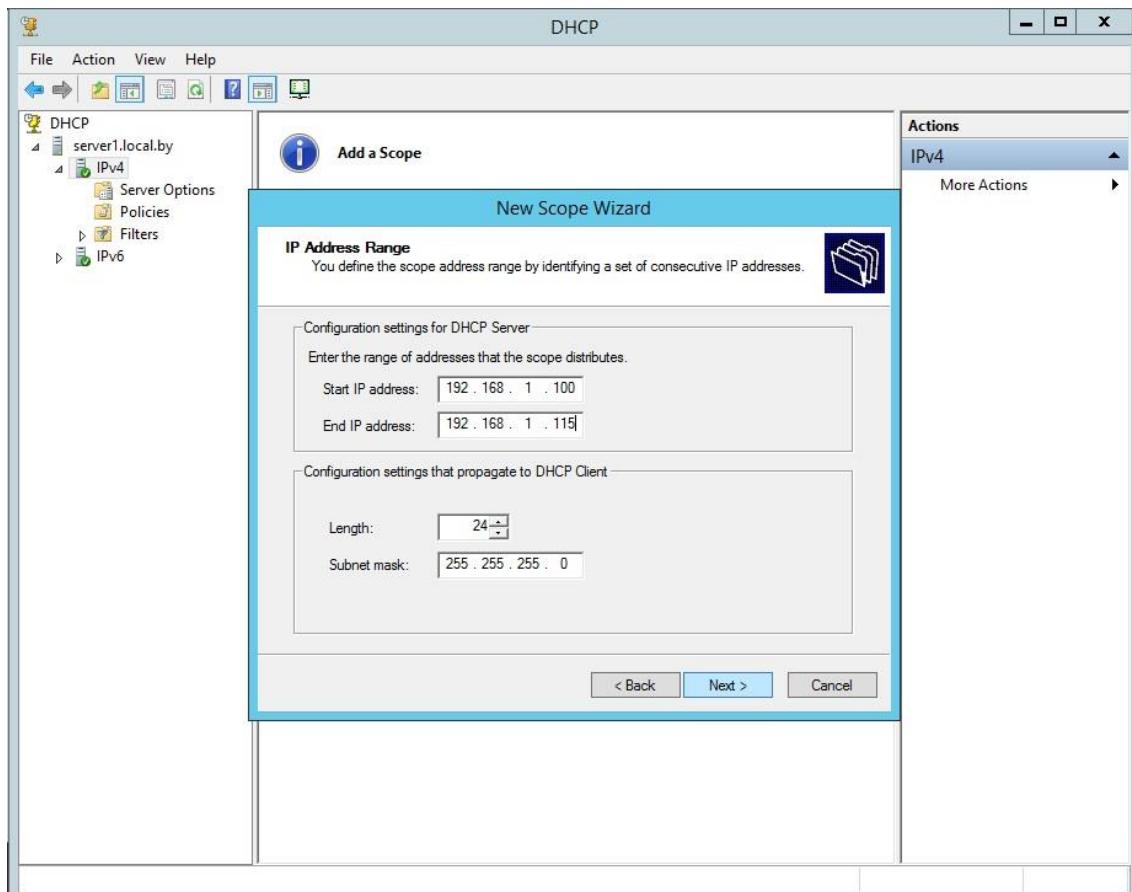


Рис. 3.13. Определение диапазона адресов области

Добавление исключений. На данном шаге задаются диапазоны IP-адресов, которые будут исключены из процесса выдачи адресов клиентам (все статические IP-адреса должны быть обязательно исключены из действующего диапазона адресов). В рассмотренном на рис. 3.14 примере исключаются адреса обоих серверов: 192.168.100 и 192.168.1.101.

Срок действия аренды. Стандартный срок действия – 8 дней (рис. 3.15).

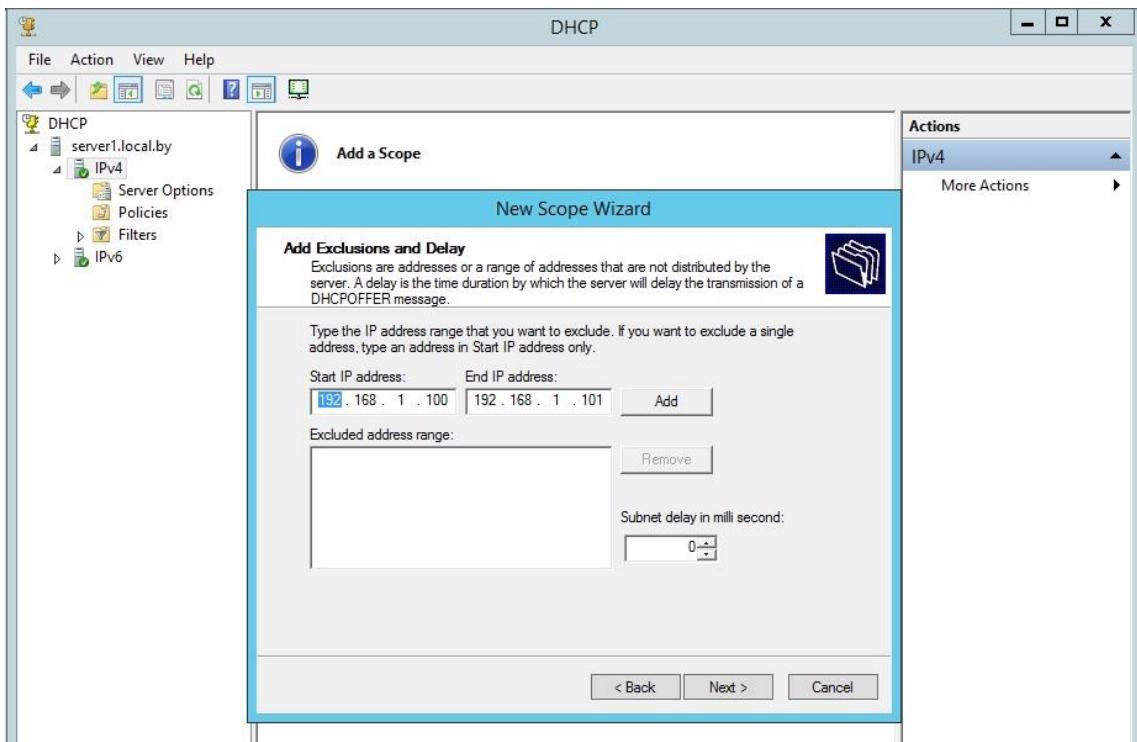


Рис. 3.14. Добавление исключающего диапазона адресов области

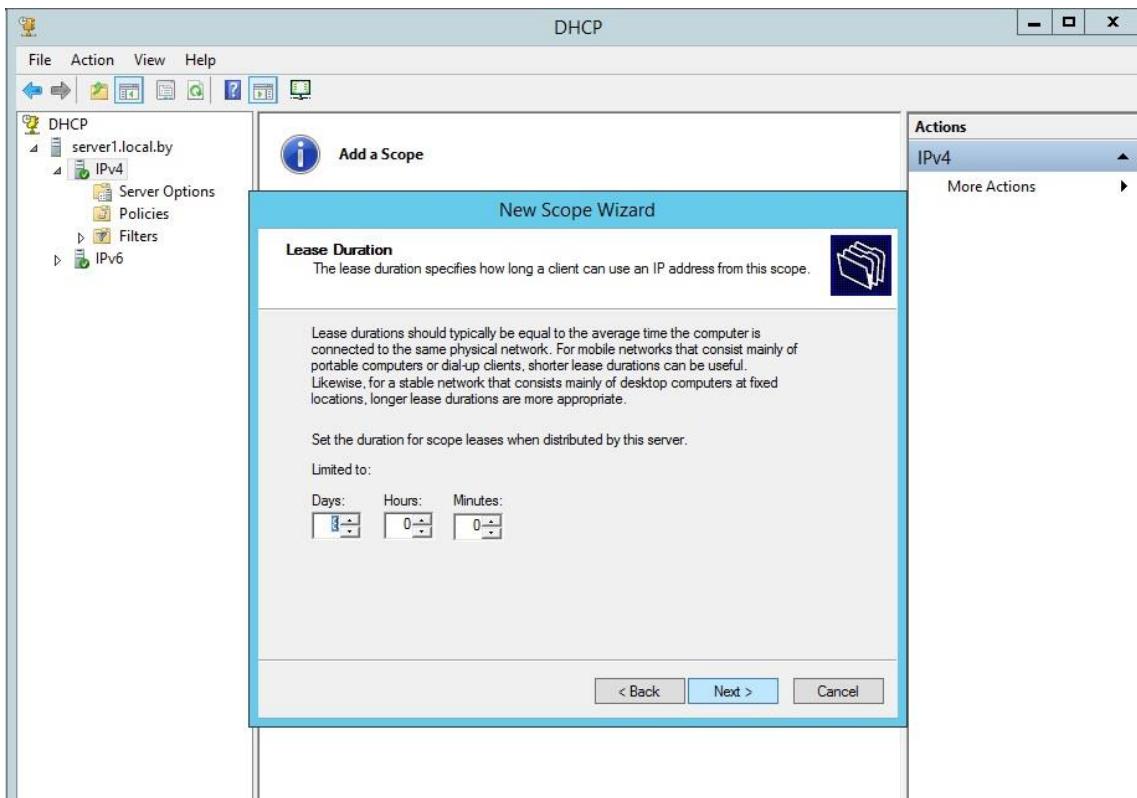


Рис. 3.15. Определение срока аренды клиентом адресов

Если в сети редко происходят изменения (добавление или удаление сетевых узлов, перемещение сетевых узлов из одной подсети в другую), то срок действия можно увеличить, это сократит количество запросов на обновление аренды. Если же сеть более динамичная, то срок аренды можно сократить, это позволит быстрее возвращать в пул IP-адреса, которые принадлежали компьютерам, уже удаленными из данной подсети.

Далее мастер предложит настроить параметры, специфичные для узлов IP-сети, относящихся к данной области, например, маршрутизатор (основной шлюз), адрес DNS-сервера (можно назначить несколько адресов, рис. 3.16); адрес WINS-сервера (аналогично серверу DNS; можно также назначить несколько адресов).

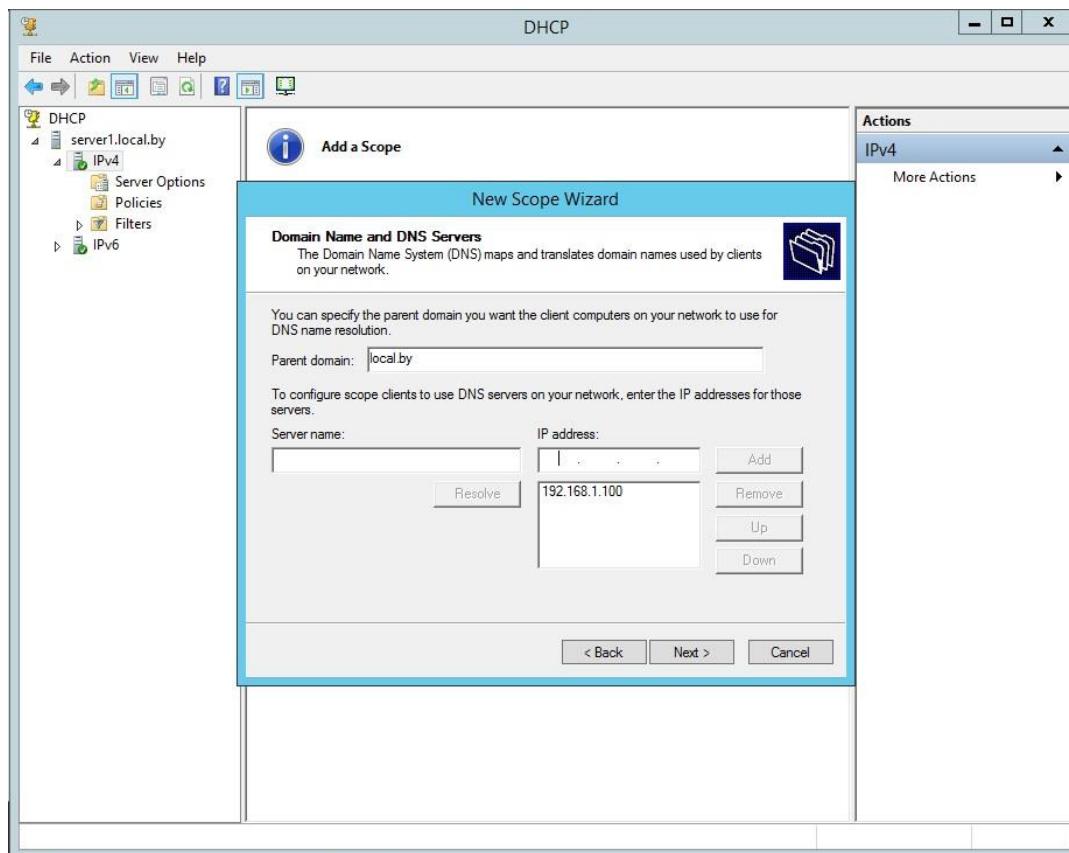


Рис. 3.16. Добавление адреса DNS-сервера, распределяемого областью

Запрос на активацию области. IP-адреса, заданные в созданной области, не будут выдаваться клиентам, пока область не будет активирована (рис. 3.17).

Далее завершаем работу мастера и область готова к использованию. Если какие-либо параметры (например, адреса серверов DNS

или WINS) являются общими для всех областей, управляемых данным DHCP-сервером, то такие параметры лучше определить не в разделе параметров каждой области, а в разделе параметров самого сервера.

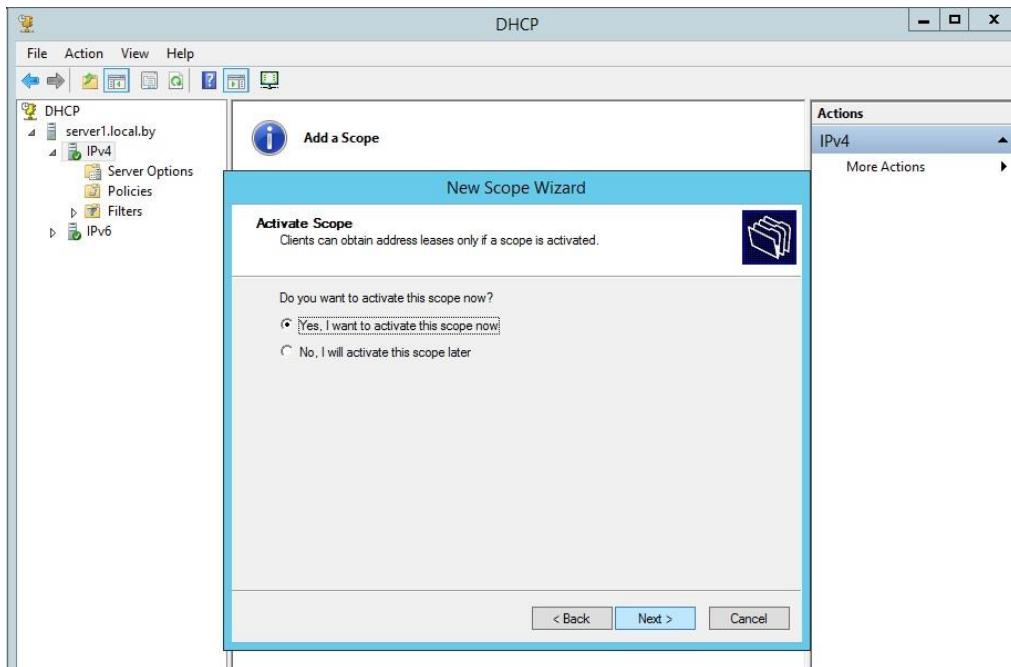


Рис. 3.17. Запрос на активацию области DHCP-сервера

Однако использование DHCP несет в себе как положительные моменты (централизация управления), так и некоторые проблемы. Во-первых, это проблема согласования информационной адресной базы в службах DHCP и DNS. Как известно, DNS служит для преобразования символьных имен в IP-адреса. Если IP-адреса будут динамически изменяться сервером DHCP, то эти изменения необходимо также динамически вносить в базу данных сервера DNS (будет рассмотрено в следующей лабораторной работе).

Во-вторых, нестабильность IP-адресов усложняет процесс управления сетью – необходимо выполнить резервирование IP-адресов по MAC-адресам. Это реализуется через заполнение поля reservations в соответствующей области (scope) DHCP-сервера, тем самым формируя таблицу соответствия MAC и IP-адресов. Аналогичные проблемы возникают и при конфигурировании фильтров маршрутизаторов, которые оперируют с IP-адресами.

Наконец, централизация процедуры назначения адресов снижает надежность системы: при отказе DHCP-сервера все его клиенты оказываются не в состоянии получить IP-адрес и другую информацию о

конфигурации. Последствия такого отказа могут быть уменьшены путем использования в сети нескольких серверов DHCP (полноценное резервирование DHCP-серверов реализовано лишь начиная с Windows Server 2012).

3.3. Лабораторная работа № 4–5

Цель: изучение методов организации клиент-серверной сети на базе операционных систем Windows с использованием динамической адресации.

Задание: организация клиент-серверной сети между двумя хостами (целесообразно использовать виртуальные ОС из лабораторной работы № 2–3) и одним сервером с централизованным управлением IP-адресацией. В качестве хостов и сервера должны выступать виртуальные операционные системы типа Windows. Диапазон адресов HOST ID, IP-адрес DHCP-сервера выбираются по таблице согласно варианту. DHCP-сервер выступает в роли шлюза (IP-адрес шлюза должен передаваться клиентам в IP-конфигурации). Предполагается также, что в сети имеются два DNS-сервера (IP-адреса выбираются по таблице согласно варианту), адреса которых должны быть присвоены IP-конфигурациям клиентов. Все статические адреса сети, входящие в scope, должны быть занесены в исключения.

На сервере для гостевой учетной записи должны быть созданы три папки:

- read (только чтение);
- full (полный доступ);
- limit (разрешена работа с файлами (но запрещено удаление файлов), запрещена работа с каталогами (создание, удаление, но разрешен их просмотр)). Тонкая настройка прав доступа представлена в подразделе 2.4.

Результаты продемонстрировать на примере нескольких клиентских ОС, используя утилиту *ipconfig*.

Варианты заданий для лабораторной работы № 4–5

| № варианта | Диапазон HOST ID | | IP-адрес DHCP-сервера | IP-адреса DNS-серверов |
|------------|-------------------|-------------------|-----------------------|--------------------------------------|
| | начальный | конечный | | |
| 1 | 172.16.194.1/20 | 172.16.194.10/20 | 172.16.194.1/20 | 172.16.194.1/20; 172.16.194.5/20 |
| 2 | 172.16.194.11/20 | 172.16.194.20/20 | 172.16.194.11/20 | 172.16.194.11/20; 172.16.194.15/20 |
| 3 | 172.16.194.21/20 | 172.16.194.30/20 | 172.16.194.21/20 | 172.16.194.21/20; 172.16.194.25/20 |
| 4 | 172.16.194.31/20 | 172.16.194.40/20 | 172.16.194.31/20 | 172.16.194.31/20; 172.16.194.35/20 |
| 5 | 172.16.194.41/20 | 172.16.194.50/20 | 172.16.194.41/20 | 172.16.194.41/20; 172.16.194.45/20 |
| 6 | 172.16.194.51/20 | 172.16.194.60/20 | 172.16.194.51/20 | 172.16.194.51/20; 172.16.194.55/20 |
| 7 | 172.16.194.61/20 | 172.16.194.70/20 | 172.16.194.61/20 | 172.16.194.61/20; 172.16.194.65/20 |
| 8 | 172.16.194.71/20 | 172.16.194.80/20 | 172.16.194.71/20 | 172.16.194.71/20; 172.16.194.75/20 |
| 9 | 172.16.194.81/20 | 172.16.194.90/20 | 172.16.194.81/20 | 172.16.194.81/20; 172.16.194.85/20 |
| 10 | 172.16.194.91/20 | 172.16.194.100/20 | 172.16.194.91/20 | 172.16.194.91/20; 172.16.194.95/20 |
| 11 | 172.16.194.101/20 | 172.16.194.110/20 | 172.16.194.101/20 | 172.16.194.101/20; 172.16.194.105/20 |
| 12 | 172.16.194.111/20 | 172.16.194.120/20 | 172.16.194.111/20 | 172.16.194.111/20; 172.16.194.115/20 |
| 13 | 172.16.194.121/20 | 172.16.194.130/20 | 172.16.194.121/20 | 172.16.194.121/20; 172.16.194.125/20 |
| 14 | 172.16.194.131/20 | 172.16.194.140/20 | 172.16.194.131/20 | 172.16.194.131/20; 172.16.194.135/20 |
| 15 | 172.16.194.141/20 | 172.16.194.150/20 | 172.16.194.141/20 | 172.16.194.141/20; 172.16.194.145/20 |

4. СИМВОЛЬНАЯ АДРЕСАЦИЯ В СЕТЯХ С КЛИЕНТ-СЕРВЕРНОЙ АРХИТЕКТУРОЙ

4.1. Символьный адрес DNS

В стеке протоколов TCP/IP, как уже ранее говорилось, используются три типа адресов – физические, IP-адреса и символьные доменные имена. Физические адреса служат для адресации на канальном уровне. IP-адреса применяются на сетевом уровне. Доменные имена кажутся в этом ряду необязательными, ведь сеть будет работать и без них. Однако пользователю сети неудобно запоминать числовые IP-адреса, ассоциируя их с конкретными сетевыми объектами. Все привыкли к символьным именам, и именно поэтому в стек TCP/IP была введена система доменных имен DNS (Domain Name System). Она описывается в RFC 1034 и RFC 1035. Полное название доменных имен – FQDN (Fully Qualified Domain Name – полностью определенное имя домена). Кроме DNS-имен операционные системы Windows Server поддерживают символьные имена NetBIOS.

DNS (Domain Name System) – это распределенная база данных, поддерживающая иерархическую систему имен для идентификации узлов в сети Internet.

Служба DNS предназначена для автоматического поиска IP-адреса по известному символьному имени узла. DNS требует статической конфигурации своих таблиц, разрешающих имена компьютеров в IP-адреса.

Протокол DNS является служебным протоколом прикладного уровня. Этот протокол несимметричен – в нем определены DNS-серверы и DNS-клиенты.

DNS-серверы хранят часть распределенной базы данных о соответствии символьных имен и IP-адресов. Эта база данных распределена по административным доменам сети Internet. Клиенты сервера DNS знают IP-адрес сервера DNS своего административного домена и по протоколу IP передают запрос, в котором сообщают известное символьное имя и просят вернуть соответствующий ему IP-адрес. Если данные о запрошенном соответствии хранятся в базе данного DNS-сервера, то он сразу посыпает ответ клиенту, если же нет – то он посыпает запрос DNS-серверу другого домена, который может сам обработать запрос либо передать его другому DNS-серверу. Все DNS-серверы соединены иерархически в соответствии с иерархией доменов сети Internet. Клиент опрашивает эти серверы имен, пока не найдет нужные отображения.

Этот процесс ускоряется из-за того, что серверы имен постоянно кэшируют информацию, предоставляемую по запросам. Клиентские компьютеры могут использовать в своей работе IP-адреса нескольких DNS-серверов для повышения надежности своей работы.

База данных DNS имеет структуру дерева, называемого *доменным пространством имен*, в котором каждый домен (узел дерева) имеет имя и может содержать поддомены.

Имя домена идентифицирует его положение в этой базе данных по отношению к родительскому домену, причем точки в имени отделяют части, соответствующие узлам домена.

Корень базы данных DNS управляет центром Internet Network Information Center. Домены верхнего уровня назначаются для каждой страны, а также на организационной основе. Имена этих доменов должны отвечать международному стандарту ISO 3166. Для обозначения стран используются трехбуквенные и двухбуквенные аббревиатуры, а для различных типов организаций применяются следующие аббревиатуры:

- *.com* – коммерческие организации (например, microsoft.com);
- *.edu* – образовательные (например, mit.edu);
- *.gov* – правительственные организации (например, nsf.gov);
- *.org* – некоммерческие организации (например, fidonet.org);
- *.net* – организации, поддерживающие сети (например, nsf.net).

Каждый домен DNS администрируется отдельной организацией, которая обычно разбивает свой *домен* на *поддомены* и передает функции администрирования этих поддоменов другим организациям. Каждый домен имеет уникальное имя, а каждый из поддоменов имеет уникальное имя внутри своего домена. Имя домена может содержать до 63 символов. Каждый хост в сети Internet однозначно определяется своим полным доменным именем (Fully Qualified Domain Name, FQDN), которое включает имена всех доменов по направлению от хоста к корню.

В процессе разрешения участвуют DNS-клиент и DNS-сервер. Системный компонент DNS-клиента, называемый DNS-распознавателем, отправляет запросы на DNS-серверы. Бывает двух видов:

- интерактивный – DNS-сервер обращается к DNS-серверу с просьбой разрешить имя без обращения к другим DNS-серверам;
- рекурсивный – всю работу по разрешению имени выполняет DNS-сервер путем отправки запросов другим DNS-серверам. DNS-сервер всегда сначала ищет имя в собственной базе данных или в кэше, в случае отсутствия обращается к другим серверам.

В основном DNS-клиентами используются рекурсивные запросы. На рис. 4.1 проиллюстрирован процесс разрешения доменного имени с помощью рекурсивного запроса.

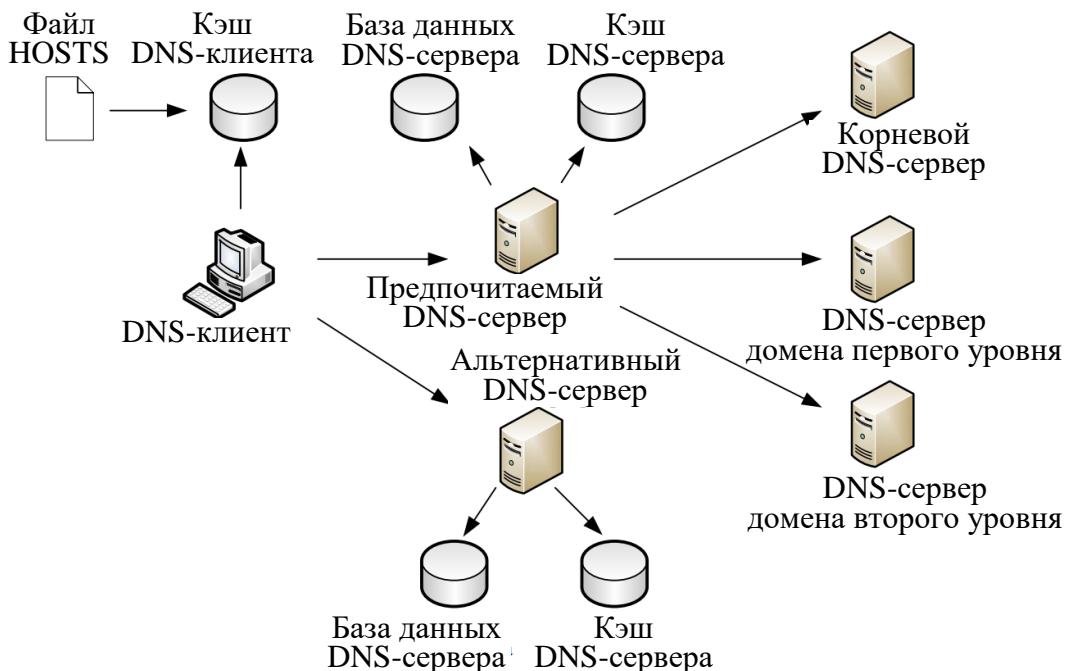


Рис. 4.1. Процесс рекурсивного разрешения имен

Сначала DNS-клиент осуществляет поиск в собственном локальном кэше DNS-имен. Это память для временного хранения ранее разрешенных запросов. В эту же память переносится содержимое файла hosts (каталог windows/system32/drivers/etc). Утилита IPconfig с ключом /displaydns отображает содержимое DNS-кэша. Если кэш не содержит требуемой информации, DNS-клиент обращается с рекурсивным запросом к предпочтительному DNS-серверу (Preferred DNS server), адрес которого указывается при настройке стека TCP/IP. DNS-сервер просматривает собственную базу данных, а также кэш-память, в которой хранятся ответы на предыдущие запросы, отсутствующие в базе данных. В том случае, если запрашиваемое доменное имя не найдено, DNS-сервер осуществляет итеративные запросы к DNS-серверам верхних уровней, начиная с корневого DNS-сервера.

Рассмотрим процесс разрешения доменного имени на примере. Пусть требуется разрешить имя www.microsoft.com. Корневой домен содержит информацию о DNS-сервере, содержащем зону .com. Следующий запрос происходит к этому серверу, на котором хранятся данные о всех поддоменах зоны .com, в том числе о домене microsoft и его

DNS-сервере. Сервер зоны microsoft.com может непосредственно разрешить имя www.microsoft.com в IP-адрес. Обращение к альтернативному серверу осуществляется, только если основной сервер недоступен.

Просмотр DNS-кэша осуществляется утилитой *ipconfig /displaydns*, очистка кэша – *ipconfig /flushdns*.

4.2. Символьный адрес NetBIOS

Протокол *NetBIOS* (Network Basic Input/Output System – сетевая базовая система ввода/вывода) был разработан в 1984 г. для корпорации IBM как сетевое дополнение стандартной BIOS на компьютерах IBM PC. В операционных системах Microsoft Windows NT, а также в Windows 98 протокол и имена NetBIOS являлись основными сетевыми компонентами. Начиная с Windows 2000, операционные системы Microsoft ориентируются на глобальную сеть Интернет, в этой связи фундаментом сетевых решений стали протоколы TCP/IP и доменные имена. Однако поддержка имен NetBIOS осталась и в операционной системе Windows Server 2008, а также Windows Server 2012.

Система имен NetBIOS представляет собой простое неиерархическое пространство, т. е. в имени NetBIOS отсутствует структура, деление на уровни, как в DNS-именах. Длина имени не более 15 символов (плюс один служебный).

Для преобразования NetBIOS-имен в IP-адреса в операционной системе Windows Server используется служба *WINS* – Windows Internet Naming Service (служба имен в Интернете для Windows).

Служба WINS работает, как и служба DNS, по модели клиент – сервер. WINS-клиенты используют WINS-сервер для регистрации своего NetBIOS-имени и преобразования неизвестного NetBIOS-имени в IP-адрес. Функции сервера NetBIOS-имен описаны в RFC 1001 и 1002.

Процесс разрешения имен в пространстве NetBIOS может быть выполнен одним из трех способов:

- 1) широковещательный запрос;
- 2) обращение к локальной базе данных NetBIOS -имен (*LMhosts*), хранящихся в папке, где файл *hosts*, отображающий FQDN-имена;
- 3) обращение к централизованной базе данных имен NetBIOS, хранящихся на сервере WINS.

В зависимости от типа узла NetBIOS разрешение имен осуществляется с помощью различной комбинации перечисленных способов. Выделяют четыре типа узла:

- b-узел (broadcast node, широковещательный) – разрешает имена в ip-адресах посредством широковещательных сообщений broadcast node;
- p-узел (peer node) – разрешает имена в IP-адреса с помощью WINS-сервера;
- m-узел (mixed node, смешанный узел) – комбинирует запросы b- и p-узлов, первоначально узел пытается применить широковещательный запрос, а в случае неудачи – обращается к WINS-серверу;
- h-узел (hybrid node, гибридный) – комбинирует запросы b- и p-узлов, но при этом сначала обращается к WINS-серверу, а при неудаче выполняет широковещательную рассылку.

Наиболее эффективным является h-узел. Тип узла определяется следующим образом: если в свойствах протокола TCP/IP нет адреса WINS-сервера, то данный компьютер считается b-узлом, в противном случае является h-узлом. Использование других типов узлов настраивается через реестр Windows.

В больших сетях для распределения нагрузки по регистрации и разрешению NetBIOS-имен необходимо использовать несколько WINS-серверов. Считается, что один WINS-сервер должен обслуживать порядка нескольких сотен компьютеров. При использовании нескольких серверов часть клиентов настраивается на регистрацию и разрешение имен на один WINS-сервер, вторая – на другой, а между серверами, по аналогии с системой DNS, настраивается репликация.

