Белорусский национальный технический университет
Факультет информационных технологий и робототехники
Кафедра программного обеспечения информационных систем и технологий

ЭЛЕКТРОННЫЙ УЧЕБНО-МЕТОДИЧЕСКИЙ КОМПЛЕКС ПО УЧЕБНОЙ ДИСЦИПЛИНЕ

Компьютерные системы и сети

для специальности:
I – 40 01 01 «Программное обеспечение информационных технологий»

Составитель: Белова С.В.

Минск ◊ БНТУ ◊ 2022

Перечень материалов

Электронный учебно-методический комплекс включает:

- программу дисциплины «Компьютерные системы и сети»,
- теоретический раздел,
- лабораторный практикум,
- список литературы,
- экзаменационные вопросы,
- тесты к разделам курса.

Пояснительная записка

Электронный учебно-методический комплекс разработан для студентов специальности I – 40 01 01 «Программное обеспечение информационных технологий». Информационное наполнение ЭУМК соответствует программе дисциплины «Компьютерные системы и сети».

ЭУМК может использоваться как при проведении занятий по дисциплине «Компьютерные системы и сети», так и для организации самостоятельной работы студентов. Внедрение ЭУМК будет способствовать более эффективному овладению теоретическими и практическими основами построения и функционирования компьютерных сетей и сетевого программирования.

Информация в ЭУМК хорошо структурирована. Теоретический раздел включает основные темы курса. Лабораторный практикум содержит необходимый базовый методический материал (цель лабораторной работы, краткие теоретические сведения, задания на лабораторную работу, контрольные вопросы). В ЭУМК приводится также список литературы и актуальная программа дисциплины. Модуль контроля знаний состоит из экзаменационных вопросов и тестов по основным разделам курса.

ЭУМК разработан в виде pdf-файла и не требует установки специального программного обеспечения.

СОДЕРЖАНИЕ

1 Программа дисциплины «Компьютерные системы и сети»

Учебная программа учреждения высшего образования по учебной дисциплине «Компьютерные системы и сети» разработана для специальности 1-40 01 01 Программное обеспечение информационных технологий специализация 1-40 01 01 05 Управление качеством и тестирование программного обеспечения.

Целью изучения учебной дисциплины является овладение теоретическими и практическими основами построения и функционирования компьютерных систем и сетей, а также сетевого программирования.

Основными задачами преподавания учебной дисциплины являются: получение студентами систематизированных знаний о классификации, архитектуре и топологии компьютерных систем и сетей; принципах построения параллельных вычислительных систем различных классов; методах, алгоритмах и протоколах функционирования взаимодействующих систем; средствах организации и поддержки распределенных действий; современных сетевых технологиях и основах сетевого программирования. Теоретические знания должны быть закреплены на практике в ходе выполнения лабораторных работ и курсового проекта.

В результате изучения учебной дисциплины «Компьютерные системы и сети» студент должен:

знать:

- типовые организации компьютеров и компьютерных систем;
- архитектуру и принципы функционирования локальных и глобальных сетей;
- принципы работы стека TCP/IP и формат пакетов;
- протоколы прикладного уровня;
- принципы организации DNS; - основы сетевого программирования; **уметь:**
- разрабатывать программы, управляющие подключением устройств к современным интерфейсам;
- управлять сетевыми соединениями и службами, выполнять администрирование и конфигурирование сети;
- разрабатывать серверные и клиентские программы, функционирующие в локальных и глобальных сетях;

владеть:

- программными средствами администрирования компьютерных систем и сетей;
- современными сетевыми технологиями.

Освоение данной учебной дисциплины должно обеспечить формирование следующих компетенций:

АК-1. Уметь применять базовые научно-теоретические знания для решения теоретических и практических задач.

АК-4. Уметь работать самостоятельно.

АК-5. Быть способным порождать новые идеи (обладать креативностью).

АК-6. Владеть междисциплинарным подходом при решении проблем.

АК-7. Иметь навыки, связанные с использованием технических устройств, управлением информацией и работой с компьютером.

АК-11. Владеть основными методами, способами и средствами получения, хранения, переработки

СЛК-6. Уметь работать в коллективе.

ПК-1. Владеть современными методами, языками, технологиями и инструментальными средствами проектирования и разработки программных продуктов.

ПК-2. Владеть принципами и основными навыками, приемами, методами настройки, адаптации и сопровождения программных средств.

ПК-4. Программировать на профессиональном уровне с учетом ресурсов и возможностей конкретного компьютера, требований стандарта, ограничений проекта.

ПК-5. Осуществлять контроль эффективности использования вычислительных средств и информационных систем в профессиональной деятельности.

ПК-6. Владеть современными технологиями тестирования, отладки, верификации, аттестации и оценки качества программных средств.

ПК-10. Владеть компьютерными методами сбора, хранения и обработки информации в сфере своей профессиональной деятельности.

ПК-11. Владеть методами эффективной эксплуатации программных средств.

ПК-12. Администрировать компьютерные системы и сети.

ПК-13. Конфигурировать компьютерные системы и сети для конкретных задач определенного круга пользователей.

ПК-22. Взаимодействовать со специалистами смежных профилей.

Согласно учебному плану на изучение учебной дисциплины отведено всего 194 ч., из них аудиторных 86ч. На курсовой проект отведено 60ч. самостоятельной работы.

Распределение аудиторных часов по курсам, семестрам и видам занятий приведено ниже в таблице 1.

Таблица 1.

Очная форма получения высшего образования				
Курс	Семестр	Лекции, ч.	Лабораторные занятия, ч.	Форма текущей аттестации
2	4	34	34	экзамен

СОДЕРЖАНИЕ УЧЕБНОГО МАТЕРИАЛА

Тема 1. Введение в компьютерные системы и сети

Предмет и содержание дисциплины, ее цель и задачи. Понятие компьютерной системы и компьютерной сети. Эволюция компьютерных систем и сетей. Современные тенденции развития. Основные понятия и компоненты компьютерной сети. Виды сетевого программного обеспечения.

Тема 2. Классификация компьютерных сетей. Технологии разработки распределенных программ.

Основы классификации компьютерных сетей. Локальные и глобальные сети. Интернет и ее элементы. Сети операторов связи и корпоративные сети. Понятие логической архитектуры компьютерной сети. Одноранговая и клиент-серверная архитектура. Технологии разработки распределенных программ. Основы технологии сокетов. RPC. WCF.

Тема 3. Методы адресации узлов сети

Адресное пространство. Методы адресации узлов сети. Протоколы разрешения адресов (ARP, DNS). Одноадресные, групповые и многоадресные типы адресов. Аппаратные адреса. Числовые составные адреса. Структура IPv4-адреса. Понятие маски. Диапазоны IPv4-адресов. Протокол DHCP. IPv6адреса. Символьные адреса и их виды.

Тема 4. Эталонная модель взаимодействия открытых систем

Основные понятия. Многоуровневый подход к стандартизации сетевого взаимодействия. Концепция семиуровневой модели. Сетенезависимые уровни OSI. Сетезависимые уровни OSI. Виды протоколов. Сетевые стандарты. Стандартные стеки коммуникационных протоколов.

Тема 5. Стек протоколов TCP/IP.

История создания и основные характеристики стека TCP/IP. Архитектура стека TCP/IP. Характеристика уровней и сравнение с эталонной моделью OSI. Типы адресов и основные протоколы стека TCP/IP.

Тема 6. Протокол IP. Маршрутизация IP-пакетов.

Протоколы IPv4. Структура IP-пакета. Протокол IPv6. Использование масок. Принципы разбиения на подсети. Понятие IP-маршрутизации. Таблицы маршрутизации. Протокол ICMP. Технологии перехода на IPv6. Технология CIDR. Трансляция сетевых адресов, протокол NAT.

Тема 7. Транспортные протоколы

Протоколы с установлением и без установления соединения. Протокол TCP. Формат TCP-сегмента. Метод скользящего окна. Способы борьбы с перегрузками в TCP. Протокол UDP. Протоколы маршрутизации.

Тема 8. Служба WWW

Сервис WWW. Универсальный идентификатор ресурсов URI. Протокол HTTP. Структура HTTP-запроса и HTTP-ответа. Методы HTTP. Заголовки HTTP. Постоянные соединения и кэширование в HTTP. HTTP 2.0

Тема 9. Прикладные протоколы. Сетевые службы

Принципы работы электронной почты. Протоколы электронной почты SMTP, POP3, IMAP. Служба передачи файлов FTP. Протокол удаленного терминала Telnet. Создание сетевой службы.

Тема 10. Служба DNS

Система DNS. Иерархическое доменное пространство имен. DNSсерверы. Зоны и ресурсные записи. Методы разрешения DNS-имен. Разрешение имен в ОС Windows. Протоколы LLMNR и NetBIOS.

Тема 11. Характеристика линий связи.

Классификация и характеристики линий связи. Типы кабелей: преимущества, недостатки, применение. Структурированные кабельные системы. Беспроводные линии связи и их особенности. Понятие сетевой топологии. Базовые топологии компьютерных сетей.

Тема 12. Методы передачи данных и методы коммутации в компьютерных сетях.

Методы передачи данных на физическом уровне. Физическое кодирование данных. Логическое кодирование. Понятие коммутации. Коммутация каналов. Коммутация пакетов. Технологии мультиплексирования.

Тема 13. Аппаратное оборудование сети

Виды коммуникационного оборудования. Физическая и логическая структуризация сети. Сетевые адаптеры. Концентраторы. Принципы работы мостов. Архитектура и характеристики коммутаторов. Интеллектуальные функции коммутаторов. Маршрутизаторы. Шлюзы.

Тема 14. Базовые технологии локальных сетей

Общая характеристика и стандарты протоколов локальных сетей.

Структура стандартов 802.x. Технология Ethernet на разделяемой среде. Метод доступа CSMA/CD. Коммутируемый Ethernet. Fast Ethernet. Gigabit Ethernet. 10 Gigabit Ethernet. Другие технологии локальных сетей. Особенности беспроводных сетей. Беспроводные локальные сети стандарта 802.11. Персональные локальные сети. Технология Bluetooth.

Тема 15. Принципы организации и технологии глобальных сетей

WAN: структура, функции, типы. Глобальные сети с коммутацией пакетов. Технологии X.25, Frame Relay, ATM. Защита информации и безопасность компьютерных сетей

ТРЕБОВАНИЯ К КУРСОВОМУ ПРОЕКТУ

Цель курсового проектирования – получение практических навыков проектирования и разработки сетевого программного обеспечения и распределенных программных средств.

Задачами курсового проектирования являются:

- самостоятельное выполнение задания;
- работа с литературой, изучение теоретических аспектов поставленной задачи;
- систематизация и закрепление полученных во время учебы теоретических и практических навыков;

- умение грамотно составлять техническую документацию на разных стадиях разработки программного обеспечения.

Примерный объем пояснительной записки 25-27 страниц.

Количество часов на выполнение курсового проекта в соответствии с учебным планом – 60.

СРЕДСТВА ДИАГНОСТИКИ РЕЗУЛЬТАТОВ УЧЕБНОЙ ДЕЯТЕЛЬНОСТИ

Оценка уровня знаний студента производится по десятибалльной шкале в соответствии с критериями, утвержденными Министерством образования Республики Беларусь.

Для оценки достижений студента используется следующий диагностический инструментарий:

- устный опрос во время лабораторных занятий;
- выполнение тестов;
- проведение текущих контрольных работ (заданий) по отдельным темам;
- защита выполненных на лабораторных занятиях индивидуальных заданий;
- защита выполненных в рамках самостоятельной работы
- индивидуальных заданий;
- собеседование при проведении индивидуальных и групповых консультаций;
- выступление студента на конференции по подготовленному реферату;
- защита курсового проекта; сдача экзамена по дисциплине.

2 Теоретический раздел

2.1 Введение в компьютерные сети

Эволюция сетевых технологий

История любой отрасли науки и техники позволяет не только удовлетворить естественное любопытство, но и глубже понять сущность основных достижений в этой отрасли, изучить существующие тенденции и оценить перспективные направления развития.

Компьютерные сети появились сравнительно недавно, в конце 60-х годов прошлого столетия. Они отнюдь не являются единственным видом сетей (электрические, телефонные, радиосети, телевизионные).

Компьютерные сети или сети передачи данных являются логическим результатом эволюции двух важнейших технических отраслей: 1) вычислительной техники; 2) телекоммуникационных сетей.

С одной стороны, компьютерные сети возникли как развитие вычислительной техники. С другой стороны, они являются средством передачи информации на большие расстояния, для чего в них применяются методы коммутации и мультиплексирования, получившие развитие в телекоммуникационных системах.

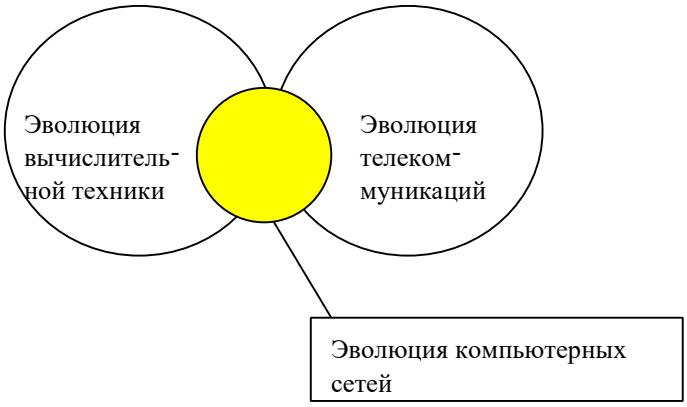


Рис. 1.1 Возникновение компьютерных сетей Рассмотрим

первый аспект.

В 50-е годы 20-го века на заре развития вычислительной техники использовались мэйнфреймы – мощные, но громоздкие и дорогие компьютеры универсального назначения. Число их было невелико, а программы работали часами и сутками. Первые компьютеры предназначались для небольшого числа избранных пользователей, в них использовались системы пакетной обработки. Это было связано с тем, что пакетный режим – самый эффективный

режим использования вычислительной мощности. Он позволяет выполнять в единицу времени больше пользовательских задач, чем любые другие режимы. Во главу угла ставилась эффективность работы самого дорогого устройства вычислительной машины – процессора, в ущерб эффективности работы специалистов.

В начале 60-х годов по мере удешевления процессоров появились новые способы организации вычислительного процесса, которые позволили учесть интересы пользователей. В это время стали создаваться интерактивные многотерминальные системы разделения времени. В такой системе компьютер работал сразу с несколькими пользователями, сидевшими за своими терминалами. Терминалы, выйдя за пределы вычислительного центра, рассредоточились по всему предприятию. Пользователь мог получить доступ к общим файлам и периферийным устройствам. Внешне работа была похожа на работу в современной локальной сети. Это и был прообраз сети. Однако, до появления локальных сетей нужно было пройти еще большой путь. В этот период был справедлив так называемый закон Гроша, который отражал уровень технологии того времени: производительность компьютера пропорциональна квадрату его стоимости.

Первые компьютерные сети были созданы в конце 60-х годов для обеспечения совместного доступа пользователей к вычислительным мощностям. Началось все с решения задачи доступа к компьютеру с удаленного за сотни километров терминала. Для этого начали использовать телефонные сети и модемы. Т.о. хронологически первыми появились глобальные компьютерные сети. Именно при построении глобальных сетей были впервые предложены и отработаны основные идеи и концепции современных компьютерных сетей.

В 1969 году Министерство обороны США инициировало работы по объединению в сеть суперкомпьютеров оборонных и научно-исследовательских центров. Эта сеть, получившая название ARPANET, стала отправной точкой для создания первой и самой известной ныне глобальной сети – Интернет.

Глобальные компьютерные сети очень многое унаследовали от гораздо более старых и распространенных телефонных сетей. Прогресс ГКС во многом определялся прогрессом телефонных сетей.

Важное событие, повлиявшее на эволюцию КС – появление в начале 70х годов БИС и основанных на них миникомпьютеров. Они были способны решать задачи уровня подразделения предприятия. Крупное предприятие могло иметь несколько таких компьютеров. Следовательно, возникла необходимость обмена данными между ними. Так возникли первые локальные

сети. Для соединения компьютеров использовались разнообразные нестандартные коммуникационные устройства и сетевые технологии.

Сетевая технология – согласованный набор программных и аппаратных средств, а также механизмов передачи данных по линиям связи, достаточный для построения вычислительной сети.

В середине 80-х годов положение дел в локальных сетях кардинально изменилось:

- утвердились стандартные технологии объединения компьютеров в сеть: Ethernet, Arcnet, Token Ring;
- появились достаточно мощные массовые ПК, они стали идеальными элементами для построения сетей. ПК начали преобладать в локальных сетях не только в качестве клиентских компьютеров, но и в качестве центров хранения и обработки данных (серверов), потеснив миникомпьютеры и мэйнфреймы.

Кабели и сетевые адаптеры первого поколения обеспечивали скорость передачи в локальных сетях до 10 Мбит/с. Для глобальных сетей скорость передачи по телефонным линиям порядка 1200 бит/с.

Понятие и компоненты компьютерной сети. Основные термины.

Понятие компьютерная сеть довольно общее. Это и Internet и два компьютера, соединенные нуль-модемным кабелем через COM-порты. Компьютерные сети называют также вычислительными сетями, или сетями передачи данных. Возможны следующие определения.

Компьютерная сеть - комплекс взаимосвязанных и согласованно функционирующих программных и аппаратных компонентов.

Компьютерная сеть – совокупность компьютеров, соединенных каналами связи.

Основным назначением компьютерной сети является:

- ***совместное использование информации и ресурсов;***
- ***совместное использование оборудования и ПО.*** Например, использование удаленных специализированных устройств. Через сеть пользователи могут обращаться к специализированным устройствам, отсутствующим на их локальных компьютерах – например, принтерам.;
- ***ускорение вычислений – распределение загрузки***. С использованием сети могут быть организованы распределенные вычисления, в которых каждый узел сети решает свою часть задачи, благодаря чему вычисления могут быть значительно ускорены.
- ***повышение надежности – обнаружение отказа машины, реинтеграция отказавшей машины***. Сетевые архитектуры позволяют в

случае сбоев или отказов одного из узлов сети (например, сервера) перераспределить его рабочую нагрузку на другой аналогичный узел сети и вывести дефектный узел из конфигурации сети, с целью его последующего ремонта или замены.

- ***коммуникация – с помощью передачи сообщений***. Сеть – удобный способ коммуникации, делового и личного общения. - ***централизованное администрирование и обслуживание.***

В терминах сетевой обработки все компьютеры и устройства, присоединенные к сети, называются ***узлами***. Можно также встретить термины: ***станция данных, оконечная система, абонентская система.*** Узлы соединены ***каналами связи***, которые могут быть кабелями или беспроводными соединениями.

Изучение сети в целом предполагает знание принципов работы ее основных компонентов. Компьютерная сеть состоит из аппаратных и программных компонентов.

Основными аппаратными компонентами сети являются:

1. ***Абонентские системы***: компьютеры (рабочие станции или клиенты и серверы); принтеры; сканеры и др. виды узлов.
2. ***Сетевое коммуникационное оборудование***: сетевые адаптеры; повторители; концентраторы; мосты; коммутаторы, маршрутизаторы, шлюзы и др.
3. ***Коммуникационные каналы***: кабели; разъемы; устройства передачи и приема данных в беспроводных технологиях.

В настоящее время в сетях широко и успешно применяются компьютеры различных классов - от персональных компьютеров до мэйнфреймов и суперЭВМ. Набор компьютеров в сети должен соответствовать набору разнообразных задач, решаемых сетью.

Для обозначения компьютеров, подключенных к сети, используются термины клиент и сервер. Если компьютер предоставляет свои ресурсы другим компьютерам сети, то он называется ***сервером***, а если он их потребляет - ***клиентом***. Иногда один и тот же компьютер может одновременно играть роли и сервера, и клиента.

Компьютер с установленной на нем серверной ОС, занимающийся исключительно обслуживанием запросов других компьютеров, называют **выделенным сервером**. Т.е. выполнение каких-либо серверных функций является его основным назначением. Конкретный сервер характеризуется видом ресурса, которым он владеет. Например, если ресурсом является база данных, то речь идет о сервере базы данных, если ресурс – файловая система, то говорят о файл-сервере. Аналогично - факс-сервер, сервер печати, сервер приложений, сервер доступа.

Хотя компьютеры и являются центральными элементами обработки данных в сетях, в последнее время не менее важную роль стали играть коммуникационные устройства. повторители, мосты, коммутаторы, маршрутизаторы и модульные концентраторы из вспомогательных компонентов сети превратились в основные наряду с компьютерами и системным программным обеспечением как по влиянию на характеристики сети, так и по стоимости.

Программные компоненты сети – это сетевое ПО:

1. Сетевые операционные системы
2. Сетевые службы
3. Сетевые приложения.

Сетевое программное обеспечение

Сетевое ПО включает следующие компоненты:

1) сетевые ОС,
2) сетевые службы,
3) сетевые приложения.

Операционную систему компьютера можно определить как взаимосвязанный набор системных программ, который обеспечивает эффективное управление ресурсами компьютера, а также предоставляет пользователям удобный интерфейс для работы с аппаратурой компьютера и разработки приложений.

Сетевая операционная система – ОС компьютера, которая помимо управления локальными ресурсами предоставляет пользователям и приложениям доступ к информационным и аппаратным ресурсам других компьютеров сети. Сегодня практически все ОС являются сетевыми.

Различия между существующими ОС не столь существенны, как это может показаться на первый взгляд. Разработчики ОС живут в едином мире без жестких границ, не препятствующих миграции хорошо зарекомендовавших себя концепций и механизмов из одной системы в другую. Функциональные компоненты сетевой ОС представлены на рисунке.

Средства управления локальными ресурсами реализуют все функции ОС автономного компьютера (распределение ОП между процессами, планирование и диспетчеризация процессов, управление процессорами, управление внешней памятью, интерфейс с пользователем и т.д.).

Из рисунка видно, что к сетевой ОС добавлены сетевые службы и транспортные средства.

Сетевые службы по своей природе являются клиент-серверными системами.

Сетевая служба - пара модулей «клиент - сервер», обеспечивающих совместный доступ пользователей к определенному типу ресурсов через сеть.

Клиентская часть сетевой службы – средства запроса доступа к удаленным ресурсам и услугам.

Серверная часть сетевой службы – средства предоставления локальных ресурсов и услуг в общее пользование.

Сетевая служба предоставляет пользователям сети некоторый набор услуг. Эти услуги называют **сетевыми сервисами**.

Сетевая служба – системная распределенная программа, реализующая сетевой сервис.

«Сервис» - описание набора услуг. «Служба» - сетевой компонент, который реализует набор услуг.

От того, насколько богатый набор сетевых служб предлагает ОС конечным пользователям, приложениям и администраторам сети, зависит ее место в ряду сетевых ОС.

Обычно сетевая операционная система поддерживает основные виды сетевых служб - **файловую службу** и **службу печати**. Вспомогательные

службы поддерживаются системными сетевыми приложениями, или утилитами, - **служба баз данных**, **факса**. Примеры сетевых служб – **WWW, FTP, UseNet, служба электронной почты, служба удаленного доступа** и т. д.. Среди сетевых служб можно выделить административные, предназначенные для организации работы сети. Например, **централизованная справочная служба,** или **служба каталогов**. Она предназначена для администрирования учетных записей пользователей сети, всех ее программных и аппаратных компонентов (Например, Active Directory компании Microsoft). **Служба мониторинга сети** позволяет захватывать и анализировать сетевой трафик. Существуют также **служба безопасности**, **служба резервного копирования и архивирования**.