4.3. Настройка DNS-сервера

Рассмотрим организацию DNS-адресации в локальной сети на примере Windows Server 2012 R2. Для организации DNS-адресации необходимо выполнить определенные действия на двух серверах (с именами Server1 и Server2) и клиенте.

1. Установка DNS-сервера

Установка службы DNS производится в целом аналогично установке DHCP-сервера, описанной ранее в подразделе 3.2, с той лишь разницей, что выбирается установка службы DNS.

2. Создание основной зоны прямого просмотра

На сервере DC1 создадим стандартную основную зону, например, с именем world.ru:

- откройте консоль DNS (рис. 4.2);
- выберите раздел *Forward Lookup Zones* (*Зоны прямого просмотра*) и запустите мастер создания зоны (тип зоны – *Primary (Основная)*).

новная), динамические обновления – разрешить, остальные параметры – по умолчанию) (рис. 4.3);

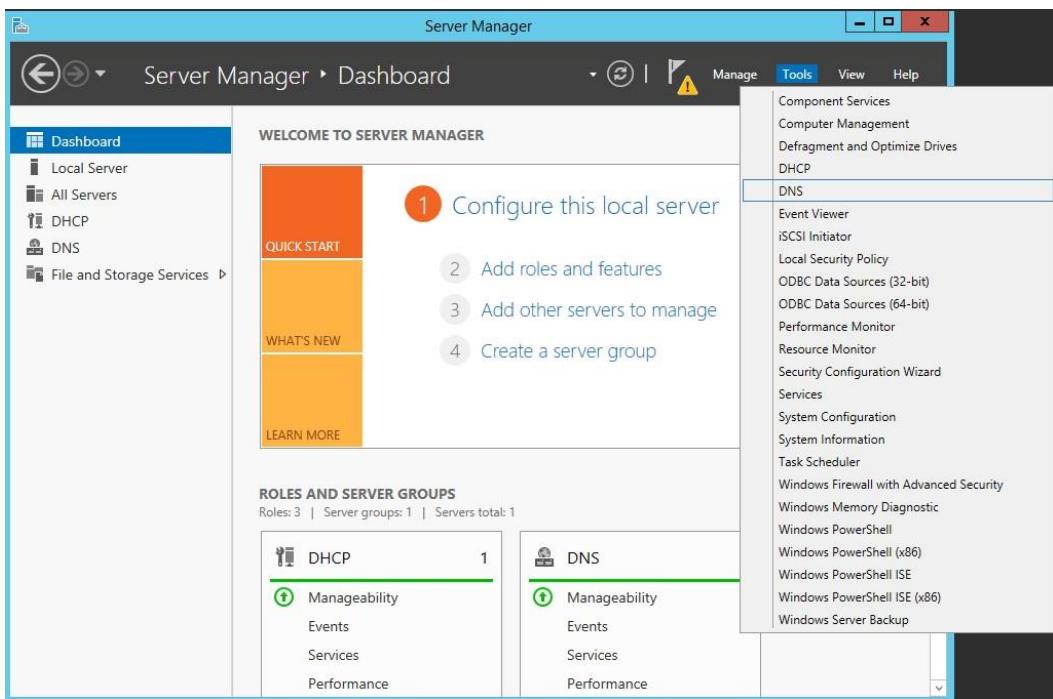


Рис. 4.2. Открытие консоли DNS-сервера

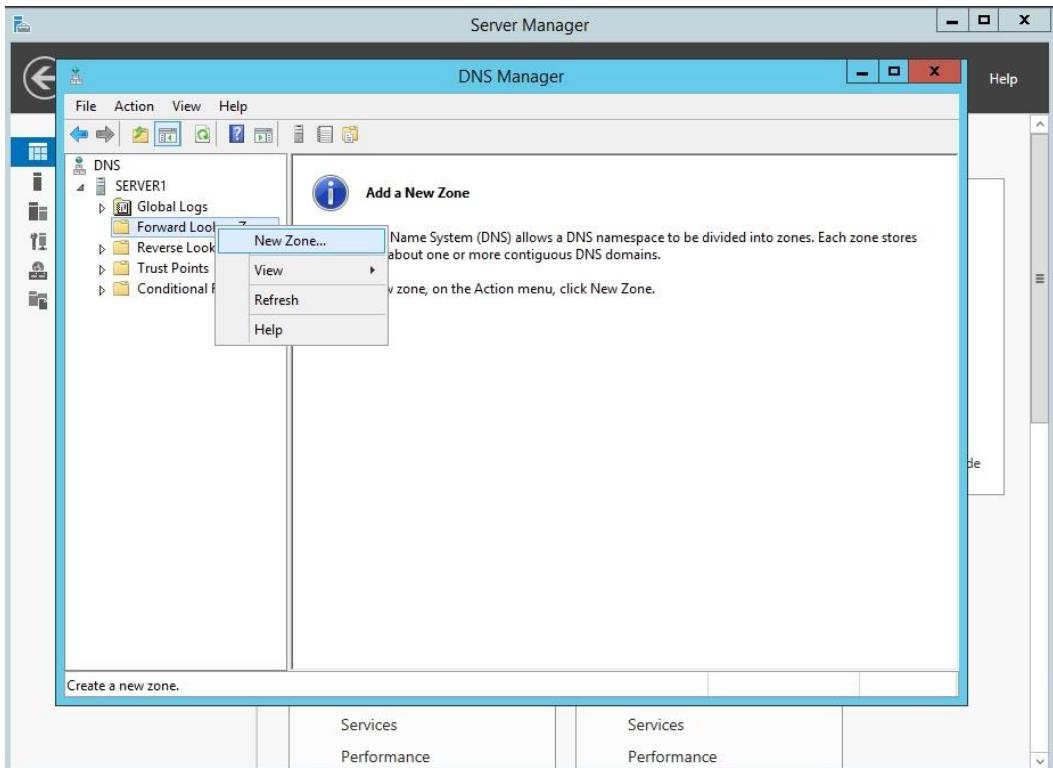


Рис. 4.3. Запуск мастера для создания новой зоны DNS-сервера

- введите тип зоны и имя – в примере используется название local.by (рис. 4.4 и 4.5); имя файла, хранящего информацию о зоне, сформируется автоматически (рис. 4.6);

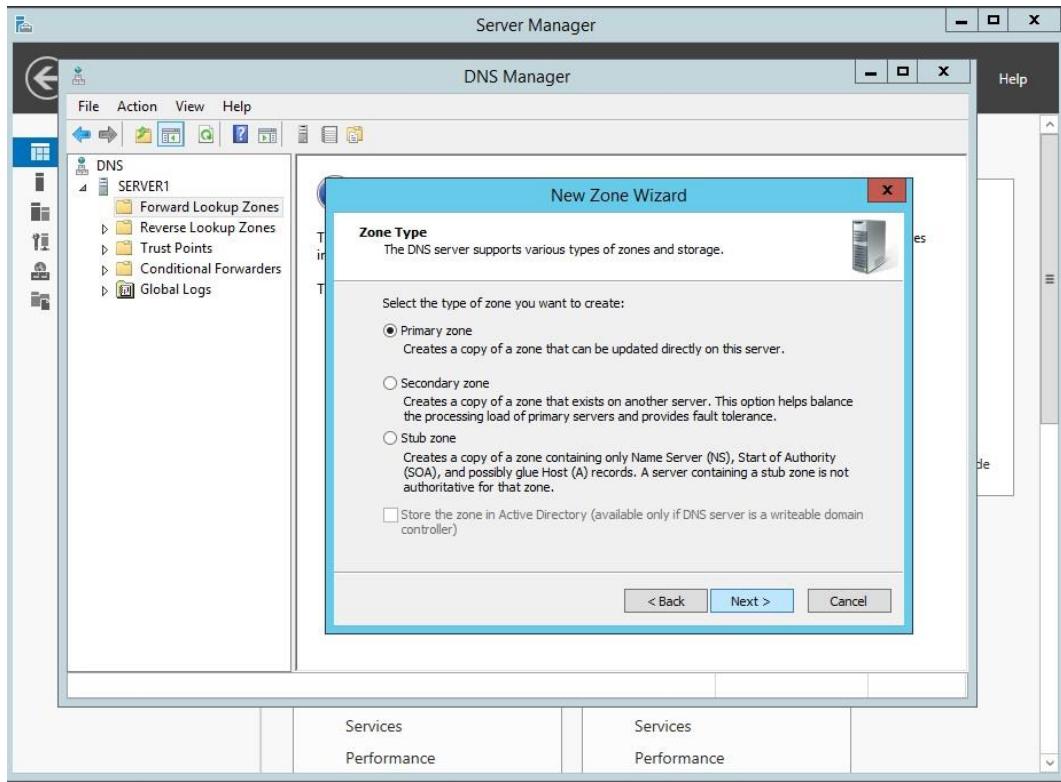


Рис. 4.4. Выбор типа новой зоны DNS-сервера

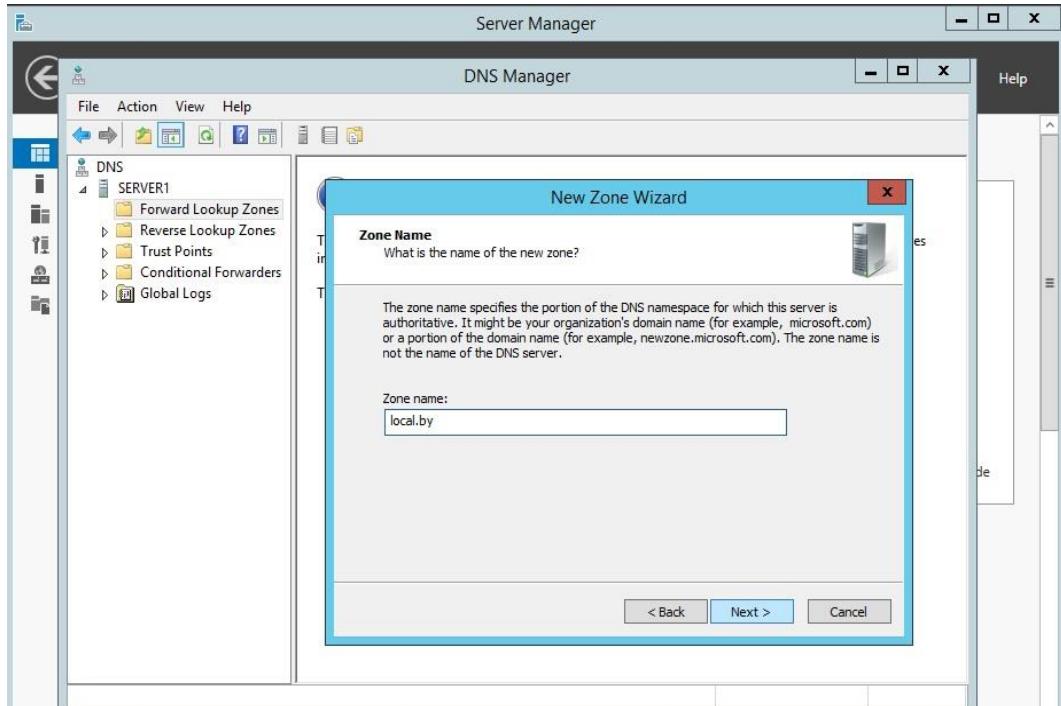


Рис. 4.5. Выбор названия новой зоны DNS-сервера

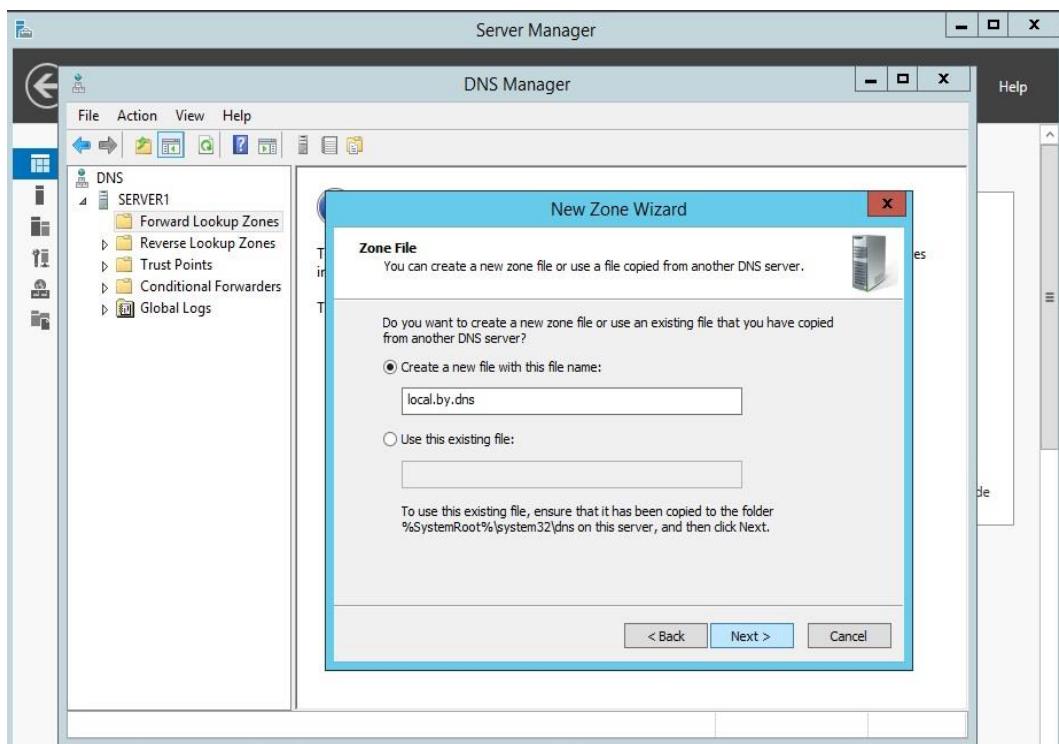


Рис. 4.6. Название файла новой зоны DNS-сервера

- разрешите передачу данной зоны на любой сервер DNS (консоль DNS – зона local.by – *Properties* (*Свойства*) – закладка *Zone Transfers* (*Передачи зон*) – Отметьте *Allow zone transfers* (*Разрешить передачи*) и *To any server* (*На любой сервер*)) (рис. 4.7).

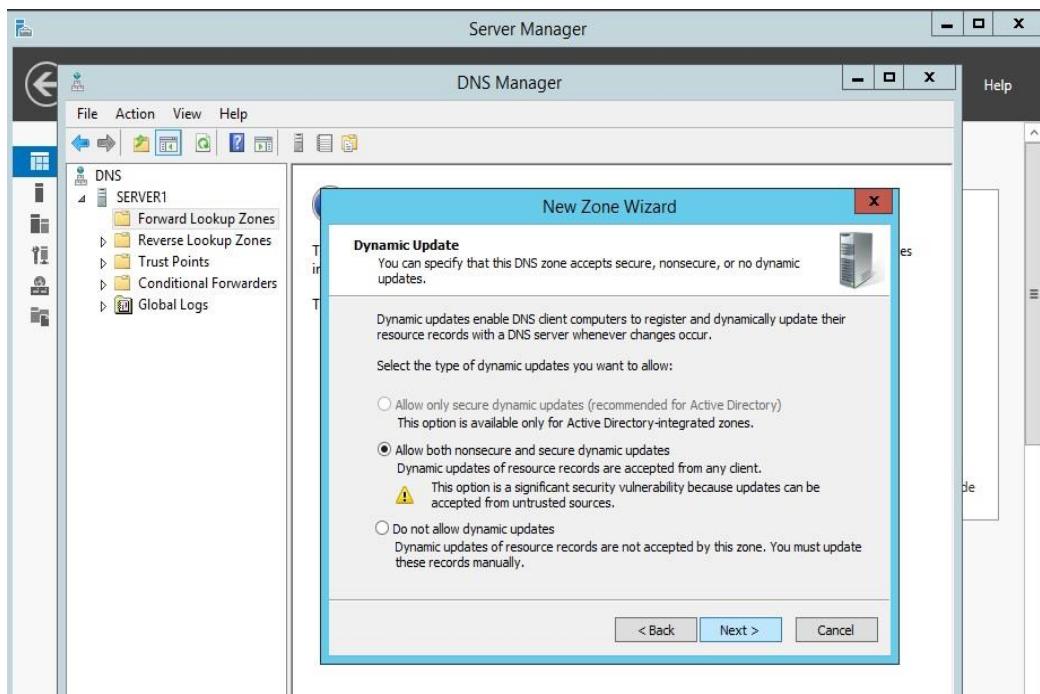


Рис. 4.7. Разрешение на передачу зоны на другой сервер

В итоге получим зону прямого просмотра DNS-сервера, как показано на рис. 4.8.

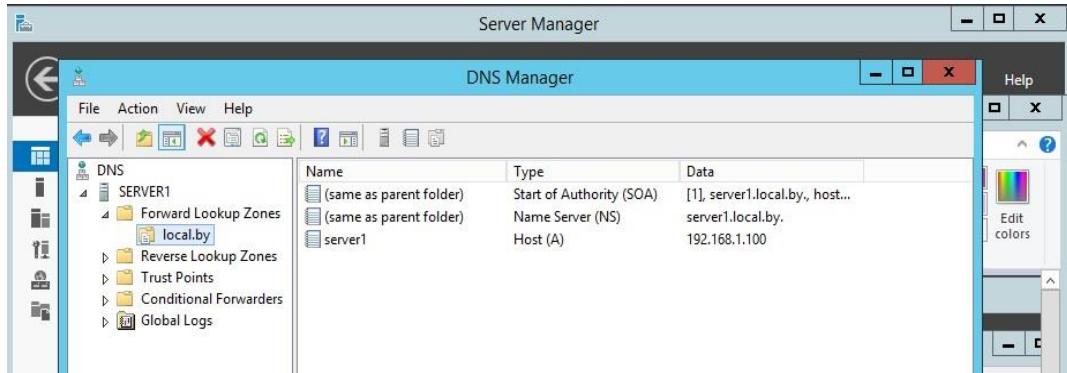


Рис. 4.8. DNS-сервер с созданной зоной прямого просмотра

Чтобы на DNS-сервере автоматически зарегистрировалось имя сервера (в нашем случае `server1`), необходимо указать в свойствах компьютера DNS-суффикс (рис. 4.9), а также в IP-конфигурации должен быть указан адрес DNS-сервера (в нашем случае это все тот же 192.168.1.100).

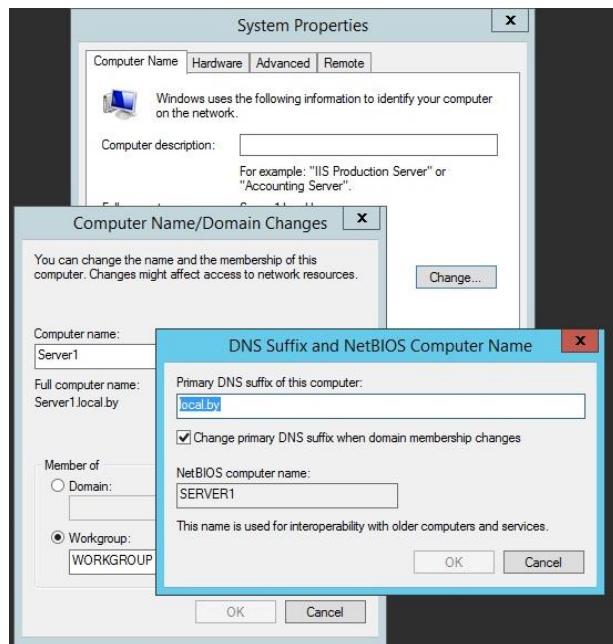


Рис. 4.9. DNS-суффикс для символьного имени компьютера

3. Создание дополнительной зоны прямого просмотра

На втором сервере создадим стандартную дополнительную зону с именем `local.by` (все действия выполняются на втором сервере анало-

гично установке службы DNS на первом сервере с отличием типа зоны прямого просмотра):

- откройте консоль DNS;
- выберите раздел *Primary Zone* (*Зоны прямого просмотра*);
- запустите мастер создания зоны (выберите: тип зоны – *Secondary Zone* (*Дополнительная зона*), IP-адрес master-сервера (с которого будет копироваться зона) – адрес сервера *server1*, остальные параметры – по умолчанию);
- введите имя зоны – local.by.

В итоге получим совместную работу DNS-серверов с реализацией функции резервирования.

4. Настройка узлов для выполнения динамической регистрации на сервере DNS

Для выполнения данной задачи нужно выполнить ряд действий как на серверах (если требуемые настройки не были выполнены ранее), так и в настройках клиента DNS. Рассмотрим пример настройки клиента с его регистрацией в DNS-сервере.

На сервере *DNS* должна быть создана соответствующая зона, а также разрешены динамические обновления.

На клиенте *DNS* необходимо сделать следующее:

- указать в настройках протокола TCP/IP адрес предпочтаемого DNS-сервера – тот сервер, на котором разрешены динамические обновления (в нашем примере – сервер с адресом 192.168.1.100);
- в полном имени компьютера указать соответствующий DNS-суффикс (в нашем примере – local.by). Для этого последовательно инициировать: *Мой компьютер – Свойства – закладка Имя компьютера – кнопка Изменить – Кнопка Дополнительно – в пустом текстовом поле вписать название домена local – кнопка OK (3 раза)*) (рис. 4.10).

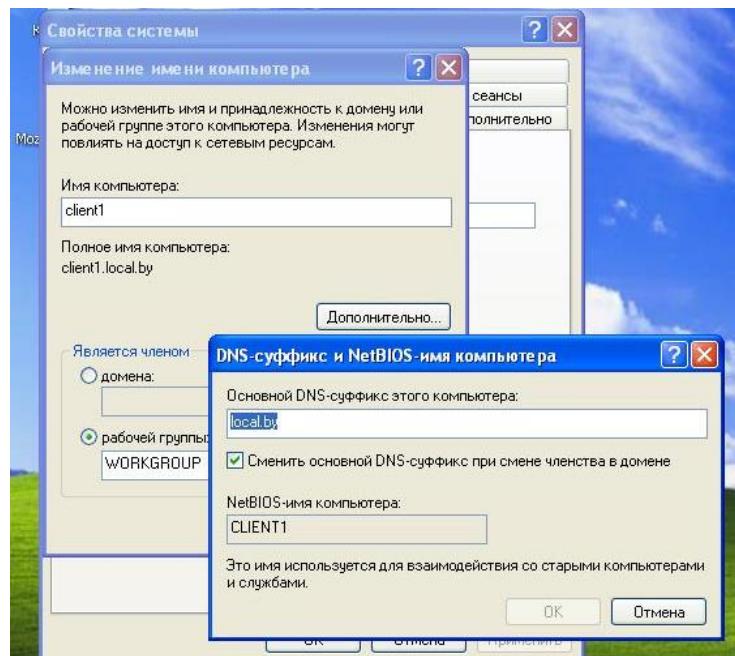


Рис. 4.10. Заполнение поля *DNS-суффикс* на клиенте

После этого система предложит перезагрузить компьютер. После выполнения перезагрузки на сервер DNS в зоне local.by автоматически создадутся записи типа A для наших серверов (рис. 4.11). В случае не создания записи для клиента (в нашем примере это client1) можно на стороне клиента в командной строке выполнить команду *ipconfig /registerdns*.

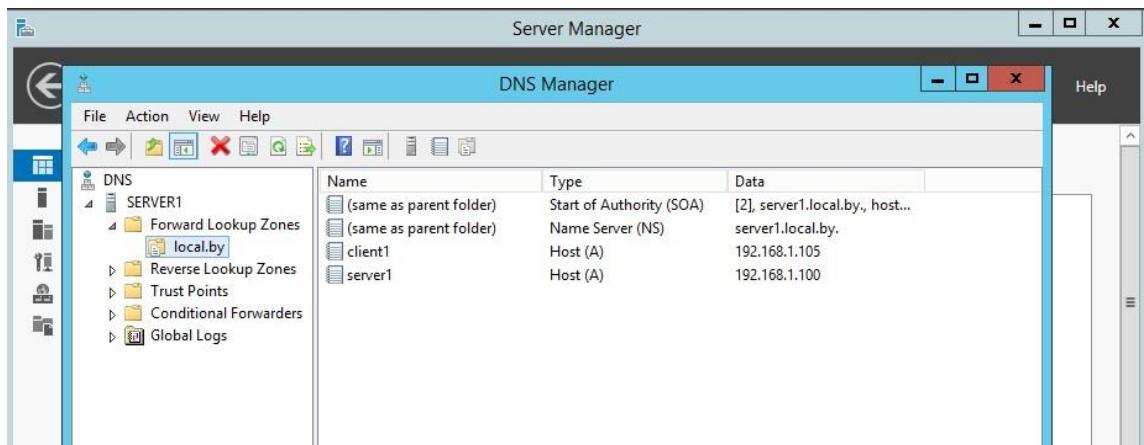


Рис. 4.11. Пример DNS-сервера с записями для клиента и сервера

Аналогичные операции необходимо выполнить на всех компьютерах сети.

Если автоматически записи не создались, то их можно создать вручную (рис. 4.12), однако при этом могут возникнуть сложности с автоматическим обновлением записей при изменении IP-адресов.

5. Создание зоны обратного просмотра

Выполняется по следующим шагам:

- откройте консоль DNS;
- выберите раздел *Reverse Lookup Zone* (*Зоны обратного просмотра*);
 - запустите мастер создания зоны (выберите: тип зоны – *Primary* (*Основная*), динамические обновления – разрешить, остальные параметры – по умолчанию) (рис. 4.13);
 - в поле *Код сети (ID)* введите параметры идентификатора сети – «192.168.1», а затем выполните команду принудительной регистрации компьютеров на сервере DNS – *ipconfig / registerdns*.

В итоге компьютеры зарегистрируются в обратной зоне DNS.

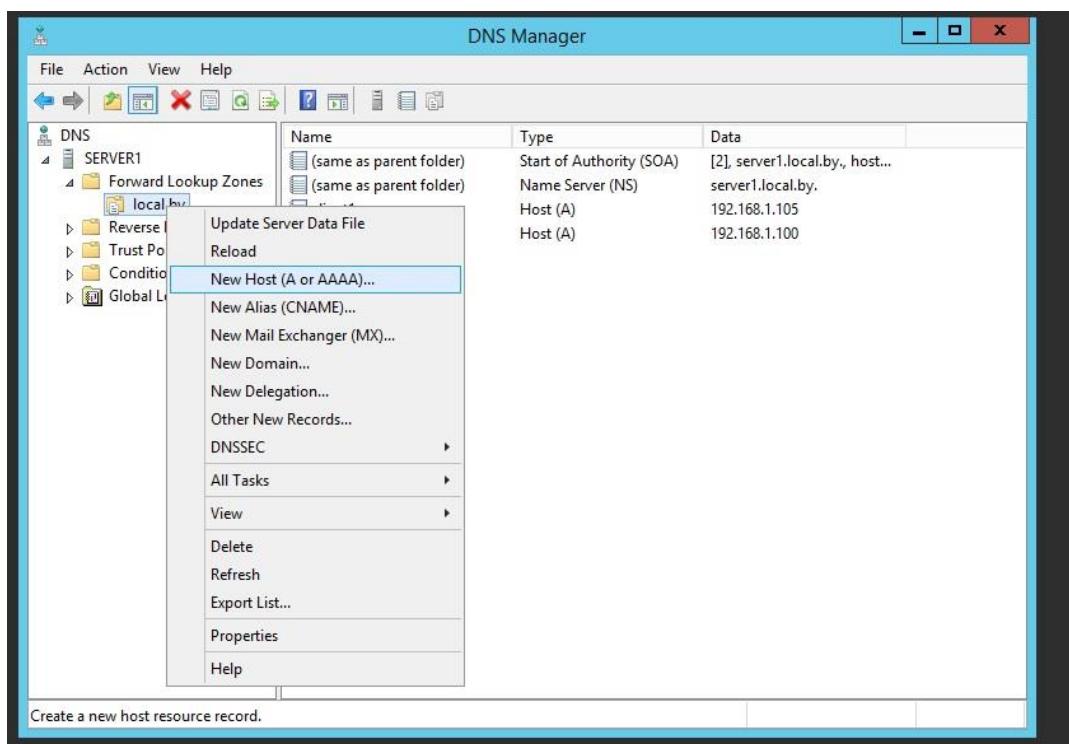


Рис. 4.12. Создание записи типа А на DNS-сервере вручную

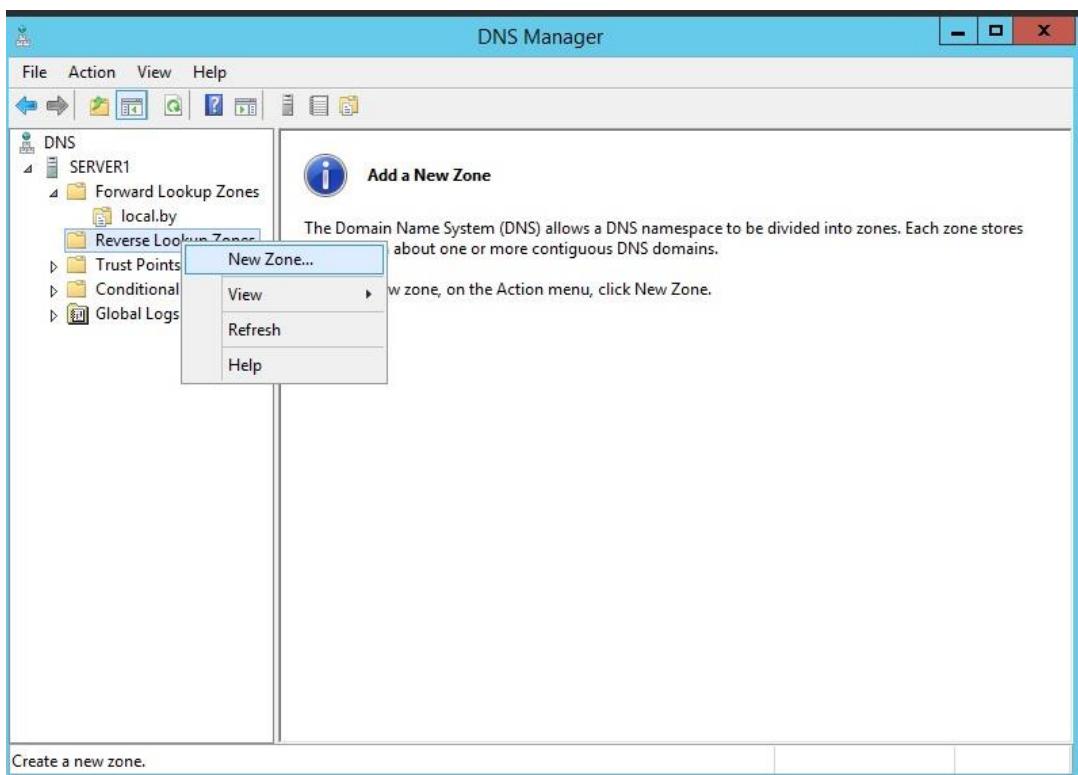


Рис. 4.13. Создание зоны обратного просмотра

4.4. Лабораторная работа № 6

Цель: изучение методов организации символьной адресации в клиент-серверной сети на базе операционных систем Windows с использованием DNS-сервера.

Задание: настройка в сети с клиент-серверной архитектурой, организованной при выполнении лабораторной работы № 4–5, DNS-сервера и регистрации DNS-клиентов. В качестве хостов должны выступать виртуальные операционные системы типа Windows с организованной динамической адресацией. DNS-сервер должен использовать статический адрес (согласно лабораторной работе № 4–5). Имена доменов выбрать согласно варианту (таблица).

Варианты заданий для лабораторной работы № 6

| № варианта | Наименование домена |
|------------|---------------------|
| 1 | ISiT1.by |
| 2 | ISiT2.by |
| 3 | ISiT3.by |

Окончание таблицы

| № варианта | Наименование домена |
|------------|---------------------|
| 4 | ISiT4.by |
| 5 | ISiT5.by |
| 6 | ISiT6.by |
| 7 | ISiT7.by |
| 8 | ISiT8.by |
| 9 | ISiT9.by |
| 10 | ISiT10.by |
| 11 | ISiT11.by |
| 12 | ISiT12.by |
| 13 | ISiT13.by |
| 14 | ISiT14.by |
| 15 | ISiT15.by |

5. ДИАГНОСТИКА TCP/IP И DNS

5.1. Утилиты диагностики TCP/IP и DNS

Любая операционная система имеет набор диагностических утилит для тестирования сетевых настроек и функционирования коммуникаций. Большой набор диагностических средств есть и в системах семейства Windows (как графических, так и в режиме командной строки).

Утилиты командной строки, являющиеся инструментами первой необходимости для проверки настроек протокола TCP/IP и работы сетей и коммуникаций, представлены в таблице. Подробное описание данных утилит содержится в системе интерактивной помощи Windows (вызывается нажатием кнопки *F1*). В таблице указаны основные и наиболее часто используемые параметры этих команд, а также дано их краткое описание.

Утилиты диагностики TCP/IP и DNS

| Название утилиты | Параметры | Комментарии |
|------------------|---|---|
| 1 | 2 | 3 |
| ipconfig | <code>/?</code> – Отобразить справку по команде <code>/all</code> – Отобразить полную информацию о настройке параметров всех адаптеров <code>/release</code> – Освободить динамическую IP-конфигурацию <code>/renew</code> – Обновить динамическую IP-конфигурацию с DHCP-сервера <code>/flushdns</code> – Очистить кэш разрешений DNS <code>/registerdns</code> – Обновить регистрацию на DNS-сервере <code>/displaydns</code> – Отобразить содержимое кэша разрешений DNS | Служит для отображения всех текущих параметров сети TCP/IP и обновления параметров DHCP и DNS. При вызове команды ipconfig без параметров выводятся IP-адрес, маска подсети и основной шлюз для каждого сетевого адаптера |
| arp | <code>-a</code> – Отображает текущие ARP-записи | Отображение и изменение ARP-таблиц |
| ping | Формат команды: «ping <сетевой узел> параметры» Параметры: <code>-t</code> – Бесконечная (до нажатия клавиш <code><Ctrl>+<Break></code>) отправка пакетов на указанный узел | Мощный инструмент диагностики (с помощью протокола ICMP). Команда ping позволяет проверить: работоспособность IP-соединения; правильность |

Продолжение таблицы

| 1 | 2 | 3 |
|-----------------|---|---|
| | <p>-a – Определение имени узла по IP-адресу</p> <p>-n <число> – Число отправляемых запросов</p> <p>-l <размер> – Размер буфера отправки</p> <p>-w <таймаут> – Таймаут ожидания каждого ответа в миллисекундах</p> | настройки протокола TCP/IP на узле; работоспособность маршрутизаторов; работоспособность системы разрешения имен FQDN или NetBIOS; доступность и работоспособность какого-либо сетевого ресурса |
| tracert | <p>-d – Без разрешения IP-адресов в имена узлов</p> <p>-h <максЧисло> – Максимальное число прыжков при поиске узла</p> <p>-w <таймаут> – Таймаут каждого ответа в миллисекундах</p> | Служебная программа для трассировки маршрутов, используемая для определения пути, по которому IP-дейтаграмма доставляется по месту назначения |
| pathping | <p>-n – Без разрешения IP-адресов в имена узлов</p> <p>-h максЧисло – Максимальное число прыжков при поиске узла</p> <p>-q <число_запросов> – Число запросов при каждом прыжке</p> <p>-w <таймаут> – Таймаут каждого ответа в миллисекундах</p> | Средство трассировки маршрута, сочетающее функции программ <i>ping</i> и <i>tracert</i> и обладающее дополнительными возможностями. Эта команда показывает степень потери пакетов на любом маршрутизаторе или канале, с ее помощью легко определить, какие маршрутизаторы или каналы вызывают неполадки в работе сети |
| netstat | <p>-a – Отображение <i>всех</i> подключений и ожидающих (слушающих) портов</p> <p>-n – Отображение адресов и номеров портов в числовом формате</p> <p>-o – Отображение кода (ID) процесса каждого подключения</p> <p>-r – Отображение содержимого локальной таблицы маршрутов</p> | Используется для отображения статистики протокола и текущих TCP/IP-соединений |
| nbtstat | <p>-n – Выводит имена пространства имен NetBIOS, зарегистрированные локальными процессами</p> <p>-c – Отображает кэш имен NetBIOS (разрешение NetBIOS-имен в IP-адреса)</p> <p>-R – Очищает кэш имен и перезагружает его из файла Lmhosts</p> | Средство диагностики разрешения имен NetBIOS |

Окончание таблицы

| 1 | 2 | 3 |
|-----------------|--|---|
| | -RR – Освобождает имена NetBIOS, зарегистрированные на WINS-сервере, а затем обновляет их регистрацию | |
| hostname | Ниаких ключей для данной утилиты не предусмотрено | Это самая простая утилита – она выводит на экран имя компьютера |

Рассмотрим примеры использования утилит командной строки для диагностики протокола TCP/IP и символьной адресации (DNS).

1. Использование команды *ipconfig* (без параметров и с параметром */all*) представлено на рис. 5.1.

Рис. 5.1. Использование команды *ipconfig*

2. Использование команды *arp*.

Пусть в сети только два узла (сервер DC1 и сервер DC2). Тогда в кэше сервера DC1 будет только одна запись – отображение IP-адреса сервера DC2 на MAC-адрес сетевого адаптера (рис. 5.2).

```
C:\>arp -a
Интерфейс: 192.168.0.1 --- 0x10003
  IP-адрес          Физический адрес      Тип
  192.168.0.2        00-03-ff-e7-14-f4    динамический
```

Рис. 5.2. Использование команды *arp*

3. Использование команды *ping*.

Существуют различные варианты использования данной утилиты (в сети существуют два компьютера с именами DC1 и DC2, настроена DNS-адресация):

- ping <IP-адрес> (рис. 5.3);

```
C:\>ping 192.168.0.2
Обмен пакетами с 192.168.0.2 по с 32 байт данных:

Ответ от 192.168.0.2: число байт=32 время<1мс TTL=128

Статистика Ping для 192.168.0.2:
  Пакетов: отправлено = 4, получено = 4, потеряно = 0
            (0% потеря)
Приблизительное время приема-передачи в мс:
  Минимальное = 0мсек, Максимальное = 0 мсек, Среднее = 0 мсек
```

Рис. 5.3. Использование команды *ping* с заданным IP-адресом

- ping <NetBIOS-имя узла>, когда в зоне сервера DNS нет записи для сервера DC2 (поиск IP-адреса производится широковещательным запросом) (рис. 5.4);

```
C:\>ping dc2
Обмен пакетами с dc2 [192.168.0.2] с 32 байт данных:

Ответ от 192.168.0.2: число байт=32 время=16мс TTL=128
Ответ от 192.168.0.2: число байт=32 время<1мс TTL=128
Ответ от 192.168.0.2: число байт=32 время<1мс TTL=128
Ответ от 192.168.0.2: число байт=32 время<1мс TTL=128

Статистика Ping для 192.168.0.2:
  Пакетов: отправлено = 4, получено = 4, потеряно = 0
            (0% потеря)
Приблизительное время приема-передачи в мс:
  Минимальное = 0мсек, Максимальное = 16 мсек, Среднее = 4 мсек
```

Рис. 5.4. Использование команды *ping* с заданным NetBIOS-именем узла
(с широковещательным запросом)

- ping <NetBIOS-имя узла>, когда в зоне сервера DNS есть запись для сервера DC2 (надо обратить внимание на подстановку клиентом DNS суффикса домена в запросе на имя узла, т. е. в команде используется краткое NetBIOS-имя сервера, а в статистике команды выводится полное имя) (рис. 5.5);

```
C:\>ping dc2
Обмен пакетами с dc2.world.ru [192.168.0.2] с 32 байт данных:
Ответ от 192.168.0.2: число байт=32 время=16мс TTL=128
Ответ от 192.168.0.2: число байт=32 время<1мс TTL=128
Ответ от 192.168.0.2: число байт=32 время<1мс TTL=128
Ответ от 192.168.0.2: число байт=32 время<1мс TTL=128
Статистика Ping для 192.168.0.2:
    Пакетов: отправлено = 4, получено = 4, потеряно = 0
              (0% потеря)
Приблизительное время приема-передачи в мс:
    Минимальное = 0мсек, Максимальное = 16 мсек, Среднее = 4 мсек
```

Рис. 5.5. Использование команды *ping* с заданным NetBIOS-именем узла (при условии существования записи на DNS-сервере)

- ping <FQDN-имя узла>, когда в зоне сервера DNS нет записи для сервера DC2 (узел DC2 не будет найден в сети) (рис. 5.6);

```
C:\>ping dc2.world.ru
При проверке связи не удалось обнаружить узел dc2.world.ru. Проверьте имя узла и повторите попытку.
```

Рис. 5.6. Использование команды *ping* с заданным FQDN-именем узла (при условии отсутствия записи на DNS-сервере)

- ping <FQDN-имя узла>, когда в зоне сервера DNS есть запись для сервера DC2 (узел успешно найден) (рис. 5.7);

```
C:\>ping dc2.world.ru
Обмен пакетами с dc2.world.ru [192.168.0.2] с 32 байт данных:
Ответ от 192.168.0.2: число байт=32 время=16мс TTL=128
Ответ от 192.168.0.2: число байт=32 время<1мс TTL=128
Ответ от 192.168.0.2: число байт=32 время<1мс TTL=128
Ответ от 192.168.0.2: число байт=32 время<1мс TTL=128
Статистика Ping для 192.168.0.2:
    Пакетов: отправлено = 4, получено = 4, потеряно = 0
              (0% потеря)
Приблизительное время приема-передачи в мс:
    Минимальное = 0мсек, Максимальное = 16 мсек, Среднее = 4 мсек
```

Рис. 5.7. Использование команды *ping* с заданным FQDN-именем узла (при условии существования записи на DNS-сервере)

- ping -a <IP-адрес> (обратное разрешение IP-адреса в имя узла) (рис. 5.8).

```
C:\>ping -a 192.168.0.2

Обмен пакетами с dc2.world.ru [192.168.0.2] с 32 байт данных:

Ответ от 192.168.0.2: число байт=32 время<1мс TTL=128

Статистика Ping для 192.168.0.2:
    Пакетов: отправлено = 4, получено = 4, потеряно = 0
              (0% потеря)
Приблизительное время приема-передачи в мс:
```

Рис. 5.8. Использование команды *ping* с обратным разрешением IP-адреса в имя узла

4. Использование команд *tracert* и *pathping*.

На рис. 5.9 и 5.10 приведены примеры трассировки маршрута до узла www.ru (если в вашем распоряжении только одна IP-сеть, то изучить работу данных команд будет невозможно).

```
C:\>tracert -d www.ru

Трассировка маршрута к www.ru [194.87.0.50]
с максимальным числом прыжков 30:

 1      17 ms      <1 мс      <1 мс      192.168.0.1
 2      1 ms       <1 мс      1 ms       217.1.1.33
 3      3 ms       3 ms       3 ms       217.1.10.1
 4      *           *           *           Превышен интервал ожидания для за-
проса.
 5      *           *           *           Превышен интервал ожидания для
запроса.
 6      10 ms      11 ms      10 ms      217.150.36.190
 7      *           *           *           Превышен интервал ожидания для
запроса.
 8      13 ms      13 ms      15 ms      194.87.0.83
 9      17 ms      12 ms      12 ms      194.87.0.50

Трассировка завершена.
```

Рис. 5.9. Использование команды *tracert*

```

C:\> pathping -n www.ru
Трассировка маршрута к www.ru [194.87.0.50]
с максимальным числом прыжков 30:
  0  192.168.0.1
  1  217.1.1.33
  2  217.1.10.1
  3  *           *
Подсчет статистики за: 100 сек. ...
      Исходный узел      Маршрутный узел
Прыжок   RTT    Утер./Отпр. %    Утер./Отпр. %    Адрес
  0     0мс      0/ 100 = 0%      0/ 100 = 0%  192.168.0.1
                                         0/ 100 = 0% |
  1     2мс      0/ 100 = 0%      0/ 100 = 0%  217.1.1.33
                                         0/ 100 = 0% |
  2     5мс      0/ 100 = 0%      0/ 100 = 0%  217.1.10.1
                                         100/ 100 =100% |
  3   ---      100/ 100 =100%      0/ 100 = 0%  0.0.0.0

Трассировка завершена.

```

Рис. 5.10. Использование команды *pathping*

Отметим, что в представленных примерах приведены лишь некоторые варианты использования команд. Отработку навыков их использования с иными ключами необходимо выполнить самостоятельно (при необходимости пользоваться внутренней справкой *команда/?*).

5.2. Разработка программы эхо-запроса для диагностики TCP/IP соединения

В сетевых протоколах существуют штатные средства, позволяющие выполнять диагностику функционирования системы. При использовании протокола TCP/IP таковым средством является протокол *ICMP* (*Internet Control Message Protocol*). Он описан в RFC792 и является частью уровня IP. Сообщения ICMP инкапсулируются в IP-дейтаграммы, так что они могут распространяться по TCP/IP-сетям.

Протокол ICMP используется для построения и поддержания в актуальном состоянии таблиц маршрутизации, определения параметра PMTU (Path Maximum Transmission Unit), диагностики сетевых проблем, осуществления контроля загруженности маршрутизаторов, исследования маршрутов передачи пакетов.

Существует возможность определения доступности и работоспособности сетевого устройства через *sockets* (необходимы права администратора). Однако в данном разделе мы будем использовать ICMP API. Для этого необходимы три функции. Первая из них (*IcmpCreateFile*) создает соединение, с которым мы собираемся рабо-

тать. Вторая закрывает его (`IcmpCloseHandle`), а третья посыпает через установленное соединение соответствующие данные (`IcmpSendEcho`).

Перед обращением к функциям ICMP следует указать следующие директивы для подключения библиотек, содержащих структуры и функции, предоставляемые Windows Sockets API (`ws2_32.lib`) и IP helper API (`iphlpapi.lib`) (подробнее о их назначении будет рассказано в следующих разделах):

```
.....  
#include <winsock2.h>  
#include <iphlpapi.h>  
#include <icmpapi.h>  
  
#pragma comment(lib, "iphlpapi.lib")  
#pragma comment(lib, "ws2_32.lib")
```

Обратите внимание, что директива `#include <iphlpapi.h>` должна предшествовать `#include <icmpapi.h>`.

Определите статусы ответов (полный перечень приведен в приложении):

```
#define IP_STATUS_BASE 11000  
#define IP_SUCCESS 0  
#define IP_DEST_NET_UNREACHABLE 11002  
#define IP_DEST_HOST_UNREACHABLE 11003  
#define IP_DEST_PROT_UNREACHABLE 11004  
#define IP_DEST_PORT_UNREACHABLE 11005  
#define IP_REQ_TIMED_OUT 11010  
#define IP_BAD_REQ 11011  
#define IP_BAD_ROUTE 11012  
#define IP_TTL_EXPIRED_TRANSIT 11013
```

Напишите функцию `Ping`. Первый параметр – адрес хоста, который будет пинговаться, второй – время ожидания ответа (в мс) после запроса, третий – количество посыпаемых запросов:

```
void Ping ( const char* cHost,  
            unsigned int Timeout,  
            unsigned int RequestCount) {  
    //  
}
```

Первым шагом должно стать получение манипулятора для выполнения ICMP-запросов. Возвращается манипулятор `hIP` функцией `IcmpCreateFile` без параметров:

```

// Создать файл сервиса
    HANDLE hIP = IcmpCreateFile();
    if (hIP == INVALID_HANDLE_VALUE) {
        WSACleanup();
        return;
}

```

Код ошибки может быть извлечен вызовом функции `WSAGetLastError` (основные коды ошибок перечислены в приложении).

Выполняем подготовительные действия. Определяем данные эхо-запроса в переменной `SendData`. Задаем буфер для эхо-ответа `pIpe` и выделяем для него память. Определяем переменные для статистики `LostPacketsCount`, `MaxMS`, `MinMS`. Определяем размер буфера эхо-ответа как сумму размера буфера эхо-запроса и размера структуры `ICMP_ECHO_REPLY`, возвращаемой в ответ на эхо-запрос, и выделяем из глобальной кучи память данного размера. Сохраняем идентификатор выделенного блока глобальной памяти (`pIpe`). В случае ошибки вызываем функцию `IcmpCloseHandle(hIP)` – закрывает ICMP и функцию `WSACleanup()` – завершает использование Winsock 2 DLL (`Ws2_32.dll`).

IP-адрес опрашиваемого узла `cHost` преобразуем к типу `unsigned long` при помощи функции `inet_addr`.

```

char SendData[32] = "Data for ping";//буфер для передачи
int LostPacketsCount = 0; // кол-во потерянных пакетов
unsigned int MaxMS = 0;// максимальное время ответа (мс)
int MinMS = -1; // минимальное время ответа (мс)

// Выделяем память под пакет – буфер ответа
PICMP_ECHO_REPLY pIpe =
    (PICMP_ECHO_REPLY)GlobalAlloc(GHND,
                                sizeof(ICMP_ECHO_REPLY)
                                + sizeof(SendData));
if (pIpe == 0) {
    IcmpCloseHandle(hIP);
    WSACleanup();
    return;
}
pIpe->Data = SendData;
pIpe->DataSize = sizeof(SendData);
unsigned long ipaddr = inet_addr(cHost);

```

Также необходимо настроить опции для включения в заголовок IP пакета. Например:

```
IP_OPTION_INFORMATION option = { 255, 0, 0, 0, 0 };
```

Здесь структура `IP_OPTION_INFORMATION` (для 64-разрядных систем используется `IP_OPTION_INFORMATION32`) имеет следующие параметры:

```
typedef struct {
    unsigned char Ttl;           // время доставки
    unsigned char Tos;          // тип сервиса
    unsigned char Flags;         // флаги IP заголовка
    unsigned char OptionsSize;   // размер опций в байтах
    unsigned char *OptionsData;  // указатель на опции
} IP_OPTION_INFORMATION, *PMP_OPTION_INFORMATION;
```

Третий шаг – приступаем к отправке эхо-запросов. Для этих целей служит функция `IcmpSendEcho` со следующим прототипом:

```
DWORD IcmpSendEcho
(
    __in     HANDLE IcmpHandle,           // дискриптор полученный
                                            // функцией IcmpCreateFile
    __in     IPAddr DestinationAddress,  // IP адрес запроса
    __in     LPVOID RequestData,        // указатель на буфер эхо-запроса
    __in     WORD RequestSize,          // размер буфера эхо-запроса
    __in     PIP_OPTION_INFORMATION RequestOptions, // указатель на структуру с опциями запроса
    __inout  LPVOID ReplyBuffer,        // указатель на буфер эхо-ответа
    __in     DWORD ReplySize,           // размер буфера эхо-ответа
    __in     DWORD Timeout            // таймаут в миллисекундах
);
```

Она отправляет ICMP эхо-запрос по указанному IP-адресу и возвращает любые ответы, полученные в пределах заданного тайм-аута (Timeout) и до исчерпания пространства буфера. Эта функция синхронна (то есть приостанавливает выполнение процесса до завершения своей работы), и во избежание блокировки процесс должен перед ее вызовом породить соответствующую нить. Функция `IcmpSendEcho` использует для передачи IPv4 ICMP. Существуют еще версии функций: `IcmpSendEcho2Ex` и `IcmpSendEcho2`. Для IPv6 используйте функции `Icmp6CreateFile` и `Icmp6SendEcho2`.

```
for (unsigned int c = 0; c < RequestCount; c++) {  
    int dwStatus = IcmpSendEcho( hIP,  
                                ipaddr,  
                                SendData,  
                                sizeof(SendData),  
                                &option,  
                                pIpe,  
                                sizeof(ICMP_ECHO_REPLY) +  
                                sizeof(SendData),  
                                Timeout);  
    if (dwStatus > 0) {  
        for (int i = 0;i< dwStatus;i++) {  
            unsigned char* pIpPtr =  
                (unsigned char*)&pIpe->Address;  
            cout << "Ответ от " << (int)*pIpPtr  
            << "." << (int)*(pIpPtr + 1)  
            << "." << (int)*(pIpPtr + 2)  
            << "." << (int)*(pIpPtr + 3)  
            << ":" число байт = " << pIpe->DataSize  
            << " время = " << pIpe->RoundTripTime  
            << "mc TTL = " << (int)pIpe->Options.Ttl;  
            MaxMS = (MaxMS > pIpe->RoundTripTime) ?  
                MaxMS : pIpe->RoundTripTime;  
            MinMS = (MinMS < (int)pIpe->RoundTripTime  
                && MinMS >= 0) ?  
                MinMS : pIpe->RoundTripTime;  
            cout << endl;  
        }  
    }  
}
```

Первым параметром функции `IcmpSendEcho` служит манипулятор, полученный на первом шаге инициализации `hIP`. Вторым – IP-

адрес опрашиваемого узла `ipaddr`. Третьим и четвертым параметрами являются указатель на отправляемые в эхо-запросе данные `SendData` и размер этих данных соответственно `sizeof(SendData)`. Пятый параметр является указателем `option` на структуру, содержащую дополнительные опции запроса (в том числе и `TTL – Time to live` – время жизни пакета). Его можно установить в `NULL`. Шестой и седьмой параметры описывают соответственно указатель на буфер для эхо-ответов `pIpe` и размер этого буфера. Тайм-аут указывается восьмым параметром (`Timeout`) и является входным параметром функции `Ping`.

Возвращаемое функцией `IcmpSendEcho` значение `dwStatus` представляет собой количество полученных эхо-ответов. По завершении работы функции `IcmpSendEcho` параметр `pIpe`, буфер ответа, будет заполнен массивом структур типа `ICMP_ECHO_REPLY`, за которым последуют опции и данные ответов. Он должен быть достаточно большим, чтобы разместить хотя бы одну структуру `ICMP_ECHO_REPLY` (для 64-разрядных – `ICMP_ECHO_REPLY32`) плюс `Max (sizeof(SendData), 8)` байт, так как сообщение об ошибке ICMP занимает 8 байт. Структура `ICMP_ECHO_REPLY` описывается следующим образом:

```
typedef struct icmp_echo_reply
{
    IPAddr          Address;           // адрес ответившего узла
    ULONG           Status;            // статус ответа
    ULONG           RoundTripTime;     // время прохождения запроса в мс
    USHORT          DataSize;          // размер данных ответа
    USHORT          Reserved;          // зарезервировано
    PVOID           Data;              // указатель на данные ответа
    Struct          IP_OPTION_INFORMATION Options;
                           // опции ответа
} ICMP_ECHO_REPLY, *PICMP_ECHO_REPLY;
```

Обращаясь к полям структуры, выводим на консоль IP-адрес, время, прошедшее с момента отправки эхо-запроса до получения эхо-ответа (RTT), число переданных байт и т. п. Фиксируем максимальное и минимальное время.

В случае ошибки считаем число потерянных пакетов и выводим код ошибки:

```
else {
    if (pIpe->Status) {
        LostPacketsCount++;

        switch (pIpe->Status) {
            case IP_DEST_NET_UNREACHABLE:
            case IP_DEST_HOST_UNREACHABLE:
            case IP_DEST_PROT_UNREACHABLE:
            case IP_DEST_PORT_UNREACHABLE:
                cout<<"Remote host may be down."<<endl;
                break;
            case IP_REQ_TIMED_OUT:
                cout<< "Request timed out."<<endl;
                break;
            case IP_TTL_EXPIRED_TRANSIT:
                cout<< "TTL expired in transit."<<endl;
                break;
            default:
                cout<< "Error code: %ld"
                    << pIpe->Status<<endl;
                break;
        }
    }
}
```

Для корректного завершения работы с ICMP-функциями с помощью вызова `IcmpCloseHandle()` освобождается ICMP-манипулятор:

```
IcmpCloseHandle(hIP);
WSACleanup();
```

В заключении можно вывести статистику:

```
if (MinMS<0) MinMS = 0;
unsigned char* pByte = (unsigned char*)&pIpe->Address;
cout<<"Статистика Ping      "
     << (int)*(pByte)
     << "." << (int)*(pByte + 1)
     << "." << (int)*(pByte + 2)
     << "." << (int)*(pByte + 3)<<endl;
```

```

cout << "\tПакетов: отправлено = "<< RequestCount
    << ", получено = "
    << RequestCount - LostPacketsCount
    << ", потеряно = " << LostPacketsCount
    << "<" <<(int)(100/(float)RequestCount)*
        LostPacketsCount
    << " % потерь>, "<< endl;

cout<<"Приблизительное время приема-передачи:"
    << endl << "Минимальное = " << MinMS
    << "мс, Максимальное = " << MaxMS
    << "мс, Среднее = " << (MaxMS + MinMS) / 2
    << "мс" << endl;

```

Вызов функции выглядит следующим образом:

```

int main(int argc, char **argv)
{
    setlocale(LC_ALL, "RUS");
    Ping("81.19.70.1", 60, 10);
    getch();
    return 0;
}

```

Результат работы программы представлен на рис. 5.11.

```

Ответ от 81.19.70.1: число байт = 32 время = 35мс TTL = 55
Ответ от 81.19.70.1: число байт = 32 время = 35мс TTL = 55
Ответ от 81.19.70.1: число байт = 32 время = 39мс TTL = 55
Ответ от 81.19.70.1: число байт = 32 время = 37мс TTL = 55
Ответ от 81.19.70.1: число байт = 32 время = 37мс TTL = 55
Ответ от 81.19.70.1: число байт = 32 время = 37мс TTL = 55
Ответ от 81.19.70.1: число байт = 32 время = 38мс TTL = 55
Ответ от 81.19.70.1: число байт = 32 время = 37мс TTL = 55
Ответ от 81.19.70.1: число байт = 32 время = 38мс TTL = 55
Request timed out.
Статистика Ping 0.0.0.0
Пакетов: отправлено = 10, получено = 9, потеряно = 1<10 % потерь>,
Приблизительное время приема-передачи:
Минимальное = 35мс, Максимальное = 39мс, Среднее = 37мс

```

Рис. 5.11. Результат работы программы

5.3. Лабораторная работа № 7

Цель: изучение методов диагностики TCP/IP сетей.

Задание:

1. Изучение использования команд *ipconfig*, *ping*, *tracert*, *pathping*, *arp*, *nbtstat* с использованием различных ключей в клиент-серверной сети с применением динамической адресацией и DNS-сервером. Приобретение навыков определения символьных и физических адресов удаленных компьютеров по известному IP-адресу.

2. Разработка программы, реализующей функции эхозапроса; проверка ее работы в клиент-серверной сети, настроенной в предыдущих лабораторных работах.

Поменяйте параметры *IP_OPTION_INFORMATION option*.

Переделайте программу таким образом, чтобы параметры функции Ping считывались через аргументы командной строки. Продемонстрируйте выполнение программы.

6. АДРЕСАЦИЯ В СЛОЖНОСТРУКТУРИРОВАННЫХ СЕТЯХ

Под сложноструктурированной сетью будем понимать несколько логических сегментов сети со своей адресацией, соединенных оборудованием с элементами коммутации и маршрутизации. Важнейшую роль при взаимодействии хостов в подобной сети играет определение *идентификатора подсети (Network ID)* и *идентификатора хоста (Host ID)*.

Для определения того, какая часть IP-адреса отвечает за ID подсети, а какая за ID хоста, применяются два способа: с помощью классов и с помощью масок.

6.1. Классы IP-адресов

Существует пять классов IP-адресов: *A*, *B*, *C*, *D* и *E* (рис. 6.1).

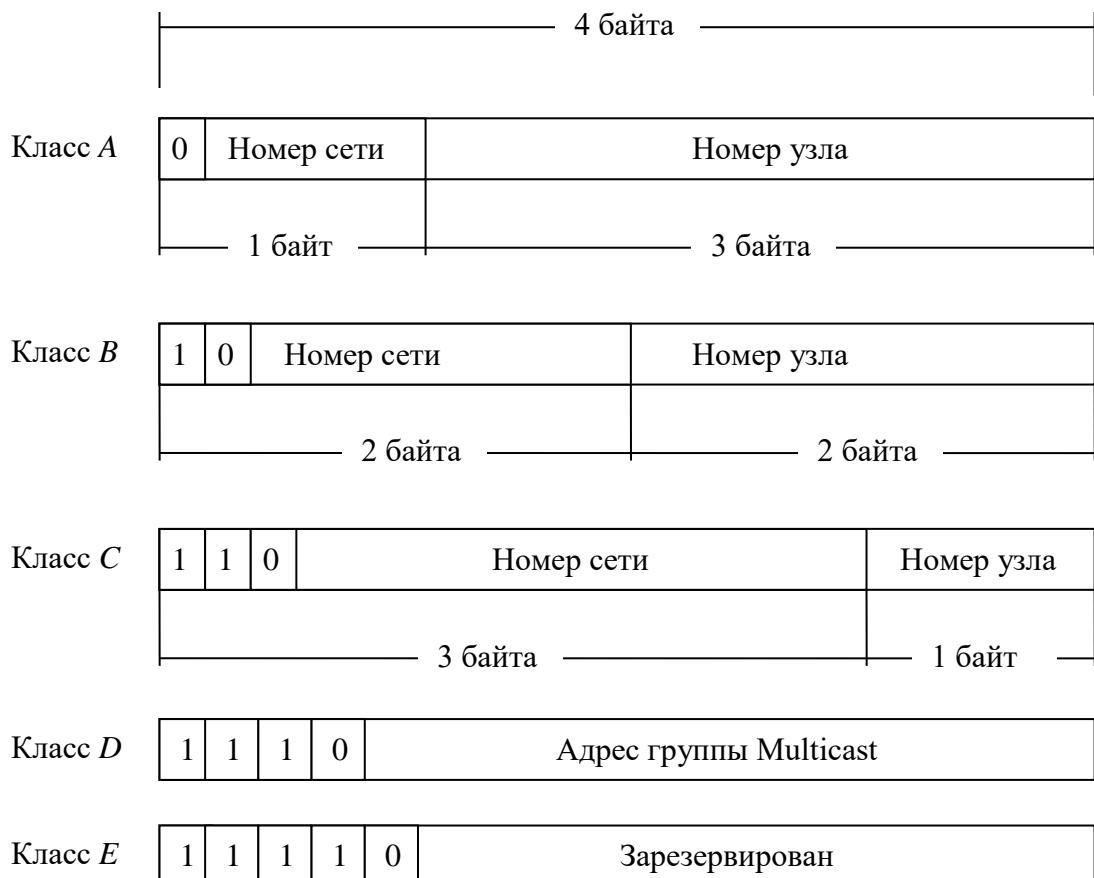


Рис. 6.1. Классы IP-адресов

За принадлежность к тому или иному классу отвечают первые биты IP-адреса. Деление сетей на классы описано в RFC 791 (документ описания протокола IP).

Целью такого деления являлось создание малого числа больших сетей (*класс A*), умеренного числа средних сетей (*класс B*) и большого числа малых сетей (*класс C*).

Если адрес начинается с 0, то сеть относят к *классу A* и номер сети занимает один байт, остальные 3 байта интерпретируются как номер узла в сети. Сети класса А имеют номера в диапазоне от 1 до 126. Сетей класса А немного, зато количество узлов в них может достигать $2^{24} - 2$, то есть 16 777 214 узлов.

Если первые два бита адреса равны 10, то сеть относится к *классу B*. В сетях класса В под номер сети и под номер узла отводится по 16 бит, то есть по 2 байта. Таким образом, сеть класса В является сетью средних размеров с максимальным числом узлов $2^{16} - 2$, что составляет 65 534 узлов.

Если адрес начинается с последовательности 110, то это сеть *класса C*. В этом случае под номер сети отводится 24 бита, а под номер узла – 8 бит. Сети этого класса наиболее распространены, число узлов в них ограничено $2^8 - 2$, то есть 254 узлами.

Адрес, начинающийся с 1110, обозначает особый, *групповой адрес* (multicast). Пакет с таким адресом направляется всем узлам, которым присвоен данный адрес.

Адреса *класса E* в настоящее время не используются (зарезервированы для будущих применений).