Как правило взаимодействие между клиентскими и серверными частями ОС стандартизируется, так что один тип сервера м.б. рассчитан на работу с клиентами разного типа, реализованными разными способами и даже разными производителями. Единственное условие для этого – клиент и сервер должны поддерживать общий стандартный протокол взаимодействия.

Программные коммуникационные (транспортные) средства обеспечивают совместно с аппаратными коммуникационными средствами передачу сообщений, которыми обмениваются клиентские и серверные части сетевых служб. Сюда относятся драйверы и протокольные модули. Они выполняют такие функции, как формирование сообщений, разбиение сообщения на части (пакеты, кадры), преобразование имен компьютеров, дублирование сообщений в случае ошибки, определение маршрута в сложной сети и т.д..

Сетевые службы и транспортные средства могут являться встроенными компонентами ОС или существовать в виде отдельных программных продуктов. (Например, файловая служба и браузер) Типичная сетевая ОС имеет в своем составе широкий набор драйверов и протокольных модулей. У пользователя есть возможность дополнить этот стандартный набор необходимыми программами.

Сетевая служба может быть представлена в ОС либо обеими (клиентской и серверной) частями, либо только одной из них.

В первом случае ОС называется **одноранговой.** Она позволяет обращаться к ресурсам других компьютеров сети и предоставляет собственные ресурсы в распоряжение других пользователей. (Например, клиенты и серверы файловой службы) Компьютеры, совмещающие функции клиента и сервера, называют одноранговыми узлами.

ОС, которая преимущественно содержит клиентские части сетевых служб, называется **клиентской**.

Серверная ОС ориентирована на обработку запросов из сети к ресурсам своего компьютера и включает в себя в основном серверные части сетевых служб.

Клиентская ОС должна обеспечивать для локально выполняющихся программ прозрачный доступ к удаленным файлам, размещенным на других узлах. Соответствующий компонент ОС называется **редиректор** (перенаправитель). Прозрачность состоит в том, что для локальной программы обращение к удаленным файлам выглядит так же, как к локальным. Основнаяе функция редиректора - перехват обращений к удаленным файлам и перенаправление их на другой узел.

Серверная ОС должна обеспечивать доступ к локальным файлам для программ, выполняющихся на других узлах. Соответствующий компонент ОС называется **сервер**. Основная функция сервера – получение от редиректоров других узлов запросов на чтение/запись данных и перенаправление их к соответствующим локальным файлам, а затем формирование ответов и передача их в сеть.

Сетевые приложения

Существуют следующие виды приложений:

1) **Локальное приложение** целиком выполняется на данном компьютере и использует только локальные ресурсы. Такому приложению не требуются сетевые средства.

2) **Централизованное сетевое приложение** целиком выполняется на данном компьютере, но обращается к ресурсам других компьютеров сети. Например, приложение, которое выполняется на клиентском компьютере, обрабатывает данные из файла, хранящегося на сервере. Работа такого приложения невозможна без сетевых средств.

3) **Распределенное (сетевое) приложение** состоит из нескольких взаимодействующих частей, расположенных на разных компьютерах, каждая из которых выполняет свою функцию. Части распределенного приложения взаимодействуют друг с другом, используя сетевые службы и транспортные средства ОС.

Преимуществом распределенных приложений является возможность распараллеливания вычислений, а также специализация компьютеров.

Сетевые характеристики

К компьютерной сети предъявляется на самом деле единственное требование – она должна обеспечивать нам нужный набор услуг. Чтобы нам было удобно и комфортно.

Основные требования, характеризующие качество работы сети:

- производительность;
- надежность;
- безопасность;
- характеристики сети поставщика услуг (расширяемость, масштабируемость, управляемость, совместимость, прозрачность).

Производительность.

Представляет собой интегральную характеристику сети, состоящую из нескольких других характеристик:

- скорость передачи трафика;
- пропускная способность;
- задержка передачи;
- джиттер (вариация задержки).

Скорость передачи данных – это объем данных, переданных сетью или ее частью в единицу времени. Измеряется в битах в секунду, либо в пакетах в секунду.

Пропускная способность может быть мгновенной, максимальной, средней.

Средняя скорость вычисляется путем деления общего объема переданных данных на время передачи. Причем выбирается достаточно длинный промежуток времени – час, день, неделя.

Мгновенная скорость - это пропускная способность на очень маленьком промежутке времени – 10мс, 1с.

Пиковая скорость – это наибольшая скорость, которую разрешается достигать пользовательскому потоку в течение оговоренного периода времени Т.

Пропускная способность – максимально возможная скорость сетевой технологии, определенная стандартом, на котором построена сеть.

Время реакции сети – интервал времени между возникновением запроса пользователя к какой-либо сетевой службе и получением ответа на запрос.

Это пользовательская характеристика. Она зависит от типа службы и текущего состояния элементов сети. Время реакции складывается из нескольких составляющих: время подготовки запроса на клиентском компьютере, время передачи запроса между клиентом и сервером, время обработки запроса на сервере, время передачи ответа клиенту, время обработки ответа на клиентском компьютере. Знание сетевых составляющих времени реакции дает возможность оценить производительность отдельных элементов сети, выявить узкие места и, если необходимо, выполнить модернизацию сети.

Задержка передачи – интервал времени между моментом поступления данных на вход системы (сети, компьютера, коммуникационного устройства) и моментом их появления на выходе.

Вариация задержки - разница между минимальной и максимальной задержкой на каком-то промежутке времени.

Не все виды трафика одинаково чувствительны к задержкам (передача электронной почты и передача голоса).

Надежность.

Для оценки надежности сети чаще всего используются два показателя:

- коэффициент доступности; - коэффициент доставки пакетов.

Коэффициент доступности (готовности) – доля времени, в течение которого система находится в работоспособном состоянии.

КД = (Период – ВремяНедоступности)/Период *100%

Это долговременная статистическая характеристика (день, месяц, год). Пример высокого уровня доступности – коммуникационное оборудование телефонных сетей (доступность «пять девяток» - 0,99999, это 5 мин. простоя в год). Для компьютерных сетей достигнут рубеж трех девяток.

Коэффициент доставки пакетов – вероятность доставки на текущий момент времени. КДП = (КоличествоОтправлПакетов – КолПотерПакетов)/КОП*100%

Доля потерянных пакетов - отношение количества потерянных пакетов к общему количеству переданных пакетов.

Отказоустойчивость – способность системы работать в условиях отказа отдельных элементов.

Характеристики сети поставщика услуг.

Эти характеристики часто являются качественными, т. е. не могут быть выражены числами и соотношениями.

Расширяемость – возможность сравнительно легкого добавления отдельных элементов сети (пользователей, компьютеров, приложений, сервисов), наращивания длины сегментов сети и замены существующей аппаратуры более мощной.

Масштабируемость означает, что сеть позволяет наращивать количество узлов и протяженность связей в очень широких пределах без потери производительности.

Расширяемость и масштабируемость – разные характеристики сети. Сеть может иметь плохую масштабируемость при хорошей расширяемости. Например, локальная сеть Ethernet на основе коаксиального кабеля обладает хорошей расширяемостью, так как легко подключать новые станции. Но

существует ограничение в 30 станций на 185м кабеля, иначе резко снижается производительность сети. Следовательно такая сеть имеет плохую масштабируемость. Хорошей масштабируемостью обладает многосегментная сеть, построенная с использованием коммутаторов и маршрутизаторов и имеющая иерархическую структуру.

Прозрачность – свойство сети скрывать от пользователя детали своего внутреннего устройства и функционирования, упрощая его работу.

Прозрачность может быть достигнута на двух уровнях: -

на уровне пользователя;

- на программном уровне.

На уровне пользователя прозрачность означает, что для работы с удаленными ресурсами могут быть использованы те же команды, что и для работы с локальными ресурсами. Такая прозрачность достигается при разработке приложения. Например, копирование ресурсов в Explorer.

На программном уровне прозрачность означает, что приложению для доступа к удаленным ресурсам требуются те же вызовы, что и для доступа к локальным ресурсам. Такая прозрачность осуществляется сетевой операционной системой.

Прозрачность – цель, к которой стремятся разработчики современных сетей.

Управляемость – возможность централизованно контролировать состояние основных элементов сети, выявлять и решать возникающие проблемы, выполнять анализ производительности и планировать развитие сети.

Совместимость (интегрируемость) – означает способность сети включать в себя разнообразное аппаратное и программное обеспечение, различные операционные системы, стеки коммуникационных протоколов и т.д..

Сеть, состоящая из разнотипных элементов, называется ***гетерогенной***. Если гетерогенная сеть работает без проблем, то она является совместимой. Основной путь построения интегрированных сетей – использование открытых стандартов и спецификаций.

2.2 Классификация компьютерных сетей

Деление по охвату территории.

Среди технологических характеристик основная - размер территории, охватываемой сетью. По величине территории, которую покрывает сеть различают локальные, глобальные и городские сети.

Локальные сети - Local Area Networks (LAN) - сети, сосредоточенные на небольшой территории (обычно в радиусе не более 1-2 км). В общем случае локальная сеть принадлежит одной организации. Из-за коротких расстояний в локальных сетях имеется возможность использования высококачественных линий связи, которые позволяют достигать высоких скоростей обмена данными, порядка 100 Мбит/с. Особенности LAN:

- высококачественные линии связи; - низкий уровень ошибок передачи;
- высокие скорости обмена данными;
- работа в режиме on-line;
- прозрачность для пользователя;
- точно определенное число абонентов; - стандартные сетевые технологии.

Стандартные сетевые технологии превратили процесс построения локальной сети из искусства в рутинную работу. Для создания сети достаточно приобрести стандартный кабель, сетевые адаптеры, соединить их стандартными разъемами и установить на компьютеры одну из популярных ОС.

Конец 90-х г.г. выявил явного лидера среди технологий локальных сетей – технологию Ethernet. Успеху способствовали простые алгоритмы работы, низкая стоимость оборудования, широкий диапазон иерархии скоростей.

Глобальные сети - Wide Area Networks (WAN) - объединяют территориально-рассредоточенные компьютеры, которые могут находиться по всему миру. Так как прокладка высококачественных линий связи на большие расстояния обходится очень дорого, в глобальных сетях часто используются уже существующие линии связи, изначально предназначенные совсем для других целей. Например, многие глобальные сети строятся на основе телефонных и телеграфных каналов общего назначения. Особенности WAN:

- невысокое качество линий связи;
- невысокие скорости передачи;
- режим off-line;
- ограниченный набор предоставляемых услуг; - высокая стоимость.

Городские сети (или сети мегаполисов) - Metropolitan Area Networks (MAN) - эти сети появились сравнительно недавно, как одно из проявлений сближения локальных и глобальных сетей. Они предназначены для обслуживания территории крупного города - мегаполиса. Сети мегаполисов занимают промежуточное положение между локальными и глобальными сетями. Они используют цифровые магистральные линии связи, часто оптоволоконные, со скоростями от 45 Мбит/с, и предназначены для связи локальных сетей в масштабах города и соединения локальных сетей с глобальными.

Основные отличия локальных и глобальных сетей:

- *Протяженность, качество и способ прокладки линий связи.*
- *Сложность методов передачи и оборудования.*
- *Скорость обмена данными.* (Для локальных сетей - 10, 16, 100, 1000 Мбит/с. Для глобальных сетей типичны гораздо более низкие скорости передачи данных - 2400,9600, 28800,33600 бит/с, 56 и 64

  Кбит/с и только на магистральных каналах - от 2 Мбит/с до 10Гбит/с.)
- *Разнообразие предоставляемых услуг.*
- *Масштабируемость.* "Классические" локальные сети обладают плохой масштабируемостью. Глобальным же сетям присуща хорошая масштабируемость.

В настоящее время ГКС по разнообразию и качеству предоставляемых услуг догнали локальные сети, которые долго лидировали в этом отношении.

PAN (Personal Area Network) – личная (персональная) сеть. Такие сети работают на расстояниях порядка 1-20 м. Например, технология Bluetooth.

Сети операторов связи и корпоративные сети.

По назначению предоставляемых услуг сети делятся на:

1) сети операторов связи (сети поставщиков услуг) – оказывают общедоступные услуги;
2) корпоративные сети – услуги сотрудникам своего предприятия.

Специализированное предприятие, которое создает телекоммуникационную сеть для оказания общедоступных услуг, владеет этой сетью и поддерживает ее работоспособность, называется **оператором связи**.

Операторы связи отличаются друг от друга:

- набором предоставляемых услуг;
- территорией, в пределах которой предоставляются услуги;
- типом клиентов, на которых ориентируются услуги;

- имеющейся во владении инфраструктурой (линиями связи, коммутационным оборудованием, информационными серверами).

Потребители телекоммуникационных услуг – клиенты – могут быть разделены на массовых ***индивидуальных клиентов*** и ***корпоративных клиентов***. Корпоративные клиенты – это предприятия и организации различного профиля.

Крупные предприятия, состоящие из нескольких территориально рассредоточенных отделений или имеющие сотрудников, работающих дома, нуждаются в расширенном наборе услуг. Прежде всего, это **виртуальная частная сеть** (**VPN**, Virtual Private Network), в которой оператор связи создает для предприятия иллюзию того, что все его отделения и филиалы соединены частной сетью, полностью принадлежащей и управляемой предприятиемклиентом.

От наличия или отсутствия собственной транспортной инфраструктуры зависит название предприятия, оказывающего информационнокоммуникационные услуги. Традиционно такие предприятия называют «оператор связи». В последние десятилетия появилось новое название – поставщик (провайдер) услуг. Говоря «оператор связи», обычно подчеркивают, что компания владеет собственной транспортной инфраструктурой. В названии «поставщик услуг» акцент делается на то, что предприятие оказывает высокоуровневые услуги, например, доступ в Internet, размещение в своей сети информационных ресурсов (web-сайтов или баз данных предприятий), и не обязательно владеет собственной развитой транспортной инфраструктурой, так как часто для их эффективной реализации достаточно арендованных сетевых ресурсов.

Оператора, который предоставляет услуги другим операторам связи, называют ***оператором операторов***.

По степени покрытия территории, на которой услуги предоставляются, **операторы делятся на локальных, региональных, национальных и транснациональных.** Локальный оператор работает на территории города или сельского района. Региональные и национальные операторы оказывают услуги на большой территории, располагая соответствующей инфраструктурой. Это операторы операторов. Их клиентами являются локальные операторы или крупные предприятия, имеющие отделения и филиалы в различных городах и регионах. В современном конкурентном телекоммуникационном мире нет строгой иерархии операторов, взаимосвязи между ними и их сетями могут быть достаточно сложными.

Для подключения оборудования клиентов операторы связи организуют **точки присутствия** (**POP**, Point Of Presents) – здания или помещения, в

которых размещается оборудование доступа для подключения большого количества клиентов.

Корпоративная сеть – это сеть, главным назначением которой является поддержание работы конкретного предприятия, владеющего данной сетью. Пользователями корпоративной сети являются только сотрудники данного предприятия. В отличие от сетей операторов связи, корпоративные сети, в общем случае, не оказывают услуг сторонним организациям и пользователям.

По производственному признаку (в зависимости от масштаба производственного подразделения, в пределах которого действует сеть, а также от сложности и многообразия решаемых задач) различают:

1) сети отделов и рабочих групп;
2) сети зданий и кампусов; 3) сети масштаба предприятия.

Сети отделов – это сети, которые используются небольшой группой сотрудников, работающих в одном отделе предприятия и, решающих общие задачи (бухгалтерия, маркетинг). Главной целью сети отдела является разделение локальных ресурсов – дорогостоящих периферийных устройств, приложений, данных, модемов. Обычно сети отделов имеют около 30 рабочих станций, создаются на основе одной сетевой технологии (Ethernet), не разделяются на подсети, могут включать 1-2 файловых сервера, используют одну операционную систему (**гомогенные**).

Сеть отдела может входить в состав сети кампуса или же представлять собой сеть удаленного офиса предприятия. В первом случае она подключается к сети кампуса по локальной технологии. Во втором случае – непосредственно к магистрали сети с помощью WAN-технологии, например, Frame Relay. ***Сети рабочих групп*** – это совсем небольшие сети, включающие 10-20 компьютеров. Их характеристики близки к характеристикам сетей отделов.

Сети зданий и кампусов объединяют множество сетей различных отделов одного предприятия в пределах отдельного здания или в пределах одной территории (кампуса), покрывающей площадь в несколько квадратных километров. Сети кампусов получили свое название от английского слова campus - студенческий городок. Сейчас это название используют для обозначения сетей любых предприятий и организаций. Для построения сетей кампусов используются технологии локальных сетей, возможностей которых достаточно, чтобы обеспечить такую зону покрытия. Обычно сеть имеет иерархическую структуру (от Ethernet до Gigabit Ethernet и выше). Важные службы, предоставляемые сетями кампусов - доступ к корпоративным базам данных, высокоскоростным принтерам, модемам, факс-серверам. На уровне сети кампуса возникают проблемы интеграции неоднородного аппаратного и

программного обеспечения. Типы компьютеров, сетевых операционных систем, сетевого аппаратного обеспечения могут отличаться в каждом отделе. Для логической структуризации используется разнообразное коммуникационное оборудование. Отсюда вытекают сложности управления сетями кампусов.

Сети масштаба предприятия или корпоративные сети (в узком смысле) могут покрывать город, регион или даже континент. Они связывают отдельные рассредоточенные локальные сети предприятия в единую *инфокоммуникационную сеть*. Эти сети предоставляют транспортные и информационные услуги. Но на первый план в них выходят именно информационные услуги.

Особенности корпоративных сетей:

- *Масштабность* - число пользователей и компьютеров может измеряться тысячами, а число серверов – сотнями.
- *Высокая степень гетерогенности* - используются различные типы компьютеров, операционных систем и множество различных приложений.
- *Использование глобальных связей* для соединения удаленных локальных сетей и отдельных компьютеров. Применяются разнообразные телекоммуникационные средства – телефонные каналы, радиоканалы, спутниковая связь.

Сложность управления и централизованного администрирования.

По мере увеличения масштабов сети повышаются требования к ее надежности и производительности, а также к защите данных. Поэтому корпоративные сети строятся на основе наиболее мощного и разнообразного оборудования и программного обеспечения.

Понятие логической архитектуры компьютерной сети.

Разновидности логических архитектур. Различают физическую и логическую архитектуру компьютерных сетей.

Физическая архитектура определяется структурой, назначением и взаимосвязями аппаратных средств компьютерной сети, а также программными реализациями протоколов нижнего и среднего уровней ЭМВОС.

Логическая архитектура описывает структуру, назначение и взаимосвязи программных средств, реализующих протоколы верхних уровней ЭМВОС.

Правильно выбранная архитектура компьютерной сети позволяет достигнуть выдвинутых требований по общей производительности,

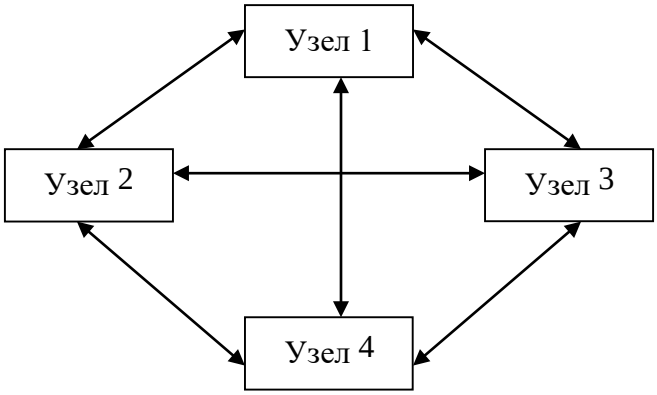
надежности защиты сетевых ресурсов, гибкости настройки, а также минимизации денежных затрат на построение и администрирование сети.

В настоящее время различают следующие ***логические архитектуры*** компьютерных сетей:

- одноранговая архитектура (peer-to-peer (с глазу на глаз, один-на один), пиринговые сети, P2P-сети, децентрализованные);
- классическая архитектура “клиент-сервер” (server based);
- архитектура «клиент-сервер», основанная на Web-технологии или Intranet-архитектура.

Появление каждой из перечисленных архитектур связывают с отдельными этапами эволюции вычислительных систем.

На первом этапе эволюции вычислительных систем компьютерные сети имели ***одноранговую архитектуру***. Термин – с 1984г., IBM (peer-to-peer).



Особенности одноранговой архитектуры:

1) Все компьютеры равноправны. Отсутствуют выделенные серверы.

2) Каждый компьютер функционирует и как клиент, и как сервер. Пользователи сами решают, какие данные на своем компьютере сделать доступными по сети (выделить в общее пользование).

Любая машина может связаться с любой. В качестве клиента (потребителя ресурсов) каждая из машин может посылать запросы на предоставление каких-либо ресурсов другим машинам в пределах этой сети и получать их. Как сервер, каждая машина должна обрабатывать запросы от других машин в сети, отсылать то, что было запрошено, а также выполнять некоторые вспомогательные и административные функции.

Локальные одноранговые сети:

3) Не более 10 компьютеров. Отсюда другое название — рабочая группа (workgroup). 4) Низкая производительность.

5) Низкие требования к защищенности сетевого программного обеспечения.

6) Не требуется дополнительного сетевого программного обеспечения и соответственно администратора.

Преимущество одноранговой архитектуры: в отличие от архитектуры клиент-сервер, такая организация позволяет сохранять работоспособность сети при любом количестве и любом сочетании доступных узлов.

Одноранговая архитектура используется и в настоящее время. Интернет дает жизнь новым одноранговым приложениям. Любой член данной сети не гарантирует никому своего присутствия на постоянной основе. Он может появляться и исчезать в любой момент времени. Но при достижении определённого критического размера сети наступает такой момент, что в сети одновременно существует множество серверов с одинаковыми функциями

Помимо чистых P2P-сетей, существуют так называемые гибридные сети, в которых существуют сервера, используемые для координации работы, поиска или предоставления информации о существующих машинах сети и их статусе (on-line, off-line и т. д.). Гибридные сети сочетают скорость централизованных сетей и надёжность децентрализованных благодаря гибридным схемам с независимыми индексационными серверами, синхронизирующими информацию между собой. При выходе из строя одного или нескольких серверов, сеть продолжает функционировать. К частично децентрализованным файлообменным сетям относятся например EDonkey, BitTorrent.

Пиринговая файлобменная сеть.

Одна из областей применения технологии пиринговых сетей — это обмен файлами. Выглядит это так: пользователи сети выкладывают какие-либо файлы в «расшаренную» (англ. *share*, делиться) папку. Какой-нибудь другой пользователь сети посылает запрос на поиск какого-либо файла. Программа ищет у клиентов сети файлы, соответствующие запросу, и показывает результат. После этого пользователь может скачать файлы у найденных источников. Современные файлообменные сети позволяют скачивать один файл сразу с нескольких источников (так быстрее и надёжнее). Чтобы убедиться, что этот файл у всех источников одинаковый, производится сравнение не обязательно по названию файла, а и по контрольным суммам или хэшам типа MD4, SHA-1. Во время скачивания файла пользователем (и после его окончания) этот файл у него могут скачивать и другие клиенты сети, в результате чего особенно популярные файлы могут в итоге быть доступными для скачивания с многих источников одновременно. Обычно в таких сетях обмениваются фильмами и музыкой.

Технология пиринговых сетей применяется также для распределённых вычислений. Они позволяют в сравнительно очень короткие сроки выполнять поистине огромный объём вычислений, который даже на суперкомпьютерах

потребовал бы, в зависимости от сложности задачи многих лет и даже столетий работы. Такая производительность достигается благодаря тому, что некоторая глобальная задача разбивается на большое количество блоков, которые одновременно выполняются сотнями тысяч компьютеров, принимающими участие в проекте.