Характеристики адресов разных классов представлены в табл. 6.1.

Таблица 6.1

Характеристики IP-адресов разных классов

| Класс | Первые биты | Наименьший номер сети | Наибольший номер сети | Количество сетей | Максимальное число узлов в сети |
|-------|-------------|-----------------------|-----------------------|------------------|---------------------------------|
| A | 0 | 1.0.0.0 | 126.0.0.0 | 126 | $2^{24} - 2 = 16777214$ |
| B | 10 | 128.0.0.0 | 191.255.0.0 | 16384 | $2^{16} - 2 = 65534$ |
| C | 110 | 192.0.0.0 | 223.255.255.0 | 2097152 | $2^8 - 2 = 254$ |
| D | 1110 | 224.0.0.0 | 239.255.255.255 | Групповой адрес | |
| E | 11110 | 240.0.0.0 | 247.255.255.255 | Зарезервирован | |

Применение классов в целом характеризуется неэффективностью распределения IP-адресов. Например, если организации требуется ты-

сяча IP-адресов, ей выделяется сеть класса *B*, при этом 64 534 адреса не будут использоваться.

Существует два основных способа решения этой проблемы:

- более эффективная схема деления на подсети с использованием масок (RFC 950);
- применение протокола IP версии 6 (IPv6).

6.2. Использование масок

Маска подсети (subnet mask) – это число, которое используется в паре с IP-адресом; двоичная запись маски содержит единицы в тех разрядах, которые должны в IP-адресе интерпретироваться как номер сети.

Для стандартных классов сетей маски имеют следующие значения:

- *класс A* – 11111111.00000000.00000000.00000000 (255.0.0.0);
- *класс B* – 11111111.11111111.00000000.00000000 (255.255.0.0);
- *класс C* – 11111111.11111111.11111111.00000000 (255.255.255.0).

Маска подсети записывается либо в виде, аналогичном записи IP-адреса, например, 255.255.255.0, либо совместно с IP-адресом с помощью указания числа единичных разрядов в записи маски, например 192.168.1.1/24, т. е. в маске содержится 24 единицы (255.255.255.0).

При использовании масок можно вообще отказаться от понятия классов.

Пример 1. Пусть задан IP-адрес 17.239.47.94, маска подсети 255.255.0.0 (другая форма записи: 17.239.47.94/16).

Требуется определить ID подсети и ID хоста в обеих схемах адресации.

1) *Адресация с использованием классов.* Двоичная запись IP-адреса имеет вид:

00010001.11101111.00101111.01011110.

Так как первый бит равен нулю, адрес относится к *классу A*. Следовательно, первый байт отвечает за ID подсети, остальные три байта – за ID хоста:

ID подсети: 17.0.0.0. ID хоста: 0.239.47.94.

2) *Адресация с использованием масок.* Запишем IP-адрес и маску подсети в двоичном виде:

IP-address: 17.239.47.94 = 00010001.11101111.00101111.01011110 ,
Subnet mask: 255.255.0.0 = 11111111.11111111.00000000.00000000.

Вспомнив определение маски подсети, можно интерпретировать номер подсети как те биты, которые в маске равны 1, т. е. первые два байта. Оставшаяся часть IP-адреса будет номером узла в данной подсети.

ID подсети: 17.239.0.0. ID хоста: 0.0.47.94.

Номер подсети можно получить другим способом, применив к IP-адресу и маске операцию логического умножения или *конъюнкции* (AND):

$$\begin{array}{r} \text{AND} \quad 00010001. \underline{11101111. 00101111. 01011110} \\ \underline{11111111. 11111111. 00000000. 00000000} . \\ 00010001. 11101111. 00000000. 00000000 \\ 17 \qquad 239 \qquad 0 \qquad 0 \end{array}$$

В масках количество единиц в последовательности, определяющей границу номера сети, не обязательно должно быть кратным 8.

Пример 2. Задан IP-адрес 192.168.89.16, маска подсети – 255.255.192.0 (другая форма записи: 192.168.89.16/18).

Требуется определить ID подсети и ID хоста. Воспользуемся операцией AND:

$$\begin{array}{l} \text{IP-address: } 192.168.89.16 = \text{AND } 11000000.10101000.01011001.00010000 \\ \text{Subnet mask: } 255.255.192.0 = \underline{11111111.11111111.11000000.00000000} . \\ \text{subnet ID: } \qquad \qquad \qquad 11000000.10101000.01000000.00000000 \\ \qquad \qquad \qquad 192 \qquad 168 \qquad 64 \qquad 0 \end{array}$$

Чтобы получить номер узла, нужно в битах, отвечающих за номер подсети, поставить нули:

Host ID: 00000000.00000000.00011001.00010000 = 0.0.25.16.

Ответ: ID подсети = 192.168.64.0, ID хоста = 0.0.25.16.

Для масок существует важное правило: разрывы в последовательности единиц или нулей недопустимы.

Например, не существует маски подсети, имеющей следующий вид:

1111111.1110111.00000000.00001000 (255.247.0.8),

так как последовательности единиц и нулей не являются непрерывными.

С помощью масок администратор может структурировать свою сеть, не требуя от поставщика услуг дополнительных номеров сетей.

Допустим, организации выделена сеть класса B : 160.95.0.0 (рис. 6.2).

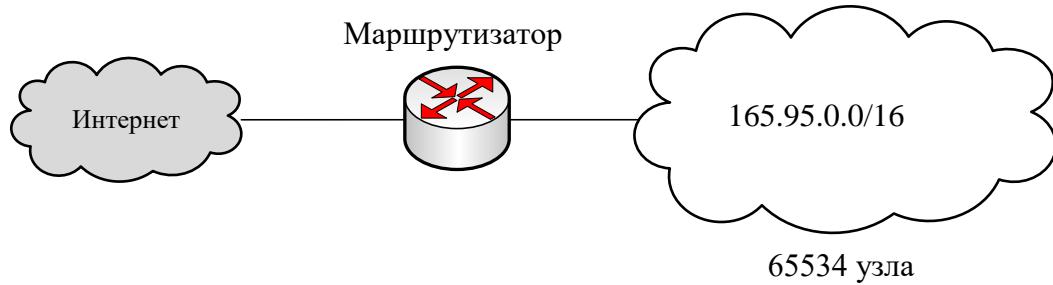


Рис. 6.2. Сеть класса B до деления на подсети

В такой сети может находиться до 65 534 узлов. Однако организации требуется 3 независимые сети с числом узлов в каждой не более 254. В этой ситуации можно применить деление на подсети с помощью масок. Например, при использовании маски 255.255.255.0 третий байт адреса будет определять номер внутренней подсети, а четвертый байт – номер узла (рис. 6.3).

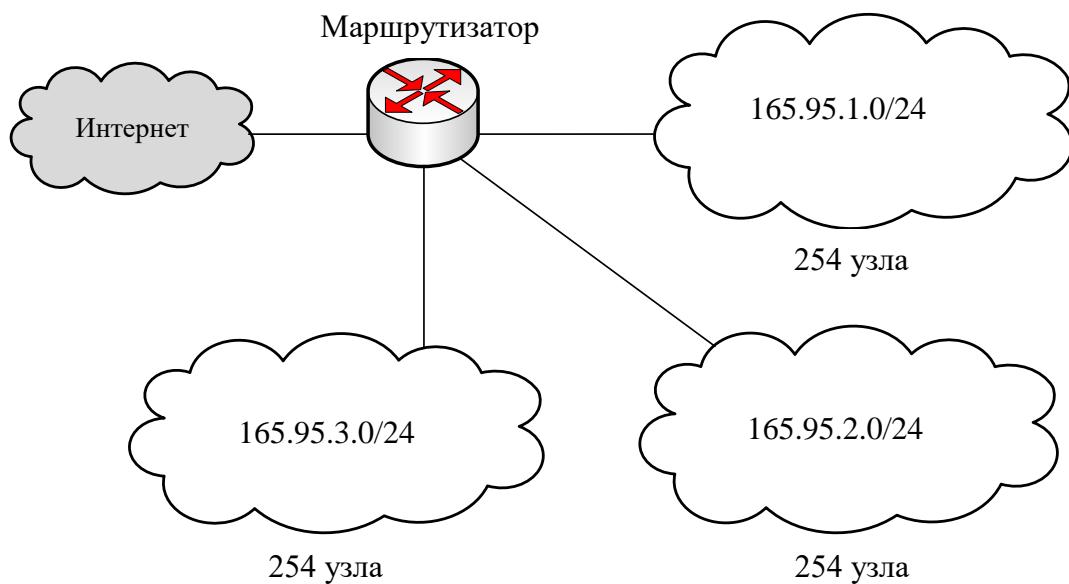


Рис. 6.3. Сеть класса B после деления на подсети

Маршрутизаторы во внешней сети (Интернет) ничего «не знают» о делении сети 160.95.0.0 на подсети, все пакеты направляются на маршрутизатор организации, который переправляет их в требуемую внутреннюю подсеть.

6.3. Особые IP-адреса

Некоторые IP-адреса являются особыми, они не должны применяться для идентификации обычных сетей.

1. Если первый октет ID сети начинается с 127, такой адрес считается адресом машины-источника пакета. В этом случае пакет не выходит в сеть, а возвращается на компьютер-отправитель. Такие адреса называются **loopback** («петля», «замыкание на себя») и используются для проверки функционирования стека TCP/IP.

2. Если все биты IP-адреса равны нулю, адрес обозначает узел- отправитель и используется в некоторых сообщениях ICMP.

3. Если все биты адреса равны 1, адрес называется **ограниченным широковещательным** (limited broadcast). Пакеты, направленные по такому адресу, рассылаются всем узлам той подсети, в которой находится отправитель пакета. Также возможна другая форма записи ограниченного широковещательно адреса – все биты ID хоста равны 1, а все биты ID-сети равны 0.

4. Если все биты ID хоста равны 1 (при этом все биты ID-сети не равны 0, т. е. задается определенная сеть, а не сеть отправителя), адрес называется **широковещательным** (broadcast); пакеты, имеющие широковещательный адрес, доставляются всем узлам подсети назначения.

5. Если все биты ID хоста равны 0, адрес считается **идентификатором подсети** (subnet ID).

Наличие особых IP-адресов объясняет, почему из диапазона доступных адресов исключаются два адреса – это случаи, когда все биты ID хоста равны 1 или 0. Например, в сети *класса С* не 256, а 254 узлов.

6.4. Разработка программы определения ID подсети и ID хоста по заданному IP-адресу и маске подсети

Входными данными программы являются IP-адрес (переменная типа `unsigned long ip;`) и маска подсети (`unsigned long mask;`). Необходимо определить ID подсети (`unsigned long subnet;`) и ID хоста (`unsigned long host;`).

Вспомнив определение маски, можно интерпретировать номер подсети как те биты, которые в маске равны 1. Тогда ID подсети можно получить, применив к IP-адресу и маске операцию логического умножения, или конъюнкции (AND):

```
subnet = ip & mask;
```

Чтобы получить ID хоста, нужно в битах, отвечающих за номер подсети, поставить нули:

```
host = ip & ~mask;
```

Чтобы получить broadcast-адрес, необходимо выполнить:

```
broadcast = ip & mask | ~mask;
```

Пусть задан IP-адрес 192.168.89.16, маска подсети 255.255.0.0.

192.168.89.16 = 11000000.10101000.01011001.00010000,
255.255.0.0 = 11111111.11111111.00000000.00000000 .

Получим ID подсети: 192.168.0.0. ID хоста: 0.0.89.16 Broadcast: 192.168.255.255 (рис. 6.4)

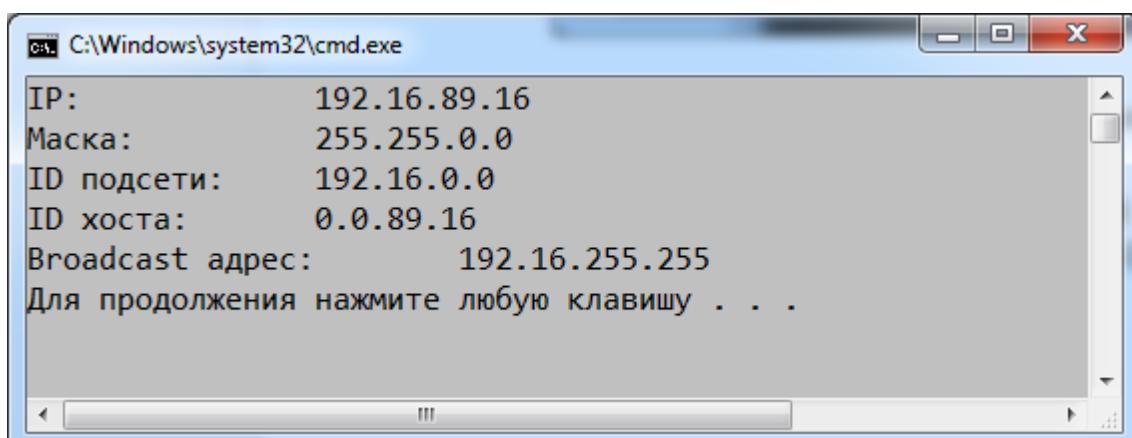


Рис. 6.4. Результат выполнения программы

В ходе выполнения задания необходимо проверить пределы допустимого диапазона октета: от 0 до 255, что и выполняет функция CheckAddress:

```
bool CheckAddress(char* ip_)
{
    int points=0,                      // количество точек
        numbers=0;                     // значение октета
    char* buff;                        // буфер для одного октета
    buff = new char[3];
```

```

for(int i=0;ip_[i]!='\0';i++)
{
    // для строки IP-адреса
    if(ip_[i]<='9'&& ip_[i]>='0') // если цифра
    {
        if(numbers>3) return false;
        //если больше трех чисел в октете - ошибка
        buff[numbers++]=ip_[i];
        //скопировать в буфер
    }
    else
        if(ip_[i]=='.') // если точка
        {
            if(atoi(buff)>255)
                // проверить диапазон октета
                return false;
            if(numbers==0)
                //если числа нет - ошибка
                return false;
            numbers=0;
            points++;
            delete[]buff;
            buff = new char[3];
        }
        else return false;
    }

    if(points!=3)
    // если количество точек в IP-адресе не 3 - ошибка
    return false;
    if(numbers==0||numbers>3)
        return false;
    return true;
}

```

Затем необходимо выполнить преобразование строки IP-адреса (16 символов) в десятично-точечной нотации, содержащей 4 группы по 3 символа (октета), разделенных точками, в число `unsigned long`. Это можно выполнить, написав собственную функцию проверки и преобразования `CharToULong`:

```

ul CharToLong(char* ip_)
{
    ul out=0;//число для IP-адреса
    char *buff;
    buff=new char[3];
    //буфер для хранения одного октета

```

```

for(int i=0, j=0, k=0; ip_[i]!='\0'; i++, j++)
{
    if(ip_[i]!='.')          //если не точка
        buff[j]=ip_[i]; // записать символ в буфер
    if(ip_[i]=='.'||ip_[i+1]=='\0')
    {
        // если следующий октет или последний
        out<<=8;           //сдвинуть число на 8 бит
        if(atoi(buff)>255)
            return NULL;
        // если октет больше 255 -
ошибка
        out += (ul)atoi(buff);
        //преобразовать и добавить
        //к числу IP-адреса
        k++;
        j=-1;
        delete[]buff;
        buff=new char[3];
    }
}
return out;
}

```

Аналогично для маски следует проверить пределы допустимого диапазона октета: от 0 до 255 и выполнить преобразование в число. После чего надо проверить корректность задания маски подсети.

```

ul ip,mask,host,subnet, broadcast;
char *ip_*,*mask_;
bool flag=true;
ip_=new char[16];
mask_=new char[16];

do
{
    if(!flag) cout<<"Неверно введён адрес!"<<endl;
    cout<<"IP: ";
    cin>>ip_;
}while(!(flag=CheckAddress(ip_)));

```

```

ip=CharToLong(ip_);
flag=true;

do
{
    if(!flag) cout<<"Неправильная маска!"<<endl;
    flag=true;
    do
    {
        if(!flag) cout<<"Неверно введена маска!"<<endl;
        cout<<"Маска: ";
        cin>>mask_;
    }while(!(flag=CheckAddress(mask_)));

    mask=CharToLong(mask_);
}while(!(flag=CheckMask(mask_)));

```

Выполнить вычисления ID подсети, ID хоста; преобразовать их к форматированной строке и вывести результат (рис. 6.4).

6.5. Лабораторная работа № 8

Цель: изучение методов определения идентификатора сети и идентификатора хоста.

Задание: разработка программы, выполняющей по введенному IP-адресу и маске определение Network ID и Host ID.

Дополнительные задания:

1. Проверка маски на непрерывность единиц.
2. Проверка правильности IP-адреса (четыре октета чисел от 0 до 255, разделенных точкой) и маски на разрешенные символы, на длину (не более 32 бит), количество октетов и т. д.

В табл. 6.2 представлены примеры корректных и некорректных IP-адресов и масок.

Таблица 6.2
Примеры правильных и неправильных IP-адресов и масок

| | |
|------------------------|--|
| Правильные IP-адреса | 172.16.192.1 192.168.0.1 10.0.0.1 |
| Неправильные IP-адреса | 256.0.0.1 1.1.1.1.1 1.0.1 1:1:1:1 |

Окончание табл. 6.2

| | |
|--------------------|---|
| Правильная маска | 255.0.0.0 255.240.0.0 255.255.128.0 255.255.255.240 |
| Неправильная маска | 0.255.0.0 255.0.255.0 255.240.255.0 1.1.1.1 0.0.0.128 |

3. Расчет широковещательного (broadcast) адреса по известному IP-адресу и маске.

7. РАЗРЕШЕНИЕ АДРЕСОВ В IP-СЕТЯХ

7.1. Определение MAC-адреса удаленного хоста по IP-адресу

7.1.1. Физический адрес

Физический, или локальный адрес узла определяется технологией, с помощью которой построена сеть, в которую входит узел. Для узлов, входящих в локальные сети, это MAC-адрес сетевого адаптера или порта маршрутизатора.

В качестве стандартного выбран 48-битный формат адреса, что соответствует примерно 280 триллионам различных адресов. Понятно, что столько сетевых адаптеров никогда не будет выпущено.

С тем, чтобы распределить возможные диапазоны адресов между многочисленными изготовителями сетевых адаптеров, была предложена следующая структура адреса (рис. 7.1).

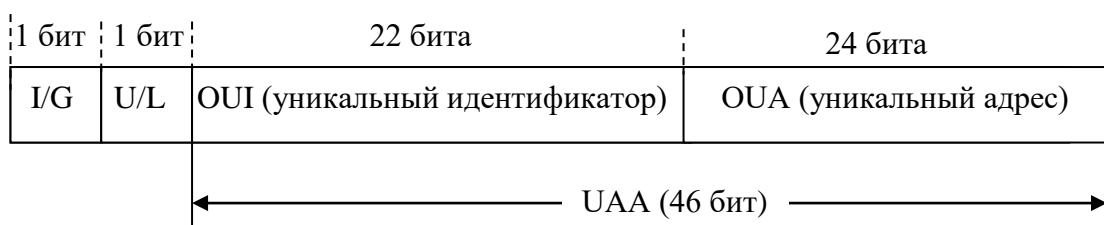


Рис. 7.1. Структура 48-битного стандартного MAC-адреса

Младшие 24 разряда кода адреса называются OUA (Organizationally Unique Address) – *уникальный адрес*. Именно их присваивает каждый из зарегистрированных производителей сетевых адаптеров. Всего возможно свыше 16 миллионов комбинаций, то есть каждый изготовитель может выпустить 16 миллионов сетевых адаптеров.

Следующие 22 разряда кода называются OUI (Organizationally Unique Identifier) – *уникальный идентификатор*. IEEE присваивает один или несколько OUI каждому производителю сетевых адаптеров. Это позволяет исключить совпадения адресов адаптеров от разных производителей. Всего возможно свыше 4 миллионов разных OUI, это означает, что теоретически может быть зарегистрировано 4 миллиона производителей. Вместе OUA и OUI называются UAA (Universally Administered Address) – *универсально управляемый адрес* или IEEE-адрес.

Два старших разряда адреса управляющие, они определяют тип адреса, способ интерпретации остальных 46 разрядов. Старший бит

I/G (Individual/Group) указывает на тип адреса. Если он установлен в 0, то индивидуальный, если в 1, то групповой (многопунктовый или функциональный). Пакеты с групповым адресом получат все имеющие этот групповой адрес сетевые адаптеры. Причем групповой адрес определяется 46-ю младшими разрядами. Второй управляющий бит U/L (Universal/Local) называется флагком универсального/местного управления и определяет, как был присвоен адрес данному сетевому адаптеру. Обычно он установлен в 0. Установка бита U/L в 1 означает, что адрес задан не производителем сетевого адаптера, а организацией, использующей данную сеть. Это случается довольно редко.

Для широковещательной передачи (то есть передачи всем абонентам сети одновременно) применяется специально выделенный *сетевой адрес*, все 48 битов которого установлены в единицу. Его принимают все абоненты сети независимо от их индивидуальных и групповых адресов.

7.1.2. Разработка программы определения МАС-адреса компьютера по IP-адресу и наоборот

Windows Sockets API

В ходе выполнения поставленной задачи необходимо использовать интерфейс Windows Sockets API (WAS).

Winsock (Windows socket) – это интерфейс программирования приложений (API), созданный для реализации сетевых приложений на основе протокола TCP/IP. Интерфейс реализуется библиотекой Wsock32.dll, экспортирующей функции работы с сокетами.

Сокет – это конечная точка в сетевом взаимодействии. Главная задача сокета – инкапсулировать реализацию установления соединения, обмена пакетами, разрыв и оповещение о происходящих ошибках в процессе сетевого взаимодействия в различных протоколах, предоставив разработчику простой и удобный интерфейс. Сокет характеризуется различными параметрами (семейство адресов, тип протоколов для передачи данных и др.), среди которых сетевой адрес и порт (при использовании транспортного протокола, требующего установления соединения).

Для подключения библиотек, содержащих необходимые структуры и функции, следует указать следующие директивы:

```
#include <Winsock2.h>
#include <Iphlpapi.h>
```

```
#pragma comment(lib, "ws2_32.lib")
#pragma comment(lib, "IPHlpApi.Lib")
```

Включение заголовочного файла `<Winsock2.h>` необходимо для использования функций Windows Socket версии 2. Единственное различие с заголовочным файлом `<Winsock.h>` то, что он содержит больше кодов ошибок новых API-функций. Заголовочный файл `<Iphlpapi.h>` нужен для использования API IP helper. Он также подключает другие заголовочные файлы, содержащие нужные структуры и перечисления.

Приложение, использующее Winsock, должно быть скомпоновано с библиотекой экспорта `"ws2_32.lib"`. Для сборки также требуется файл `"IPHlpApi.Lib"`.

Инициализация Winsock

Чтобы начать использовать интерфейс сокетов, необходимо инициализировать библиотеку `ws2_32.dll` с помощью вызова функции `WSAStartup`, передав ей в старшем байте номер требуемой версии, а в младшем – подверсии (первый аргумент):

```
WSADATA wsaData;
WSAStartup(0x0202, &wsaData);
```

Таким образом, первый параметр функции `0x0202` запрашивает версию 2.2 Winsock (можно использовать макрос `MAKEWORD(2, 2)`).

В большинстве случаев при написании новых приложений целесообразно загружать последнюю доступную версию библиотеки Winsock. Если используется, например, версия 3, приложение, использующее версию 2.2, должно выполняться корректно. При запросе более поздней версии Winsock, не поддерживаемой вашей платформой, `WSAStartup` вернет ошибку, а в поле `wHighVersion` структуры `wsaData` появится номер последней версии библиотеки, поддерживаемой данной системой.

Аргумент `wsaData` должен указывать на структуру `WSADATA`, в которую при успешной инициализации будет занесена информация о производителе библиотеки. Индивидуальные поля структуры `WSADATA` содержат: версию Winsock; высшую версию Winsock, поддерживающую загруженной библиотекой; текстовое описание загруженной библиотеки; текстовую строку с соответствующей информацией о состо-

янии или конфигурации; максимальное количество сокетов; максимальный размер дейтаграммы UDP; информацию об изготовителе.

Если инициализация не удалась, то функция возвращает ненулевое значение. Например:

```
// Инициализация Winsock
WSADATA WsaData;

if (WSAStartup(0x0202, &WsaData) !=NULL)
    cout<<"WSA: Ошибка !\n";
```

С этого момента можно начинать использовать весь спектр функций, предоставляемых WSA.

Создание сокета

Сокет представлен отдельным типом – SOCKET и создается одной из двух функций:

```
SOCKET WSASocket
(
    __in int af,
        // определяет семейство адресов протокола
    __in int type,
        // тип сокета для данного протокола
    __in int protocol,
        // указывает конкретный транспорт
    __in LPWSAPROTOCOL_INFO lpProtocolInfo,
    __in GROUP g,
        // параметр группы всегда равен 0
    __in DWORD dwFlags
        // флаги
);
```

ИЛИ

```
SOCKET WSAAPI socket
(
    __in int af,
        // определяет семейство адресов протокола
    __in int type,
        // тип сокета для данного протокола
    __in int protocol
        // указывает конкретный транспорт
);
```

Первый аргумент указывает на семейство используемых протоколов, константа AF_INET используется, чтобы сослаться на протокол IP (IPv4) (AF_INET6 для IPv6) и создать UDP- или TCP-сокет.

Второй аргумент – тип создаваемого сокета. Принимает значения: SOCK_STREAM (потоковый), или SOCK_DGRAM (дейтаграммный), или SOCK_RAW (сырой) и др (подробный перечень значений приведен в MSDN).

Последний аргумент задает тип создаваемого сокета. Нулевое значение соответствует выбору по умолчанию (система выбирает поставщика транспорта исходя из других двух параметров af и type): TCP – для потоковых, UDP – для дейтаграммных.

Если функция завершилась успешно, то она возвращает дескриптор сокета. Например пользователь вводит IP адрес, проверяем корректность структуры и диапазона заданного адреса (CheckAddr) и открываем сокет:

```
char *ip_, *ip_end_;
DWORD ip, ip_end;
bool flag=true;

ip_=new char[16];

SOCKET udp_s;
SOCKADDR_IN udp_sin;

do
{
    if(!flag) cout<<"Неверный IP"<<endl;
    cout<<"Введите IP: ";
    cin>>ip_;
}while(!(flag=CheckAddr(ip_)));

udp_s=socket(AF_INET,SOCK_DGRAM, IPPROTO_UDP);

if (udp_s!=SOCKET_ERROR)
{
    //Сокет успешно открыт
}
else
{
    cout<<"Ошибка открытия socket\n";
}
```

Установка соединения

Адресация. Сетевым адаптерам назначается IP-адрес, состоящий из 32 бит, официально называемый IP-адресом версии 4 (IPv4). Для

взаимодействия с сервером по TCP или UDP клиент должен указать IP-адрес сервера и номер порта. Чтобы прослушивать входящие запросы клиента, сервер также должен указать свой IP-адрес и номер порта.

В Winsock IP-адрес и порт задают в структуре `sockaddr_in`:

```
struct sockaddr_in
{
    short   sin_family;
           //семейство адресов
    u_short sin_port;
           //коммуникационный порт TCP или UDP,
           //который будет использован
           //для идентификации службы сервера
    struct  in_addr sin_addr;
           //адрес
    char    sin_zero[8];
};
```

Поле `sin_family` должно быть равно `AF_INET`: этим Winsock сообщает об использовании семейства адресов IP.

Порты

При использовании TCP и UDP приложение решает, через какой порт установить связь. Существуют стандартные номера портов, зарезервированные для служб сервера, поддерживающих протоколы более высокого уровня, чем TCP. Например, порт 21 зарезервирован для FTP, 80 – для HTTP. Стандартные службы обычно используют порты 1-1023. Для того чтобы узнать номера портов, используемых стандартными службами, можно вызвать функцию `getservbyname` или `WSAAsyncGetServByName`. Эти функции просто извлекают статическую информацию из файла с именем `services`.

Во избежание накладок с портами, уже занятymi системой или другим приложением, ваша программа должна выбирать зарегистрированные порты в диапазоне 1024–49151. Порты 49152–65535 также можно задействовать свободно – с ними не связаны никакие стандартные службы.

Поле `sin_addr` хранит IP-адрес в 4-байтном виде с типом `unsigned long int` (может представлять и локальный, и удаленный IP-адрес). IP-адреса задают в десятично-точечной нотации типа `a.b.c.d`. Здесь каждая буква представляет число для каждого байта и

задается слева направо (все четыре байта с типом `unsigned long int`). И наконец, поле `sin_zero` играет роль простого заполнителя, чтобы структура `sockaddr_in` по размеру равнялась исходной структуре `sockaddr`.

В разрабатываемой программе создадим объект `udp_sin`, содержащий структуру типа `sockaddr_in`, и выполним инициализацию:

```
.....  
if(udp_s!=SOCKET_ERROR)  
{  
    ip=inet_addr(ip_);  
    udp_sin.sin_family = AF_INET; //семейство  
    udp_sin.sin_port = htons(5234); //порт  
    udp_sin.sin_addr.S_un.S_addr = ip; //IP  
    адресс  
}
```

При инициализации определяется семейство адреса, IP-адрес и порт для соединения. Здесь `_ip` должен быть предварительно сформирован на основе введенных пользователем данных.

Преобразование адреса

Полезная вспомогательная функция `inet_addr` преобразует IP-адрес из точечной нотации в 32-битное длинное целое без знака и имеет формат:

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

Поле `cp` является строкой, заканчивающейся нулевым символом. Функция в качестве результата возвращает IP-адрес, представленный 32-битным числом с сетевым порядком следования байт (network-byte order). Функция является устаревшей, для подавления предупреждений установите:

```
#define _WINSOCK_DEPRECATED_NO_WARNINGS
```

Для конвертации IP адреса желательно использовать функции:

```
#include <arpa/inet.h>
const char *inet_ntop (int af, const void *src,
                      char *dst, socklen_t size);
int inet_pton (int af, const char *src, void *dst);
```

Порядок байт

Разные процессоры в зависимости от конструкции представляют числа в одном из двух порядков байт big-endian или little-endian. Номер порта и IP-адрес хранятся в памяти компьютера как многобайтные числа и представляются в системном порядке (host-byte-order). Когда IP-адрес или номер порта задаются по сети, стандарты Интернета требуют, чтобы многобайтные значения представлялись от старшего байта к младшему (в порядке big-endian), что обычно называется сетевым порядком (network-byte order).

Есть целый ряд функций для преобразования многобайтных чисел из системного порядка в сетевой, и обратно. Например следующая API-функция htons преобразует числа из системного порядка в сетевой:

```
u_short WSAAPI htons
(
    __in u_short hostshort // двухбайтное число
                           // с системным порядком
);
```

Функция htons возвращает число как двухбайтное значение с сетевым порядком.

Передача данных

По сути, в сетевом программировании самое главное – уметь отправлять и принимать данные. Для пересылки данных по сокету используются функции send, sendto и WSASend. Аналогично, для приема данных существуют функции recv, recvfrom и WSARecv. Все буферы, используемые при отправке и приеме данных, состоят из элементов типа char.

Будем использовать функцию sendto, она имеет следующий формат:

```

int sendto
(
    __in SOCKET s,    //дескриптор сокета
    __in const char *buf,
                    //указатель на буфер,
                    //содержащий данные для пересылки
    __in int len,
                    //длина в байтах передаваемого буфера
    __in int flags,
                    //флаги, которые определяют
                    //способ вызова
    __in const struct sockaddr *to,
                    //указатель на структуру SOCKADDR
    __in int tolen
                    //размер структуры SOCKADDR
);

```

В приложении отсылаем сообщение. Для этого мы вызовем функцию `sendto`.

```

if
    (sendto(udp_s,"TEST",5,NULL,(SOCKADDR*)&udp_sin,
           sizeof(udp_sin))>0)
{
}

else
{
    cout << "Ошибка передачи! \n " << WSAGetLastError();
}

```

Здесь `udp_s` – идентификатор определенного ранее сокета. Если ошибок не было, `sendto` возвратит общее число переданных байт, которое может быть меньше, чем указано в третьем аргументе функции. В противном случае будет возвращено значение `SOCKET_ERROR`, а код ошибки может быть извлечен вызовом функции `WSAGetLastError` или функцией `GetLastError`.

Для TCP-клиента (UDP-клинета) можно использовать функцию `connect` (версия Winsock 1) или `WSAConnect` (версия Winsock 2). У нее первый элемент – это дескриптор сокета для установления соединения, второй – структура адреса сокета `sockaddr`, содержащая адрес (IP), порт, к которому подключаются, и последний аргумент, сообщающий о размере `sockaddr`. Если по каким-то причинам

установить соединение не удается, то функция возвращает ненулевое значение:

```
if (connect (udp_sock, (sockaddr*)& udp_sin,
             sizeof (udp_sin)) )
{
    cout << "Ошибка соединения! \n "
        << WSAGetLastError();
    return -1;
}
```

Для получения MAC-адреса в дальнейшем будем использовать Internet Protocol Helper (IP helper) API. Он позволяет извлекать и модифицировать сетевую конфигурацию для локального компьютера.

Получение таблицы ARP

Функция GetIpNetTable (или GetIpNetTable2) извлекает IPv4 таблицу маппирования физических адресов. Прототип функции:

```
DWORD GetIpNetTable
(
    __out     PMIB_IPNETTABLE pIpNetTable,
                //указатель на структуру
    __inout   PULONG pdwSize,
                //размер структуры
    __in      BOOL bOrder
                //порядок сортировки по IP
);
```

В случае успеха функция возвращает NO_ERROR или ERROR_NO_DATA, если функция выполнилась успешно, но данных нет. Таблица возвращается через первый аргумент функции в виде структуры MIB_IPNETTABLE, которая содержит записи Address Resolution Protocol (ARP) для IPv4 адресов:

```
typedef struct _MIB_IPNETTABLE
{
    DWORD dwNumEntries;
                //количество записей в таблице
    MIB_IPNETROW table[ANY_SIZE];
                //указатель на массив MIB_IPNETROW структур
```

```
} MIB_IPNETTABLE, *PMIB_IPNETTABLE;
```

В свою очередь вложенная структура MIB_IPNETROW содержит:

```
typedef struct _MIB_IPNETROW
{
    DWORD dwIndex;
        //индекс адаптера
    DWORD dwPhysAddrLen;
        //длина физического адреса в байтах
    BYTE bPhysAddr[MAXLEN_PHYSADDR];
        //физический адрес
    DWORD dwAddr;
        //IPv4 адрес
    DWORD dwType;
        //тип ARP-записи
} MIB_IPNETROW, *PMIB_IPNETROW;
```

Для получения MAC-адреса вызываем функцию GetIpNetTable:

```
MIB_IPNETTABLE * pIpNetTable =
    (MIB_IPNETTABLE *) new char[0xFFFF];
ULONG cbIpNetTable = 0xFFFF;

if(NO_ERROR ==
    GetIpNetTable(pIpNetTable, &cbIpNetTable, true))
{
    for(DWORD i = 0; i < pIpNetTable->dwNumEntries; i++)
    {
        //если адрес в поле dwAddr совпадает с искомым ip
        //и dwType отличен от 2.
        //значение 2 соответствует недоступной или
        //незаконченной ARP записи

        if(pIpNetTable->table[i].dwAddr == ip
            && pIpNetTable->table[i].dwType!=2)
        {
            //извлекаем и выводим MAC
            printf("%s : %X-%X-%X-%X-%X-%X\n", ip_,
                pIpNetTable->table[i].bPhysAddr[0],
                pIpNetTable->table[i].bPhysAddr[1],
                pIpNetTable->table[i].bPhysAddr[2],
                pIpNetTable->table[i].bPhysAddr[3],
                pIpNetTable->table[i].bPhysAddr[4],
```

```
    pIpNetTable->table[i].bPhysAddr[5]);  
    //освобождаем таблицу  
    delete[] pIpNetTable;  
}  
}  
}
```

Освобождение Winsock

По завершении работы с библиотекой Winsock вызовите функцию для выгрузки библиотеки и освобождения ресурсов. Функция WSACleanup заканчивает использование Winsock DLL (Ws2_32.dll) (при удачном вызове возвращает ноль):

```
if (WSACleanup ()== SOCKET_ERROR)  
    cout<< "ERROR Closesocket      "  
        <<WSAGetLastError ( );
```

Для каждого вызова WSASStartup необходимо согласованно вызывать WSACleanup, так как каждый стартовый вызов увеличивает значение эталонного счетчика ссылок на загруженные Winsock DLL. Чтобы уменьшить значение счетчика, требуется равное количество вызовов WSACleanup.

Обратите внимание: Winsock 2 полностью совместим со всеми вызовами функций Winsock 1.1. Поэтому приложение, написанное для Winsock 1.1, будет работать и с библиотекой Winsock 2 – функции Winsock 1.1 сопоставляются с их эквивалентами в Winsock 2.

Используя описанные в разделе функции и структуры, можно выполнить обратное определение: по заданному MAC-адресу определить IP-адрес.

На рис. 7.2 приведен результат работы программы, выводящей MAC-адрес для заданного IP-адреса, на основе описанного подхода.

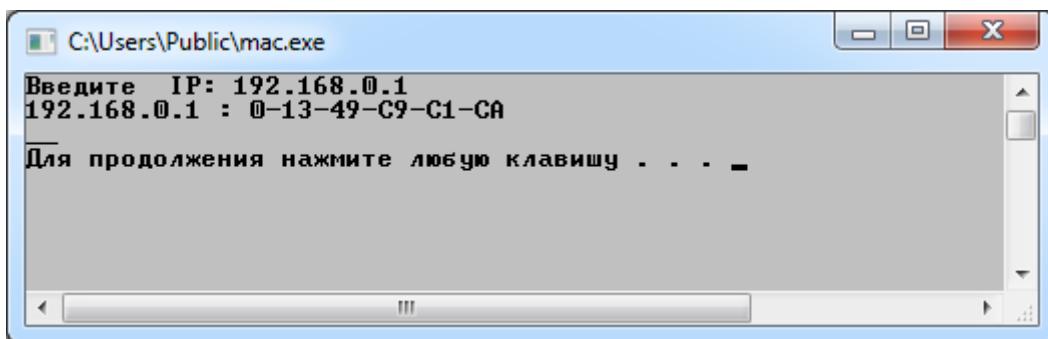


Рис. 7.2. Результат работы программы

SendARP

Для определения MAC-адреса можно использовать еще одну полезную функцию SendARP, которая посылает Address Resolution Protocol (ARP) запрос. На основе полученных данных можно получить MAC адресата по заданному IPv4 адресу:

```
DWORD SendARP
(
    __in     IPAddr DestIP,
    //IPv4 адрес адресата
    __in     IPAddr SrcIP,
    //IPv4 адрес источника
    __out    PULONG pMacAddr,
    //указатель на массив физического адреса
    __inout   PULONG PhyAddrLen
    //максимальная длина буфера массива физ. адреса
);
```

В случае успеха функция возвращает значение NO_ERROR. Рассмотрим другой способ получения MAC:

```
#include <iostream>
using namespace std;
#include <winsock2.h>
#include <iphlpapi.h>
#include <stdio.h>
#include <conio.h>
#include <Ws2tcpip.h>

#pragma comment(lib , "iphlpapi.lib")
#pragma comment(lib , "ws2_32.lib")

void GetMacAddress(unsigned char *, struct in_addr);
int main()
{
    setlocale(LC_ALL, "RUS");
    unsigned char mac[6];
    struct in_addr srcip = { 0 };
    struct sockaddr_in sa;
    char ip_address[32];

    WSADATA firstsock;
    if (WSAStartup(MAKEWORD(2, 2), &firstsock) != 0)
    {
        cout<<"Ошибка инициализации winsock";
```

```

        cout<< WSAGetLastError();
        return -1;
    }

    cout<<"Введите IP : ";
    cin>> ip_address;

    //преобразование IP адреса другим способом
    //srcip.s_addr = inet_addr(ip_address);

    inet_pton(AF_INET, ip_address, &(sa.sin_addr));

    //Получение MAC по IP
    GetMacAddress(mac, sa.sin_addr);
    //GetMacAddress(mac, srcip);
printf("MAC адрес : %.2X-%.2X-%.2X-%.2X-%.2X-%.2X",
      mac[0], mac[1], mac[2], mac[3], mac[4], mac[5]);
printf("\n");
getch();
    return 0;
}

void GetMacAddress(unsigned char *mac, struct in_addr
destip)
{
    DWORD ret;
    IPAddr srcip;
    ULONG MacAddr[2];
    ULONG PhyAddrLen = 6;
    int i;

    srcip = 0;

    //Послать ARP пакет
    ret = SendARP((IPAddr)destip.S_un.S_addr,
                  srcip, MacAddr, &PhyAddrLen);

    //Преобразовать адрес
    if (PhyAddrLen)
    {
        BYTE *bMacAddr = (BYTE *)& MacAddr;
        for (i = 0; i < (int)PhyAddrLen; i++)
        {
            mac[i] = (char)bMacAddr[i];
        }
    }
}

```

На рис. 7.3 приведен результат работы программы, выводящей MAC-адрес для заданного IP-адреса, на основе описанного подхода.

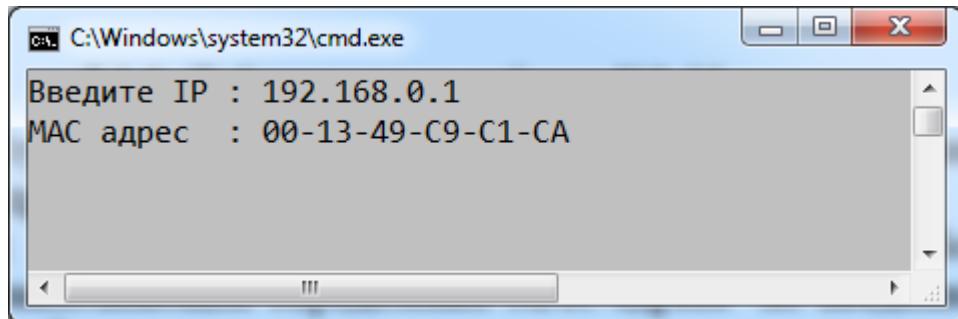


Рис. 7.3. Результат работы программы

7.1.3. Лабораторная работа № 9–10

Цель: изучение методов определения MAC-адреса удаленного хоста по IP-адресу и наоборот.

Задание: разработка программы, выполняющей определение MAC-адреса по введенному IP-адресу и наоборот.

Дополнительное задание: поиск MAC-адреса должен осуществляться для диапазона IP-адресов различных классов.

7.2. Определение символьного адреса удаленного хоста по IP-адресу

7.2.1. Разработка программы определения имени (DNS, NetBIOS) компьютера по IP-адресу, и наоборот

Интерфейс NetBIOS

Network Basic Input/Output System (NetBIOS) – стандартный интерфейс прикладного программирования (API), разработанный Sytek Corporation для IBM. NetBIOS определяет программный интерфейс для сетевой связи, но не обуславливает физический способ передачи данных по сети. В Win32 интерфейс NetBIOS обеспечивает обратную совместимость со старыми приложениями.

Ключ к пониманию NetBIOS – номера сетевых адаптеров (LAN Adapter, LANA). В первоначальных реализациях NetBIOS каждому физическому сетевому адаптеру присваивалось уникальное значение – номер LANA. В Win32 это стало проблематичным, так как рабочая

станция может иметь и множество сетевых протоколов, и множество плат сетевого интерфейса.

Номер LANA соответствует уникальным сочетаниям сетевого адаптера с транспортным протоколом. Так, если рабочая станция имеет две сетевые платы и два поддерживающих NetBIOS транспорта (например, TCP/IP и NetBEUI), будет присвоено четыре номера LANA. Номера LANA лежат в диапазоне от 0 до 9, и операционная система назначает их без какого-либо определенного порядка, кроме LANA 0, который имеет особый смысл – это номер «по умолчанию».

Есть два типа имен NetBIOS – уникальное и групповое. Никакой другой процесс в сети не может зарегистрировать уже имеющееся уникальное имя – будет выдана ошибка дублирования имени. Имена компьютеров в сетях Microsoft – имена NetBIOS. Когда компьютер загружается, он регистрирует свое имя на локальном сервере Windows Internet Naming Server (WINS), который сообщает об ошибке, если другой компьютер уже использует то же имя. Сервер WINS поддерживает список всех зарегистрированных имен NetBIOS.

Вместе с именем могут храниться и сведения о протоколе. Например, в сетях TCP/IP WINS запоминает IP-адрес компьютера, зарегистрировавшего имя NetBIOS.

API-интерфейс NetBIOS содержит только одну функцию:

```
UCHAR Netbios (PNCB pNCB) ;
```

Все объявления функций, константы и т. п. для NetBIOS определены в заголовочном файле. Единственная библиотека, необходимая для компоновки приложений NetBIOS – Netapi32.lib. Добавим в приложение:

```
#include <stdio.h>
#include <iostream>
#include <windows.h>
#include <Winsock.h>
#include <Wsnetbs.h>

#pragma comment(lib, "WS2_32.lib")
#pragma comment (lib, "Netapi32.lib")

using namespace std;
```

Валидация пользовательского ввода

Перед определением имен и IP-адресов необходимо добавить проверку корректности ввода пользователем IP-нотации адреса и DNS-имени. Для проверки IP добавьте в приложение функцию:

```
bool CheckAddr(char* ip)
{
    int points=0, //счетчик точек
        numbers=0;//счетчик цифр октета

    char* buff;

    buff = new char[3];

    for(int i=0;ip[i]!='\0';i++)//просмотреть IP-адрес
    {
        if(ip[i]<='9'&&ip[i]>='0') //если только цифры
        {
            if(numbers>3) return false; //не больше 3
            buff [numbers++]=ip[i];
            //записать в буфер
        }
        else if(ip[i]=='.') //если следующий октет
        {
            if(atoi(buff)>255) return false;
            //больше 255 - ошибка

            if(numbers==0) return false;
            //если цифр нет

            numbers=0; //обнулить счетчик цифр октета
            points++;
            buff = new char[3];
        }
        else return false;
    }
    if(points!=3) return false;
    //если точек меньше 3 - ошибка
    if(numbers==0||numbers>3) return false;
    return true;
}
```

Для проверки верности задания DNS-имени добавьте пользовательскую функцию:

```

bool CheckDNSName(char* dns_name)
{
    for(int i=0;dns_name[i]!='\0';i++)
        if(!(dns_name[i]>='A'&&dns_name[i]<='Z'|| 
            dns_name[i]>='a'&&dns_name[i]<='z'|| 
            dns_name[i]>='0'&&dns_name[i]<='9'|| 
            dns_name[i]=='.'|| 
            dns_name[i]=='-')) return false;

    //имя может содержать латинские символы, цифры, точки
    //тире
    return true;
}

```

Инициализация Winsock

Инициализируем Winsock следующим образом:

```

setlocale(LC_ALL, "RUS");

WSADATA WsaData;

    struct hostent *dns;
    struct hostent *netbios;
    char *host = new char[16];
    int i=0;
    bool flag=true;
    bool is_ip=true;
    in_addr addr;

    if(WSAStartup(0x0202, &WsaData)==NULL)
        cout<<"WSA done!"<<endl;

```

Выполняем ввод информации и проверку ввода:

```

do
{
    if(!flag) cout<<"Неверный IP"<<endl;
    cout<<"Введите IP-адрес или DNS-имя: ";
    cin>>host;           //ввести

    if(is_ip!=isalpha(host[0]))
        //если первый - не буква

```

```

        flag=CheckAddr(host);
                    //проверить адрес
    else
        flag=true;
}while(!flag);

if(!is_ip) //если буква
{
    do
    {
        if(!flag)
        {
            cout<<"Неверное DNS-имя"<<endl;
            cout<<"Введите DNS-имя: ";
            cin>>host;           //ввести
        }
        flag=CheckDNSName(host);
                    //проверить DNS
    }while(!flag);
}

```

Поиск сведений об узле

Если задан IP-адрес, определим DNS и NetBIOS имена. Будем использовать функцию `gethostbyaddr`, извлекающую информацию о хосте в соответствии с введенным сетевым адресом. Функция последовательно просматривает файл `/etc/hosts` с самого начала в поисках имени или адреса главной машины. Формат функции:

```

struct hostent* FAR gethostbyaddr
(
    __in const char *addr,
                    //указатель на адрес
                    //в сетевом порядке следования байт
    __in int len,
                    //длина адреса в байтах
    __in int type
                    //тип адреса
);

```

Тип адреса `type` может принимать значения `AF_INET` для семейства адресов IPv4; `AF_NETBIOS` для NetBIOS семейства; `AF_INET6` для IPv6.

В случае ошибки возвращается `NULL` указатель, а специфический код ошибки может быть получен через вызов `WSAGetLastError`. В случае удачного завершения, функция возвращает структуру `hostent`, которая используется для хранения информации о заданном хосте:

```
typedef struct hostent
{    char FAR *h_name;
     //официальное host-имя

    char FAR FAR **h_aliases;
     //список псевдонимов -
     //массив альтернативных имен
     //машины, оканчивающийся нулем
    int      h_addrtype;
     //тип адреса
     //в настоящее время всегда AF_INET
    int      h_length;
     //длина адреса в байтах
    char    FAR **h_addr_list;
     //указатель на список адресов хоста,
     //оканчивающийся нулем
} ;HOSTENT, *PHOSTENT, FAR *LPHOSTENT;
```

Поле `h_name` является официальным именем узла. Если в сети используется доменная система имен (DNS), в качестве имени сервера будет возвращено полное имя домена (Fully Qualified Domain Name, FQDN). Если в сети применяется локальный файл узлов (`hosts`, `lmhosts`) – это первая запись после IP-адреса. Поле `h_aliases` – массив, завершающийся нулем дополнительных имен узла. Поле `h_addrtype` представляет возвращаемое семейство адресов. Поле `h_length` определяет длину в байтах каждого адреса из поля `h_addrtype`. Поле `h_addr_list` – массив, содержащий IP-адреса узла (узел может иметь несколько IP-адресов). Каждый адрес в этом массиве представлен в сетевом порядке. Обычно приложение использует первый адрес из массива. При получении нескольких адресов приложение должно выбирать адрес случайным образом из числа доступных.

Приложение не может изменять структуру или освобождать поля структуры `hostent`. Только одна копия данной структуры может размещаться в потоке, поэтому нужную информацию необходимо копировать перед любым другим Windows Sockets API вызовом.

Существует асинхронная версия функции `gethostbyaddr` и `WSAAsyncGetHostByAddr`.

Разрешение имен

Если пользователь ввел IP, преобразуем IP-адрес из точечной нотации в 32-битное длинное целое без знака, вызываем функцию `gethostbyaddr`, получаем указатель на структуру `hostent`, выводим на консоль официальное `host`-имя, извлекаем список адресов хоста для вывода на консоль, используем функцию преобразования `inet_ntoa`, конвертирующую IPv4-адрес в ASCII строку десятично-точечной нотации для IP-адресов, и вызываем функцию `GetNetBiosName`, выводящую имена NetBIOS:

```
if(is_ip)
{
    addr.S_un.S_addr=inet_addr(host); //преобразование

    dns=gethostbyaddr((char*)&addr,4,AF_INET);
    if(dns!=NULL)           //если структура заполнена,
    {
        cout<<"DNS имя: "<<dns->h_name<<endl;
                    //выводим имя хоста

        while(dns->h_addr_list[i]!=0)
        {
            //выводим список адресов хоста
            addr.S_un.S_addr =
                *(u_long*) dns->h_addr_list[i++];
            cout<<" "<<inet_ntoa(addr);
        }
    }
    else
    {
        cout<<"DNS имя не найдено..."<<endl;
    }
    GetNetBiosName(inet_ntoa(addr));
}
```

На рис. 7.4 приведен результат работы программы.

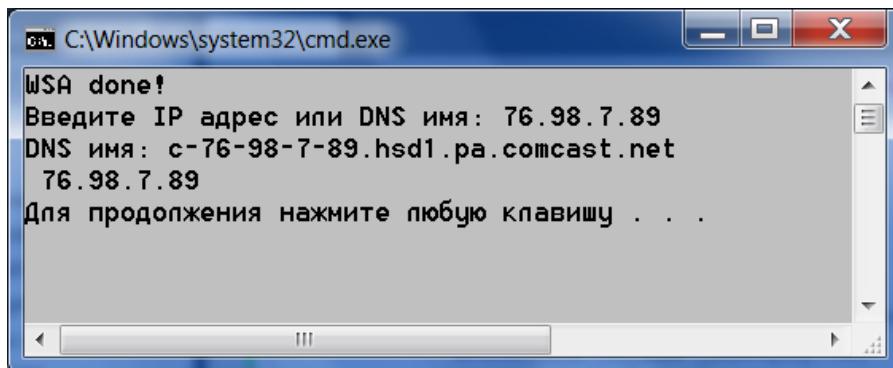


Рис. 7.4. Результат работы программы

В Winsock предусмотрены две функции для разрешения имени в IP-адрес. Функции `gethostbyname` и `WSAAAsyncGetHostByName` отыскивают в базе данных узла сведения об узле, соответствующие его имени. `WSAAAsyncGetHostByName` – асинхронная версия функции `gethostbyname`, оповещающая приложение о завершении своего выполнения с помощью сообщений Windows.

Если пользователь ввел DNS-имя, вызываем функцию `gethostbyname`, получаем указатель на структуру `hostent`, выводим на консоль официальное `host-имя`, извлекаем список адресов хоста для вывода на консоль и вызываем функцию `GetNetBiosName`:

```
else {
    dns = gethostbyname(host);
    if(dns!=NULL) {
        cout<<"DNS имя: "<<dns->h_name<<endl;
        while(dns->h_addr_list[i]!=0)
        {
            addr.S_un.S_addr =
                *(u_long*) dns->h_addr_list[i++];
            cout<<" "<<inet_ntoa(addr)<<endl;
        }
        addr.S_un.S_addr =
            *(u_long*) dns->h_addr_list[0];
        GetNetBiosName(inet_ntoa(addr));
    }
    else{
        cout<<"DNS имя не найдено..."<<endl;
    }
}
```

На рис. 7.5 приведен результат работы программы.

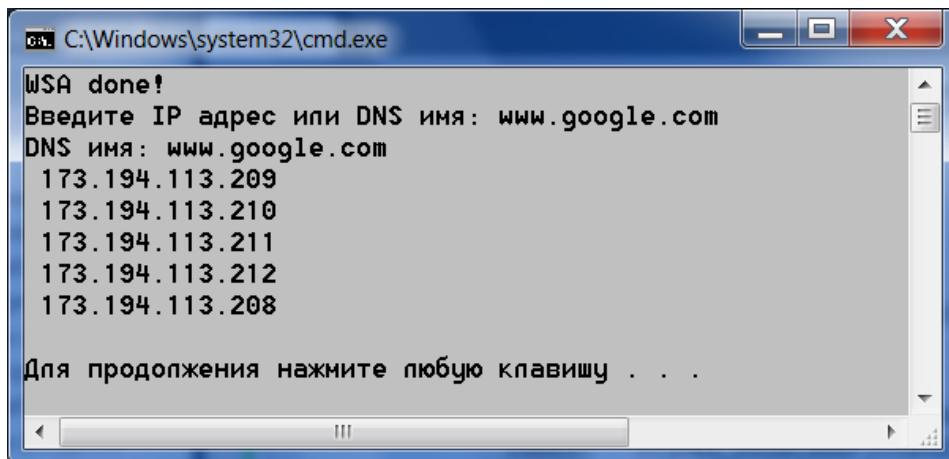


Рис. 7.5. Результат работы программы

API-интерфейс NetBIOS и блок сетевого управления (NCB)

Функция `GetNetBiosName` определяет NetBIOS имена компьютера по IP через стандартную функцию `Netbios`. Наиболее важная особенность функции `Netbios` – параметр `pNCB`, который является указателем на блок сетевого управления (network control block, NCB). Это указатель на структуру `NCB`, содержащую всю информацию, требуемую функции `Netbios` для выполнения команды. Определяется эта структура так:

```
typedef struct _NCB
{
    UCHAR ncb_command;
    UCHAR ncb_retcod;
    UCHAR ncb_lsn;
    UCHAR ncb_num;
    PUCHAR ncb_buffer;
    WORD ncb_length;
    UCHAR ncb_callname[NCBNAMSZ];
    UCHAR ncb_name[NCBNAMSZ];
    UCHAR ncb_rto;
    UCHAR ncb_sto;
    void (CALLBACK *ncb_post)( struct *NCB );
    UCHAR ncb_lana_num;
    UCHAR ncb_cmd_cplt;
    UCHAR ncb_reserve[X];
    HANDLE ncb_event;
} NCB, *PNCB;
```

Не все члены структуры будут использоваться в каждом вызове Netbios; некоторые из полей данных являются выходными параметрами (другими словами, задаются по возвращении из вызова Netbios). Важно всегда обнулять структуру NCB до вызова Netbios. Это можно сделать функцией:

```
ZeroMemory(&ncb, sizeof(NCB)) ;
```

Рассмотрим состав структуры NCB:

ncb_command – указывает выполняемую команду Netbios.
Многие команды могут выполняться синхронно или асинхронно;

ncb_retcode – определяет код возврата для данной операции;

ncb_lsn – определяет номер локального сеанса, уникально идентифицирующий его в текущем окружении. Функция возвращает новый номер сеанса после успешной команды NCBCALL (открыть сессию с другим именем) или NCBLISTEN;

ncb_num – указывает номер локального сетевого имени. Новый номер возвращается для каждого вызова команды NCBADDNAME (добавить новое имя) или NCBADDGRNAME (добавить новое групповое имя);

ncb_buffer – указывает на буфер данных. Для команд, которые отправляют данные, этот буфер содержит отправляемые данные; для команд, которые получают данные – данные, возвращаемые функцией Netbios. Для других команд, типа NCBENUM (перечисление адаптеров LAN), буфер будет предопределен структурой LANA_ENUM;

ncb_length – указывает длину буфера в байтах. Для команд приема присваивает этому полю значение, равное количеству полученных байтов, если буфер недостаточно велик, Netbios возвращает ошибку;

ncb_callname – указывает имя удаленного приложения;

ncb_name – указывает имя, под которым известно приложение;

ncb_rto – указывает время ожидания (тайм-аут) для операций приема (значение определено в 500-миллисекундных единицах);

ncb_sto – указывает время ожидания для операций отправки;

ncb_post – указывает адрес процедуры, которую надо вызвать по завершении асинхронной команды. Функция определена как void CALLBACK PostRoutine(PNCB pncb);

ncb_lana_num – указывает номер LANA для выполнения команды;

`ncb_cmd_cplt` – определяет код возврата для операции;
`ncb_reserve` – зарезервировано, должно быть равно 0;
`ncb_event` – указывает описатель объекта события Windows в свободном (nonsignaled) состоянии. Когда асинхронная команда завершается, событие переходит в занятое (signaled) состояние. Следует использовать только ручной сброс событий.

Пример определения имен NetBIOS

Теперь рассмотрим текст функции `GetNetBiosName`. Она получает все номера LANA, для каждого LAN извлекает статус адаптера, если определено имя NetBios, выводит на консоль:

```
int GetNetBiosName(char *dwIpAddr)
{
    ADAPTER_STATUS *pStatus; //Статус адаптера
    NAME_BUFFER *pNames; //Структура имя NetBios
    NCB ncb; //Структура NetBios
    HANDLE hHeap;
    WORD cbBuffer;
    char *Addr;
    UCHAR rc;
    int i;

    //Получаем IP-адрес в строковом виде
    in_addr in;
    if (dwIpAddr==0) return 0;
    in.s_addr = inet_addr(dwIpAddr);
    Addr=inet_ntoa(in);
    if (!Addr) return 0;

    //Получить все номера LANA
    LANA_ENUM lan_num; //перечисление всех lana
    //Инициализация/очистка структуры NCB
    ZeroMemory(&ncb,sizeof(NCB));
    ncb.ncb_command = NCBENUM;
    //определить код для заполнения LANA_ENUM структуры
    ncb.ncb_buffer = (unsigned char *) &lan_num;
    ncb.ncb_length = sizeof(lan_num); //размер
    rc=Netbios (&ncb); //вызвать функцию
    for (i=0;i < lan_num.length; i++)
        //для каждого LANA
    {
        //Инициализация/очистка
```

```

ZeroMemory(&ncb,sizeof(NCB));
//Сброс всех сведений о LANA, перечисленных в структуре
//LANA_ENUM и использование первого NetBIOS-имени
ncb.ncb_command = NCBRESET;
ncb.ncb_lana_num = lan_num.lana[i];
//установить номер LANA
rc=Netbios (&ncb); //вызвать функцию
hHeap = GetProcessHeap();

//буфер под статус адаптера + список имен(255 имен)
cbBuffer = sizeof (ADAPTER_STATUS)
+ 255 * sizeof (NAME_BUFFER);

//выделить блок памяти для кучи и инициализировать 0
pStatus = (ADAPTER_STATUS *) HeapAlloc
(hHeap,HEAP_ZERO_MEMORY,cbBuffer);

if (pStatus==NULL) return 0;

//Инициализация/очистка структуры NCB
ZeroMemory(&ncb,sizeof(NCB));

ncb.ncb_command = NCBASTAT;
//извлечь статус адаптера
ncb.ncb_lana_num = lan_num.lana[i];//для i LANA
ncb.ncb_buffer = (P UCHAR) pStatus;
ncb.ncb_length = cbBuffer;
strcpy((char *)ncb.ncb_callname,Addr);//для IP
rc = Netbios (&ncb); //вызвать функцию

if (ncb.ncb_retcode==NRC_GOODRET)
//если определено имя NetBios
{
    pNames = (NAME_BUFFER *) (pStatus + 1);
    printf("\n Имена NetBios:\n");

    for ( i = 0;i < pStatus->name_count;i++)
    {
        //вывести имя на консоль
        pNames->name[15]='\0';
        printf("%d: %s\n", i+1, pNames->name);
        pNames++;
    }
}
HeapFree (hHeap,0,pStatus);
}

return 0;
}

```

В конце программы не забудьте вызвать функцию WSACleanup().

7.2.2. Лабораторная работа № 11–12

Цель: изучение методов определения символьного (DNS, NetBIOS) адреса удаленного хоста, и наоборот.

Задание: разработка программы, выполняющей определение символьного (DNS, NetBIOS) адреса по введенному IP-адресу, и наоборот, для удаленного хоста по сети.

7.3. Определение MAC-адреса удаленного хоста по символьному адресу

7.3.1. Разработка программы определения MAC-адреса по DNS (NetBIOS) имени

Задача определения MAC-адреса по DNS-имени во многом аналогична уже рассмотренным задачам. Поэтому рассмотрим ее достаточно кратко (за более подробной информацией обратитесь к предыдущим разделам).