Сетевую архитектуру, появившуюся на очередном этапе эволюции компьютерных технологий (в 80-е годы), называют **классической архитектурой «клиент-сервер».**

Особенности классической архитектуры «клиент-сервер»:

1) В сети имеется компьютер-сервер и компьютеры-клиенты. Сервер оптимизирован для быстрой обработки запросов от сетевых клиентов и для повышения защищенности файлов и каталогов.
2) Более высокая производительность и скорость работы.
3) Администрирование осуществляется централизовано.
4) Обеспечивается всесторонняя и централизованная защита ресурсов.
5) Любое количество пользователей (для локальных сетей).
6) Должна быть установлена сетевая операционная система и специальное сетевое программное обеспечение.

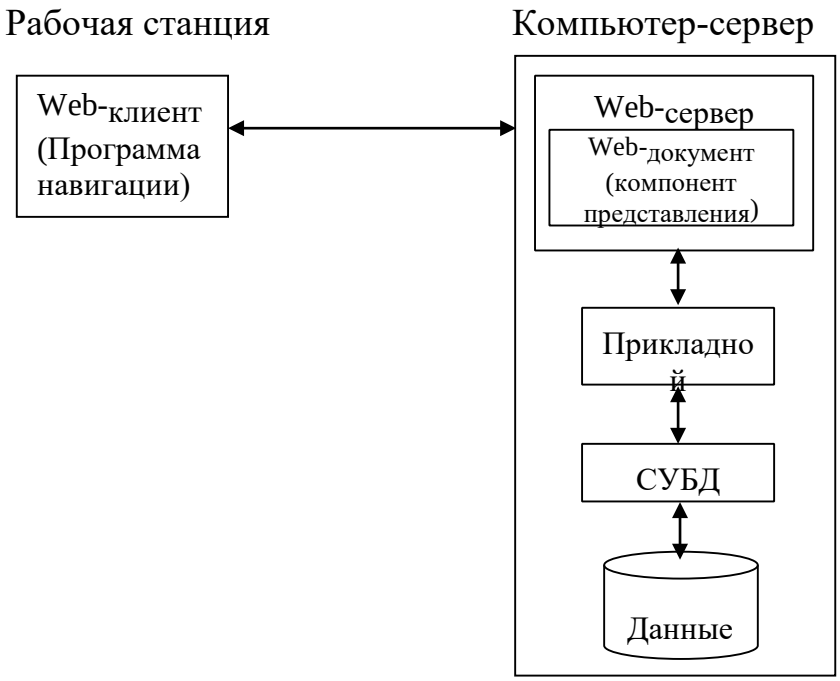
Сети на основе сервера стали промышленным стандартом.

Архитектура «клиент-сервер», основанная на Web-технологии, Intranet-архитектура или ***Web-архитектура*** появилась в 1993 году.
Основная особенность – возвращение серверам ряда функций, которые были вынесены на втором этапе.

Базисом этой архитектуры является Web-технология, пришедшая из Internet. В соответствии с ней на сервере размещаются Web-документы, которые интерпретируются программой навигации (Web-браузером), функционирующей на рабочей станции.

Web-документ реально включает одну страницу, но логически может объединять любое количество страниц, принадлежащих различным Webдокументам. В Web-страницу могут быть включены ссылки на различные объекты:

- другую часть Web-доркумента или - другой Web-документ;
- мультимедийный объект;
- программу, которая будет выполняться на сервере;
- программу, которая будет передана с сервера на рабочую станцию для запуска навигатором;
- любой другой сервис (эл.почту, поиск и т.д.).



Особенности:

1) Все информационные ресурсы и прикладная система сконцентрированы на сервере.

2) Для обмена информацией используются открытые протоколы (HTTP). Адрес ресурса указывается в формате URI.

3) Облегчено централизованное управление сервером и клиентами, т.к. ПО стандартизовано.

2.3 Методы адресации узлов сети

Виды адресации узлов сети

Каждый узел сети должен иметь адрес, чтобы была возможной передача информации и функционирование сети.

К адресу узла сети и схеме его назначения можно предъявить несколько требований: Адрес должен уникально идентифицировать компьютер в сети любого масштаба.

- Схема назначения адресов должна сводить к минимуму ручной труд администратора и вероятность дублирования адресов.
- Адрес должен иметь иерархическую структуру, удобную для построения больших сетей. В крупных сетях отсутствие иерархии адресов может привести к большим издержкам - конечным узлам и коммуникационному оборудованию придется оперировать с таблицами адресов, состоящими из тысяч записей.
- Адрес должен быть удобен для пользователей сети, т.е. должен иметь символьное представление.
- Адрес должен иметь по возможности компактное представление, чтобы не перегружать память коммуникационной аппаратуры - сетевых адаптеров, маршрутизаторов и т. п.

Перечисленные требования трудно совместить в рамках какой-либо одной схемы адресации, поэтому на практике обычно используется сразу несколько схем, так что компьютер одновременно имеет несколько адресов. Каждый адрес используется в той ситуации, когда соответствующий вид адресации наиболее удобен.

По количеству адресуемых сетевых интерфейсов адреса можно классифицировать следующим образом:

- ***Одноадресный тип*** или ***уникальный адрес (unicast)*** используется для идентификации отдельных интерфейсов (физический интерфейс между компьютером и сетью) конечного узла или маршрутизатора; позволяет пересылать сообщения в одну точку (на один конечный узел сети).
- ***Групповой адрес (multicast)*** идентифицирует сразу несколько интерфейсов, данные доставляются каждому из интерфейсов, входящих в группу; позволяет пересылать сообщения группе произвольно расположенных узлов.
- ***Широковещательный адрес (broadcast)*** используется для доставки данных всем узлам подсети;

- ***Адрес произвольной рассылки (anycast)*** задает группу интерфейсов, но данные должны быть доставлены не всем, а одному члену группы, как правило, «ближайшему» (новый тип адреса, определен в протоколе IPv6). Назначается только интерфейсам маршрутизатора.

Множество всех адресов, которые являются допустимыми в рамках некоторой схемы адресации, называется ***адресным пространством***. Адресное пространство может иметь ***плоскую (линейную)*** или ***иерархическую*** организацию. При плоской организации множество адресов никак не структурировано. При иерархической организации адресное пространство организовано в виде вложенных друг в друга подгрупп.

Наибольшее распространение получили **три схемы адресации узлов** (три типа адресов):

- локальные адреса; - числовые составные адреса; - символьные адреса или имена.

В современных сетях для адресации узлов применяются, как правило, одновременно все три приведенные выше схемы. Пользователи адресуют компьютеры символьными именами, которые автоматически заменяются в сообщениях, передаваемых по сети, на числовые номера. С помощью этих числовых номеров сообщения передаются из одной сети в другую, а после доставки сообщения в сеть назначения вместо числового номера используется аппаратный адрес компьютера.

Протоколы разрешения адресов

Для преобразования адресов из одного вида в другой используются специальные протоколы, которые называют ***протоколами разрешения адресов***.

Проблема установления соответствия между адресами различных типов может решаться **централизованными** или **распределенными средствами**. В случае централизованного подхода в сети выделяется один компьютер (сервер имен), в котором хранится таблица соответствия друг другу имен различных типов, например символьных имен и числовых номеров. Все остальные компьютеры обращаются к серверу имен, чтобы по символьному имени найти числовой номер компьютера, с которым необходимо обменяться данными.

При распределенном подходе, каждый компьютер сам решает задачу установления соответствия между именами. Недостатком распределенного подхода является необходимость рассылки широковещательных сообщений - такие сообщения перегружают сеть, так как они требуют обязательной обработки всеми узлами, а не только узлом назначения. Поэтому распределенный подход используется только в небольших локальных сетях. Для крупных сетей характерен централизованный подход. Наиболее

известной службой централизованного разрешения имен является служба **DNS (Domain Name System)** сети Internet.

Пример использования распределенного подхода – **протокол разрешения адресов ARP (Address Resolution Protocol)**, используемый стеком TCP/IP для преобразования IP-адреса в аппаратный адрес.

Необходимость обращения к протоколу ARP возникает каждый раз, когда модуль IP передает пакет на уровень сетевых интерфейсов, например драйверу Ethernet.

Работа протокола ARP начинается с просмотра **ARP-таблицы**. По таблице определяется нужный MAC-адрес. Для каждой сети, подключенной к сетевому адаптеру компьютера или порту маршрутизатора, строится отдельная ARP-таблица. Пример.

IP-адрес	MAC-адрес	Тип записи
194.85.135.75	00-80-48-EB-7E-60	динамический
194.85.135.70	08-00-5A-21-A7-22	динамический
194.85.60.21	00-80-48-EB-75-67	статический

Работа с таблицей осуществляется с помощью специальной утилиты arp. Таблица выводится на экран по команде arp –a.

Статические записи создаются вручную с помощью утилиты arp. Они находятся в кэше до перезагрузки компьютера.

Динамические записи создаются протоколом ARP, добавляются и удаляются автоматически.

Если запись в течение определенного времени не обновляется, то она исключается из таблицы. Т.о. ARP-таблица содержит записи только об узлах, активно участвующих в сетевых операциях. Поэтому ее еще называют **ARPкэш**.

Если искомого адреса в таблице нет, то протокол ARP широковещательно рассылает **ARP-запрос**, указывая IP-адрес («Чей это IP-адрес и каков ваш адрес сетевого адаптера?»). Узел, IP-адрес которого совпал с указанным в запросе, отправляет **ARP-ответ** с указанием своего локального адреса на машину, сделавшую запрос. После этого новая запись добавляется в ARPтаблицу.

Если в сети нет узла с искомым IP-адресом, ARP-ответа не будет. Протокол IP уничтожает пакеты, направленные по этому адресу.

Локальные адреса ***Аппаратные (hardware) адреса (локальные, физические, MAC-адреса)***. Эти адреса предназначены для сети небольшого или среднего размера, они не имеют иерархической

структуры, используются только аппаратурой для доставки информации в пределах подсети.

Локальный адрес – такой тип адреса, который используется средствами базовой технологии для доставки данных в пределах подсети, являющейся элементом составной сети. В разных подсетях допустимы различные сетевые технологии, следовательно, различные протоколы. Поэтому существуют разные типы локальных адресов.

Подсеть – совокупность хостов, которые взаимодействуют друг с другом не прибегая к маршрутизации. Подсети объединяются в интерсеть с помощью маршрутизаторов.

Для ЛВС локальный адрес – это MAC-адрес (Media Access Control). сетевого адаптера.

У одного узла может быть несколько сетевых интерфейсов и, следовательно, несколько аппаратных адресов.

Типичным представителем локальных адресов является ID-адрес сетевого адаптера локальной сети Ethernet. Его записывают в виде двоичного или шестнадцатеричного значения, например 11-A0-17-3D-BC-01, в ПЗУ платы сетевого адаптера на заводе изготовителе. При замене сетевого адаптера изменяется и аппаратный адрес узла сети.

Стандарты на аппаратные адреса были разработаны IEEE (Institute of Electrical and Electronics Engineers). Для всех технологий ЛВС длина аппаратного адреса 6 байт (280 триллионов адресов). Возможные диапазоны адресов распределяются между многочисленными производителями сетевых адаптеров.

IPv4-адреса

Числовые составные адреса. Во многих случаях для работы в больших сетях в качестве адресов узлов используют числовые составные адреса. Типичным представителями адресов этого типа являются IP- и IPX-адреса. В них поддерживается двухуровневая иерархия, адрес делится на старшую часть - номер сети и младшую - номер узла. Такое деление позволяет передавать сообщения между сетями только на основании номера сети, а номер узла используется только после доставки сообщения в нужную сеть.

IP-адрес – основной тип адреса, на основе которого информация передается между подсетями.

Для протокола IPv4 - это 32-битный адрес (4 байта). Он может быть представлен в двоичном или 16-ричном формате.

Для удобства чтения в технической литературе и прикладных программах IP-адреса представляются в виде 4-х десятичных чисел, разделенных точками. Каждое из чисел соответствует одному октету (8 битам) и может иметь

значения от 0 до 255. Этот формат называется **точечно-десятичным** (DecimalPant Notation).

Например: 10010001.00001010.00100010.00000011

145.10.34.3

IP-адрес состоит из 2-х логических частей: номера сети и номера узла, т.е. используется 2-хуровневая иерархия.

Поле номера сети в IP-адресе (ID сети) называется **сетевым префиксом**, оно идентифицирует подсеть.

Номера сетей назначаются либо централизованно (провайдером), если сеть является частью Internet, либо произвольно, если сеть работает автономно.

Номер узла (ID узла) идентифицирует устройство в подсети и назначается независимо от локального адреса узла.

Межсетевая схема адресации, применяемая в протоколе IP, описана в документах RFC 990, RFC 997.

Устройствами, которым назначаются IP-адреса, могут быть конечные узлы, коммуникационные серверы, маршрутизаторы. Маршрутизатор имеет несколько портов и входит сразу в несколько подсетей, поэтому каждый порт маршрутизатора имеет свой IP-адрес и свой локальный адрес в той подсети, которая к нему подключена. Конечный узел также может входить в несколько IP-сетей, следовательно, может иметь несколько IP-адресов. Т.о., IP-адрес характеризует не отдельный узел, а одно сетевое соединение.

IP-адреса делятся на классы: A, B, C, D, E.

Зная класс, вы можете определить, какая часть адреса относится к номеру сети, а какая к номеру узла, т.е. где проходит граница между сетевым префиксом и номером устройства.

Каждый класс включает в себя определенное количество возможных подсетей, в каждой из которых может содержаться определенное количество хостов. Принято говорить, что подсеть включает в себя **пул адресов** – все возможные номера устройств для данной подсети.

Класс определяется по **ключу** – значению первых бит адреса.

Если адрес начинается с 0, то сеть относят к **классу А**.

Номер сети занимает один байт, остальные 3 байта интерпретируются как номер узла. Сети класса А имеют номера в диапазоне от 1 до 126. (Номер 0 не используется, а номер 127 зарезервирован для специальных целей) Сетей класса А немного, зато количество узлов в них очень велико. На номер сети отводится 7 бит, на номер узла – 24 бита.

Максимальное количество сетей класса А: $2^7 - 2 = 126$ сетей.

Максимальное числом узлов в сети класса А: $2^{24} - 2 = 16\ 777\ 214$ узлов.

Если первые два бита адреса равны 10, то сеть относится к **классу B.**

Сеть класса B является сетью средних размеров. На номер сети отводится 14 бит, на номер узла – 16 бит.

Максимальное количество сетей класса B: $2^{14} - 2 = 16382$ сети

Максимальное числом узлов в сети класса B: $2^{16} – 2 =$ 65 534 узла.

Если адрес начинается с последовательности 110, то это сеть **класса C.**

Это многочисленные небольшие сети.

Под номер сети отводится 24 – 3 = 21 бит, а под номер узла - 8 бит. Сети этого класса наиболее распространены, но число узлов в них мало.

Максимальное количество сетей класса C: $2^{21} - 2 = 2\ 097\ 152$ сети

Максимальное числом узлов в сети класса C: $2^{8} – 2 =$ 254 узла.

Если адрес начинается с ключа 1110, то он является адресом **класса D** и обозначает особый **групповой адрес (multicast).** Адрес класса D не делится на номер сети и номер узла.

Групповой адрес идентифицирует группу узлов, которые в общем случае могут принадлежать разным сетям. Узел, входящий в группу, получает наряду с обычным индивидуальным IP-адресом еще один групповой адрес.

Если при отправке пакета в качестве адреса назначения указан адрес класса D, то такой пакет должен быть доставлен всем узлам, входящим в группу.

Если адрес начинается с ключа 11110, то он относится к классу E. Адреса этого класса зарезервированы и в настоящее время не используются.

В таблице приведены диапазоны номеров сетей и максимальное число узлов, соответствующих каждому классу сетей.

Класс	Первые биты	Наименьший номер сети	Наибольший номер сети	Максимальное число узлов в сети
A	0	1.0.0.0	126.0.0.0	2^{24}
B	10	128.0.0.0	191.255.0.0	2^{16}
C	110	192.0.1.0	223.255.255.0	2^{8}
D	1110	224.0.0.0	239.255.255.255	Multicast
E	11110	240.0.0.0	247.255.255.255	Зарезервирован

Назначение IP-адресов. Маски подсети. Шлюзы

Назначить IP-адреса узлам сети можно двумя способами. 1) Вручную администратором сети (статический адрес); 2) Автоматически.

Назначение IP-адресов узлам сети даже при не очень большом размере сети может быть сложным для администратора. Поэтому в помощь сетевым

администраторам разработаны различные программные продукты и технологии.

Например, для автоматизации назначения IP-адресов используется **протокол DHCP** (Dynamic Host Configuration Protocol) – **протокол динамического конфигурирования хоста.** Описан в RFC 2131, 2132.

Использование DHCP значительно упрощает настройку параметров TCP/IP на компьютерах-клиентах, избавляет администратора от рутинных операций, гарантирует отсутствие конфликтов.

Протокол работает по модели клиент-сервер. Клиенты, поддерживающие DHCP, могут запрашивать у сервера DHCP данные о конфигурации TCP/IP (IP-адрес, маску, адрес шлюза по умолчанию). DHCP-сервер выдает адрес клиенту из пула адресов, заданного администратором, на ограниченное время – время аренды. Если компьютер удаляется из подсети, то назначенный ему IP-адрес автоматически освобождается и может быть выдан другому компьютеру. Это свойство особенно важно для временных пользователей.

Маска – это 32-разрядное двоичное число, которое используется в паре с IP-адресом и содержит единицы в тех разрядах, которые должны в IP-адресе интерпретироваться как номер сети. Т.о. маска используется для определения части IPv4-адреса, которая представляет ID сети.

Маска может быть указана в точечно-десятичной нотации, либо как десятичное число после косой черты вслед за IP-адресом.

Например, маска подсети /16 в представлении с разделительными точками выглядит как 255.255.0.0, а маска подсети /24 - как 255.255.255.0.

В двоичном виде 11111111.11111111.0.0.

Представление с косой чертой, называется представлением бесклассовой междоменной маршрутизации CIDR (Classless Inter Domain Routing). IP адрес 129.64.134.5, маска 255.255.128.0 В двоичном виде:

IP адрес: 10000001.01000000.10000110.00000101

Маска: 11111111.11111111.10000000.00000000

Номер сети: 129.64.128.0 Номер узла: 0.0.6.5

IP адрес AND маска = Номер сети (AND – логическое умножение)

Диапазоны IPv4-адресов IPv4-адреса делятся на индивидуальные и групповые. **Групповые IPv4-адреса**

Групповые адреса начинаются с префикса 1110 (адреса класса D). Значение первого октета у них от 224-х и выше. Они не делятся на номер сети и номер узла - это особый групповой адрес multicast. Пакет с адресом multicast

должны получить все узлы, которым присвоен данный адрес. Групповая адресация широко используется в Интернет.

Индивидуальные IPv4-адреса можно разбить на публичный диапазон, частный диапазон, и адреса APIPA.

Публичные IPv4-адреса

Каждый IPv4-адрес в Интернете должен быть уникален. Распределение адресного пространства курирует **Группа Администрирования адресного пространства Интернет (Internet Assigned Numbers Authority, IANA).**

Частные IPv4-адреса

Администрация IANA зарезервировала определенные диапазоны IPv4адресов в качестве частных адресов. Они никогда не используются в Интернете. Эти частные IPv4-адреса применяются для локальных сетей, которым нужны коммуникации IPv4 без отображения в Интернет.

Диапазоны частных адресов

Начальный адрес	Конечный адрес
10.0.0.0/8	10.255.255.254
172.16.0.0/16	172.31.255.254 (16 номеров сетей)
192.168.0.0/24	192.168.255.254 (255 сетей)

Узлы с частными адресами могут подключаться к Интернету через сервер или маршрутизатор, выполняющий преобразование сетевых адресов NAT. **NAT (Network Address Translation)** - технология **трансляции адресов**– преобразование адресов с помощью специальных таблиц соответствия. В роли маршрутизатора с функцией NAT может выступать компьютер Windows Server 2008 или аппаратный маршрутизатор.

Адреса APIPA (автоматические частные IP-адреса). Если компьютеру автоматически назначается IP-адрес, то по умолчанию в случае недоступности DHCP-сервера всем сетевым подключениям назначаются адреса APIPA.
Частные адреса APIPA расположены в диапазоне от 169.254.0.1 до 169.254.255.254. Маска подсети 255.255.0.0.

Специальные IP-адреса

Некоторые IP-адреса зарезервированы для специальных целей. Такие адреса не назначаются конечным узлам и не передаются маршрутизаторами.

Адрес **0.0.0.0** обозначает все сетевые интерфейсы данного узла либо шлюз по умолчанию (default gateway). В IPv6 это адрес **::**. (IPAddress.Any).

Адрес **255.255.255.255** называется **ограниченным широковещательным сообщением** (limited broadcast). Пакет с таким адресом назначения должен рассылаться всем узлам, находящимся в той же подсети, что и источник пакета. Такие пакеты никогда не передаются через

маршрутизаторы. В IPv6 от имеет префикс 1111 1111 и относится к multicastадресам.

Если в поле номер узла назначения все двоичные единицы, то пакет рассылается всем узлам сети с заданным номером сети. Такая рассылка называется **направленным широковещательным сообщением** (broadcast или multicasting). (Например, 192.190.21.255/24)

Адрес, **первый байт** которого **равен 127 (127.0.0.0 , 127.0.0.1)** называется **loopback, адрес петли обратной связи.** Используется для тестирования программ и взаимодействия процессов в пределах одного узла. Достижим только с локальной машины. Стандартное, официально зарезервированное имя для адресов loopback - **localhost**. В IPv6 это адрес **::1**.

IPv6-адреса

В основном в связи с нерациональным использованием адресного пространства наблюдается дефицит IP-адресов. В последнее время, предлагаются более сложные варианты числовой адресации. Для решения проблемы истощения адресного пространства IPv4 была разработана версия IPv6.

Версия IPv4 обеспечивает 4,3 млрд возможных уникальных адресов. Вместо 32-битовых адресов в версии IPv6 используются 128-битовые. Адресное пространство IPv6 обеспечивает 2^{128}, или 340 282 366 920 938 463 463 374 607 431 768 211 456 ($3.4x10^{38}$) уникальных адресов, что намного больше числа IPv4-адресов.

IPv6 адрес записывается в шестнадцатиричной системе и представляет собой 8 групп по 4 16-ричных цифры. Для разделения групп используется двоеточие.

Например:

2001:00B8:0000:0000:0000:4567:89AB:CDEF

Если одна или более групп подряд равны 0000, то они могут быть опущены и заменены на двойное двоеточие (::). Такой пропуск должен быть единственным в адресе.

Например:

предыдущий адрес можно сократить до 2001:00B8::4567:89AB:CDEF адрес 0000:0000:0000:0000:0000:0000:AE21:0012 можно сократить до :: AE21:0012

Можно также опускать незначащие нули в начале каждой группы. Например вместо 00B8 записать B8.

Для сетей, поддерживающих обе версии протокола IP, разрешается использовать для младших 4 байт традиционную IPv4 запись, а для старших 12 байт – 16-ричную форму.

Например: ::AAAA:128.16.144.38

IPv6-адреса разделены на две части: 64-битовый компонент сети и 64битовый компонент узла. Компонент сети идентифицирует уникальную подсеть, и администрация IANA выделяет эти числа поставщикам ISP или крупным компаниям.