Сначала необходимо включить в программу:

```
#include <stdio.h>
#include <iostream>
#include <windows.h>
#include <Winsock.h>
#include <Wsnetbs.h>
#include <Iphlpapi.h>

#pragma comment(lib, "WS2_32.lib")
#pragma comment (lib, "Netapi32.lib")
#pragma comment(lib, "IPHlpApi.Lib")

using namespace std;
```

Инициализация Winsock

Затем вызываем функцию WSASStartup – инициализируем библиотеку WS2_32.dll:

```
int _tmain(int argc, _TCHAR* argv[])
{
    setlocale(LC_ALL, "RUS");
```

```
WSADATA WsaData;  
if(WSAStartup(0x0202, &WsaData) !=NULL)  
{  
    cout<<"WSA Error!"<<endl;  
    return -1;  
}
```

Разрешение имени в IP-адрес

Вводим DNS-имя, проверяем введенное имя функцией CheckDNSName, работа которой была рассмотрена выше. Вызываем функцию gethostbyname, результат функции будет сохранен в структуре hostent:

```
hostent *dns;  
char* host = new char[20];  
SOCKADDR_IN addr;  
SOCKET udp_sock;  
do  
{  
    cout<<"Host: ";  
    cin>>host;  
}while(!CheckDNSName(host));  
dns=gethostbyname(host);  
if(dns==NULL)  
{  
    cout<<"Не найдено"<<endl;  
    return -1;  
}
```

Создание сокета

Создаем сокет функцией socket, установив IP-адрес, извлеченный из структуры hostent:

```
udp_sock = socket(AF_INET, SOCK_DGRAM, IPPROTO_UDP);
```

Передача данных

Создаем объект, содержащий структуру sockaddr_in и выполняем его инициализации. Отсылаем сообщение по сокету функцией sendto:

```

addr.sin_family = AF_INET;
addr.sin_port = htons(1234);
addr.sin_addr.S_un.S_addr =
    *(ULONG*) dns->h_addr_list[0];
if(sendto(udp_sock, "TESR", 5, NULL, (SOCKADDR*)&addr,
           sizeof(addr))==NULL)
{
    cout<<"Send"<<endl;
    return -1;
}

```

Получение ARP-таблицы и MAC-адреса

Через вызов функции `GetIpNetTable` извлекаем IPv4-таблицу маппирования физических адресов в структуру `MIB_IPNETTABLE`:

```

MIB_IPNETTABLE *pTable =
    (MIB_IPNETTABLE*) new char[0xFFFF];
ULONG cTable=0xFFFF;

if(GetIpNetTable(pTable, &cTable, true) !=NO_ERROR)
{
    cout<<"MAC not found"<<endl;
    return -1;
}

```

Из поля `table` объекта структуры `MIB_IPNETTABLE` извлекаем физический адрес:

```

for(DWORD i=0;i<pTable->dwNumEntries;i++)
{
    if(pTable->table[i].dwAddr ==
        addr.sin_addr.S_un.S_addr
        && pTable->table[i].dwType!=2)
    {
        printf("IP: %i.%i.%i.%i\nMAC: %X-%X-%X-%X-%X-%X\n",
               (int)addr.sin_addr.S_un.S_un_b.s_b1,
               (int)addr.sin_addr.S_un.S_un_b.s_b2,
               (int)addr.sin_addr.S_un.S_un_b.s_b3,
               (int)addr.sin_addr.S_un.S_un_b.s_b4,
               pTable->table[i].bPhysAddr[0],
               pTable->table[i].bPhysAddr[1],

```

```
    pTable->table[i].bPhysAddr[2],  
    pTable->table[i].bPhysAddr[3],  
    pTable->table[i].bPhysAddr[4],  
    pTable->table[i].bPhysAddr[5]);  
}  
}  
closesocket( udp_sock );  
WSACleanup();  
return 0;  
}
```

Результат работы программы приведен на рис. 7.6.

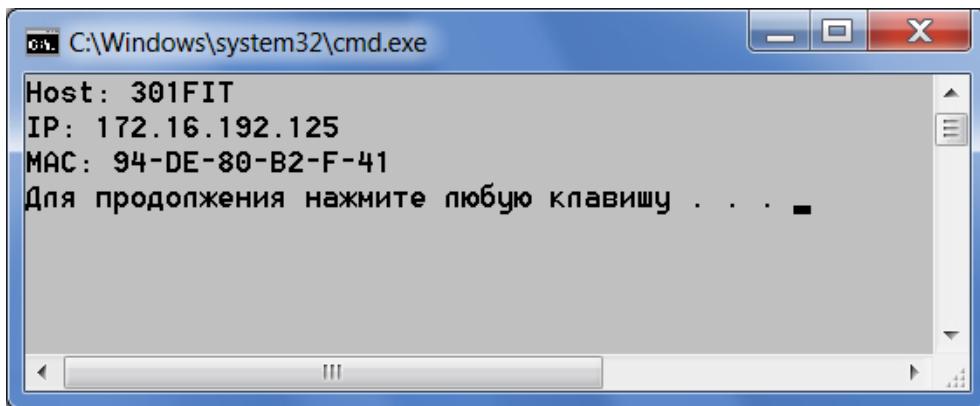


Рис. 7.6. Результат работы программы.

7.3.2. Лабораторная работа № 13-14

Цель: изучение методов определения МАС-адреса удаленного хоста по символьному адресу.

Задание: разработка программы, выполняющей определение МАС-адреса удаленного хоста по введенному символьному (DNS, NetBIOS) адресу.

8. ПЕРЕДАЧА ИНФОРМАЦИИ С ИСПОЛЬЗОВАНИЕМ ПРОТОКОЛОВ HTTP И FTP

Веб-сервер представляет собой некоторую программу (службу), принимающую HTTP-запросы от клиентов, обычно от веб-браузеров, обрабатывающую данные запросы и выдающую им ответ, как правило, вместе с HTML-страницей (сайтом), изображением, файлом, медиа-потоком или другими данными.

Если управление сетью осуществляется на основе Windows Server, то логично будет использовать веб-сервер IIS. Это весьма популярная платформа, которая позволяет работать как с большинством популярных CMS, так и имеет широкий спектр систем, предназначенных для работы именно на Windows и IIS.

Несомненным достоинством IIS является его тесная интеграция с другими технологиями и средствами разработки Microsoft. В частности веб-решения для IIS могут использовать богатые возможности .NET и легко взаимодействовать с настольными приложениями на этой платформе. К тому же для IIS имеется богатый выбор готовых CMS.

Однако далее в рамках курса «Компьютерные сети» мы прежде всего будем уделять внимание не созданию «мощных» и удобных веб-ресурсов, а базовым элементам, а именно настройке и работе по HTTP и FTP протоколам, которые в полной мере поддерживаются IIS-сервером.

8.1. Установка IIS-сервера

По своей сути установка IIS идентична установке DHCP или DNS серверов, т. е. ее удобно выполнять, используя мастер настройки сервера, а именно через добавление ролей (рис. 8.1). Необходимо отметить, что если стоит задача сразу установить HTTP и FTP составляющие IIS-сервера, то после выбора роли *Web Server (IIS)* необходимо выбрать *Add Features (Добавить компоненты)* и в появившемся окне галочками отметить, что необходимо инсталлировать (например, *FTP Server* и *Management Tools*) (рис. 8.2).

Далее необходимо будет отвечать утвердительно на всех последующих диалоговых окнах, т. е. выбирая *Next (Далее)*. После завершения установки в консоли *Server Manager* отдельно появится сервер *IIS* (рис. 8.3).

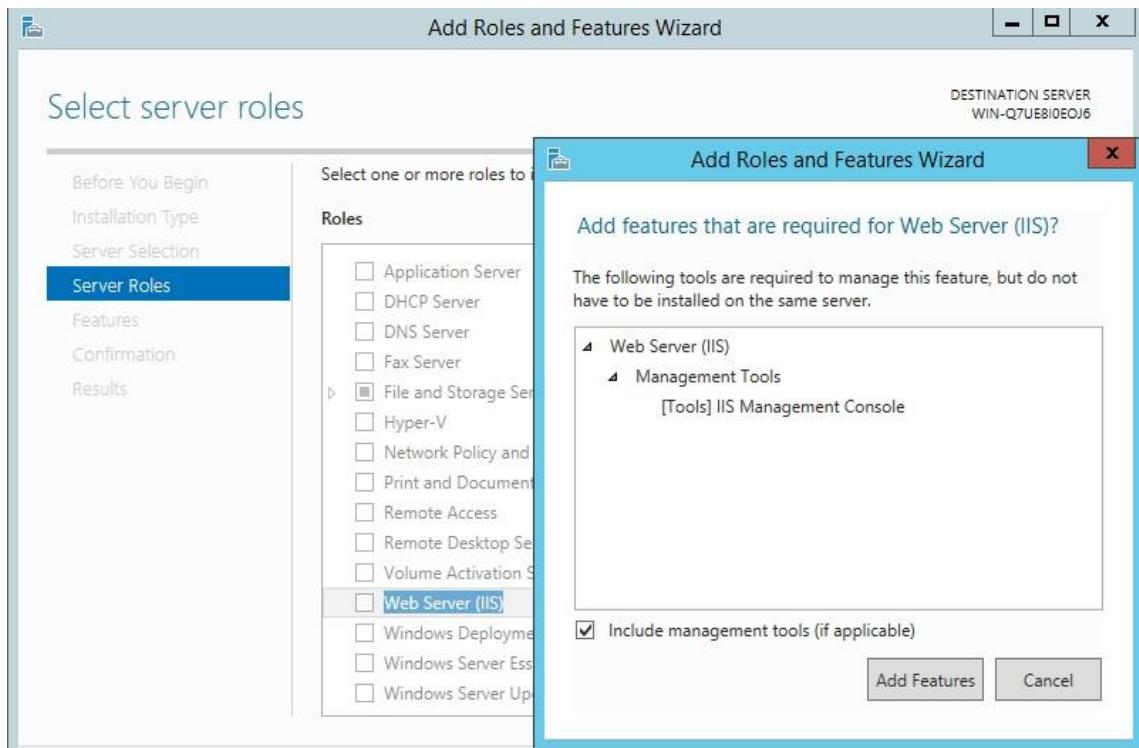


Рис. 8.1. Выбор устанавливаемой роли *Web Server (IIS)*

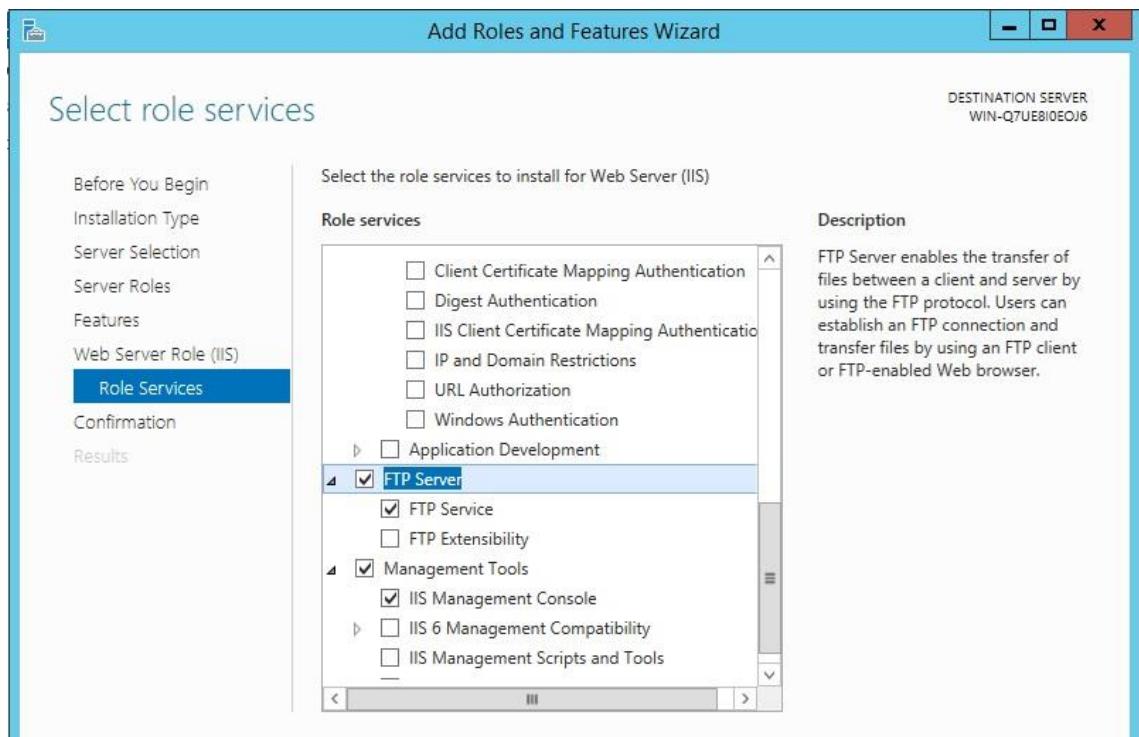


Рис. 8.2. Выбор компонент при установке роли *Web Server (IIS)*

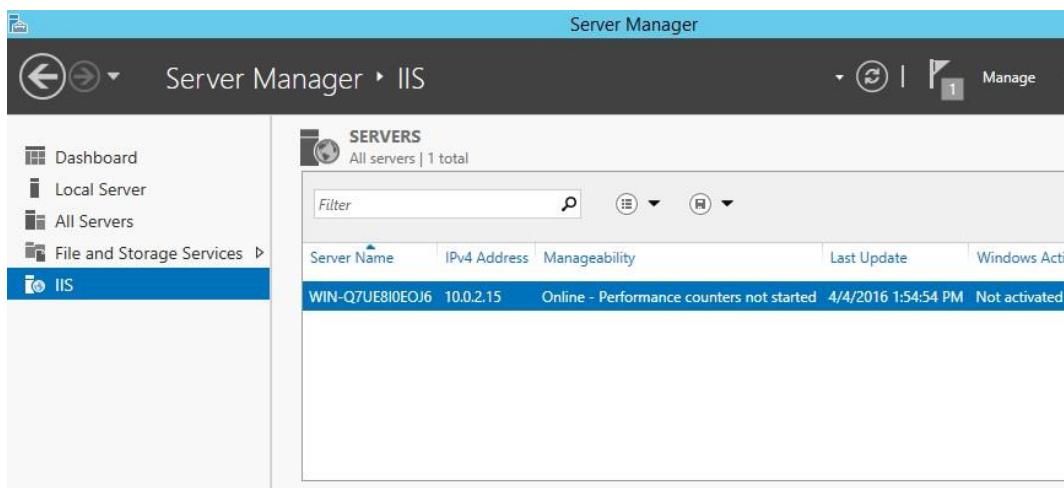


Рис. 8.3. Консоль *Server manager* с установленным *IIS*

Далее можно переходить к настройке HTTP или FTP составляющих сервера.

8.2. Настройка и передача данных по протоколу HTTP. Web-узлы и виртуальные каталоги

8.2.1. Настройка web-узлов

Для настройки HTTP или FTP составляющих сервера необходимо выбрать через контекстное меню *Internet Information Services (IIS) Manager*, или перейдя в него через *Server Manager / Tools / Internet Information Services (IIS) Manager* (рис. 8.4).

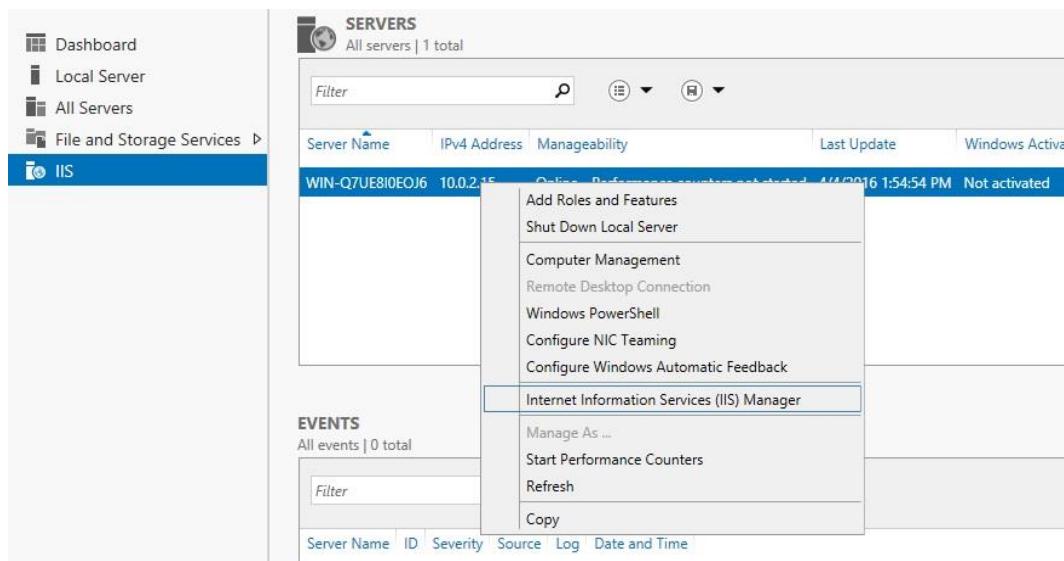


Рис. 8.4. Переход в консоль управления IIS-сервера (*Internet Information Services (IIS) Manager*)

В первоначальном виде консоль управления IIS-сервера выглядит, как показано на рис. 8.5.

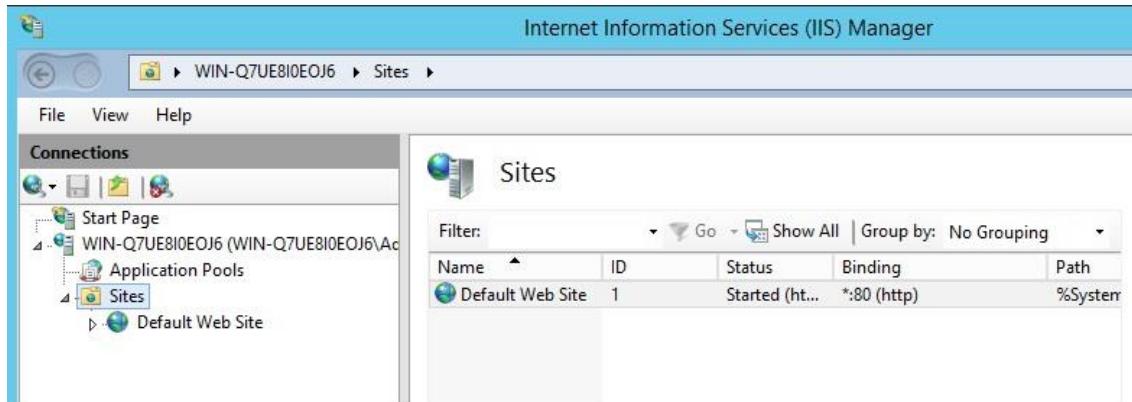


Рис. 8.5. Первоначальный вид консоли управления IIS-сервера
(*Internet Information Services (IIS) Manager*)

Прежде чем далее приступать к настройке web и ftp узлов, рекомендуется создать некоторое количество альтернативных IP-адресов, которые логически связаны с основным IP-адресом (в примере он равен 192.168.1.100/24). В качестве дополнительных (альтернативных) адресов в дальнейших примерах будут использоваться 192.168.1.120/24, 192.168.1.121/24, 192.168.1.122/24, 192.168.1.123/24 и 192.168.1.124/24 (рис. 8.6). Именно они будут присваиваться отдельным web и ftp узлам.

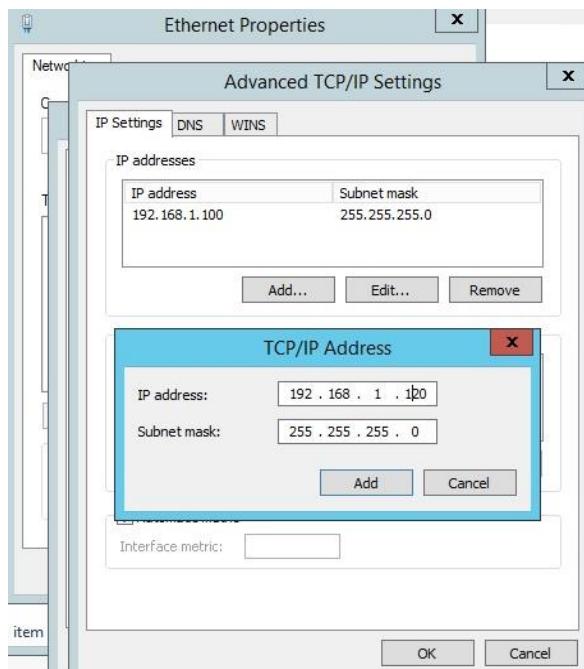


Рис. 8.6. Ввод дополнительных IP-адресов

Теперь можно непосредственно приступать к настройке, например web-узла. На каком-либо диске, например *C*, создадим папку *iis_http*, которая будет являться домашним каталогом для создаваемого web-узла *my_site* (рис. 8.7), и разместим там файл с именем *index.html* (фактически это web-страница, которая будет отображаться в браузере при обращении пользователя к web-узлу).

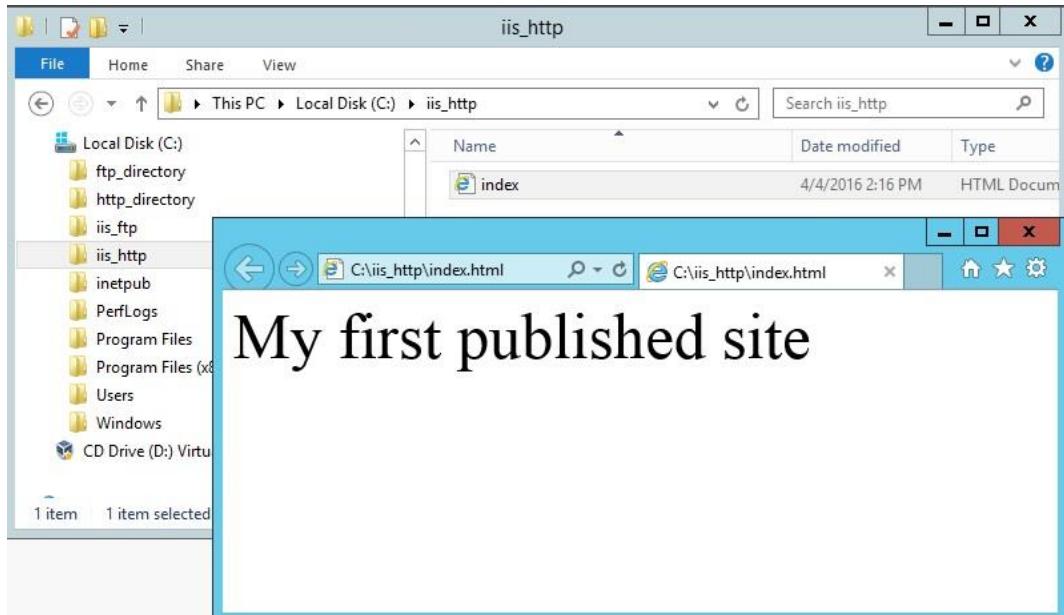


Рис. 8.7. Домашний каталог для web-узла

Далее перейдем непосредственно к созданию и настройке web-узла. Для этого в консоли управления сервером *IIS* вызываем контекстное меню для *Sites* и выбираем *Add Website...* (*Добавить Web-сайт...*), как показано на рис. 8.8.

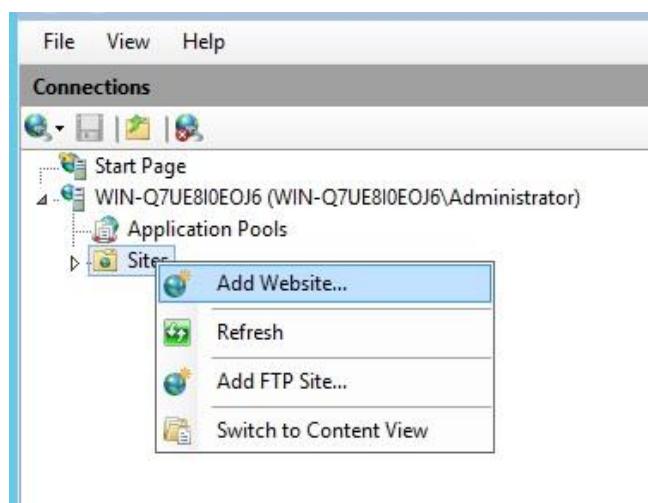


Рис. 8.8. Старт процедуры добавления нового web-узла

Далее в появившемся окне необходимо задать имя web-узла, а также физический путь к его домашней директории (каталогу) (рис. 8.9). Также следует выбрать IP-адрес web-узла (один из созданных дополнительных IP-адресов) (рис. 8.10).

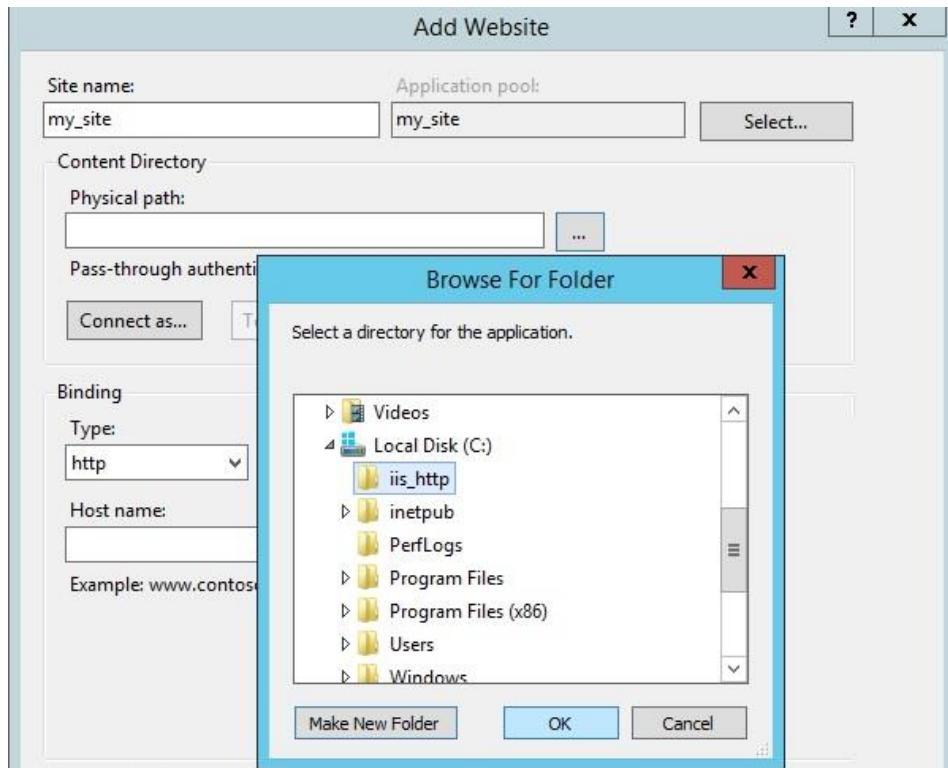


Рис. 8.9. Выбор имени web-узла и домашнего каталога

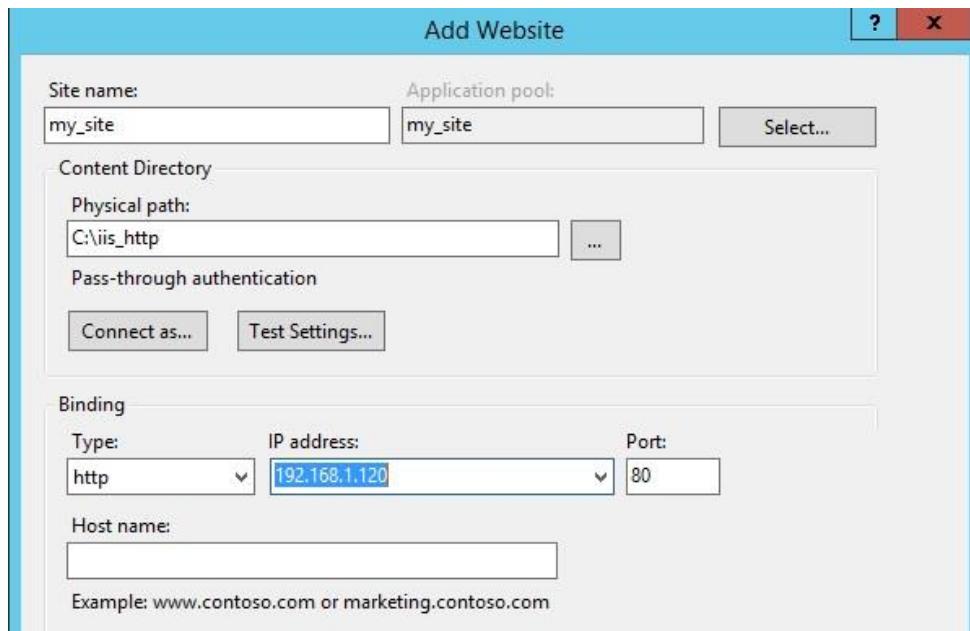


Рис. 8.10. Выбор IP-адреса web-узла

Далее необходимо будет отвечать утвердительно на всех последующих диалоговых окнах, т. е. выбирая *Next* (*Далее*). После завершения установки в консоли *IIS* в категории сайтов появится созданный web-узел с именем *my_site* (рис. 8.11). Проверить работу web-узла можно, написав в строке браузера соответствующий IP-адрес (рис. 8.12) (можно также добавить соответствующую запись в DNS-сервер, чтобы вызывать сайт по символьному имени).

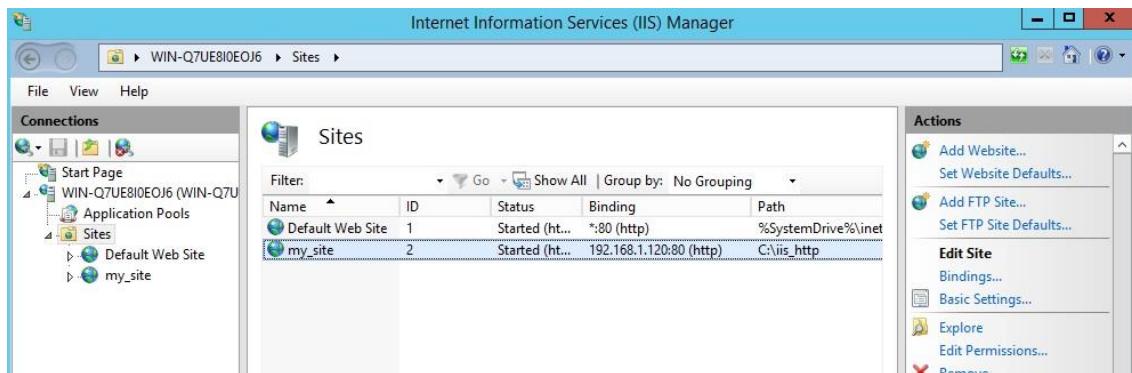


Рис. 8.11. Консоль *IIS*-сервера с созданным web-узлом *my_site*



Рис. 8.12. Просмотр сайта с помощью браузера

8.2.2. Настройка виртуальных каталогов

Виртуальный каталог – это понятное имя или псевдоним либо для физического каталога на жестком диске сервера, расположенного вне области основного каталога, либо для основного каталога на другом компьютере. Поскольку псевдоним обычно бывает короче, чем запись пути к физическому каталогу, он более удобен для ввода пользователем. Кроме того, использование псевдонимов повышает безопасность, так как не дает пользователям сведений о физическом местоположении файлов на сервере и тем самым исключает возможность изменения этих файлов пользователями. Псевдонимы упрощают также перемещение каталогов на узле. Вместо того чтобы изменять URL-адрес

для каталога, достаточно изменить сопоставление псевдонима и физического адреса каталога.

Если на веб-узле имеются файлы, размещенные вне основного каталога или на других компьютерах, то для добавления этих файлов на веб-узел следует создать виртуальный каталог. Данный метод можно применять также, например, для опубликования данных из каких-либо каталогов, размещенных вне основного каталога.

Виртуальные каталоги для web-узла бывают двух типов: локальные и сетевые. *Локальный виртуальный каталог* физически расположен на той же виртуальной машине (операционной системе), что и непосредственно web-сервер, обеспечивающий работу web-узла. *Сетевой виртуальный каталог* физически расположен на любом сетевом компьютере, но не на компьютере с web-сервером. При этом правила доступа к каталогам обоих типов для пользователей одинаковы – в строке браузера необходимо ввести *http://адрес web-узла/имя виртуального каталога*.

Для создания виртуального каталога необходимо перейти в консоль управления IIS-сервером (*Internet Information Services (IIS) Manager*) и в контекстном меню для web-узла выбрать *View Virtual Directories* (*Посмотреть виртуальные каталоги*) (рис. 8.13), тем самым мы перейдем в окно с имеющимися виртуальными каталогами для выбранного web-узла (в примере на рис. 8.14 виртуальные каталоги пока не создавались). Далее также через контекстное меню, вызываемое нажатием правой кнопки мыши, выбираем команду *Add Virtual Directory* (*Добавить виртуальный каталог*) (рис. 8.15), чтобы запустить мастер создания и настройки виртуальных каталогов, который представлен на рис. 8.16. В процессе создания (рис. 8.15) необходимо ввести имя каталога, под которым его будет видеть пользователь (имя виртуального каталога может не совпадать с физическим именем каталога), а также путь к нему. Для виртуального локального каталога вводится локальный путь, например, *C:\http_virtual\local* (показано на рис. 8.16), а в случае настройки сетевого виртуального каталога – сетевой путь *\\\имя компьютера\имя сетевой папки*, например *\\\192.168.1.200\net*, где 192.168.1.200 – IP-адрес компьютера, где физически расположен каталог, net – сетевое имя открытой для доступа по сети папки на компьютере с IP-адресом компьютера 192.168.1.200. Соответственно для данной папки должен предварительно открыт доступ по сети с установкой соответствующих прав для выбранного пользователя. В процессе настройки виртуального каталога необходимо будет ввести имя данного пользователя и его пароль (возможно

потребуется создание пользователя с совпадающим именем и паролем на компьютере с web-сервером, т. к. клиент, обращаясь к виртуальному каталогу, будет в него заходить фактически с правами данного пользователя).

Проверка работы виртуального каталога осуществляется, как показано на рис. 8.17 (виртуальный каталог с именем *local_directory* ассоциирован с web-узлом с адресом 192.168.1.120).

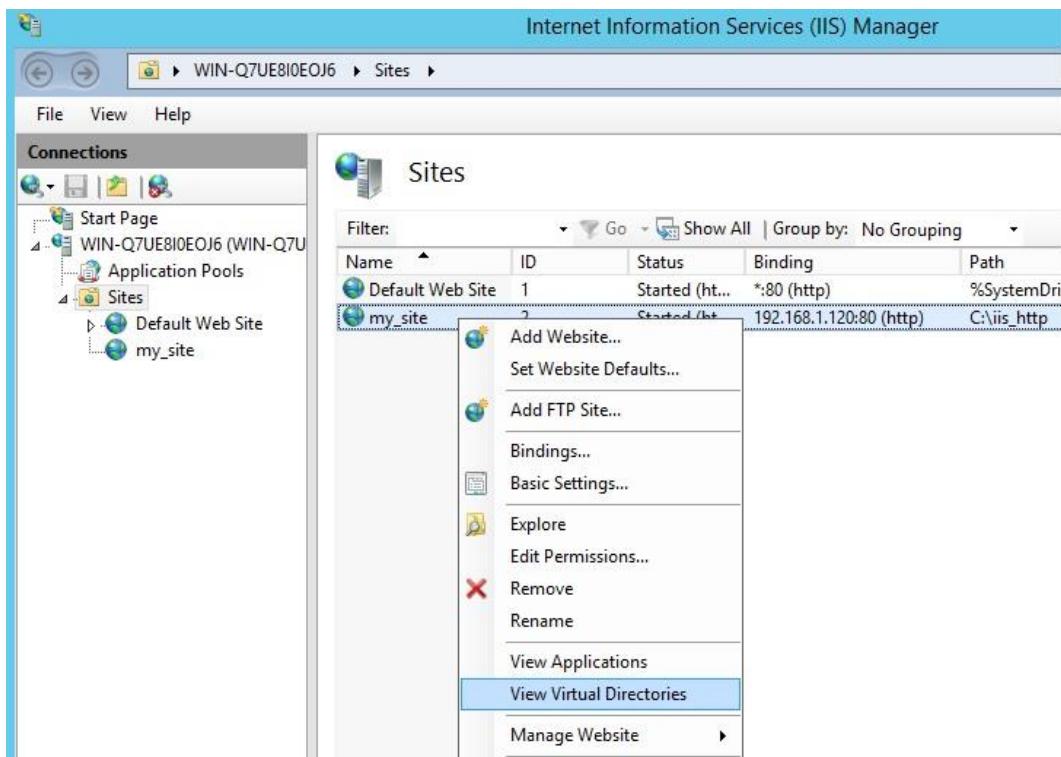


Рис. 8.13. Переход в окно с виртуальными каталогами web-узла



Рис. 8.14. Окно с виртуальными каталогами web-узла



Рис. 8.15. Запуск мастера для создания виртуального каталога web-узла

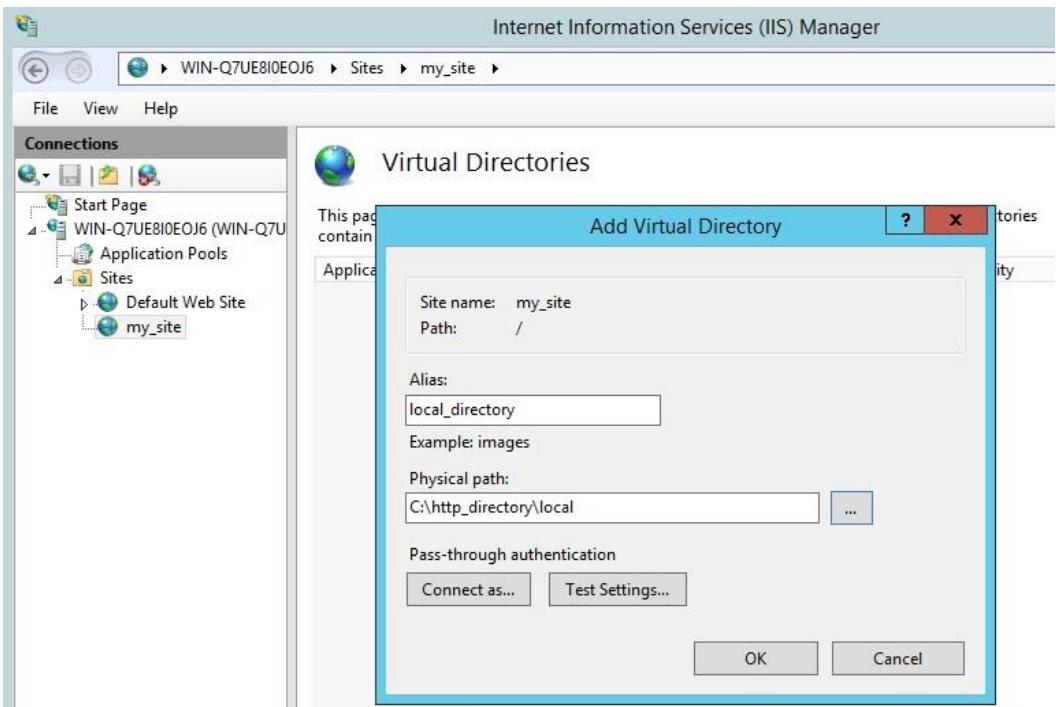


Рис. 8.16. Процесс создания виртуального каталога web-узла



Рис. 8.17. Проверка работы виртуального каталога web-узла

8.2.3. Лабораторная работа № 15

Цель: изучение методов настройки IIS-сервера для передачи данных по протоколу http.

Задание: настройка следующих элементов.

1. Web-узла с анонимным доступом, тем самым должен быть опубликован web-сайт, разработанный студентом.
2. Виртуального локального каталога для созданного web-узла с анонимным доступом).
3. Виртуального сетевого каталога для созданного web-узла с анонимным доступом).

Доступ к web-ресурсам (сайт, каталоги) необходимо демонстрировать с помощью браузера любого компьютера в сети (например, второй виртуальной машины).

8.3. Настройка и передача данных по протоколу FTP

Если протокол http прежде всего используется для доступа к web-ресурсам в виде отдельных web-страниц, то FTP предназначен для организации хранения и передачи различных типов файлов (графических, текстовых, мультимедийных и т. д.). На базе данного протокола как правило организуются файловые хранилища.

Web-сервер IIS позволяет организовывать и настраивать FTP-узлы трех типов:

- с гостевым (анонимным) доступом;
- с авторизированным доступом;
- с изоляцией пользователей.

Далее рассмотрим процесс создания и использования каждого типа узла отдельно.

Отметим, что служба FTP, как часть веб сервера IIS, уже была установлено совместно с HTTP составляющей, как показано в подразделе 8.2.

8.3.1. Создание FTP-узла с гостевым (анонимным) и авторизированным доступом

Для запуска мастера настройки ftp необходимо в консоли управления сервером IIS вызвать контекстное меню (для веб-сервера IIS) и выполнить команду Add FTP Site... (Добавить FTP-сайт), как показано на рис. 8.18.

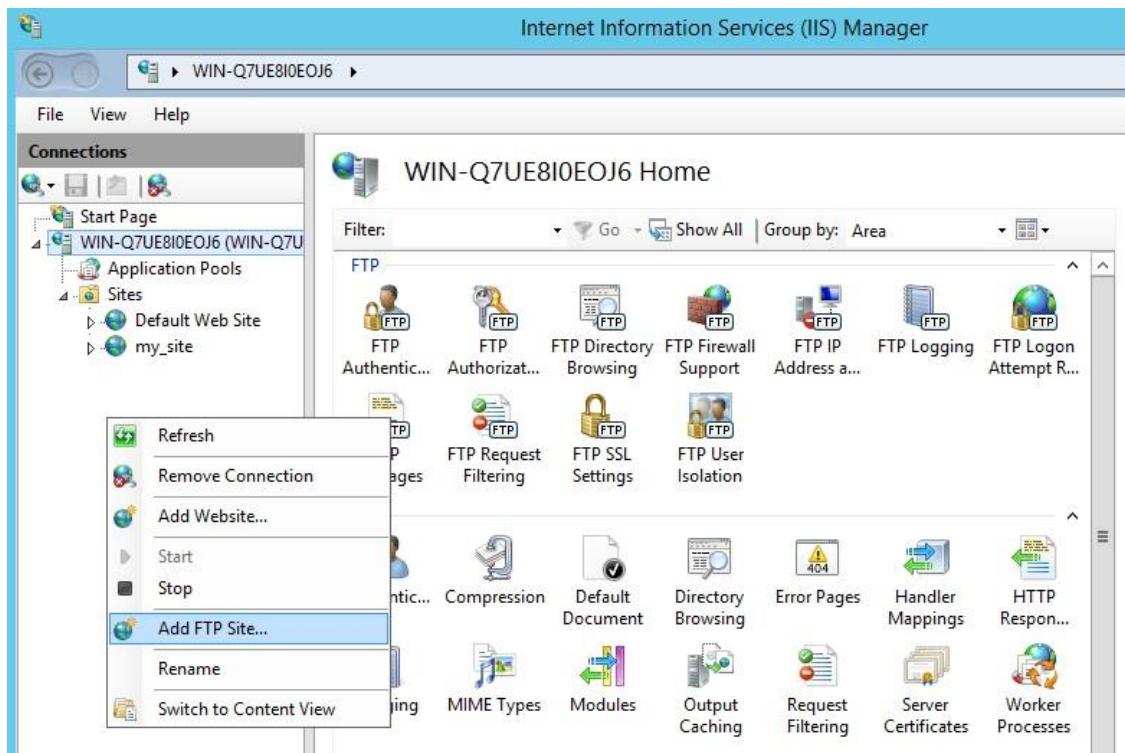


Рис. 8.18. Вызов мастера настройки службы FTP

Далее необходимо по аналогии с настройкой web-узла заполнить с ходу диалога с «мастером» соответствующие поля: выбрать имя ftp-узла и путь к его домашней директории (рис. 8.19 и 8.20); выбрать IP-адрес узла из доступных в выпадающем списке и способ шифрования передаваемых данных (на рис. 8.21 выбран вариант без SSL); способ аутентификации пользователя (рис. 8.22 – выбран вариант анонимной аутентификации, т. е. гостевой доступ). При необходимости можно также определить вариант доступа (полный, чтение) к ftp-узлу. Отметим, что такая настройка будет достаточно «грубой», поэтому целесообразно ее выполнять через управление доступом к домашней директории ftp-узла в файловой системе NTFS (как было показано в разделе 2.4).

После того, как все будет настроено, ftp-узел появится в консоли управления IIS-сервера, как показано на рис. 8.23. Подключиться же к ресурсам данного ftp-узла можно через браузер, введя в адресной строке соответствующий IP-адрес (рис. 8.24 и 8.25).

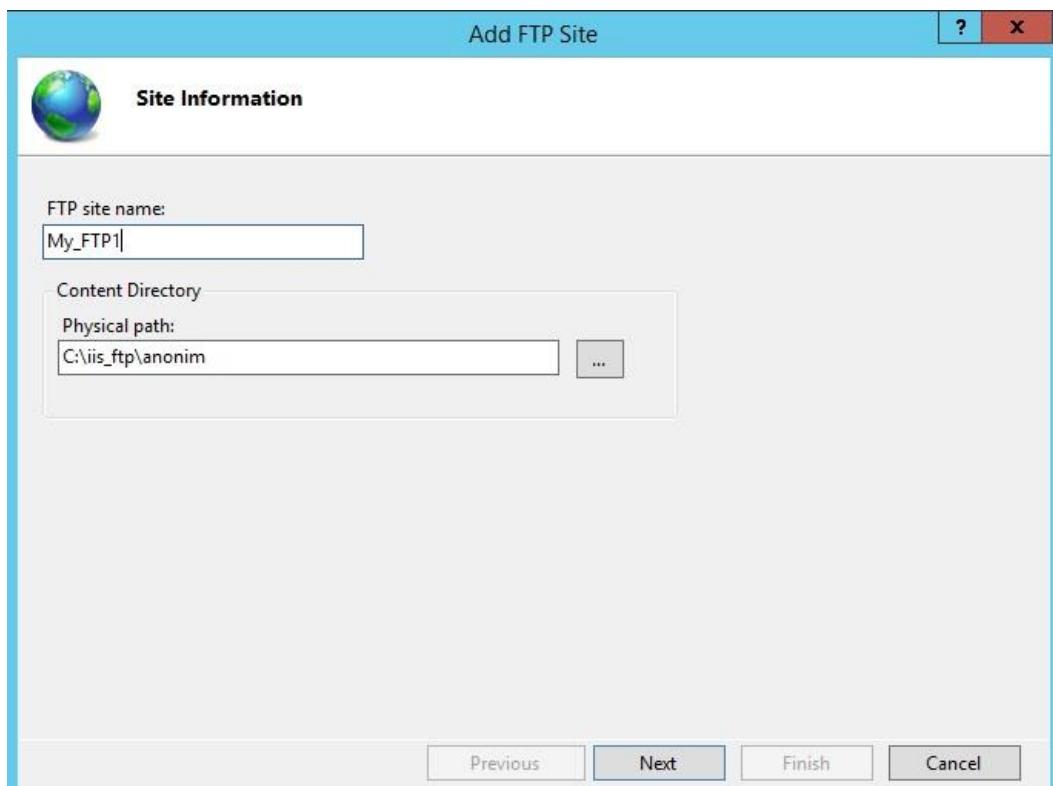


Рис. 8.19. Диалог для ввода имени и домашней директории ftp-узла

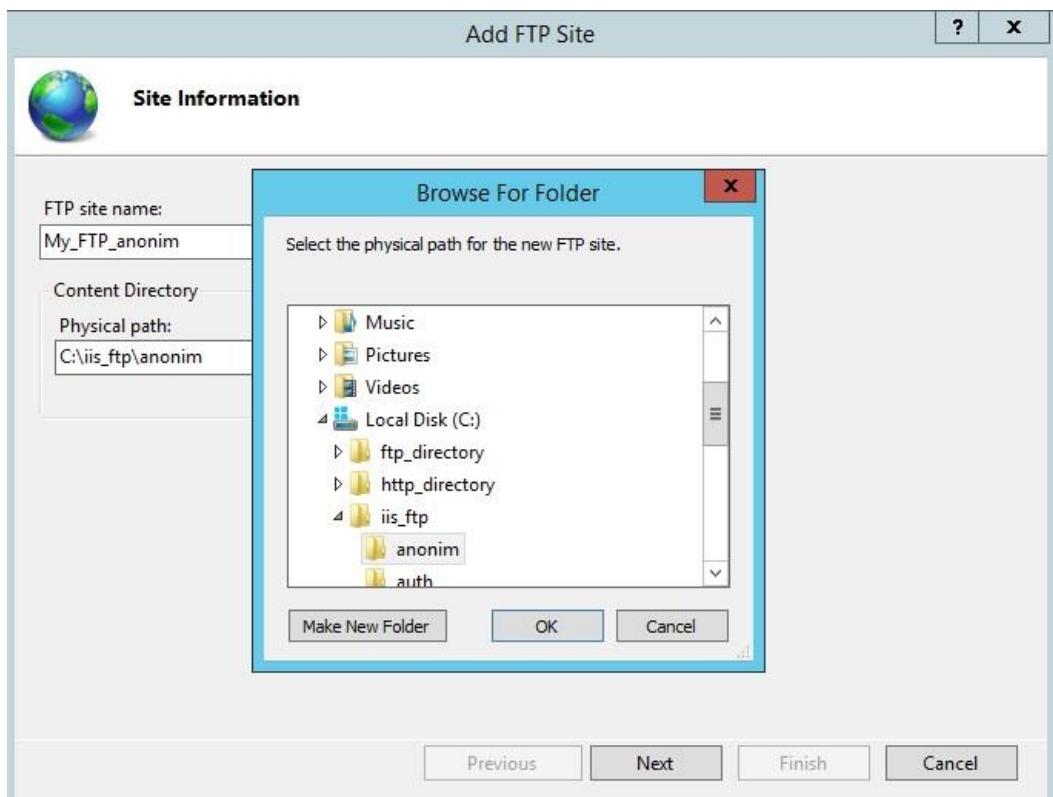


Рис. 8.20. Диалог для пути к домашней директории ftp-узла

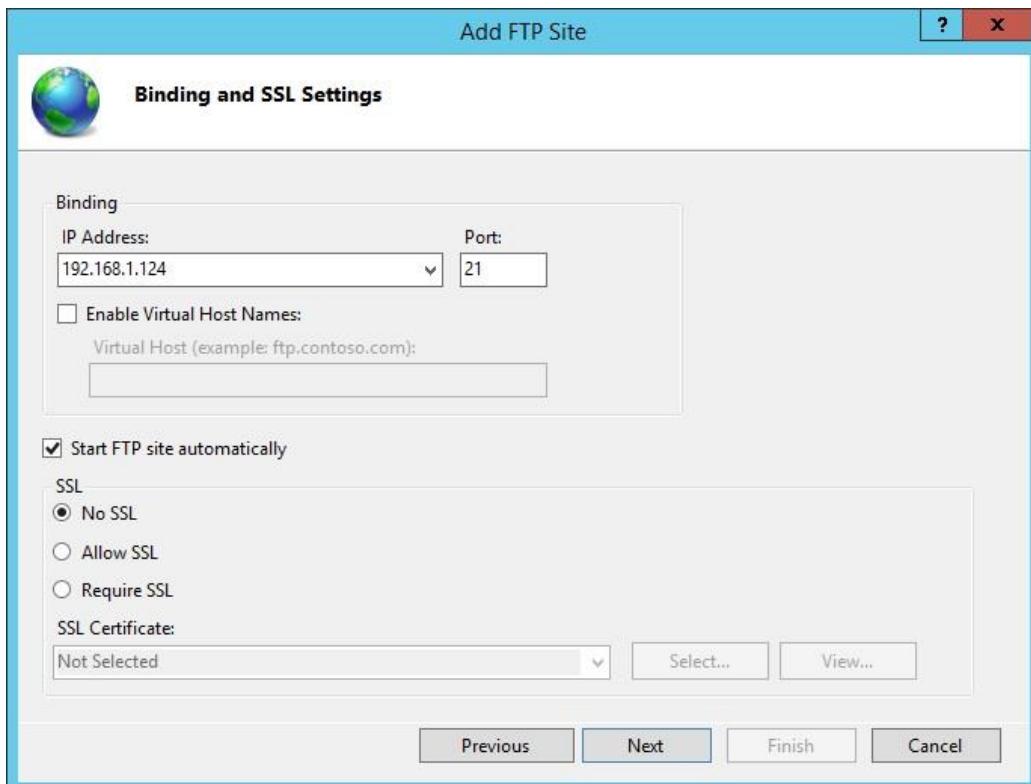


Рис. 8.21. Выбор IP-адреса ftp-узла

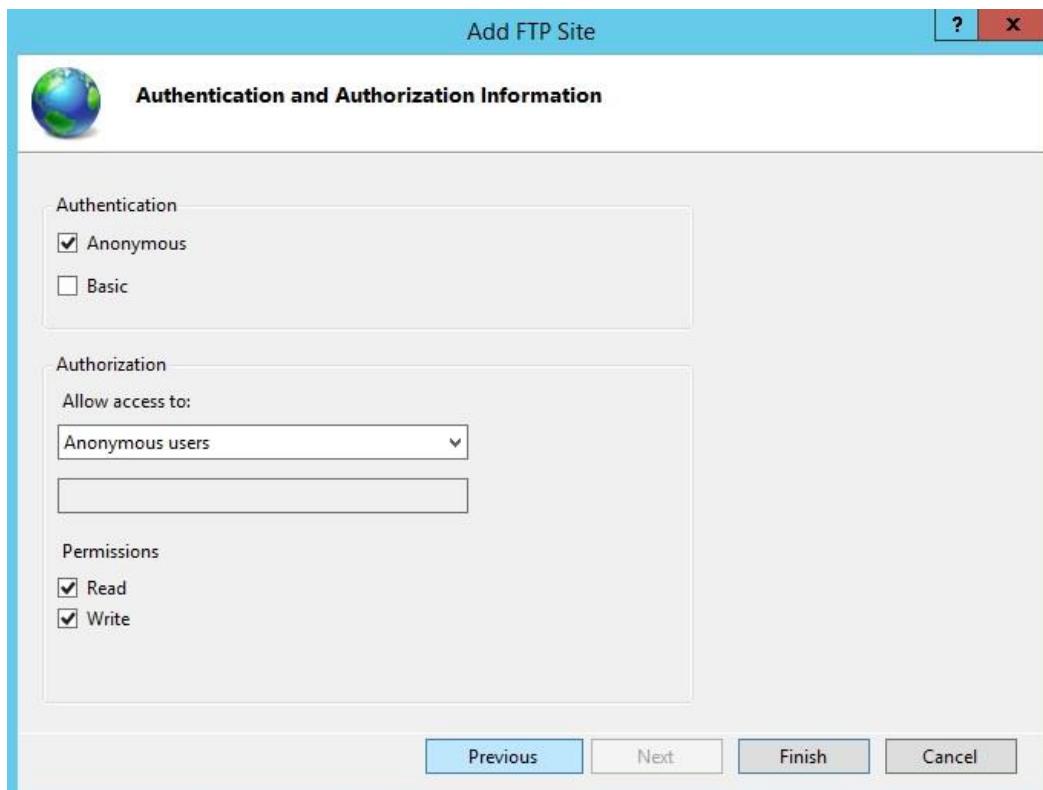


Рис. 8.22. Выбор способа аутентификации при подключении к ftp-узлу

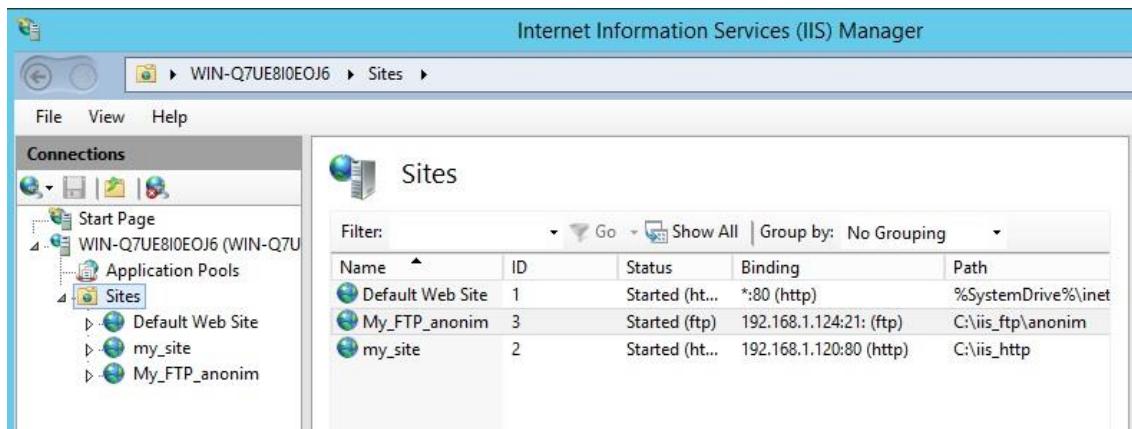


Рис. 8.23. Общий вид консоли управления IIS-сервером с настроенным ftp-узлом



Рис. 8.24. Подключение к ftp-узлу через браузер

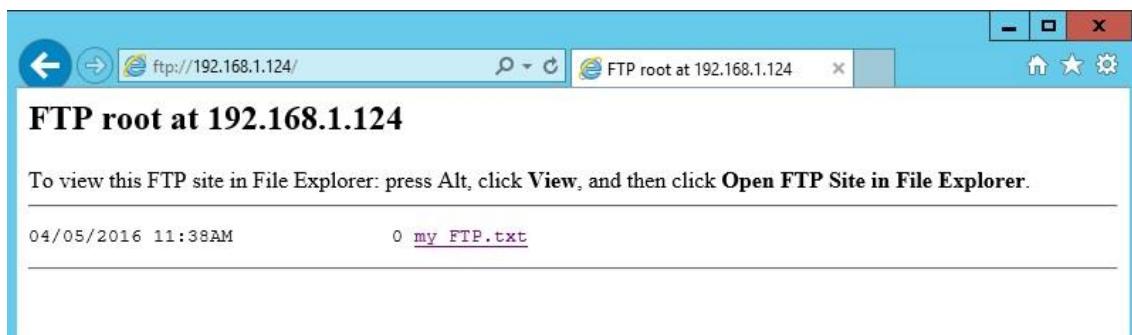


Рис. 8.25. Просмотр содержимого ftp-узла

Отметим, что именно «тонкая» настройка доступа к домашней директории может позволить создавать ftp-узлы с «интересным» функционалом, например «черный ящик». Подключения к такому ресурсу осуществляются под гостевой учетной записью (без запроса логина и пароля) таким образом, чтобы пользователь мог записывать файлы, но не видел содержимого «черного ящика». То есть для госте-

вой учетной записи IIS-сервера должна применяться настройка, показанная на рис. 8.26. При этом целесообразно оставить доступ к домашней директории только гостевой учетной записи IIS-сервера и группе администраторов (рис. 8.27). Поэтому если осуществить подключение под логином пользователя, имеющего административные права (принадлежит группе *Администраторы*), то ему будет доступна вся информация, хранящаяся на данном ftp-узле, а также любые операции с ней (запись, удаление, изменение, просмотр и т. д.)

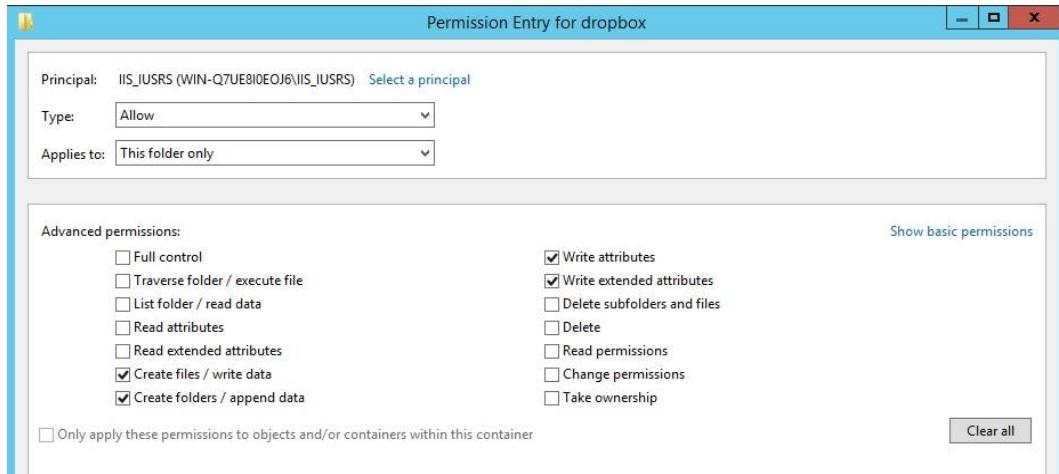


Рис. 8.26. Настройка доступа к домашней директории для ftp-узла типа «черный ящик» для гостевой учетной записи IIS-сервера

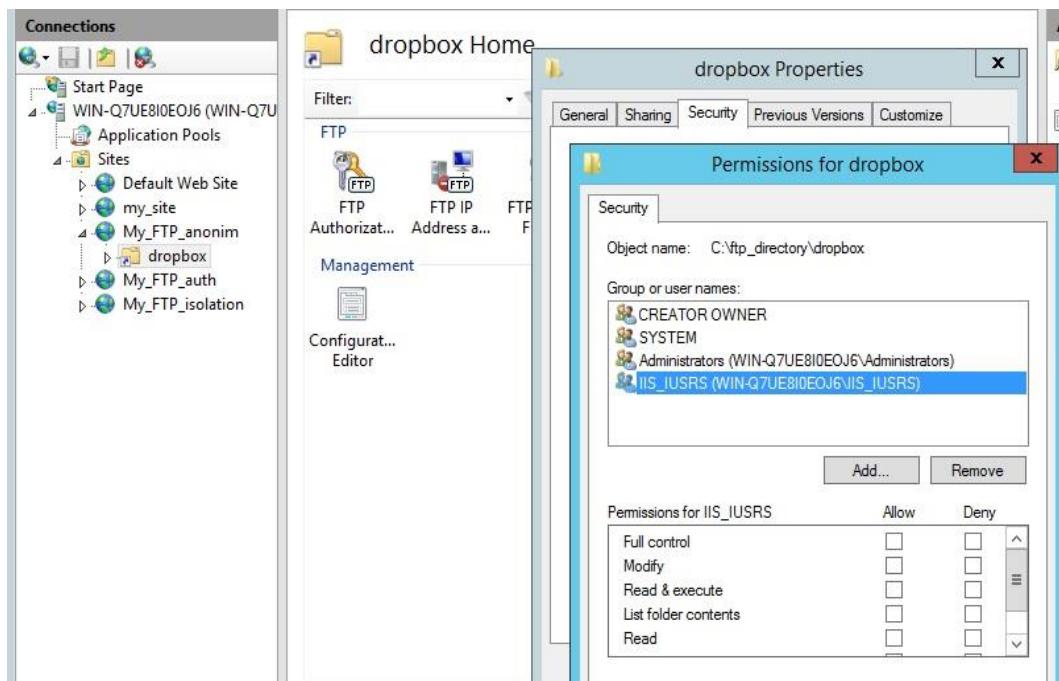


Рис. 8.27. «Тонкая» настройка доступа к домашней директории для ftp-узла типа «черный ящик»

Процесс создания ftp-узла с авторизированным доступом осуществляется аналогичным образом с той лишь разницей, что необходимо выбрать тип авторизации basic и указать пользователей, которым будет разрешен доступ (рис. 8.28). Данные пользователи должны быть предварительно созданы, как показано на рис. 8.29– 8.31, а также выполнена «тонкая» настройка доступа данным пользователям к домашнему каталогу через файловую систему NTFS. При доступе к ftp-узлу с авторизацией (например, через браузер) обязательно будет осуществлен запрос на ввод логина и пароля.