Компонент узла, как правило, основан на уникальном 48-битовом MACадресе сетевого адаптера или генерируется случайным образом.

Для одноадресных типов IPv6 не поддерживает идентификаторы подсетей переменной длины, а число битов, используемых для идентификации сети одноадресного типа IPv6-адреса, всегда равно 64 (первая половина адреса). Поэтому для представления одноадресных типов IPv6 нет необходимости указывать маску подсети, поскольку компьютеры распознают маску /64.

IPv6-адреса используют маски, выражаемые в представлении с косой чертой, однако лишь для описания маршрутов и диапазонов адресов, а не для указания ID сети. Например, в таблице маршрутизации IPv6 можно встретить такую запись: 2001:DB8:3FA9::/48.

В отличие от IPv4, версия IPv6 не использует широковещание в сети. Вместо широковещания в IPv6 применяется многоадресная или групповая передача.

Версия IPv6 изначально проектировалась для обеспечения более простого конфигурирования узлов, чем IPv4. Хотя IPv6 можно конфигурировать и вручную (обычно это требуется для маршрутизаторов), конфигурирование IPv6 на компьютерах практически всегда выполняется автоматически. Компьютеры могут получать IPv6-адреса от соседних маршрутизаторов или DHCPv6-серверов. Кроме того, компьютеры всегда сами назначают себе адрес для использования исключительно в локальной подсети.

Символьные адреса

Компьютеры оперируют числами, в то время как люди более комфортно себя чувствуют, используя слова. Это фундаментальное различие и явилось причиной появления символьных адресов или имен.

Символьные адреса **или** ***имена***. Эти адреса предназначены для запоминания людьми и поэтому обычно несут смысловую нагрузку. Символьные адреса легко использовать как в небольших, так и крупных сетях. Для работы в больших сетях символьное имя может иметь сложную иерархическую структуру. Примеры символьных адресов: DNS, URI, NetBIOS.

Получение имени компьютера - утилита hostname, Панель управления /Система

UNC (Universal Naming Convention)- формат записи имен. UNC-имя имеет следующую структуру:

\\имя_компьютера\имя_общего_ресурса\путь\имя_файла

DNS (Domain Name System) – Доменная система имен, используемая в Internet.

Например, www.microsoft.com

219-3.povt.fitr.bntu.by

URI (Uniform Resource Identifiers) –Универсальный идентификатор ресурса.

Интернет объединяет в единую информационную среду ресурсы, использующие различные способы идентификации. Поэтому была разработана спецификация, которая включала в себя обращения к FTP, Gopher, WAIS, Usenet, E-mail, Prospero, Telnet, HTTP (WWW). URI - это универсальная спецификация, которая позволяет расширять список адресуемых ресурсов за счет появления новых схем. Примеры URI:

http:// www.microsoft.com

ftp://ftp.ncsa.uiuc.edu/Mac/Mosaic news:

msnews.microsoft.com

http://www.globalknowledge.net:80/training/generic.asp?pageid=1078&country=DACH

2.4 Эталонная модель взаимодействия открытых систем

Основные понятия

Проблема совместимости аппаратного и программного обеспечения одна из наиболее острых в компьютерных сетях. Прогресс в этой области невозможен без разработки стандартов. Идеологической основой стандартизации в компьютерных сетях является многоуровневый подход к разработке средств сетевого взаимодействия.

Архитектура подразумевает представление сети в виде системы элементов, каждый из которых выполняет свою функцию, при этом все элементы вместе решают общую задачу взаимодействия компьютеров.

Прорывом в стандартизации архитектуры компьютерной сети стала разработка модели взаимодействия открытых систем. В 1984 г. Международная организация по стандартизации ISO (International Standards Organization) выпустила ряд спецификаций, названных **эталонной моделью взаимодействия открытых систем OSI (Open Systems Interconnection).**
Модель OSI стала международным стандартом для построения сетей различных типов.

Модель OSI, была разработана на основе большого опыта, полученного при разработке компьютерных сетей в 70-е г.г.. Полное её описание занимает более1000 страниц. Модель OSI имела огромный успех. Теперь она используется как справочная модель для описания различных сетевых протоколов и их функциональных возможностей.

В широком смысле **открытой системой** называется любая система, которая построена в соответствии с открытыми спецификациями (опубликованными, общедоступными, соответствующими стандартам). **Спецификация** – формализованное описание аппаратных или программных компонентов, способов их функционирования, взаимодействия с другими компонентами, условий эксплуатации. Не всякая спецификация является стандартом.

Модель OSI касается только одного аспекта открытости – открытости средств взаимодействия устройств, связанных в компьютерную сеть. Здесь **открытая система** - это система, реализующая стандартный набор функций, поддерживаемая стандартными протоколами и отвечающая требованиям эталонной модели OSI.

Если сеть построена с соблюдением принципов открытости, это дает следующие преимущества:

1. возможность построения сети из аппаратных и программных средств различных производителей, придерживающихся одного стандарта;
2. возможность безболезненной замены одних компонентов сети другими, более совершенными;
3. легкое сопряжения одной сети с другими;
4. простота обслуживания сети.

Яркий пример открытой системы: международная сеть Internet (смогла объединить в себе самое разнообразное оборудование и ПО огромного числа сетей по всему миру).

Модель OSI стндартизирует:

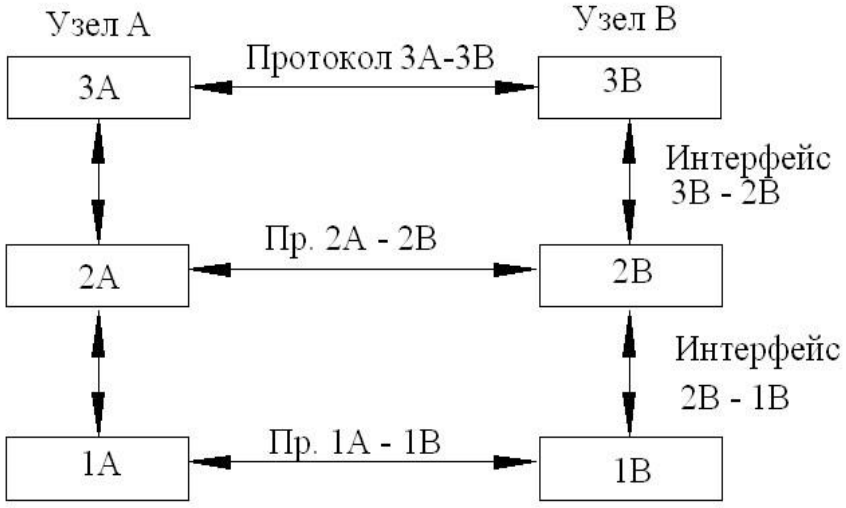
1. Понятия и основные термины, используемые при построении открытых систем;
2. Уровни взаимодействия систем в сетях с коммутацией пакетов;
3. Стандартные названия уровней;
4. Функции, которые должен выполнять каждый уровень.

Протокол – набор формализованных правил, по которым обмениваются информацией сетевые компоненты, лежащие на одном уровне, но в разных узлах сети.

Интерфейс – набор формализованных правил, по которым обмениваются информацией сетевые компоненты соседних уровней одного узла.

Таким образом, интерфейс определяет набор сервисов, предоставляемый данным уровнем соседнему уровню.

В сущности, интерфейс и протокол выражают одно и то же понятие – формализованное описание процедуры взаимодействия двух объектов. Но традиционно в сетях за ними закрепили разные области действия:



Протоколы определяют правила взаимодействия модулей одного уровня в разных узлах, а интерфейсы – модулей соседних уровней в одном узле.

Иерархически организованный набор протоколов, достаточный для организации взаимодействия узлов в сети, называют **стеком коммуникационных протоколов**.

Коммуникационные протоколы могут быть реализованы как программно, так и аппаратно. Протоколы нижних уровней часто реализуются комбинацией программных и аппаратных средств, а протоколы верхних уровней, как правило, чисто программными средствами. Программный модуль, реализующий некоторый протокол, часто то же называют протоколом.

Протоколы реализуются не только компьютерами, но и другими сетевыми устройствами (мостами, маршрутизаторами и т.д.).

Уровни OSI

Модель OSI включает **7-уровней**: прикладной, представительный, сеансовый, транспортный, сетевой, канальный, физический.

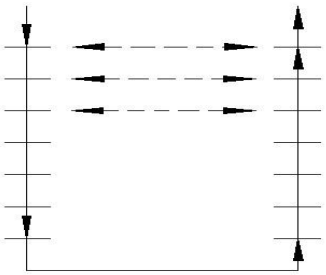
Каждый уровень выполняет свою функцию и имеет дело с определенным аспектом взаимодействия сетевых устройств.

Самый верхний уровень – прикладной, самый нижний – физический.

Обмен данными происходит путем их перемещения с верхнего уровня на нижний, транспортировки по сети и обратного воспроизведения на компьютере-получателе с нижнего на верхний.

При этом на каждом уровне к исходному сообщению, которое надо передать по сети, добавляется заголовок данного уровня, содержащий служебную информацию, необходимую для передачи. На компьютереполучателе каждый уровень в свою очередь анализирует соответствующий ему заголовок, выполняет нужные функции, а затем удаляет этот заголовок и передает сообщение вышележащему уровню.

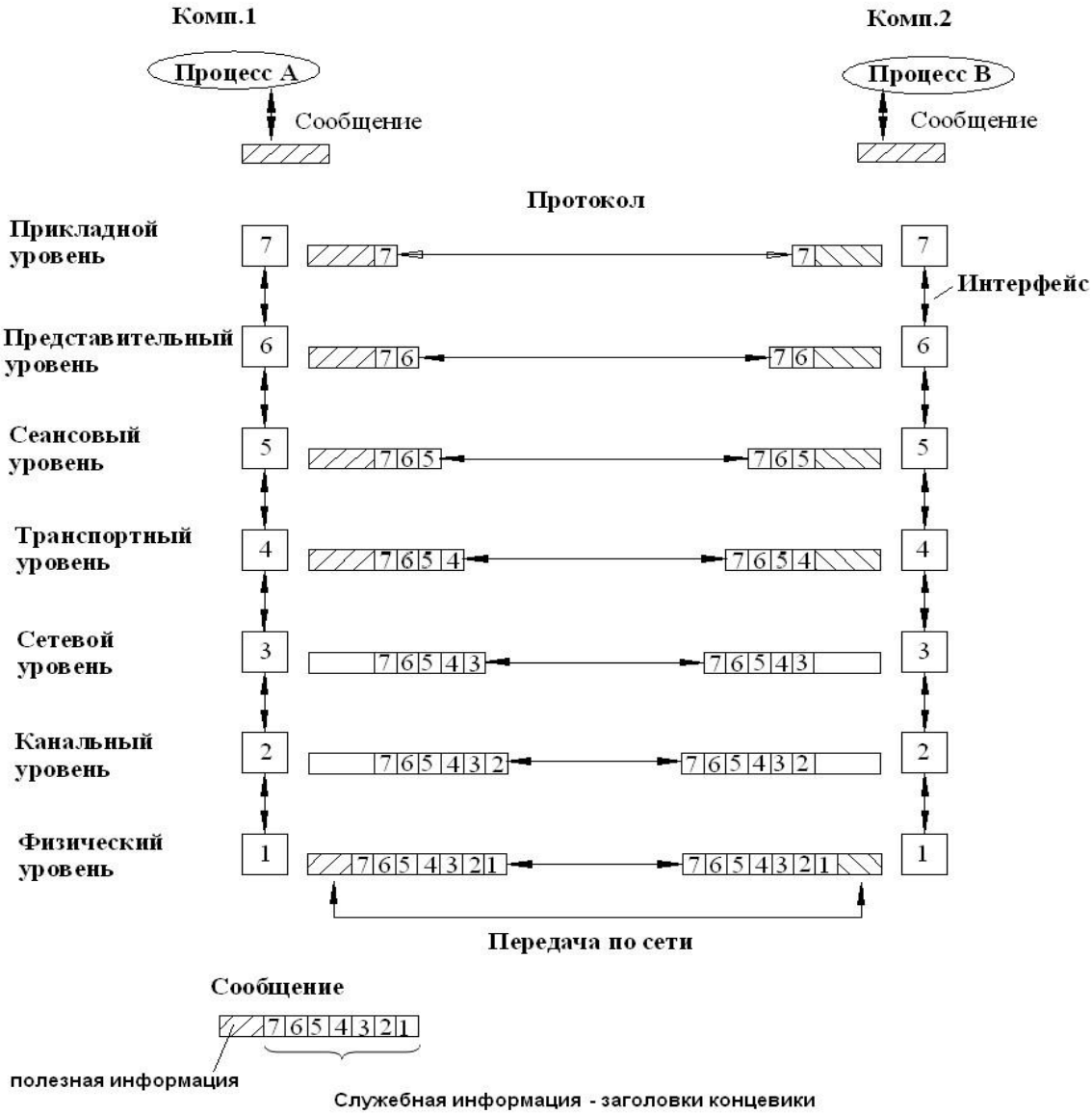
Каждый уровень информация проходит 2 раза – физическая модель. С логической точки зрения – каждый уровень взаимодействует с таким же уровнем на компьютере-получателе (виртуально).



Пример: пусть приложение обращается с запросом к прикладному уровню, например, к файловой службе. На основании этого запроса ПО прикладного уровня формирует сообщение стандартного формата. Обычно сообщение состоит из заголовка и поля данных.

Заголовок содержит служебную информацию, например, место нахождения файла тип операции, которую надо выполнить.

После формирования сообщения прикладной уровень направляет его вниз по стеку к уровню представления. Протокол представительного уровня на основании информации полученной из заголовка прикладного уровня, выполняет требуемые действия и добавляет к сообщению свою служебную информацию (заголовок представительного уровня, в котором содержатся указания для протокола представительного уровня машины адресата).



Далее сообщение передается сеансовому уровню и т.д. Наконец, сообщение достигает физического уровня, который передает его по линиям связи машине-адресату. К этому моменту сообщение «обрастает» заголовками всех уровней. Иногда служебная информация помещается в конец сообщения в виде «концевика». Когда сообщение поступает машине-адресату, оно

принимается её физическим уровнем и последовательно перемещается вверх с уровня на уровень. Каждый уровень анализирует и обрабатывает заголовок-концевик своего уровня, выполняет нужные функции, а затем удаляет этот заголовок и предает сообщение вышележащему уровню.

Модель OSI описывает только системные средства взаимодействия, реализуемые ОС, системными утилитами, системными аппаратными средствами. Модель не включает средства взаимодействия приложений конечных пользователей. Важно различать уровень взаимодействия приложений и прикладной уровень семиуровневой модели.

Для обозначения единиц данных разных уровней в процедурах обмена применяются следующие термины:

- **сообщение** (message) – единица данных прикладного уровня, это логически завершенная порция данных (например, файл);
- **кадр** (frame) - канальный уровень;
- **пакет** (packet) - сетевой уровень;
- **сегмент** (segment) – протокол TCP;
- **дейтаграмма** (datagram);
- **протокольный блок данных** (Protocol Data Unit, PDU) – в стандартах ISO.
- **MTU** (Maximum Transmission Unit) – максимальная единица передачи.

Уровни модели OSI делятся на 2 группы:

- сетезависимые уровни, зависящие от конкретной технической реализации сети;
- сетенезависимые уровни, ориентированные на работу с приложениями. На протоколы этих уровней не влияют какие бы то ни было изменения в топологии сети или сетевой технологии.

К сетезависимым относятся три нижних уровня: сетевой, канальный, физический. К сетенезависимым относятся: прикладной, представительный, сеансовый. Транспортный уровень занимает промежуточное положение между нижними и верхними уровнями.

Сетезависимые уровни OSI

Физический уровень (Physical Layer)

Физический уровень - самый нижний уровень в модели OSI. Описывает процесс прохождения сигналов через среду передачи между сетевыми устройствами (например, по сетевому кабелю). Единица данных – бит.

На этом уровне стандартизируются: характеристики физических сред передачи данных (полоса пропускания, помехозащищенность, волновое сопротивление и т.д.); характеристики электрических и оптических сигналов, передающих дискретную информацию (крутизна фронтов импульсов, уровни напряжения и тока передаваемого сигнала, тип кодирования двоичной информации, скорость передачи и т.д.); способ соединения сетевого кабеля с платой сетевого адаптера (типы разъемов, количество контактов в разъемах и их функции).

Физический уровень должен обеспечивать синхронизацию битов, гарантируя, что переданная единица будет воспринята именно как единица, а не как ноль.

Со стороны компьютера функции физического уровня выполняет сетевой адаптер или последовательный порт.

Пример спецификации физического уровня – 10Base-T (спецификация технологии Ethernet, определяет в качестве кабеля - неэкранированную витую пару категории 3 с волновым сопротивлением 100 Ом, разъем для подключения - RJ-45, максимальная длина физического сегмента кабеля - 100м, метод кодирования – немодулированная передача, манчестерский код).

Канальный уровень (Data-Link Layer)

Канальный уровень обеспечивает надежную передачу данных через физический канал. Он предлагает следующие услуги:

- установление логического соединения между взаимодействующими узлами;
- согласование в рамках соединения скоростей передатчика и приемника информации;
- обеспечение надежной передачи, обнаружение и коррекция ошибок.

Единица данных канального уровня – **кадр**.

Основное назначение канального уровня – прием кадра из сети, формирование кадра и отправка его в сеть. При выполнении этой задачи канальный уровень осуществляет:

- физическую адресацию передаваемых сообщений;
- проверку доступности среды передачи и реализацию соответствующего метода доступа к среде; - выявление неисправностей оборудования;
- обнаружение и коррекцию ошибок передачи.

Все кадры имеют одинаковую структуру.

Протокол канального уровня помещает специальную информацию в начало каждого кадра (ограничитель) и вычисляет контрольную сумму. Получатель снова вычисляет контрольную сумму и сравнивает её с суммой из кадра. Если они совпадают, то кадр принимается. Иначе – фиксируется ошибка.

В сетях, построенных на основе разделяемой среды, канальный уровень выполняет еще одну функцию – проверяет доступность разделяемой среды. Эту функцию иногда выделяют в отдельный подуровень **управления доступом к среде** (MAC).

Реализуются протоколы канального уровня в компьютерах сетевыми адаптерами и их драйверами (аппаратно-программно). Используются протоколы канального уровня конечными узлами и всеми промежуточными устройствами.

В ЛВС в протоколы канального уровня заложена определенная структура связей между компьютерами и способы их адресации.

Примеры протоколов канального уровня ЛВС: Ethernet, Token Ring, FDDI, 100VG-Any LAN.

В ГКС канальный уровень обычно обеспечивает обмен сообщениями между двумя соседними узлами (протоколы «точка-точка»).

Примеры протоколов канального уровня ГКС:

PPP, LAP-B (канальный уровень X.25), LAP-D (ISDN), LAP-F (frame relay), семейство протоколов HDLC.

Сетевой уровень (Network Layer)

Основная функция – доставка данных между сетями.

Сетевой уровень служит для образования единой транспортной системы, объединяющей несколько сетей, называемой составной сетью или интернетом. Т.е. этот уровень обеспечивает доставку данных между любыми двумя узлами в интерсети с произвольной топологией и сетевой технологией, без обеспечения надежности передачи данных.

Термин сеть имеет специфическое значение на сетевом уровне. Под **сетью** понимается совокупность компьютеров, объединенных по одной из базовых сетевых технологий и топологий.

Сетевая технология – согласованный набор стандартных протоколов и реализующих их программно-аппаратных средств, достаточный для построения компьютерной сети. «Достаточный», то есть минимальный набор, с помощью которого можно построить работоспособную сеть. Иногда сетевые технологии называют **базовыми технологиями**, так как на их основе строиться базис любой сети. Таким образом, внутри сети (с базовой

технологией и топологией) доставка данных обеспечивается канальным уровнем. А доставкой данных между сетями занимается сетевой уровень.

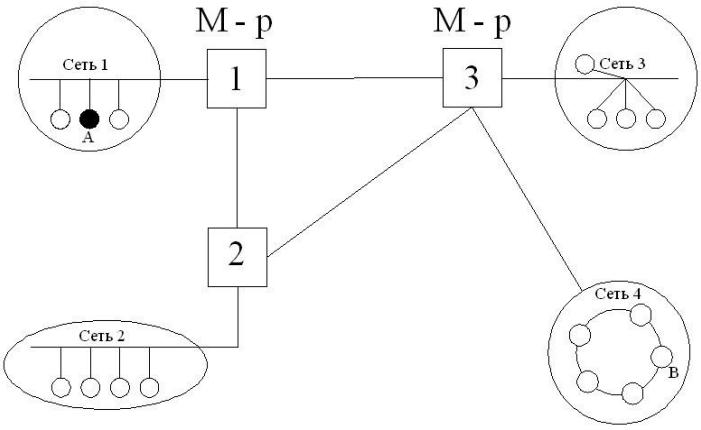
Функции сетевого уровня реализуются:

- группой протоколов;
- специальными устройствами – маршрутизаторами, а также конечными узлами сети.

Маршрутизаторы соединяют между собой сети.

Маршрутизатор – это устройство, которое собирает информацию о топологии межсетевых соединений и на её основании пересылает пакеты сетевого уровня в сеть назначения. Чтобы передать сообщение, нужно совершить некоторое количество транзитных передач между сетями – хопов (hop – прыжок), каждый раз выбирая подходящий маршрут.

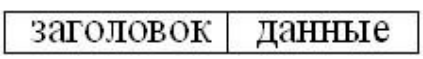
Маршрут – последовательность маршрутизаторов, через которые проходит пакет.



Между узлами А и В пролегают 2 маршрута:

- через маршрутизаторы 1 и 3;
- через маршрутизаторы 1 и 2.

Выбор маршрута называется **маршрутизацией**. Маршрутизация является главной задачей сетевого уровня. Самый короткий путь не всегда самый лучший. Критерии маршрутизации: время передачи, пропускная способность каналов, интенсивность трафика, надежность передачи. Единица данных сетевого уровня – **пакет**. Пакет имеет свою структуру.



Заголовок содержит адрес источника и назначения. Это составной числовой адрес, состоит из № сети и № узла. № сети – для передачи данных между сетями, № узла - внутри сети. Помимо адреса заголовок сетевого уровня

может содержать дополнительную информацию: время жизни пакета, информацию по качеству обслуживания, данные о фрагментации пакетов и т.д.

На сетевом уровне работают различные виды протоколов:

1. **Сетевые протоколы (маршрутизируемые протоколы)**. Служат для передачи пакетов от конечных узлов маршрутизаторам и между маршрутизаторами (IP, IPX).
2. **Протоколы маршрутизации (маршрутизирующие протоколы)**. С помощью них маршрутизаторы собирают информацию о топологии межсетевых соединений и строят таблицы маршрутизации (RIP, OSPF).
3. **Протоколы разрешения адресов**. Преобразуют адрес, используемый на сетевом уровне в локальный адрес сети (ARP: IP в ID).

Протоколы сетевого уровня реализуются драйверами ОС и программноаппаратными средствами маршрутизаторов.

Сетенезависимые уровни OSI Транспортный уровень (Transport Layer)

На пути от отправителя к получателю пакеты могут быть искажены или утеряны.

Транспортный уровень обеспечивает приложениям или верхним уровням стека передачу данных с той степенью надежности, которая им требуется.

Кроме того, транспортный уровень предназначен для:

- управления потоком данных;
- упорядочения и фрагментации пакетов;
- подтверждения передачи или приема.