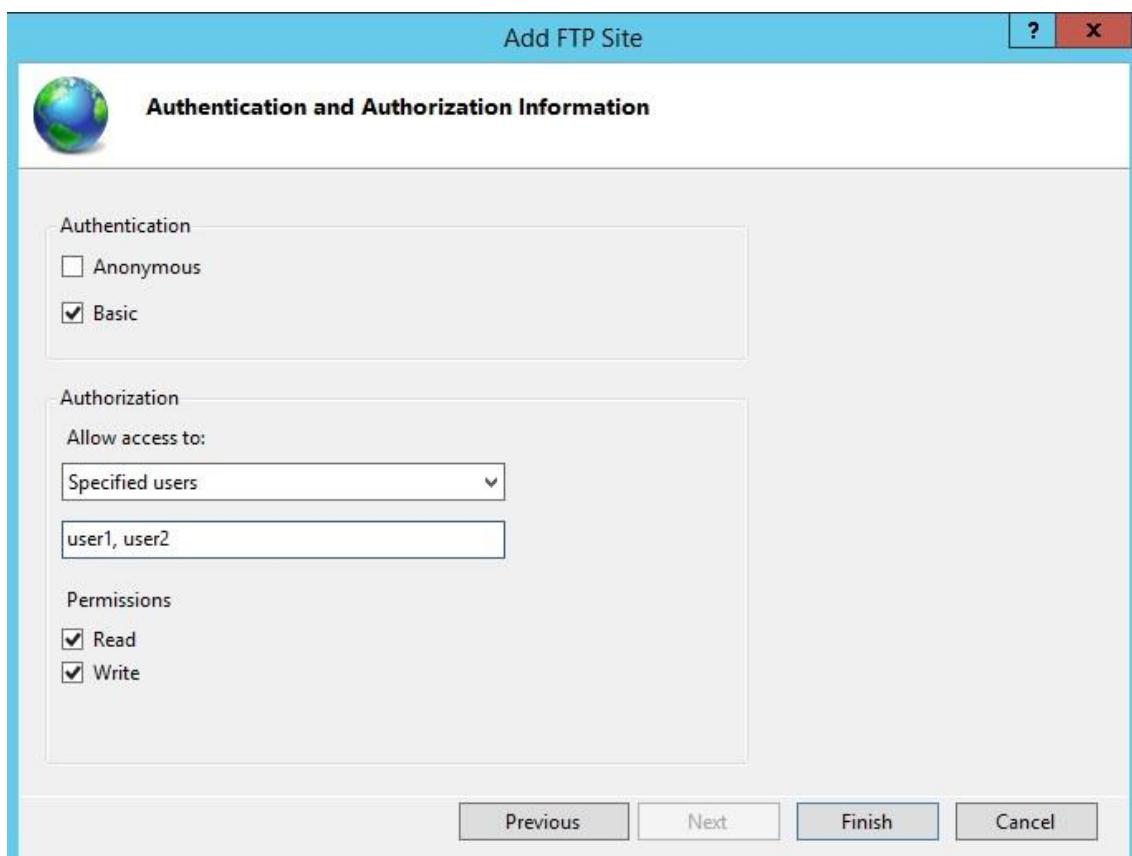


Рис. 8.28. Выбор способа аутентификации при подключении к ftp-узлу для пользователей user1 и user2

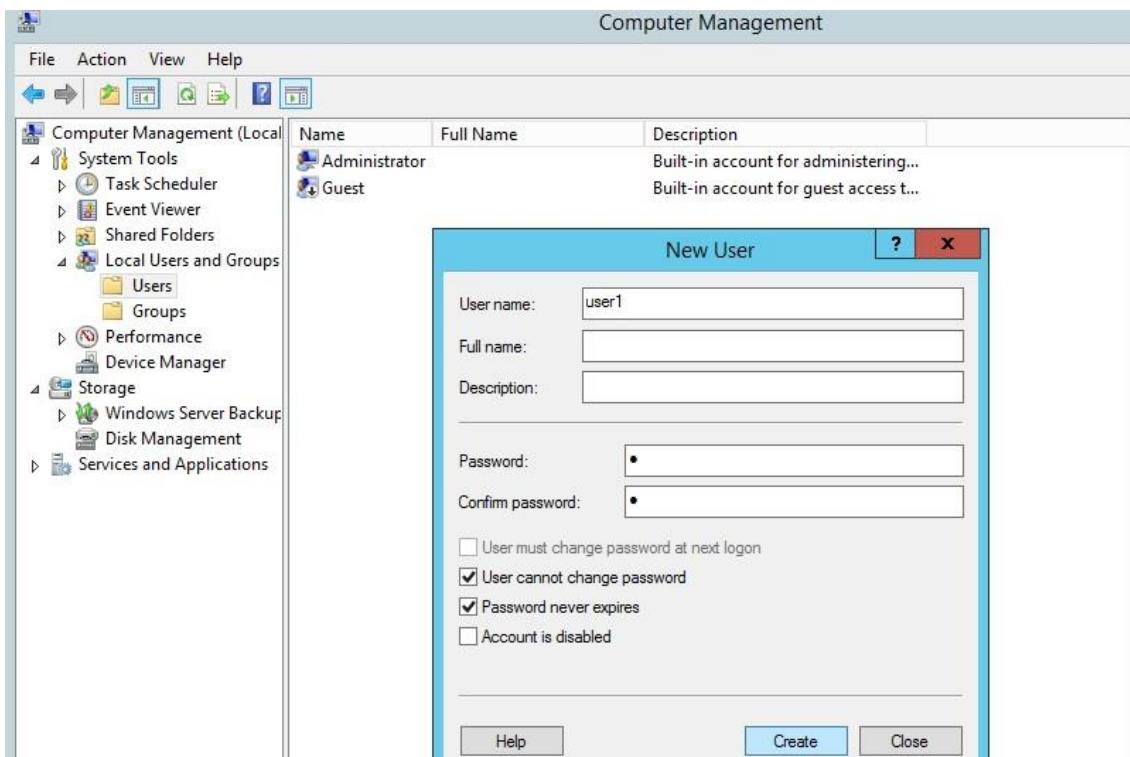


Рис. 8.29. Создание пользователя user1

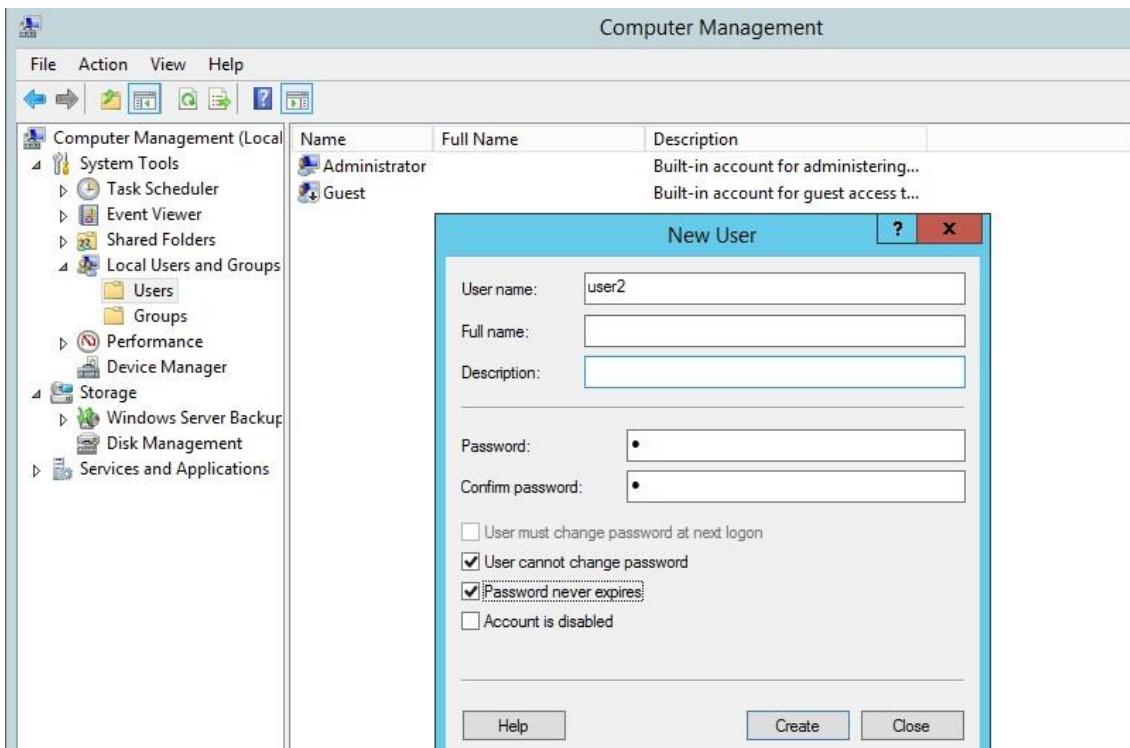


Рис. 8.30. Создание пользователя user2

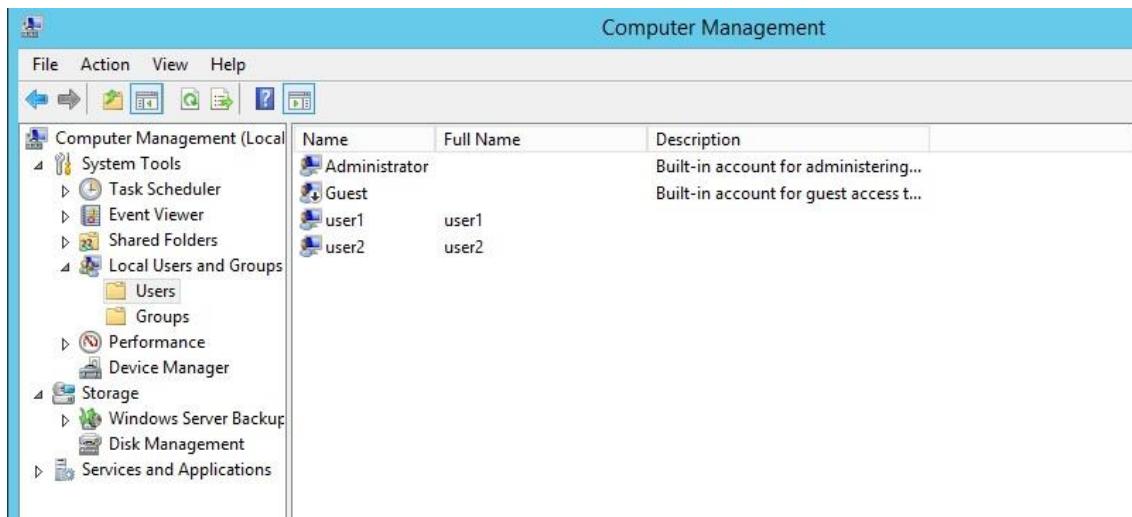


Рис. 8.31. Окно управления компьютером с созданными пользователями user1 и user2

8.3.2. Создание FTP-узла с изоляцией пользователей

Создание ftp-узла с изоляцией предполагает первоначально создание ftp-узла с авторизацией (как описано выше) с дальнейшей ее настройкой. Вызывать окно настройки (рис. 8.30) можно, предварительно выбрав ftp-узел в левой части окна с консолью управления IIS сервером и далее в правой ее части щелкнув по иконке *Ftp User Isolation* (рис. 8.31). Отметим, что для такого типа узла необходимо создать структуру каталогов в домашней директории ftp-узла (в примере на рис 8.32 это *isolation*), как показано на рис. 8.32 (имена папок в папке *localuser* должны точно совпадать с именами пользователей).

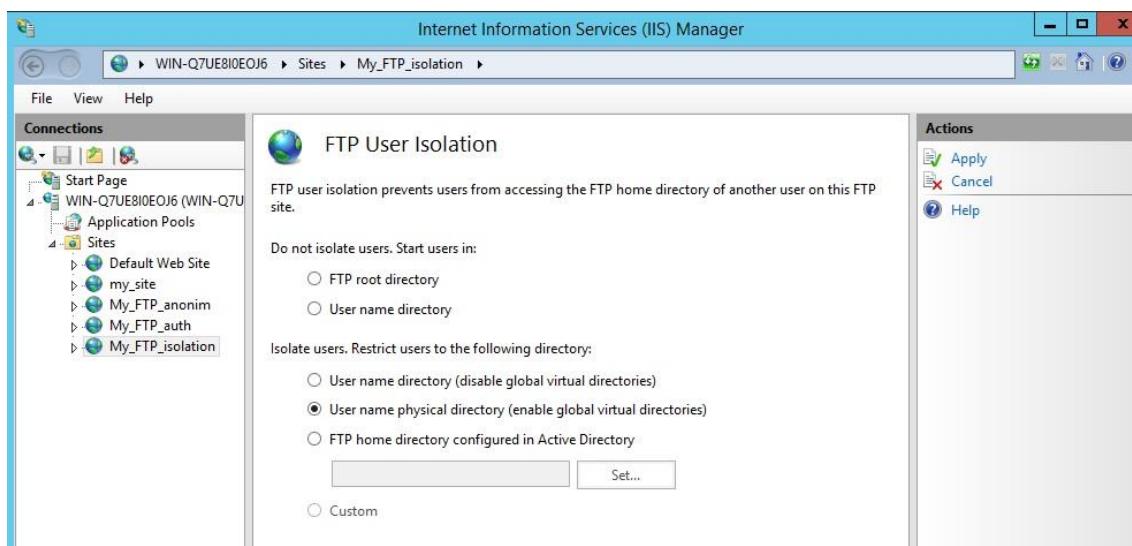


Рис. 8.32. Настройка ftp-узла с изоляцией пользователей

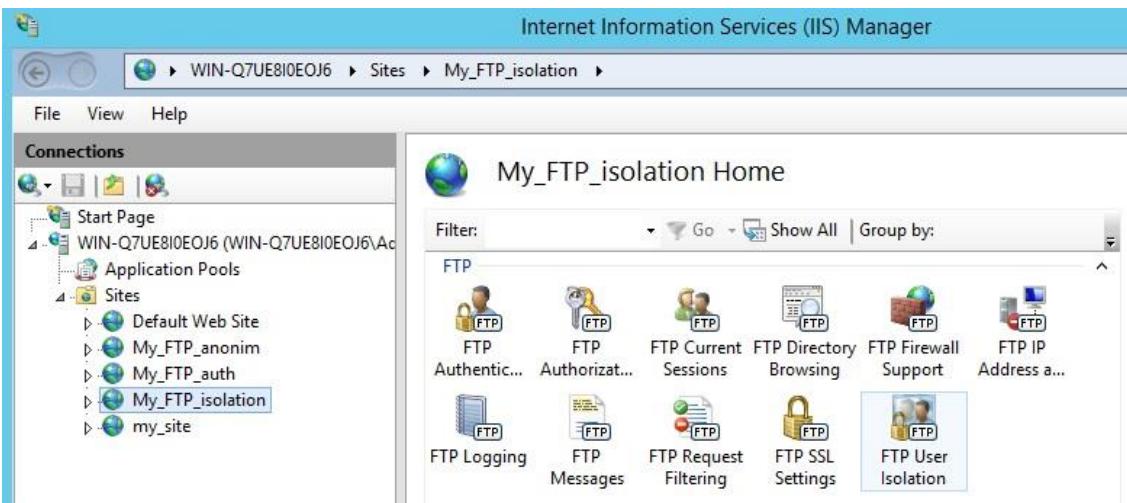


Рис. 8.33. Запуск мастера настройки ftp-узла с изоляцией пользователей

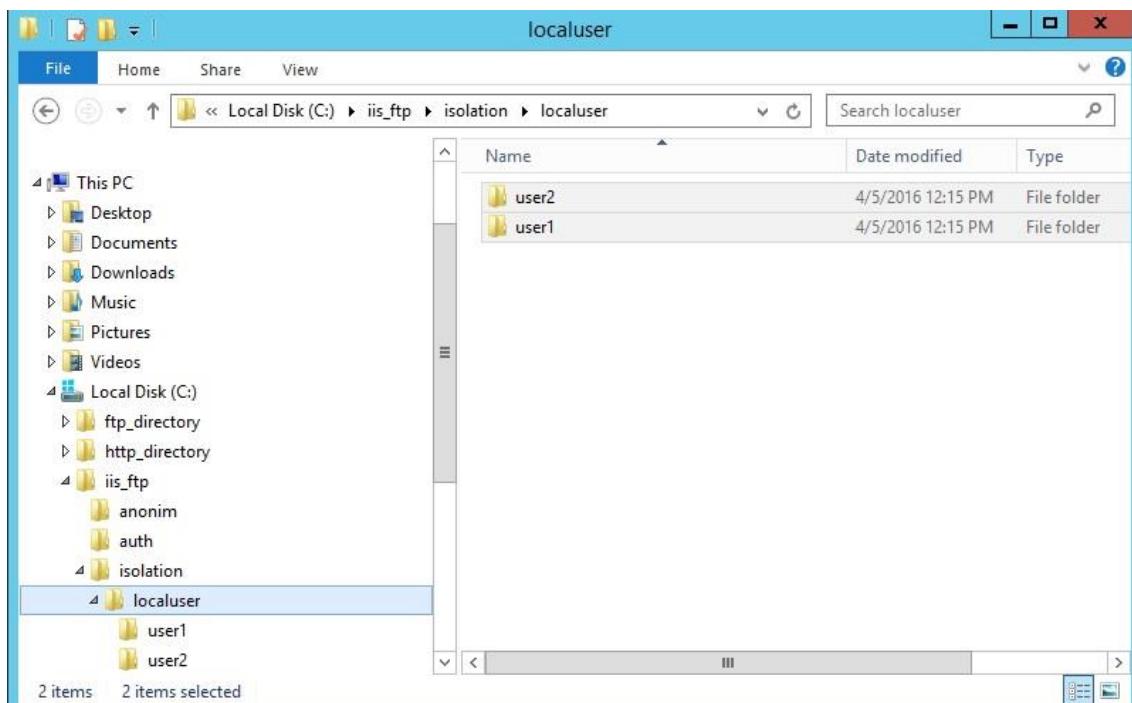


Рис. 8.34. Структура папок для ftp-узла с изоляцией
(для двух пользователей user1 и user2)

8.3.3. Лабораторная работа № 16–17

Цель: изучение методов настройки ПС-сервера для передачи данных по протоколу ftp.

Задание: настройка следующих элементов.

1. Ftp-узел с гостевым (анонимным) доступом с возможностью записи информации, но запретом на удаление.

2. Ftp-узел с авторизированным доступом для двух пользователей: первый пользователь получает полный доступ, а второй – только чтение.

3. Ftp-узел с изоляцией пользователей. Все пользователи получают полный доступ, но только к своей папке (папке с именем пользователя).

4. Ftp-узел, работающий по принципу «черного ящика». Подключения к данному ресурсу осуществляются под гостевой учетной записью (без запроса логина и пароля) таким образом, чтобы пользователь мог записывать файлы, но не видел содержимого «черного ящика». Если осуществить подключение под логином пользователя, имеющего административные права (полный доступ), то ему будет доступна вся информация, хранящаяся на данном ftp-узле, а также любые операции с ней (запись, удаление, изменение, просмотр и т. д.)

Доступ к ftp-ресурсам необходимо демонстрировать с помощью браузера (либо другого ftp-клиента, например total commander) любого компьютера в сети (например, второй виртуальной машины).

СОДЕРЖАНИЕ

| | |
|---|-----|
| ПРЕДИСЛОВИЕ..... | 3 |
| 1. МОДЕЛИРОВАНИЕ КОМПЬЮТЕРНЫХ СЕТЕЙ | 4 |
| 1.1. Понятие и виды топологии | 4 |
| 1.1.1. Топология «общая шина» | 4 |
| 1.1.2. Кольцевая топология..... | 5 |
| 1.1.3. Топология типа «звезда»..... | 6 |
| 1.1.4. Древовидные топологии | 8 |
| 1.1.5. Комбинированные топологии | 9 |
| 1.1.6. Ячеистые топологии..... | 11 |
| 1.1.7. Многозначность понятия топологии | 12 |
| 1.2. Использование пакета NetCracker для моделирования компьютерных сетей | 14 |
| 1.3. Лабораторная работа № 1 | 17 |
| 2. СТАТИЧЕСКАЯ IP-АДРЕСАЦИЯ В СЕТЯХ С ОДНОРАНГОВОЙ АРХИТЕКТУРОЙ | 19 |
| 2.1. Понятие одноранговой архитектуры | 19 |
| 2.2. Виртуализация операционных систем..... | 20 |
| 2.2.1. Настройка аппаратной части виртуальной машины | 25 |
| 2.2.2. Настройка операционной системы виртуальной машины | 36 |
| 2.3. Организация одноранговой сети | 39 |
| 2.4. Распределение ресурсов по сети | 45 |
| 2.5. Лабораторная работа № 2–3 | 53 |
| 3. ДИНАМИЧЕСКАЯ IP-АДРЕСАЦИИ В СЕТЯХ С КЛИЕНТ-СЕРВЕРНОЙ АРХИТЕКТУРОЙ | 56 |
| 3.1. Понятие архитектуры клиент – сервер | 56 |
| 3.2. Настройка динамической адресации в клиент-серверных компьютерных сетях | 58 |
| 3.3. Лабораторная работа № 4–5 | 70 |
| 4. СИМВОЛЬНАЯ АДРЕСАЦИЯ В СЕТЯХ С КЛИЕНТ-СЕРВЕРНОЙ АРХИТЕКТУРОЙ | 72 |
| 4.1. Символьный адрес DNS | 72 |
| 4.2. Символьный адрес NetBIOS | 75 |
| 4.3. Настройка DNS-сервера..... | 76 |
| 4.3. Лабораторная работа № 6 | 84 |
| 5. ДИАГНОСТИКА TCP/IP И DNS | 86 |
| 5.1. Утилиты диагностики TCP/IP и DNS | 86 |
| 5.2. Разработка программы эхо-запроса для диагностики TCP/IP соединения.. | 92 |
| 5.3. Лабораторная работа № 7 | 100 |
| 6. АДРЕСАЦИЯ В СЛОЖНОСТРУКТУРИРОВАННЫХ СЕТЯХ | 101 |
| 6.1. Классы IP-адресов | 101 |
| 6.2. Использование масок | 103 |
| 6.3. Особые IP-адреса | 106 |
| 6.4. Разработка программы определения ID подсети и ID хоста по заданному IP-адресу и маске подсети | 106 |

| | |
|--|------------|
| 6.5. Лабораторная работа № 8 | 110 |
| 7. РАЗРЕШЕНИЕ АДРЕСОВ В IP-СЕТЯХ..... | 112 |
| 7.1. Определение MAC-адреса удаленного хоста по IP-адресу..... | 112 |
| 7.1.1. Физический адрес | 112 |
| 7.1.2. Разработка программы определения MAC-адреса компьютера по IP-адресу и наоборот | 113 |
| 7.1.3. Лабораторная работа № 9–10 | 126 |
| 7.2. Определение символьного адреса удаленного хоста по IP-адресу | 126 |
| 7.2.1. Разработка программы определения имени (DNS, NetBIOS) компьютера по IP-адресу, и наоборот | 126 |
| 7.2.2. Лабораторная работа № 11–12 | 138 |
| 7.3. Определение MAC-адреса удаленного хоста по символьному адресу | 138 |
| 7.3.1. Разработка программы определения MAC-адреса по DNS (NetBIOS) имени | 138 |
| 7.3.2. Лабораторная работа № 13–14 | 141 |
| 8. ПЕРЕДАЧА ИНФОРМАЦИИ С ИСПОЛЬЗОВАНИЕМ ПРОТОКОЛОВ HTTP И FTP | 142 |
| 8.1. Установка IIS-сервера | 142 |
| 8.2. Настройка и передача данных по протоколу HTTP. Web-узлы и виртуальные каталоги | 144 |
| 8.2.1. Настройка web-узлов..... | 144 |
| 8.2.2. Настройка виртуальных каталогов | 148 |
| 8.2.3. Лабораторная работа № 15 | 152 |
| 8.3. Настройка и передача данных по протоколу FTP | 152 |
| 8.3.1. Создание FTP-узла с гостевым (анонимным) и авторизированным доступом | 152 |
| 8.3.2. Создание FTP-узла с изоляцией пользователей | 160 |
| 8.3.3. Лабораторная работа № 16–17 | 162 |
| ПРИЛОЖЕНИЕ | 168 |
| ЛИТЕРАТУРА | 165 |

ПРИЛОЖЕНИЕ

Следующая таблица описывает возможные коды ошибок, которые возвращает функция WSAGetLastError. Ошибки перечислены в возрастающем порядке. Некоторые коды ошибок, указанные в заголовочном файле Winsock2.h, не возвращаются ни одной из функций.

Коды ошибок Winsock

| Код возврата/ значение | Описание |
|---------------------------|---|
| WSA_INVALID_HANDLE/6 | Указан неверный объект события. Ошибка появляется при использовании функций Winsock, соответствующих функциям Win32 |
| WSA_NOT_ENOUGH_MEMORY/8 | Недостаточно памяти |
| WSA_INVALID_PARAMETER/87 | Один или несколько неверных параметров |
| WSA_OPERATION_ABORTED/995 | Перекрытая операция прервана |
| WSA_IO_INCOMPLETE/996 | Объект события для перекрытого ввода/вывода не свободен |
| WSAEBADF/10009 | Неверный дескриптор файла. Такую ошибку может возвратить функция <code>socket</code> , она означает, что общий последовательный порт занят |
| WSAEACCES/10013 | Доступ запрещен. Была сделана попытка обращения к запрещенному сокету. Обычно эта ошибка возникает при попытке использовать широковещательный адрес в функциях <code>sendto</code> или <code>WSASendTo</code> |
| WSAEFAULT/10014 | Недопустимый адрес. Функции Winsock передан недопустимый указатель адреса. Ошибка также возникает, когда указан слишком маленький буфер |
| WSAEINVAL/10022 | Недопустимый параметр |
| WSAEMFILE/10024 | Открыто слишком много файлов. Открыто слишком много сокетов |
| WSAEWOULDBLOCK/10035 | Ресурс временно недоступен. Ошибка наиболее часто возвращается неблокирующими сокетами, на которых запрошеннную операцию (например, вызов функции <code>connect</code>) нельзя выполнить немедленно |

| Код возврата/ значение | Описание |
|-----------------------------------|---|
| WSAEMSGSIZE/10040 | Слишком длинное сообщение. Сообщение отправляется на дейтаграммный сокет размером больше внутреннего буфера. Ошибка также возникает, если сообщение больше, чем позволяют ограничения сети. Еще одна причина появления данной ошибки — буфер слишком мал, чтобы вместить полученную дейтаграмму |
| WSAEPROTOTYPE / 10041 | Неверный тип протокола для сокета. При вызове <code>socket</code> или <code>WSASocket</code> указан протокол, не поддерживающий семантику данного типа сокета |
| WSAENOPROTOOPT /10042 | Неверный параметр протокола. Неизвестный, неподдерживаемый или неверный параметр сокета |
| WSAEPROTOONSUPPORT/ 10043 | Неподдерживаемый протокол. Запрошенный протокол не установлен в системе. Например, попытка создать TCP- или UDP-сокет, при том, что в системе не установлен протокол TCP/IP |
| WSAESOCKTNOSUPPORT/ 10044 | Неподдерживаемый тип сокета |
| WSAEOPNOTSUPP/10045 | Не поддерживаемая операция. Обычно возникает при попытке вызвать функции Winsock на сокете, не поддерживающем данную операцию |
| WSAEPFNOSUPPORT/10046 | Неподдерживаемое семейство протоколов |
| WSAEAFNOSUPPORT/10047 | Семейство адресов не поддерживает запрошенную операцию. Ошибка возникает при вызове <code>socket</code> или <code>WSASocket</code> с указанием неверной комбинации семейства адресов, типа сокета и протокола |
| WSAEADDRINUSE/10048 | Адрес уже используется. При обычных обстоятельствах только один сокет может использовать каждый адрес сокета. Например, адрес IP-сокета состоит из локального IP-адреса и номера порта. Ошибка обычно связана с функциями <code>bind</code> , <code>connect</code> и <code>WSAConnect</code> |
| WSAEADDRNOTAVAIL/10049 | Невозможно назначить запрошенный адрес. Ошибка может возникнуть при указании порта 0 удаленной машины, с которой производится соединение, в функциях <code>connect</code> , <code>WSAConnect</code> , <code>sendto</code> , <code>WSASendTo</code> |

| Код возврата/ значение | Описание |
|-----------------------------------|--|
| WSAENETDOWN / 10050 | Сеть не работает. Операция не может быть выполнена из-за неполадок в сети, стеке, сетевом интерфейсе или с локальной сети |
| WSAENETUNREACH / 10051 | Сеть недоступна. Попытка произвести операцию в недоступной сети: локальный узел не знает, как достичь удаленный |
| WSAENETRESET / 10052 | Сеть разорвала соединение при сбросе |
| WSAECONNABORTED /10053 | Преждевременный разрыв соединения из-за ошибки ПО |
| WSAECONNRESET/10054 | Соединение разорвано партнером по связи |
| WSAENOBUFS/10055 | Свободное место в буфере закончилось |
| WSAEISCONN/10056 | С сокетом уже установлено соединение. Попытка установить соединение с сокетом, который уже используется. Ошибка может произойти как на дейтаграммных, так и на потоковых сокетах |
| WSAENOTCONN/10057 | Нет соединения с сокетом |
| WSAESHUTDOWN/10058 | Отправление невозможно после отключения сокета |
| WSAETIMEDOUT /10060 | Время ожидания операции истекло |
| WSAECONNREFUSED/10061 | В соединении отказано |
| WSASYSNOTREADY/10091 | Подсистема сети недоступна |
| WSAVERNOTSUPPORTED/10092 | Некорректная версия Winsock.dll |
| WSANOTINITIALISED/10093 | Winsock не инициализирован. Вызов WSASStartup не удался |
| WSAEDISCON/10101 | Идет корректное завершение работы |
| WSAENOMORE/10102 | Более записей не найдено |
| WSAECANCELLED/10103 | Операция отменена |
| WSASERVICE_NOT_FOUND/10108 | Не найдено такой службы |
| WSATYPE_NOT_FOUND / 10109 | Не найден тип класса |
| WSA_E_NO_MORE/10110 | Записей более не найдено |
| WSA_E_CANCELLED/10111 | Операция отменена |
| WSAEREFUSED/10112 | Запрос отклонен |
| WSAHOST_NOT_FOUND/11001 | Узел не найден. Ошибку возвращают gethostbyname и gethostbyaddr |
| WSATRY AGAIN/11002 | Неполномочный узел не найден |
| WSANO_RECOVERY/11003 | Произошла неустранимая ошибка |
| WSANO_DATA /1100 | Не найдено записей данных запрошенного типа |

ЛИТЕРАТУРА

1. Урбанович, П. П. Компьютерные сети: учебное пособие для студентов специальности 1-40 01 02 «Информационные системы и технологии» / П. П. Урбанович, Д. М. Романенко, Е. В. Кабак. – Минск: БГТУ, 2011. – 350 с.
2. Олифер, В. Г. Компьютерные сети. Принципы, технологии, протоколы: учебник для ВУЗов / В. Г. Олифер, Н. А. Олифер. – 2-е изд. – СПб.: Питер, 2004.
3. Танненбаум, Э. Компьютерные сети / Э. Танненбаум. – СПб.: Питер, 2002.
4. Кульгин, М. В. Компьютерные сети. Практика построения. Для профессионалов / М. В. Кульгин. – 2-е изд. – СПб.: Питер. 2003.
5. Джонс, Э. Программирование в сетях Microsoft Windows. Мастер-класс / Э. Джонс, Д. Оланд; пер. с англ. – Спб.: Питер; М.: Издательско-торговый дом «Русская Редакция», 2002. – 608 с.
6. Network Protocols: Windows Sockets [Электронный ресурс]. – URL: <http://msdn.microsoft.com>. Data of access: 12.02.2011.
7. Верма, Р. Д. Справочник по Win32 API / Р. Д. Верма. –М.: Горячая линия, 2005. – 551 с.
8. Побегайло, А. П. Системное программирование в Windows / А. П. Побегайло. – СПб.: БХВ-Петербург, 2005. – 1056 с.

Учебное издание

Романенко Дмитрий Михайлович
Пацей Наталья Владимировна
Кудлацкая Марина Федоровна

КОМПЬЮТЕРНЫЕ СЕТИ

Лабораторный практикум

Редактор *Ю. Д. Нежикова*
Компьютерная верстка *Е. В. Ильченко*
Корректор *Ю. Д. Нежикова*

Подписано в печать .05.2016. Формат 60×84¹/₁₆.
Бумага офсетная. Гарнитура Таймс. Печать офсетная.
Усл. печ. л. . Уч.-изд. л. .
Тираж 220 экз. Заказ

Издатель и полиграфическое исполнение:
УО «Белорусский государственный технологический университет».
Свидетельство о государственной регистрации издателя,
изготовителя, распространителя печатных изданий
№ 1/227 от 20.03.2014.
Ул. Свердлова, 13а, 220050, г. Минск.