Он определяет max размер пакета для данной сетевой архитектуры, фрагментирует пакеты, если требуется, следит за их доставкой в определенном порядке, проверяет дубликаты и пересылает потерянные пакеты.

Модель OSI определяет 5 классов сервиса, предоставляемых транспортным уровнем. Они отличаются: срочностью, возможностью восстановления первичной связи, мультиплексированием нескольких соединений, способностью обнаружения и исправления ошибок передачи.

Выбор класса сервиса определяется умением приложения проверять данные и надежностью всей системы транспортировки в сети. Качество каналов связи высоко, вероятность возникновения ошибок мала, следовательно, можно использовать облегченный сервис транспортного уровня. Если сеть ненадежна, то лучше использовать сервис транспортного уровня с максимальными средствами обнаружения и устранения ошибок – с

предварительным установлением соединения, контрольной суммой, нумерацией пакетов, таймаутами и т.д..

Обычно все протоколы, начиная с транспортного уровня и выше, реализуются программными средствами ОС конечных узлов сети.

Примеры: TCP, UDP (TCP/IP). SPX (Novell).

На транспортном уровне приложения идентифицируются через так называемую **конечную точку** (endpoint). В протоколе TCP конечная точка задается комбинацией номера порта и IP-адреса (сокет).

Сеансовый уровень (Session Layer) Создает виртуальное соединение между приложениями.

Сеансовый уровень описывает процедуру установления соединения, чтобы одно приложение могло взаимодействовать с другим.

Сеансовый уровень обеспечивает управление диалогом, предоставляет средства синхронизации сообщений, устанавливает правила начала и завершения взаимодействия, определяет границы сообщений, контрольные точки, сигнализирует приложению об ошибках и т.д.

На практике приложения редко используют сеансовый уровень. Функции этого уровня часто объединяют с функциями прикладного или транспортного уровня и реализуют в одном протоколе (например, TCP).

Уровень представления (Presentation Layer)

Гарантирует, что информация, передаваемая прикладным уровнем одной системы, будет понятна прикладному уровню другой системы.

Представительный уровень работает с формой представления передаваемой информации, не изменяя её содержания. На этом уровне выполняются:

1. Преобразование форматов символов;
2. Преобразование форматов чисел;
3. Сжатие данных;
4. Шифрование и расшифрование.

Пример: SSL (Secure Socket Layer) – протокол, обеспечивает защищенный обмен сообщениями для протоколов прикладного уровня TCP/IP.

Прикладной уровень (Application Layer).

Прикладной уровень обеспечивает функции доступа к разделяемым сетевым ресурсам (сетевым сервисам) – файлам, принтерам, Web-страницам, электронной почте. Прикладной уровень служит пользовательским программым интерфейсом с сетью. Он непосредственно взаимодействует с

пользовательскими прикладными программами, предоставляя им доступ в сеть.

Протоколы прикладного уровня предоставляют приложениям набор сетевых услуг в виде сетевого интерфейса API.

Единица данных прикладного уровня – **сообщение**.

На прикладном уровне работают такие **сетевые приложения как** электронная почта (SMTP), передача файлов по сети (FTP), совместная подготовка документов, доступ к принтерам и т.д.

Примеры прикладных протоколов: NCP (Novell), SMB (MS Win NT), NFS, FTP, TFTP, telnet, SMTP (TCP/IP), HTTP, DNS. Число протоколов прикладного уровня постоянно расширяется (появляются новые сетевые сервисы).

2.5 Архитектура стека TCP/IP

С 1998 года стек TCP/IP вышел в лидеры по числу установок. Сейчас это стек № 1. Любая промышленная операционная система обязательно включает его программную реализацию. Основные идеи TCP/IP:

1. пакетный принцип передачи данных и управления;
2. адаптация длины пакета к условиям передачи (фрагментация/ дефрагментация);
3. инкапсуляция пакетов друг в друга;
4. динамическая маршрутизация.

Перечислим основные преимущества стека TCP/IP.

Независимость от сетевой технологии, т.е. физической архитектуры сети (он позволяет описать и осуществить процессы передачи данных любых типов независимо от типа оборудования, на котором эти процессы происходят). TCP/IP поддерживает практически все существующие технологии локальных и глобальных сетей.

Поддержка стандартных прикладных протоколов. Стек включает средства поддержки основных приложений Интернет: эл. почта, ftp, удал. доступ, WWW и т.д..

Открытость (информация открыто публикуется, в стек постоянно добавляются новые перспективные разработки).

Надежность (протоколы стека обеспечивают подтверждение правильности прохождения информации при обмене между отправителем и получателем) и отказоустойчивость (сеть сохраняет работоспособность, даже если часть сети выйдет из строя).

Масштабируемость.

Гибкая система адресации. Маршрутизируемость.

Так как стек TCP/IP был разработан до появления Эталонной модели OSI (1984г.), то соответствие его уровней с уровнями OSI достаточно условно. В стеке TCP/IP 4 уровня.

TCP/IP	**OSI**
1. Прикладной	Прикладной Представительный
2. Транспортный	Сеансовый Транспортный
3. Сетевой\ Уровень межсетевого взаимодействия\ Интернет-уровень	Сетевой

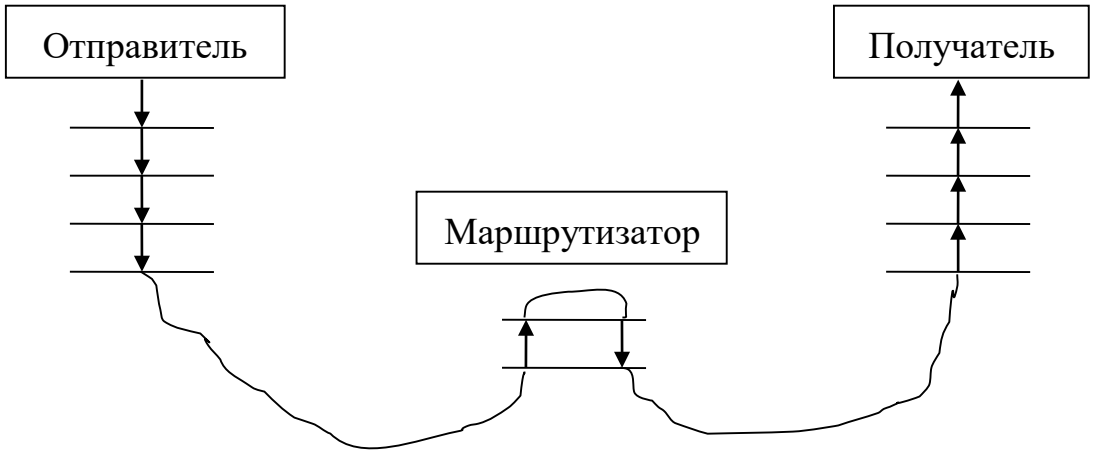
4. Уровень сетевых интерфейсов	Канальный Физический

Каждый из уровней несет часть нагрузки по решению основной задачи: организации надежной и производительной работы составной сети, части которой построены на основе разных сетевых технологий.

В структуре стека имеется явный «центр тяжести» - сетевой уровень и его протокол IP. Протокол IP может взаимодействовать с несколькими протоколами более высокого уровня и несколькими сетевыми интерфейсами.

На практике процесс передачи сообщения от одной прикладной программы к другой будет выглядеть следующим образом.

Отправитель передает сообщение, которое на уровне 3 помещается в дейтаграмму и отправляется в сеть. На промежуточных маршрутизаторах дейтаграмма передается вверх до уровня межсетевого взаимодействия, а затем спускается вниз в другую сеть. У получателя дейтаграмма передается вверх до прикладного уровня.



Уровень сетевых интерфейсов.

Это нижний уровень. Соответствует физическому и канальному уровням OSI. Он не регламентируется.

Основная задача: обеспечивает интеграцию любых типов сетей в составную сеть (отвечает за прием дейтаграмм от сетевого уровня, формирование кадров и передачу их по конкретной сети). Для каждой сетевой технологии должны быть разработаны собственные интерфейсы.

На этом уровне работают протоколы инкапсуляции IP-пакетов уровня межсетевого взаимодействия в кадры локальных технологий. Они описаны в RFC.

Поддерживаются все популярные стандарты физического и канального уровней.

Для ЛВС: Ethernet, Fast Ethernet, Gigabit Ethernet, Token Ring, FDDI. Для ГВС: X.25, Frame Relay, ATM, протоколы «точка-точка» SLIP и PPP (использ для передачи TCP/IP по телефонным линиям).

Обычно при появлении новой перспективной технологии она быстро включается в стек TCP/IP за счет разработки соответствующего RFC.

Уровень межсетевого взаимодействия (Интернет, сетевой).

Основная функция: реализует передачу пакетов в составных сетях, состоящих из большого количества ЛВС и ГВС, объединенных глобальными и локальными связями.

Протоколы:

1) Основной протокол этого уровня – IP(Internet Protocol). Реализует передачу пакетов в дейтаграммном режиме по тому маршруту, который в данный момент является наиболее рациональным.

2) Протоколы маршрутизации (протоколы сбора маршрутной информации, создают и поддерживают таблицы маршрутизации). Например: RIP, OSPF, BGP.

3) Протоколы разрешения адресов. Сюда относится ARP – преобразует IP-адрес в локальный адрес конкретной сетевой технологии.

4) Протоколы управления и обмена информацией об ошибках.

ICMP (Internet Control Message Protocol) – протокол межсетевых управляющих сообщений. Это сетевой протокол для передачи команд и сообщений об ошибках, выполняет диагностические функции и сообщает об ошибках и сбоях в сетях TCP/IP. Определен в RFC 792. Каждое сообщение ICMP передается по сети внутри IP-пакета.

IGMP (Internet Group Management Protocol) – протокол управления группами. Используется IP-узлами для сообщения поддерживающим групповую передачу маршрутизаторам о своем участии в группах.

Транспортный уровень. Функции:

1) Обеспечивает сеансы связи между узлами.

2) Обеспечивает надежность передачи информации между 2-мя конечными узлами (т.к. IP не гарантирует доставку).

На транспортном уровне в Интернет работают протоколы UDP (без установления соединения) и TCP (с установлением соединения).

TCP (Transmission Control Protocol) – протокол управления передачей.

UDP (User Datagram Protocol) – протокол дейтаграмм пользователя.

Это два принципиально разных подхода к передаче данных. В обоих случаях и передатчик, и приемник имеют индивидуальные IP-адреса и порты. Но в случае TCP они ассоциируются в соединители (socket) — две пары

IPадрес-порт, и прием/передача в рамках одной сессии происходит по схеме точка-точка. Для UDP же допускается возможность передачи одновременно нескольким приемникам (мультикастинг) и прием данных от нескольких передатчиков в рамках одной и той же сессии.

Протокол TCP используется для поточной передачи данных, при которой доставка гарантируется на протокольном уровне. Это обеспечивается обязательным подтверждением получения каждого пакета TCP.

Напротив, протокол UDP не требует подтверждения получения. В этом случае, как правило, исключается также и фрагментация пакетов, так как пакеты при схеме без установления соединения никак не связаны между собой. По этим причинам UDP в основном служит для передачи мультимедийных данных, где важнее своевременность, а не надежность доставки. Протокол TCP применяется там, где важна надежная, безошибочная доставка информации (файловый обмен, передача почтовых сообщений и WEB-технология).

TCP решает две основные задачи: 1) управление потоком пакетов, 2) контроль ошибок. Обеспечивает гарантированную доставку данных за счет образования логических соединений между удаленными процессами. Это протокол с предварительным установлением соединения, медленный. TCP делит поток байт на части – сегменты и передает их нижележащему уровню межсетевого взаимодействия. После доставки сегментов TCP снова собирает их в поток байт.

UDP обеспечивает передачу прикладных сообщений дейтаграммным способом и выполняет функции связующего звена между сетевым протоколом и службами прикладного уровня. Быстрый, т.к. не осуществляет контроля доставки пакетов, не устанавливает соединений. Используется в ситуациях, когда вероятность потери пакетов мала и подтверждение приема не нужно. Ответственность за доставку при этом несет само приложение.

Прикладной уровень.

Объединяет все службы (сервисы), предоставляемые операционной системой пользовательским приложениям.

Протоколы этого уровня занимаются работой конкретного приложения и не затрагивают способов передачи данных по сети.

Включает множество протоколов и постоянно расширяется.

FTP (File Transfer Protocol) – протокол передачи файлов между удаленными системами.

HTTP (HyperText Transfer Protocol) – протокол передачи гипертекста. **Telnet** – протокол эмуляции удаленного терминала. Обеспечивает передачу потока байтов между процессами, а также между процессом и терминалом.

При использовании Telnet пользователь фактически управляет удаленным компьютером так же, как локальный пользователь. Поэтому серверы Telnet должны как минимум использовать аутентификацию по паролю, а лучше более мощные средства защиты, например службу Kerberos.

DNS (Domain Name System) – протокол разрешения доменного имени в IP-адрес.

SMTP (Simple Mail Transfer Protocol) – простой протокол передачи почты.

SNMP (Simple Network Management Protocol) используется для организации сетевого управления. Изначально был разработан для удаленного контроля и управления маршрутизаторами Интернет. Затем его стали применять и для управления любым коммуникационным оборудованием – концентраторами, мостами, коммутаторами, сетевыми адаптерами.

2.6 Основы технологии сокетов

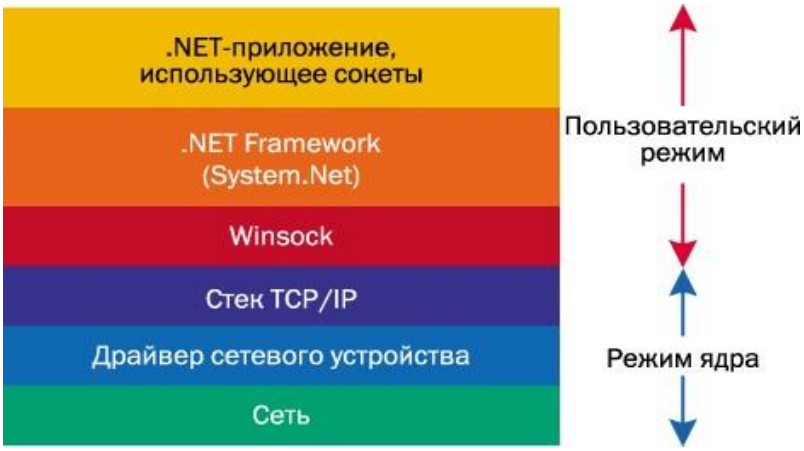
Понятие и типы сокетов ***Сокет*** (Socket - гнездо, разъем) - абстрактное программное понятие, используемое для обозначения в прикладной программе конечной точки сетевого соединения.

Технология (интерфейс) сокетов – название программного интерфейса для обеспечения обмена данными между процессами. Процессы при таком обмене могут выполняться как на одном компьютере, так и на разных, связанных между собой сетью.

Интерфейс сокетов был разработан в Калифорнийском университете в г. Беркли и встроен в ОС BSD Unix в 1983 г. (Сокеты Беркли)

Важнейшим преимуществом сокетов является предоставление единого независимого интерфейса сетевого программирования для различных сетевых протоколов (TCP/IP, IPX/SPX, NetBIOS/SMB, AppleTalk, ATM).

Библиотека Win32 Windows Sockets (Winsock API) предоставляет механизмы программирования сокетов. В Microsoft .NET Framework имеется более высокий по отношению к Winsock уровень, благодаря чему управляемые приложения также могут взаимодействовать через сокеты.



Интерфейс сокетов расположен над транспортным уровнем стека TCP/IP, но в некоторых случаях может взаимодействовать напрямую с сетевым уровнем в обход транспортного.

Существуют три основных типа сокетов: потоковые, дейтаграммные и сырые.

Потоковые сокеты – это сокеты с установлением соединения, состоящие из потока байтов, который может быть двунаправленным. Т.е. через такую конечную точку приложение может и передавать, и получать данные. Потоковый сокет гарантирует обнаружение и исправление ошибок, обрабатывает доставку и сохраняет последовательность данных. Он подходит

для передачи больших объемов данных, поскольку в этом случае накладные расходы, связанные с установлением соединения, незначительны по сравнению со временем передачи самого сообщения. Качество передачи достигается за счет использования протокола TCP.

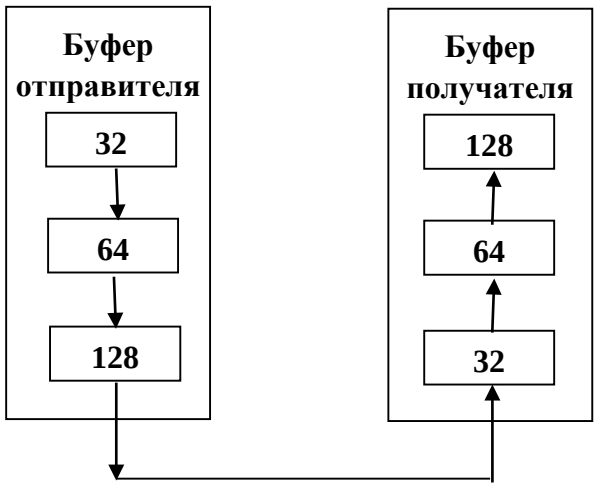
Дейтаграммные сокеты – это сокеты без установления соединения, не обеспечивающие надежность при передаче. Применяются для приложений, когда неприемлемы затраты времени, связанные с установлением явного соединения, для групповой передачи. Работают быстрее. Используется с протоколом UDP.

Сырые сокеты (raw sockets - необрабатываемые, простые) – это сокеты, которые взаимодействуют с протоколами сетевого уровня в обход протоколов транспортного уровня. Используются для непосредственного доступа приложения к IP-пакетам сетевого уровня. Использование сырых сокетов возможно при разработке низкоуровневого системного ПО. Например, сырые сокеты используют различные программы-анализаторы пакетов, сниферы, утилиты TCP/IP ping, tracert и т д.

Различают потоковые протоколы и протоколы, ориентированные на передачу сообщений.

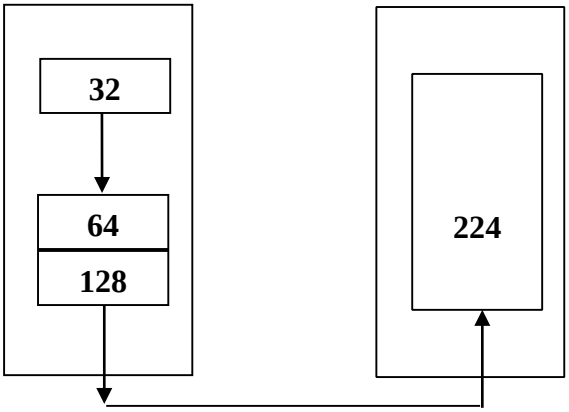
Протоколом, ориентированным на передачу сообщений, называется протокол, сохраняющий границы сообщений. Для каждой команды ввода\вывода он передает байты в отдельном сообщении.

Например, источник отправляет сообщения в 3-х командах соответственно по 128, 64 и 32 байта. Приемник с буфером в 256 байт выполняет 3 команды чтения. На каждый запрос чтения возвращается соответственно 128, 64 и 32 байта. Первый запрос чтения не возвращает сразу 3 пакета, даже если все они уже получены.



Потоковым протоколом называется протокол, не сохраняющий границы сообщений.

Потоковая служба непрерывно передает данные. Получатель считывает столько данных, сколько имеется в наличии, независимо от границ сообщений. Система может разбить исходное сообщение на части или объединить несколько сообщений и сформировать большой пакет данных как при получении, так и при отправке.



Узел может накапливать данные в буфере перед отправкой. Аналогично на стороне получателя принимающий буфер накапливает поступающие данные прежде, чем передать их конкретному процессу.

Если получатель потребует прочитать только 20 байт, система вернет 20 байт. Если получатель считывает данные в 256-байтный буфер, то все 224 байта возвращаются сразу.

Адресация сокетов. Номера портов Интерфейс сокетов поддерживает разнообразные сетевые стеки протоколов: TCP/IP, IPX/SPX, NetBIOS/SMB, AppleTalk, ATM, Infrared Sockets. Каждому из них соответствует свое семейство адресов сокетов. Напрмер, стеку TCP/IP соответствует семейство адресов InterNetwork для адресов IPv4, InterNetworkV6 для адресов IPv6. Стеку IPX/SPX соответствует семейство адресов Ipx и т.д.

Семейство адресов – важнейший параметр сокета. Он указывает используемый в настоящее время сетевой протокол и ограничивает применение других параметров сокета.

Мы рассмотрим адресацию сокетов только для стека протоколов TCP/IP, как самого распространенного на сегодняшний день.

Адрес сокета при использовании протоколов TCP/IP – это IP-адрес и номер порта прикладной службы.

IP-адрес уникален в Internet. Номер порта уникален на отдельном компьютере. Следовательно, адрес сокета будет уникален в Internet. Это позволяет удаленным процессам общаться исключительно на основе адреса сокета.

При работе с сокетами могут использоваться некоторые специальные IPадреса. Например, для отладки клиент-серверного приложения на одном компьютере применяется адрес петли обратной связи 127.0.0.1 или соответствующее ему имя localhost.

Для получения информации на всех сетевых интерфейсах локального узла используется специальный адрес 0.0.0.0.

Порт задается, чтобы решить задачу одновременного сетевого взаимодействия с несколькими приложениями на одном узле. Если на компьютере выполняется несколько приложений, то получая пакет из сети, можно идентифицировать приложение по уникальному номеру порта, который задан при установлении связи.

Программисты должны быть внимательны при выборе номера порта, поскольку некоторые доступные порты зарезервированы для использования популярными службами, такими как FTP, HTTP и т.д. Эти порты обслуживаются и распределяются центром Internet Assigned Numbers Authority (IANA).

Номера портов разделяются на 3 категории (стандартные, зарегистрированные и динамические и/или частные):

- Номера от 0 до 1023 зарезервированы для стандартных служб.
- Порты с номерами от 1024 до 49151 являются регистрируемыми.
- Порты с номерами от 49152 до 65535 - динамические и частные порты.

Во избежание накладок с портами, уже занятыми системой или другим приложением, ваша программа должна выбирать порты, начиная с 1024. Можно вместо конкретного номера порта задать 0, тогда система сама выберет произвольный неиспользуемый в данный момент номер.

Запустив утилиту netstat –a, можно увидеть перечень всех используемых в данный момент на компьютере номеров портов.

В файле services из каталога <windir>\system32\drivers\ets перечислены предопределенные пользовательские и системные номера портов, используемых стандартными службами. Если порт содержится в перечне этого файла, то утилита netstat вместо номера порта отобразит имя протокола.

Порядок байт

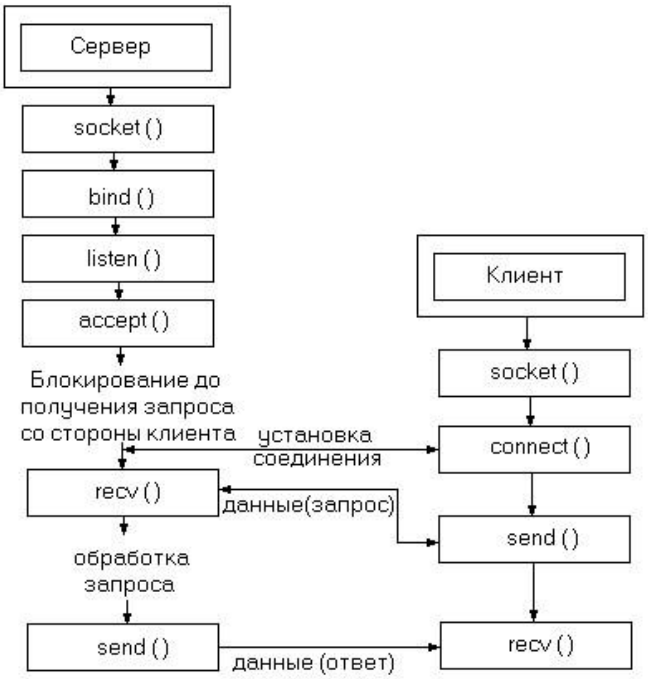
В памяти компьютера IP-адрес и номер порта представляются в ***системном порядке (host-byte-order)***. Для Intel-совместимых процессоров это

порядок от менее значимого к более значимому байту. Его называют ***обратный*** или ***little endian*** (младший байт числа расположен в младшем адресе памяти). Для других типов процессоров, например Motorola, используется ***прямой порядок*** или ***big endian*** (в младшем адресе сохраняется старший байт числа).

Сетевые стандарты требуют, чтобы многобайтные значения передавались в ***сетевом порядке (network-byte order)***. Это прямой порядок следования байтов. Поэтому в некоторых случаях, когда системный и сетевой порядок байт не совпадает, существует необходимость преобразования чисел из одной формы в другую. Для этого разработаны специальные функции и методы.

Коммуникационная модель клиент-сервер

Приложение, использующее сокеты, состоит из распределенной программы, исполняемой на обоих концах канала связи. Программу, инициирующую передачу, называют ***клиентом***. ***Сервер*** представляет собой модуль, пассивно ожидающий входящих запросов на установку соединений от удаленных клиентов. Как правило, серверное приложение загружается при запуске системы и прослушивает свой порт, ожидая входящих соединений. Клиентские приложения пытаются установить соединение с сервером, после чего начинается обмен данными. По завершении сеанса связи клиент, как правило, разрывает соединение. На рисунке представлена базовая модель взаимодействия для потоковых сокетов.



3 Лабораторный практикум

3.1 Лабораторная работа № 1 Утилиты TCP/IP

Цель работы

Познакомиться со средствами диагностики сети и поиска неисправностей стека TCP/IP.

Методические указания 1.

Диагностические утилиты TCP/IP.

Для диагностики и поиска неисправностей работы сети можно использовать специальные утилиты, предназначенные для проверки конфигурации стека TCP/IP и тестирования сетевого соединения. Список некоторых утилит приведен в таблице 1.

Таблица 1 – Диагностические утилиты TCP/IP.

Утилита	**Применение**
arp	Выводит для просмотра и изменения таблицу трансляции адресов, используемую протоколом разрешения адресов ARP (Address Resolution Protocol - определяет локальный адрес по IP-адресу).
hostname	Отображает имя локального хоста. Используется без параметров.
getmac	Отображает MAC-адреса сетевых адаптеров компьютера
ipconfig	Выводит значения для текущей конфигурации стека TCP/IP: IP-адрес, маску подсети, адрес шлюза по умолчанию, адреса WINS (Windows Internet Naming Service) и DNS (Domain Name System)
netstat	Выводит статистику и текущую информацию по соединению TCP/IP.
ping	Осуществляет тестирование сетевых подключений.
tracert	Осуществляет проверку маршрута к удаленному компьютеру путем отправки эхо-пакетов протокола ICMP (Internet Control Message Protocol). Выводит маршрут прохождения пакетов на удаленный компьютер.
pathping	Аналогична утилите tracert, предназначена для определения потерь данных на промежуточных узлах.
route	Просмотр и редактирование таблицы маршрутищации
nslookup	Определение ip-адреса по доменному имени и имен-псевдонимов

Синтаксис использования для всех утилит одинаков. В окне работы с командной строкой после приглашения операционной системы указывается имя утилиты, пробел, параметры. Команды нечувствительны к регистру.

C:\WINDOWS>ping –n 10 www.gmail.com

Справочную информацию по любой из утилит можно получить, используя /? после имени утилиты.

C:\WINDOWS>ipconfig /?

2. Проверка правильности конфигурации TCP/IP.

При устранении неисправностей и проблем в сети TCP/IP следует сначала проверить правильность конфигурации TCP/IP. Для этого используется утилита ipconfig.

С помощью утилиты ipconfig можно получить сведения обо всех сетевых интерфейсах данного узла, в частности, всю адресную информацию (имя узла, все IP-адреса узла, маски, адреса шлюзов, адреса DNS-серверов, все физические адреса узла).

Эта команда особенно полезна на компьютерах, работающих с **DHCP (Dynamic Host Configuration Protocol) – протокол динамического конфигурирования хоста.** Утилита позволяет определить, какая конфигурация сети TCP/IP и какие величины были установлены с помощью DHCP. Синтаксис:

ipconfig [/all | /renew[adapter] | /release]

Параметры:

all выдает весь список параметров. Без этого ключа отображается только IP-адрес, маска и шлюз по умолчанию; renew[adapter] обновляет параметры конфигурации DHCP для указанного сетевого адаптера; release[adapter] освобождает выделенный DHCP IP-адрес; adapter – имя сетевого адаптера; displaydns выводит информацию о содержимом локального кэша клиента DNS, используемого для разрешения доменных имен.

Таким образом, утилита ipconfig позволяет выяснить, инициализирована ли конфигурация TCP/IP и не дублируются ли IP-адреса:

- если конфигурация инициализирована, то появляется IP-адрес, маска, шлюз;
- если IP-адреса дублируются, то маска сети будет 0.0.0.0;
- если при использовании DHCP компьютер не смог получить IP-адрес, то он будет равен 0.0.0.0 .

3. Тестирование связи с использованием утилиты ping.

Утилита ping (Packet Internet Grouper) используется для тестирования сетевого соединения с удаленным узлом, сервером, маршрутизатором, диагностики ошибок соединения, а также для проверки конфигурирования TCP/IP. Она определяет доступность указанного узла и позволяет измерить время прохождения пакетов от данного узла до любого другого узла сети. Использование ping лучший способ проверки того, что между локальным компьютером и сетевым хостом существует маршрут. **Хостом** называется любое сетевое устройство (компьютер, маршрутизатор), обменивающееся информацией с другими сетевыми устройствами по TCP/IP.

Команда ping проверяет соединение с удаленным хостом. Она посылает к указанному пользователем хосту несколько IP-пакетов (по умолчанию 4) и ожидая ответы на них. При этом она измеряет интервал времени, в течение которого пакет вернулся, а также показывает соотношение количества отосланных пакетов к количеству принятых, что может служить субъективной оценкой «качества связи» между узлами. Если связь между хостами плохая, из сообщений ping станет ясно, сколько пакетов потеряно.

Ping можно использовать для тестирования как имени хоста (DNS или NetBIOS), так и его IP-адреса. Если ping с IP-адресом выполнилась успешно, а с именем – неудачно, это значит, что проблема заключается в распознавании соответствия адреса и имени, а не в сетевом соединении.

Утилита ping используется следующими способами:

1) Для проверки того, что TCP/IP установлен и правильно сконфигурирован на локальном компьютере, в команде ping задается адрес петли обратной связи (loopback address): ping 127.0.0.1

2) Чтобы убедиться в том, что компьютер правильно добавлен в сеть и IP-адрес не дублируется, в утилите ping используется IP-адрес локального компьютера:

ping IP-адрес_локального_хоста

3) Чтобы проверить, что шлюз по умолчанию функционирует и что можно установить соединение с любым хостом в локальной сети, задается IP-адрес шлюза по умолчанию: ping IP-адрес_шлюза

4) Для проверки возможности установления соединения через маршрутизатор в команде ping задается IP-адрес или имя удаленного хоста: ping IP-адрес_удаленного хоста

либо ping имя хоста

Синтаксис утилиты ping:

ping [-t] [-a] [-n count] [-l length] [-f] [-i ttl] [-v tos] [-r count] [-s count] [[-j host-list] | [-k host-list]] [-w timeout] destination-list

Параметры:

-t выполняет команду ping до прерывания. Control-Break - посмотреть статистику и продолжить. Control-C - прервать выполнение команды;

-a позволяет определить доменное имя удаленного компьютера по его IPадресу;

-n count посылает количество пакетов ECHO, указанное параметром count;

-l length посылает пакеты длиной length байт (максимальная длина 8192 байта);

-f посылает пакет с установленным флагом «не фрагментировать». Этот пакет не будет фрагментироваться на маршрутизаторах по пути своего следования;

-i ttl устанавливает время жизни пакета в величину ttl (каждый маршрутизатор уменьшает ttl на единицу);

-v tos устанавливает тип поля «сервис» в величину tos;

-r count записывает путь выходящего пакета и возвращающегося пакета в поле записи пути. Count - от 1 до 9 хостов;

-s count позволяет ограничить количество переходов из одной подсети в другую (хопов). Count задает максимально возможное количество хопов;

-j host-list направляет пакеты с помощью списка хостов, определенного параметром host-list. Последовательные хосты могут быть отделены промежуточными маршрутизаторами (гибкая статическая маршрутизация). Максимальное количество хостов в списке, позволенное IP, равно 9;

-k host-list направляет пакеты через список хостов, определенный в host-list. Последовательные хосты не могут быть разделены промежуточными маршрутизаторами (жесткая статическая маршрутизация). Максимальное количество хостов – 9;

-w timeout указывает время ожидания (timeout) ответа от удаленного хоста в миллисекундах (по умолчанию – 1сек);

destination-list указывает IP-адрес или имя удаленного хоста, к которому надо направить пакеты ping.

Пример использования утилиты ping.

C:\WINDOWS>ping –n 10 www.netscape.com

Обмен пакетами с www.netscape.com [205.188.247.65] по 32 байт:

Ответ от 205.188.247.65: число байт=32 время=194мс TTL=48
Ответ от 205.188.247.65: число байт=32 время=240мс TTL=48
Ответ от 205.188.247.65: число байт=32 время=173мс TTL=48
Ответ от 205.188.247.65: число байт=32 время=250мс TTL=48
Ответ от 205.188.247.65: число байт=32 время=187мс TTL=48
Ответ от 205.188.247.65: число байт=32 время=239мс TTL=48
Ответ от 205.188.247.65: число байт=32 время=263мс TTL=48
Ответ от 205.188.247.65: число байт=32 время=230мс TTL=48
Ответ от 205.188.247.65: число байт=32 время=185мс TTL=48
Ответ от 205.188.247.65: число байт=32 время=406мс TTL=48 Статистика Ping для 205.188.247.65:

Пакетов: послано = 10, получено = 10, потеряно = 0 (0% потерь) Приблизительное время передачи и приема:

Наименьшее = 173мс, наибольшее = 406мс, среднее =236мс

5. Изучение маршрута между сетевыми соединениями с помощью утилиты tracert.

Tracert - это утилита трассировки маршрута. Она позволяет проследить путь от данного узла до любого другого узла сети Internet. Хост за хостом показывается прохождение IP-пакетов, при этом выводится название и IP-адрес каждого пройденного хота, а также значение интервала времени, в течение которого был получен ответ.

Утилита использует поле TTL (time-to-live, время жизни) из заголовка IP-пакета и сообщения об ошибках протокола ICMP для определения маршрута от одного хоста до другого.

Утилита tracert может быть более содержательной и удобной, чем ping, особенно в тех случаях, когда удаленный хост недостижим. С помощью нее можно определить район проблем со связью (у Internet-провайдера, в опорной сети, в сети удаленного хоста) по тому, насколько далеко будет отслежен маршрут. Если возникли проблемы, то утилита выводит на экран звездочки (*), либо сообщения типа «Destination net unreachable», «Destination host unreachable», «Request time out», «Time Exeeded».

Маршрут определяется путем изучения сообщений ICMP, которые присылаются обратно промежуточными маршрутизаторами.

ПРИМЕЧАНИЕ: некоторые маршрутизаторы просто молча уничтожают пакеты с истекшим TTL и не будут видны утилите tracert. Синтаксис:

tracert [-d] [-h maximum_hops] [-j host-list] [-w timeout] имя_целевого_хоста

Параметры:

-d указывает, что не нужно распознавать адреса для имен хостов;

-h maximum_hops указывает максимальное число хопов для того, чтобы искать цель;

-j host-list указывает нежесткую статическую маршрутизацию в соответствии с host-list;

-w timeout указывает, что нужно ожидать ответ на каждый эхо-пакет заданное число мсек.

6. Определение потерь данных с помощью утилиты PathPing.

Эта утилита аналогична утилите Tracert, за исключением того, что PathPing предназначена для определения потерь данных на промежуточных узлах. Утилита PathPing отправляет пакеты на каждый маршрутизатор на пути к конечной точке и вычисляет процентное соотношение пакетов, возвращаемых в каждом прыжке. Поскольку утилита PathPing показывает степень потери пакетов на каждом маршрутизаторе или узле, с ее помощью можно точно определить маршрутизаторы и узлы, на которых возникают сетевые проблемы.

Для определения потерь данных в командную строку надо ввести команду PathPing и указать имя или адрес конечного компьютера, сервера или маршрутизатора, путь к которому необходимо отследить.

7. Утилита arp.

ARP – это имя утилиты и протокола.

Основная задача протокола разрешения адресов ARP (Address Resolution Protocol) – трансляция IPv4-адресов в соответствующий MAC-адрес (аппаратный, локальный) сетевого интерфейса. Для этого ARP-протокол использует информацию из ARP-таблицы (ARP-кэша). Если необходимая запись в таблице не найдена, то протокол ARP отправляет широковещательный запрос ко всем компьютерам локальной подсети, пытаясь найти владельца данного IP-адреса. В кэше могут содержаться два типа записей: статические и динамические. Статические записи вводятся вручную и хранятся в кэше постоянно. Динамические записи помещаются в кэш в результате выполнения широковещательных запросов. Для них существует понятие времени жизни. Если в течение определенного времени (по умолчанию 2 мин.) запись не была востребована, то она удаляется из кэша.

Утилиту arp можно использовать для отображения и редактирования ARP-кэша компьютера и для определения MAC-адресов узлов подсети. Синтаксис: arp [-s inet_addr eth_addr] | [-d inet_addr] | [-a] Параметры:

-s занесение в кэш статических записей;

-d удаление из кэша записи для определенного IP-адреса;

-a просмотр содержимого кэша для всех сетевых адаптеров локального компьютера;

inet_addr - IP-адрес; eth_addr - MAC-адрес.

Например, отобразив кэш с помощью команды arp -a, можно обнаружить проблему, связанную с тем, что соседние виртуальные машины назначили себе один и тот же виртуальный MAC-адрес (довольно распространенная ситуация).

8. Утилита netstat.

Утилита netstat производит отображение активных подключений TCP, портов, прослушиваемых компьютером, статистики Ethernet, таблицы маршрутизации IP, статистики IPv4 (для протоколов IP, ICMP, TCP и UDP) и IPv6 (для протоколов IPv6, ICMPv6, TCP через IPv6 и UDP через IPv6). Особенно она полезна на брандмауэрах, с ее помощью можно обнаружить нарушения безопасности периметра сети. Синтаксис:

netstat [-a] [-e] [-n] [-s] [-p protocol] [-r]

Параметры:

-a выводит перечень всех сетевых соединений и прослушивающихся портов локального компьютера. Если порт содержится в перечне файла services из

каталога <Windir>\system32\drivers\etc, то утилита netstat вместо номера порта отобразит имя протокола;

-e выводит статистику для Ethernet-интерфейсов (например, количество полученных и отправленных байт);

-n выводит информацию по всем активным соединениям (например, TCP) для всех сетевых интерфейсов локального компьютера. Для каждого соединения выводится информация об IP-адресах локального и удаленного интерфейсов вместе с номерами используемых портов;

-s выводит статистическую информацию для протоколов UDP, TCP, ICMP, IP. Ключ «/more» позволяет просмотреть информацию постранично;

-r выводит содержимое таблицы маршрутизации.

9. Утилита route

Утилита route предназначена для просмотра и редактирования таблицы маршрутизации (добавление маршрута, удаление маршрута, редактирование маршрута). Вывод таблицы маршрутизации на экран: route print

10. Команда nslookup

Команда nslookup используется с целью получения доменного имени, IP-адреса или другой информации из записей сервера DNS.

Команда nslookup может работать в интерактивном и неинтерактивном режимах. Интерактивный режим позволяет в режиме диалога отправлять DNS-серверу запросы о различных узлах и доменах. Неинтерактивный режим позволяет отправить один запрос об одном узле или домене. Синтаксис команды:

nslookup [опции] [имя/адрес хоста] [адрес DNS сервера]

Если в запросе указать только доменное имя хоста, то будет выведен IP-адрес используемого для ответа DNS-сервера и IP-адреса запрашиваемого хоста (тип записи «A» (DNS, IP-адрес), а также его алиасные имена. Пример команды: nslookup www.microsoft.com

Задания на лабораторную работу

Упражнение 1. Получение справочной информации по командам TCP/IP, командам ОС MS Windows, сетевым командам

Вывести на экран справочную информацию по утилитам TCP/IP. Для этого в командной строке ввести имя утилиты без параметров или с /?. Изучить ключи, используемые при запуске утилит.

Например: ipconfig /? (выводит справочную информацию по команде ipconfig)

Упражнение 2. Получение имени хоста

Вывести на экран имя локального хоста с помощью команды hostname.

Упражнение 3. Получение MAC-адресов сетевых адаптеров

Вывести на экран MAC-адреса сетевых адаптеров с помощью утилиты getmac.

Упражнение 4. Чтение результатов ipconfig

Изучить конфигурацию TCP/IP локального хоста с помощью утилиты ipconfig. Использовать утилиту без параметров и с параметром /all

1. Определить символьное имя узла.
2. Сколько физических сетевых интерфейсов у данного узла? Перечислите их. Укажите их адреса.
3. Сколько программных сетевых интерфейсов назначено узлу? Перечислите их. (* возле подключения по локальной сети означает, что это туннельный интерфейс)
4. Сколько IPv4 и IPv6-адресов назначено узлу? Перечислите их. Укажите для каждого IP-адреса основные настройки TCP/IP – маску и адрес шлюза по умолчанию.

Упражнение 5. Тестирование связи с помощью утилиты ping

1. Проверить правильность установки и конфигурирования TCP/IP на локальном компьютере.
2. Проверить, правильно ли добавлен в сеть локальный компьютер и не дублируется ли IP-адрес.
3. Проверить функционирование шлюза по умолчанию, послав 5 эхо-пакетов длиной 64 байта.
4. Проверить с помощью ping, можно ли обратиться к компьютерам в своей локальной сети по имени компьютера, по IPv4-адресу, по IPv6-адресу (указав идентификатор зоны %n своей машины).
5. Указать в ping адрес компьютера, который отключен, несуществующий адрес. Сравнить полученные результаты?
6. Проверить возможность установления соединения с различными удаленными хостами, используя DNS-имена. Определить IP-адреса этих узлов. Отметить время отклика (время кругового обращения пакета). Попробовать увеличить время отклика. Как влияет размер пакета на время кругового обращения?

Упражнение 6. Определение пути IP-пакета

1. Воспользоваться командой tracert для определения числа участков маршрута от вашего компьютера к различным хостам (локальному хосту, шлюзу по умолчанию, удаленному хосту). Отметьте, через какие промежуточные узлы проходят эхо-пакеты.
2. Сравнить значения времени кругового обращения, полученные при выполнении программы ping, с числом участков маршрута, полученным при выполнении программы tracert, для ряда адресов назначения. Существует ли зависимость между продолжительностью задержки и числом участков маршрута?

Упражнение 7. Утилита PathPing

Используя утилиту PathPing, определить потери данных на промежуточных узлах при тестировании маршрута к различным хостам. Прокомментировать полученные результаты.

Упражнение 8. Утилита arp

С помощью утилиты arp просмотреть ARP-таблицу локального узла. Какая информация в ней хранится? Для чего она нужна?

Упражнение 9. Получение информации о текущих сетевых соединениях и протоколах стека TCP/IP

1) С помощью утилиты netstat вывести на экран перечень сетевых соединений и используемых в данный момент портов локального узла. Просмотреть информацию о состоянии соединения. Выяснить, находится ли соединение в состоянии прослушивания или уже установлено.

2) Получить статистическую информацию для протоколов UDP, TCP, ICMP, IP.

3) Вывести на экран локальную таблицу маршрутизации. Изучить ее содержимое.

Контрольные вопросы

1. Дайте определение компьютерной сети. Из каких компонентов состоит компьютерная сеть? Что такое хост?
2. Как определить имя компьютера?
3. Как определить, сколько у данного компьютера физических сетевых интерфейсов, виртуальных сетевых интерфейсов? Как определить физический адрес компьютера?
4. Как узнать, в какое количество подсетей в данный момент подключен компьютер?
5. Каким образом команда ping проверяет сетевые соединения? Какой протокол использует утилита ping? Отметьте возможные причины, по которым ping не может связаться с удаленным хостом.
6. Если утилита ping с IP-адресом выполнилась успешно, а с именем хоста неудачно, что это означает?
7. Что такое «петля обратной связи»?
8. Для чего предназначена и как работает утилита tracert? Чем отличается использование утилит tracert и PathPing?
9. Каково назначение протокола ARP? Что такое ARP-кэш? Для чего используется утилита arp?
10. Как просмотреть перечень всех используемых в данный момент портов?

Дополнительные источники

1. Энциклопедия Windows. Все об использовании и настройке Windows. Утилиты для диагностики TCP/IP

Режим доступа: https://windata.ru/windows-xp/faq-xp/utility-dlya-diagnostiki-tcp-ip/

2. ИТ Проффи. Статьи и полезные заметки для системных и сетевых администраторов

Режим доступа: https://itproffi.ru

3.2 Лабораторная работа № 2 Установка и настройка ORACLE VM VIRTUALBOX

Цель работы

Познакомиться с ПО Oracle VM VirtualBox. Изучить способы создания и настройки виртуальных машин.

Методические указания

Oracle VM VirtualBox – это кросс-платформенное средство для программной виртуализации. VirtualBox может быть установлен на компьютеры с операционными системами MS Windows, Linux, Mac OS X, Solaris x86 и другими. VirtualBox позволяет одновременно на одном компьютере создавать и запускать виртуальные машины, использующие различные операционные системы. Например, можно запустить Windows и Linux на Mac, или Linux и Solaris из-под Windows, или Windows из-под Linux.

VirtualBox поддерживает различные виды сетевого взаимодействия между виртуальными машинами, а также виртуальными машинами и хостом.

Программа поддерживает цепочки сохраненных состояний виртуальной машины (snapshots), к которым может быть произведён откат из любого состояния гостевой системы Также осуществляется поддержка Shared Folders для простого обмена файлами между хостовой и гостевой системами.

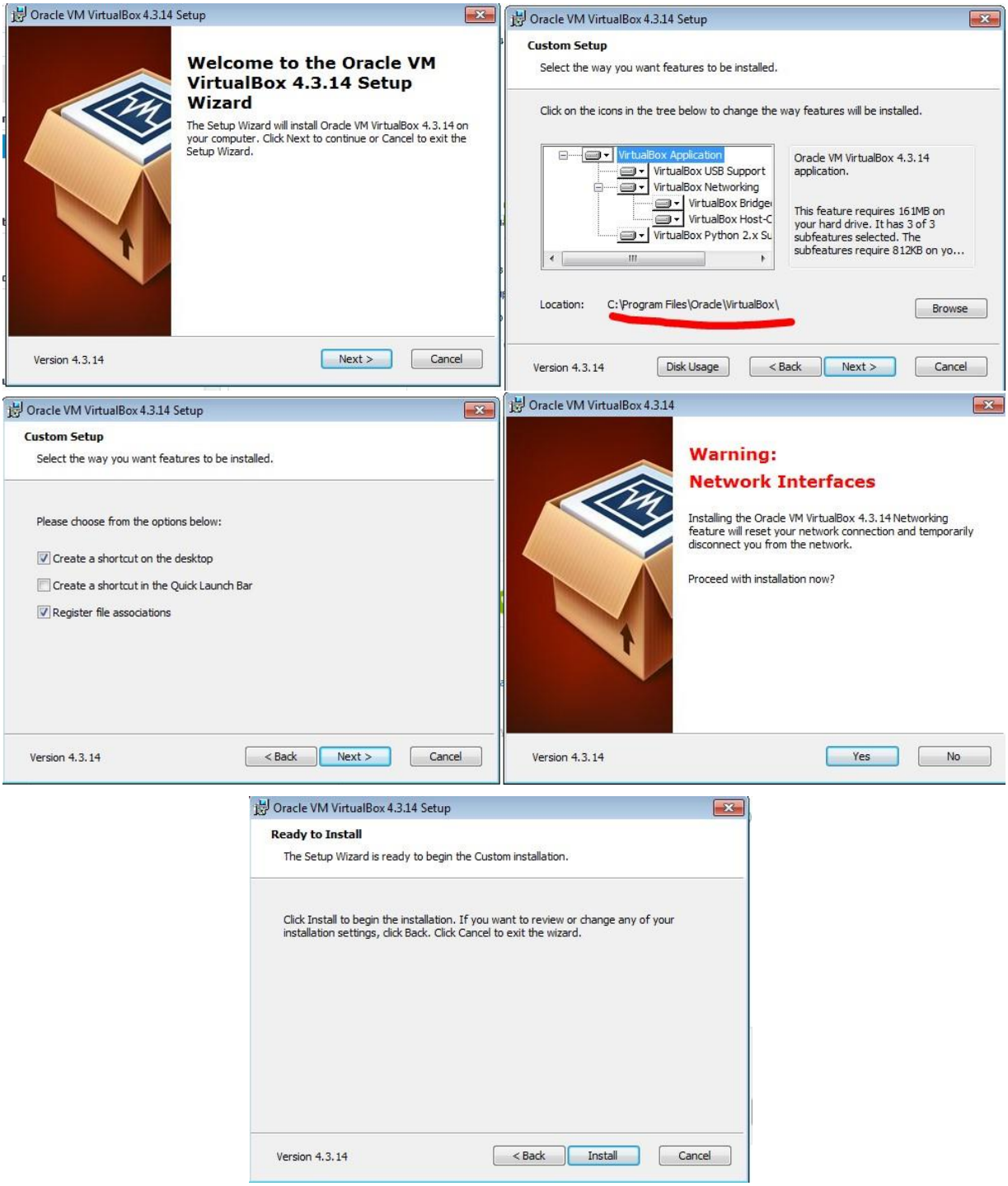
Постановка задачи

1. Установить и настроить Oracle VM VirtualBox.
2. Создать две виртуальные машины под управлением ОС Windows Server 2008. Присвоить машинам имена Serv1 и Comp1.
3. Подготовить виртуальные машины для выполнения последующих лабораторных работ.

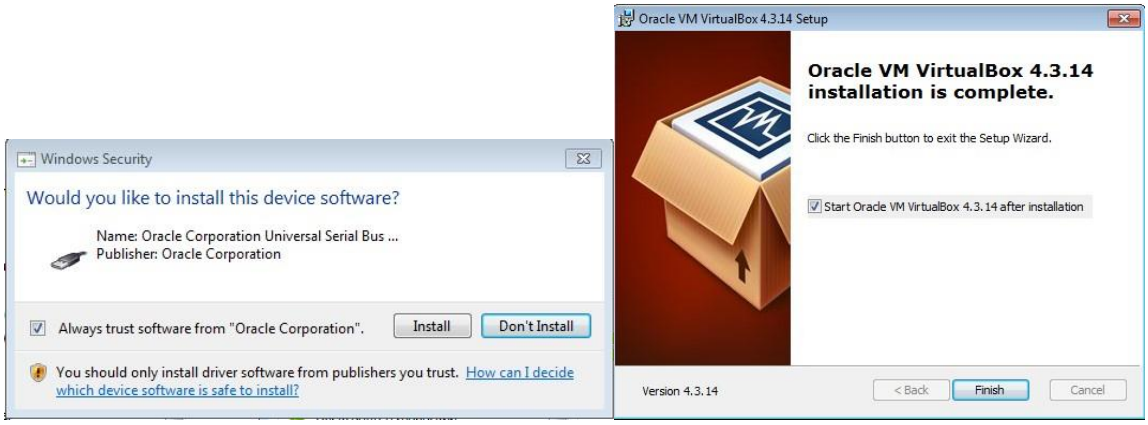
Порядок выполнения работы

1. Установка Oracle VM virtual box

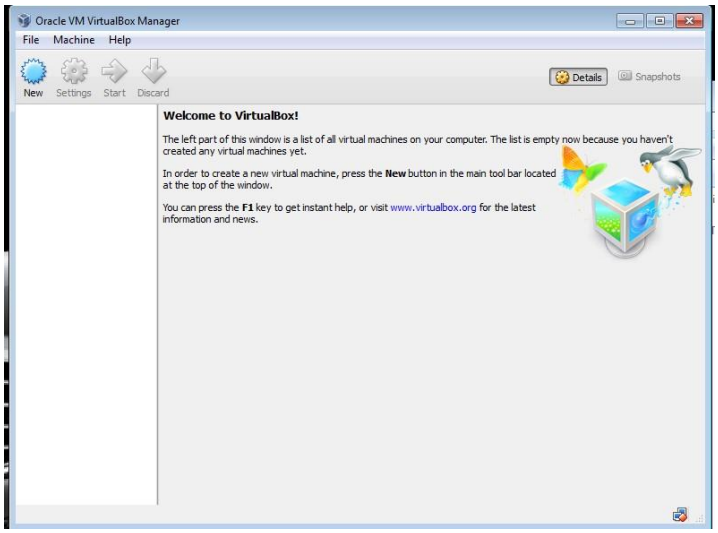
Запускаем установщик VirtualBox-4.3.14-95030-Win.exe. Жмем далее, пока не начнется процесс установки.



На вопрос в появившемся окне отвечаем install.

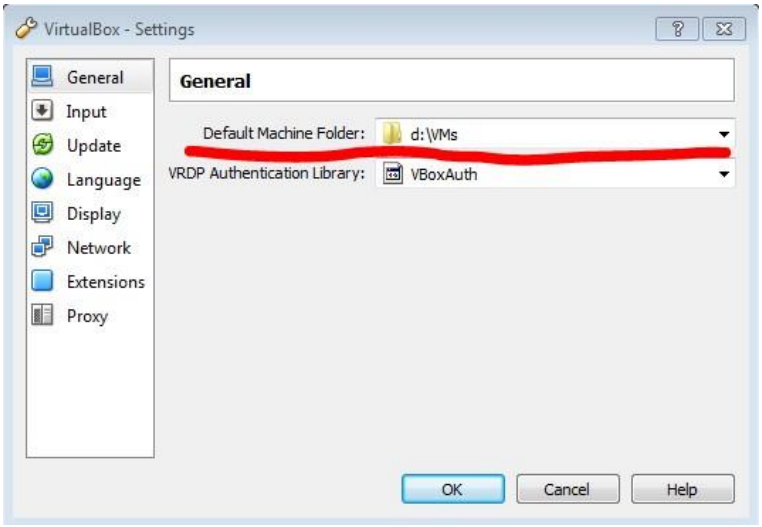


2. Настройка VirtualBox. Запускаем программу.



Переходим в настройки программы: File->Preferences (Файл->Настройки).

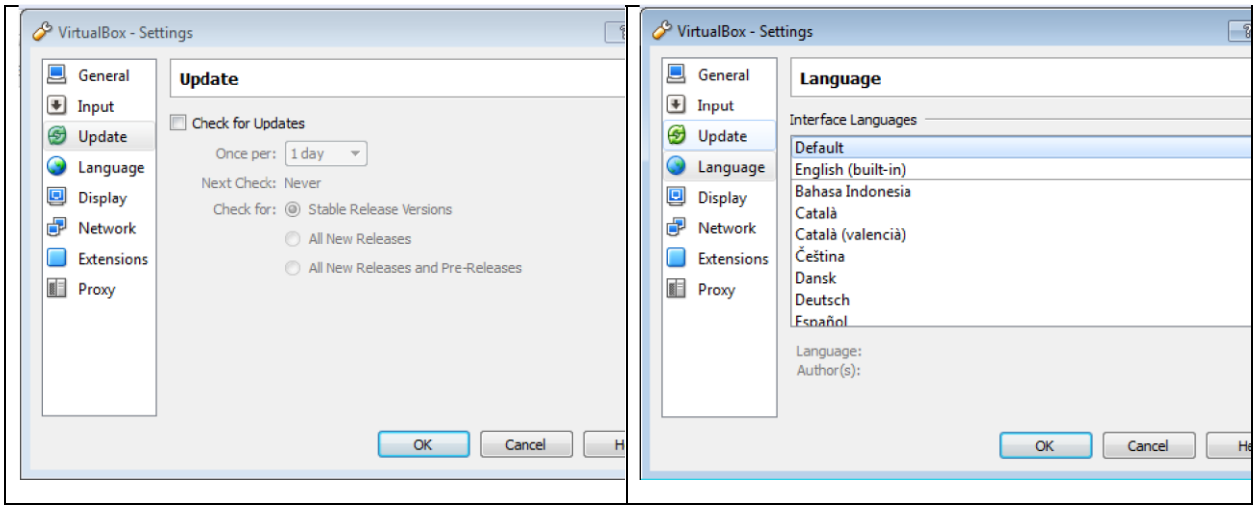
В настройках на вкладке General (Общие) выбираем папку для хранения виртуальных машин.



На вкладке Input (Ввод) ничего не изменяем.

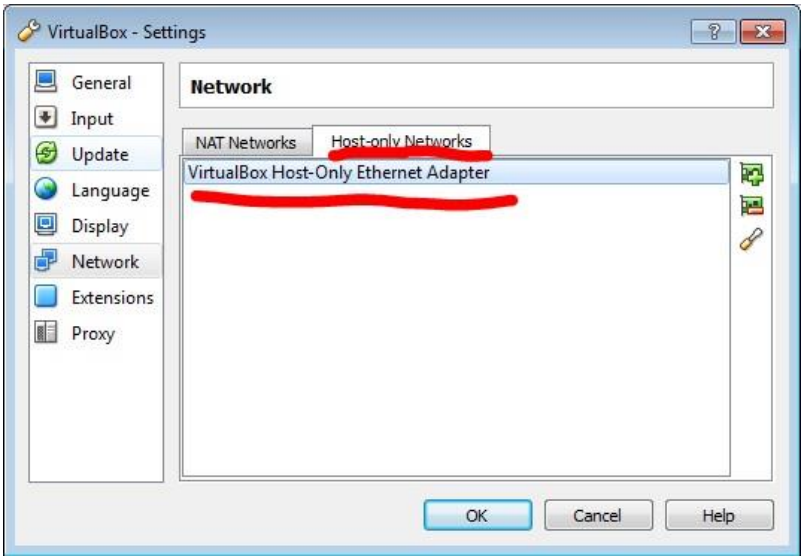
На вкладке Update (Обновления) выключаем проверку обновлений. Нам не особенно важны обновления, которые будут приходить.

На Вкладке Language (Язык) выбираем язык.

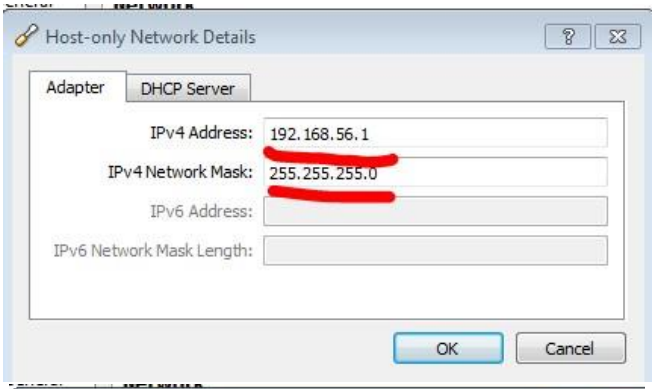


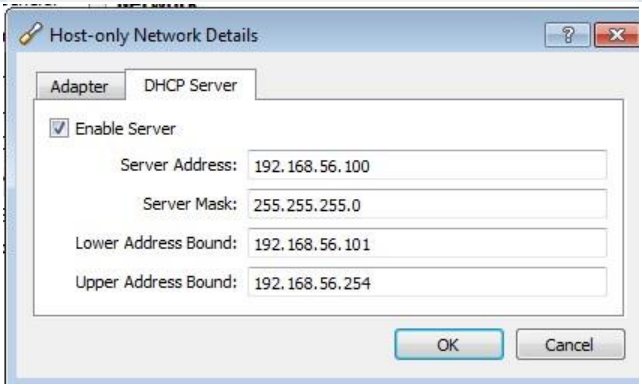
Ничего не изменяем на вкладке Display.

На вкладке Network (Сеть) переходим на вкладку Host-only Networks (Виртуальные сети хоста) и открываем настройки VirtualBox Host-Only Ethernet Adapter.



В настройках сетевого адаптера смотрим IP адрес хоста и маску сети (вкладка Adapter), а так же настройки DHCP сервера (вкладка DHCP Server).





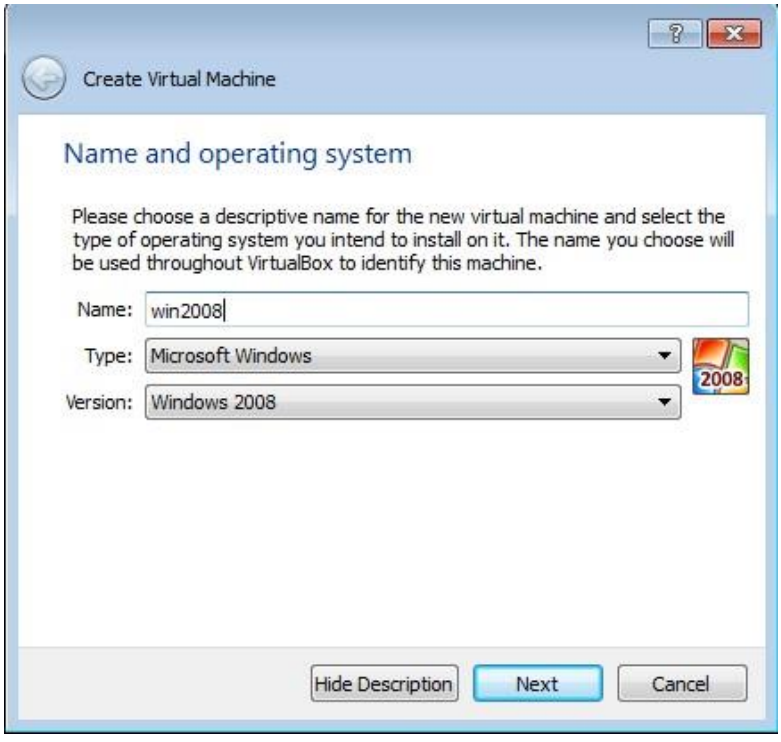
На вкладках Extensions и Proxy ничего не изменяем и нажимаем кнопку OK. Базовые настройки выполнены.

3. Создание виртуальной машины с операционной системой Windows Server 2008.

Нажимаем кнопку New 

на панели управления или выбираем в меню: Machine>new.

В диалоговом окне создания машины вводим имя машины (Serv1), выбираем семейство операционных систем (в нашем случае это Microsoft Windows) и версию: Windows 2008.



Нажимаем далее.

Выбираем объем ОЗУ (512 Мб, можно больше.). Нажимаем далее.

В появившемся окне для выбора жесткого диска выбираем пункт Create a virtual hard drive now и нажимаем Create.

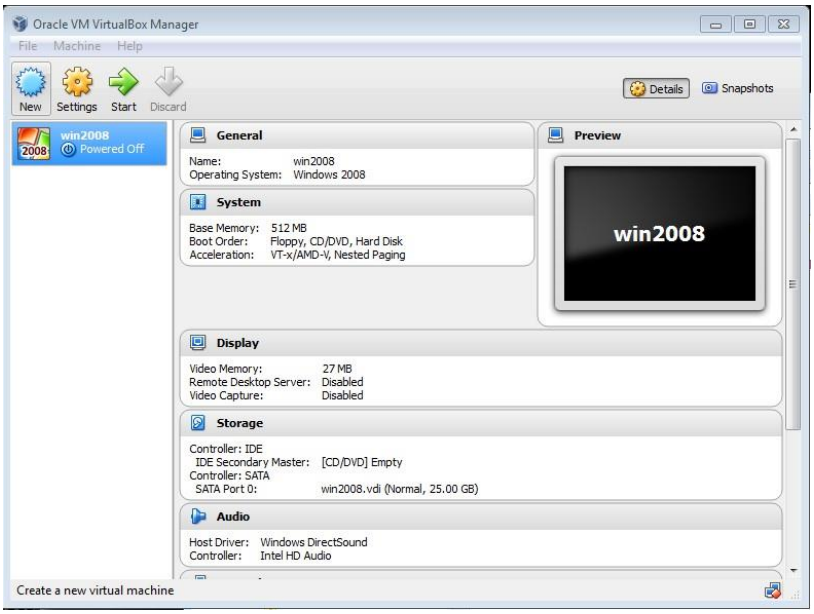
Тип диска оставляем по умолчанию VDI (VirtualBox Disk Image). Нажимаем далее.

В окне Storage on physical hard drive выбираем пункт Dynamically allocated, чтобы место под диск выделялось по мере наполнения диска, а не все сразу. Нажимаем далее.

Размещение диска оставляем по умолчанию, размер указываем около 20 Гб.

Все диск готов! Машина тоже! ☺

Созданная виртуальная машина будет видна в главном окне приложения.



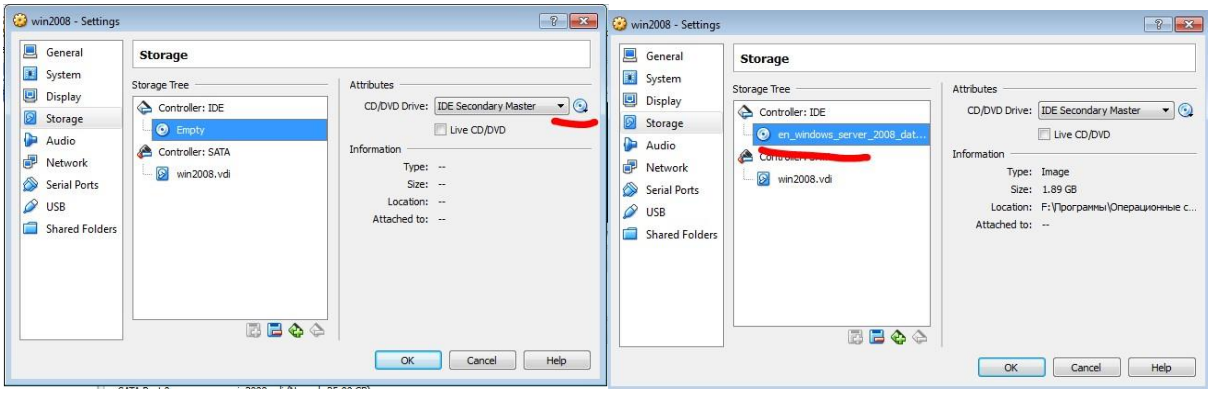
Теперь необходимо ее настроить!

4. Настройка виртуальной машины.

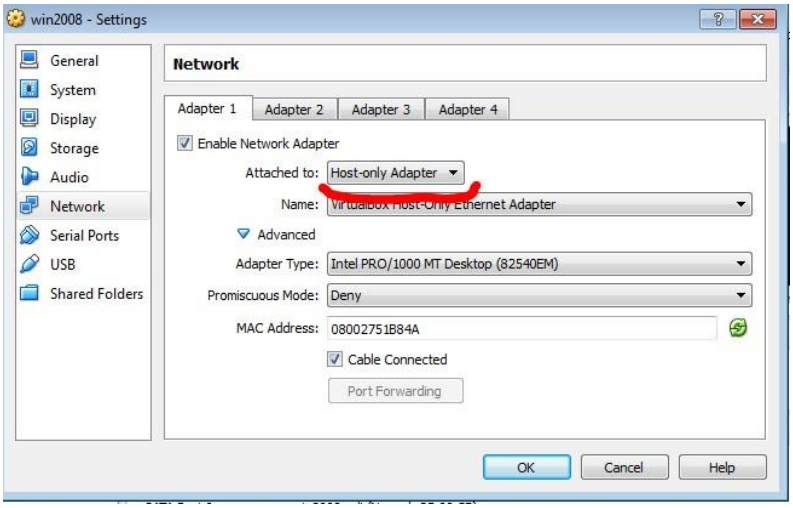
Переходим в настройки. Для этого нажимаем кнопку 

на панели управления.

В окне настроек на вкладке Storage (Носители) ставим образ ОС в привод виртуальной машины:



На вкладке Network (Сеть) настроим сетевой интерфейс.



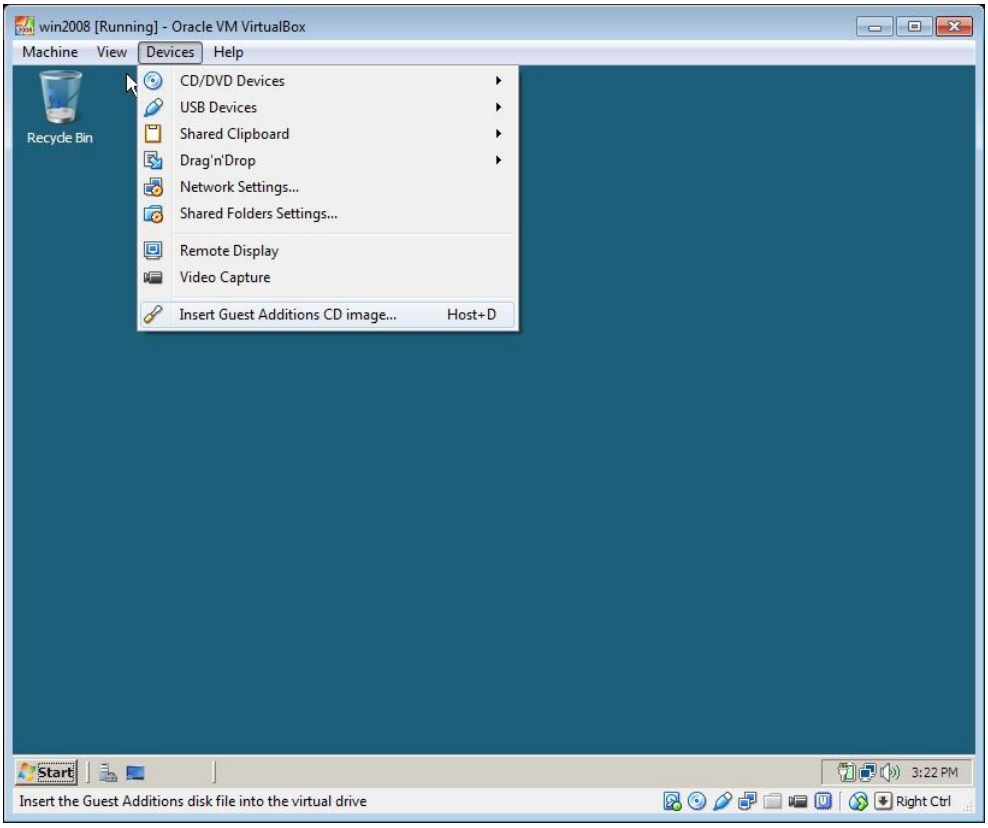
Нажимаем ОК.

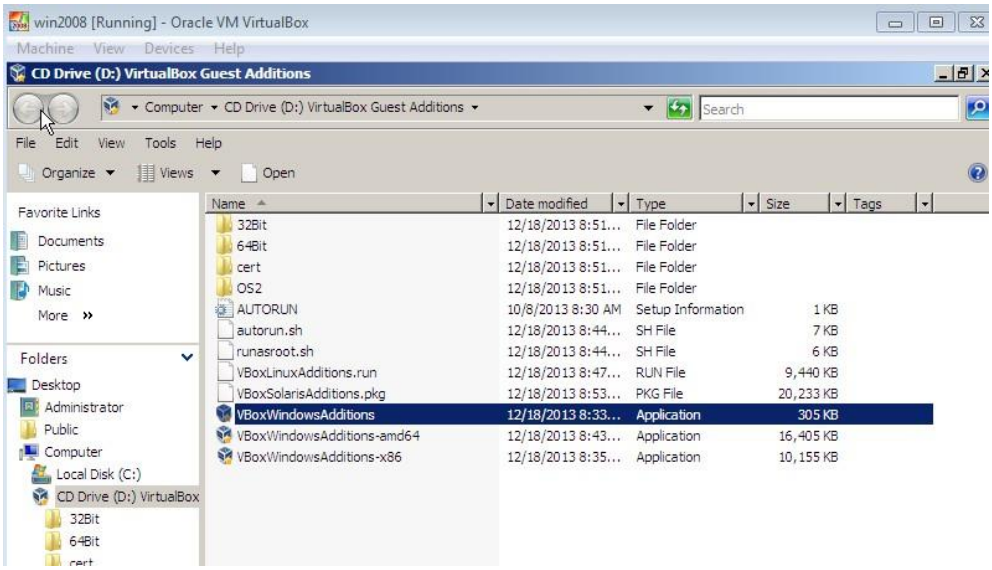
Машину можно запускать

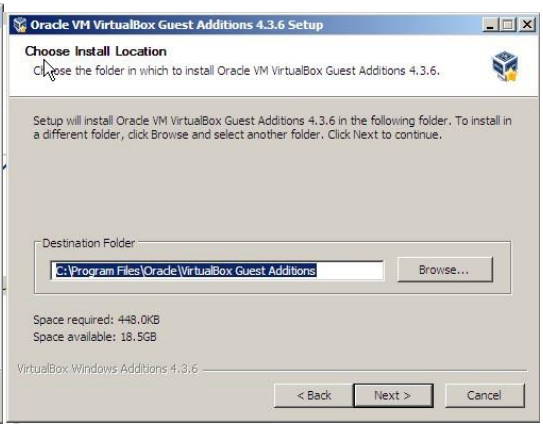
5. Запуск виртуальной машины и установка ОС.

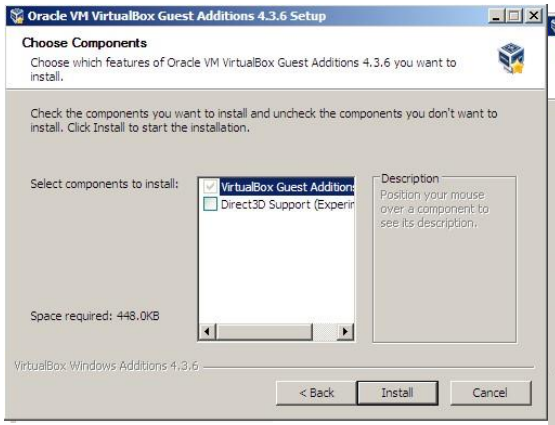
Запускаем созданную виртуальную машину и устанавливаем ОС MS Windows Server 2008. **6. Установка гостевых дополнений.**

VirtualBox Guest Additions — комплект программного обеспечения, устанавливаемый в гостевую операционную систему и расширяющий её возможности по взаимодействию с системой виртуализации и хост-системой.


win2008 [Running] - Oracle VM VirtualBox
Machine
View
Devices
Help
Recycle Bin
CD/DVD Devices
USB Devices
Shared Clipboard
Drag'n'Drop
Network Settings...
Shared Folders Settings...
Remote Display
Video Capture
Insert Guest Additions CD image... Host+D
Start
3:22 PM
Insert the Guest Additions disk file into the virtual drive
Right Ctrl



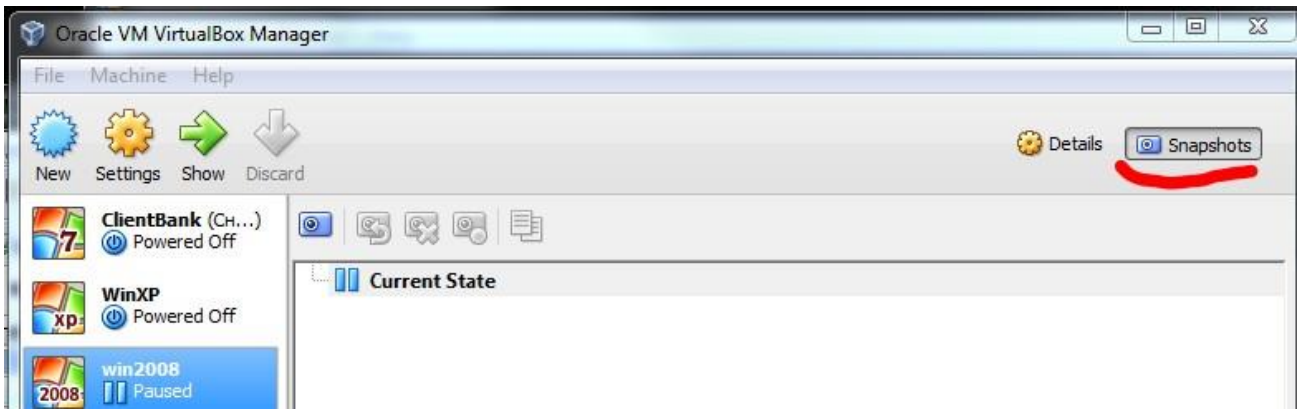




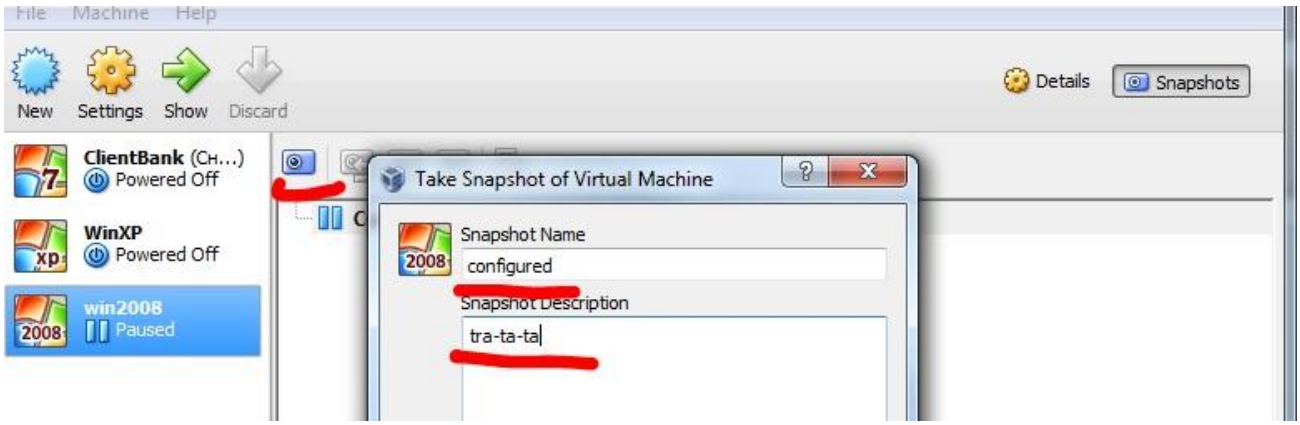
Перезапускаем систему!

7. Создание снимков

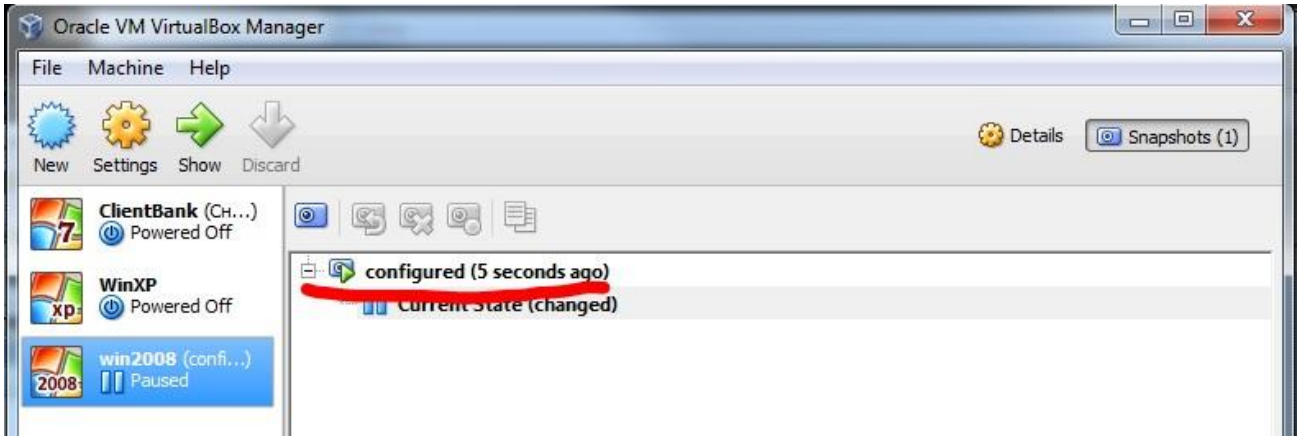
Машина сконфигурирована. Сделаем снимок ее состояния (снэпшот). Для этого переходим в консоль управления машинами. Вкладка снэпшоты.



Делаем снимок.



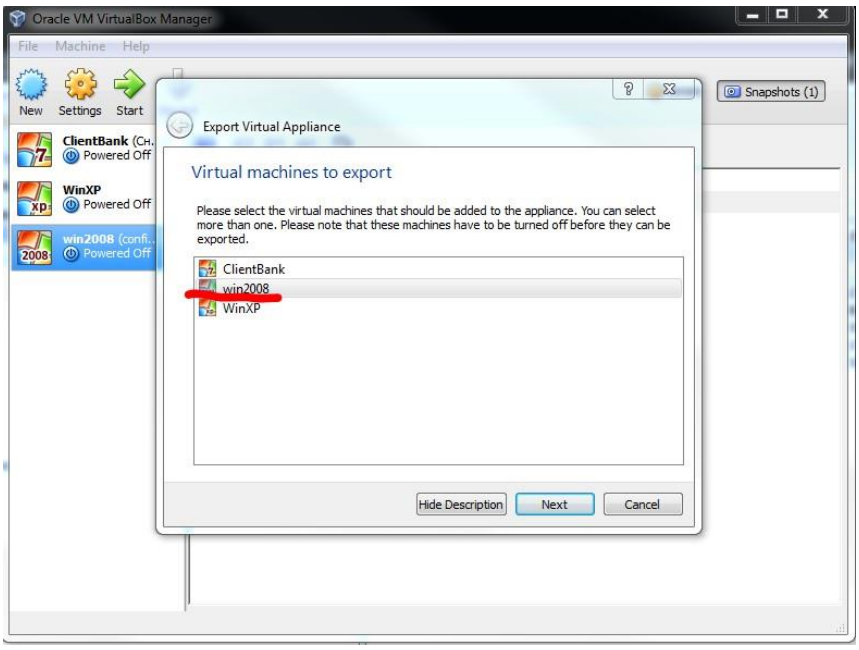
Снимок готов.



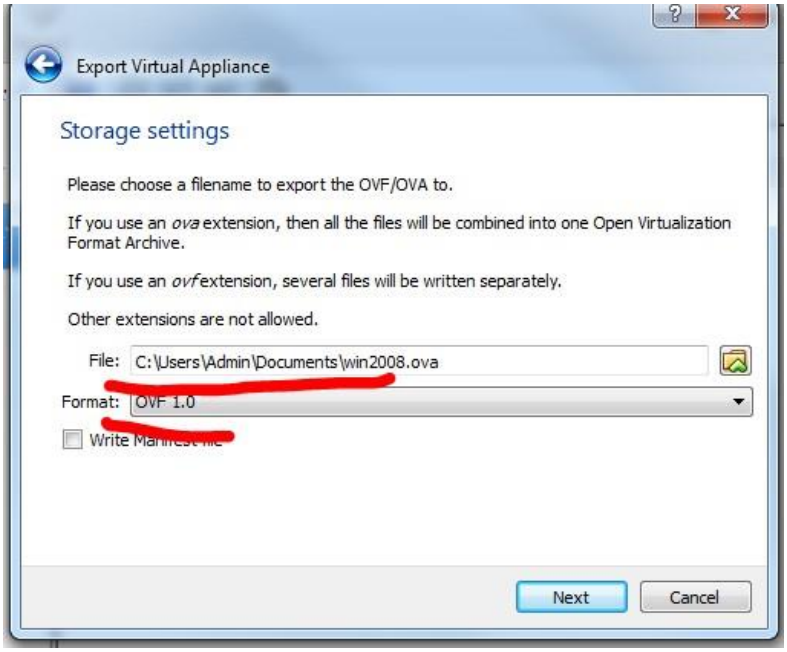
Совет: снимки желательно делать перед каждым серьезным изменением машины.

8. Бэкапирование виртуальной машины (экспорт)

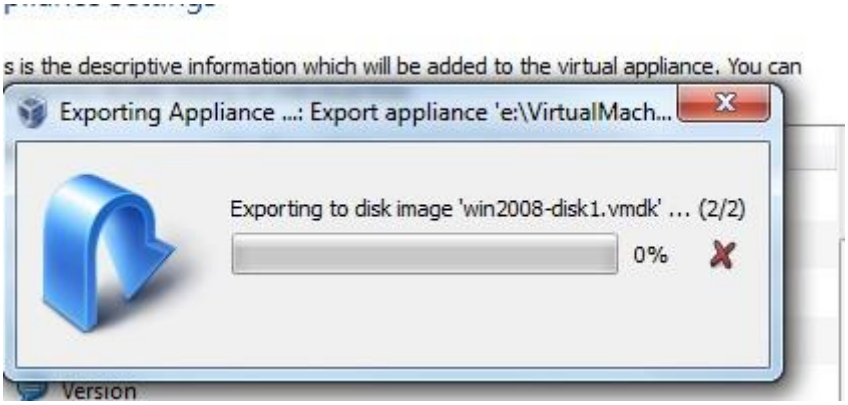
Останавливаем виртуальную машину. В меню консоли управления виртуальными машинами выбираем: File->ExportAppliance (Файл->Экспорт конфигураций). В появившемся окне выбираем нужную машину.



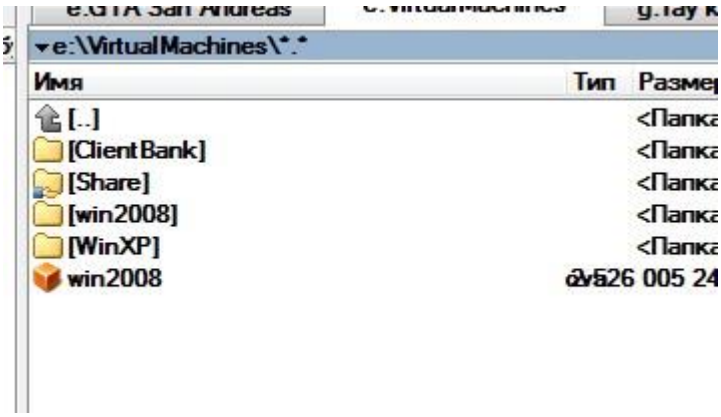
Выбираем файл, в который будет экспортироваться машина, и формат файла:



Заполняем свойства и нажимаем Export.

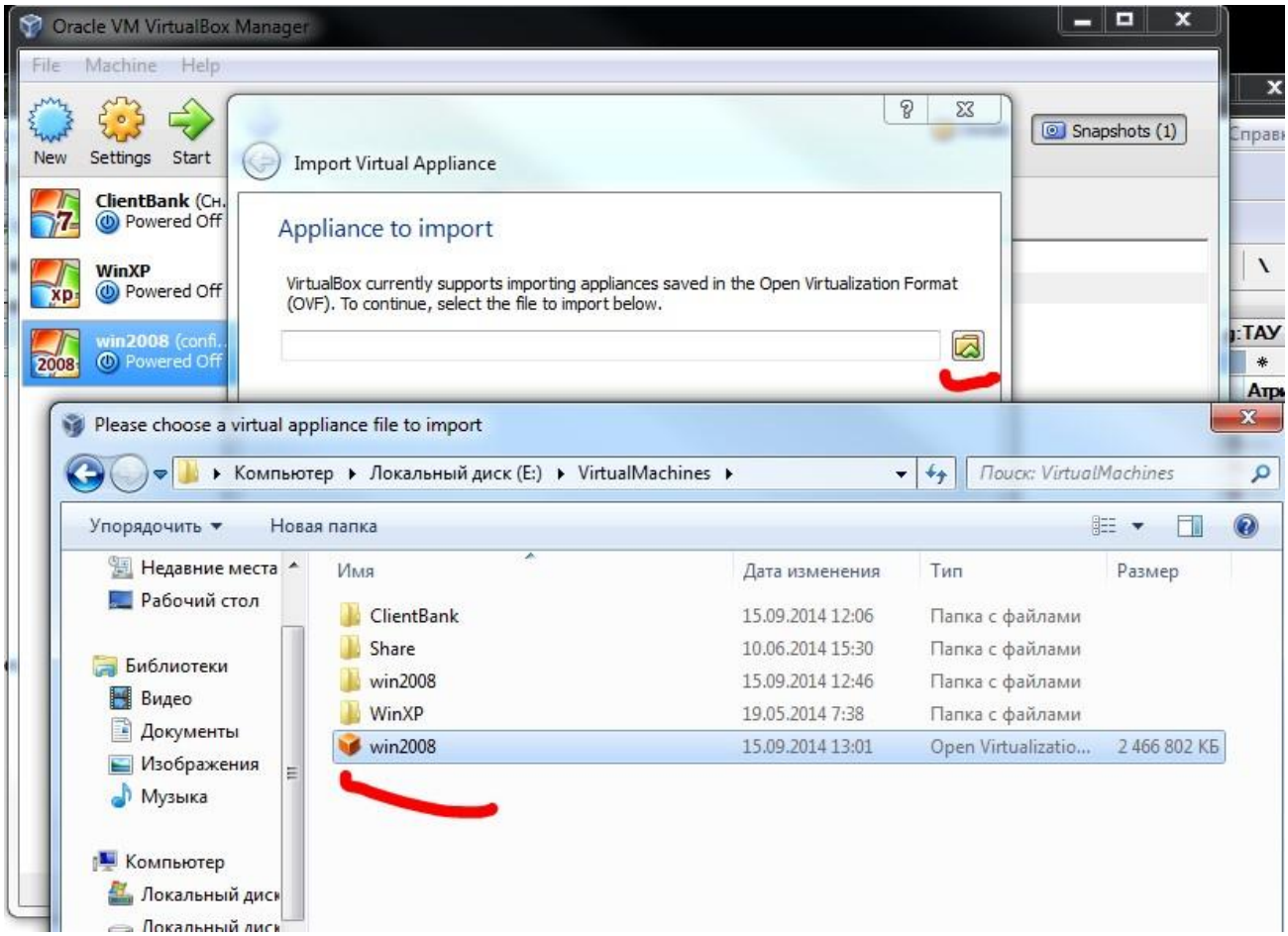


Ждем окончания. Экспорт виртуальной машины выполнен.



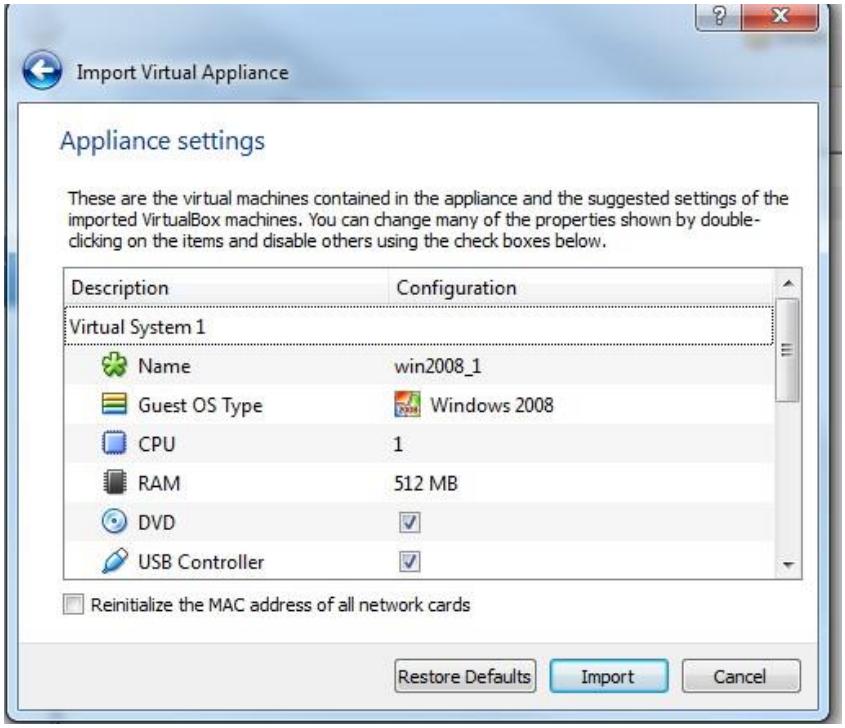
9. Развёртывание бэкапа (Импорт)

В меню консоли управления витруальными машинами выбираем: File>ImportAppliance (Файл->Импорт конфигураций). В появившемся окне выбираем бэкап.



Изменяем имя машины, проверяем свойства и нажимаем import. Устанавливаем флаг Reinitializethemacaddressofallnetworkcards в случае, если мы таким образом клонируем машины.

Совет: в случае клонирования существующей виртуальной машины для нее необходимо создать свой собственный уникальный MAC-адрес, который будет отличаться от адреса оригинальной машины.



Ждем окончания процесса, новая машина готова.

Дополнительная литература

http://www.virtualbox.org/manual/ch01.html – мануал Oracle VirtualBox

http://www.slideshare.net/puahhlss/virtualbox-step-by-step-guide – установка и настройка Oracle VirtualBox http://bagrat-oraclevmvirtualbox.blogspot.com/ – создание машины и добавление общей папки http://www.arvydas.co.uk/2011/02/forwarding-usb-devices-on-oracle-virtualbox/ – настройка перенаправления USB-устройств http://rus-linux.net/MyLDP/vm/VirtualBox-networking.html - немного о настройке сетей

3.3 Лабораторная работа № 3 Настройка сетевых подключений

Цель работы

Изучение принципов настройки сетевых подключений и конфигурирования IPадресов (на примере ОС Windows). Создание простейшей локальной сети.

Методические указания 1.

Настройка сетевых свойств клиента

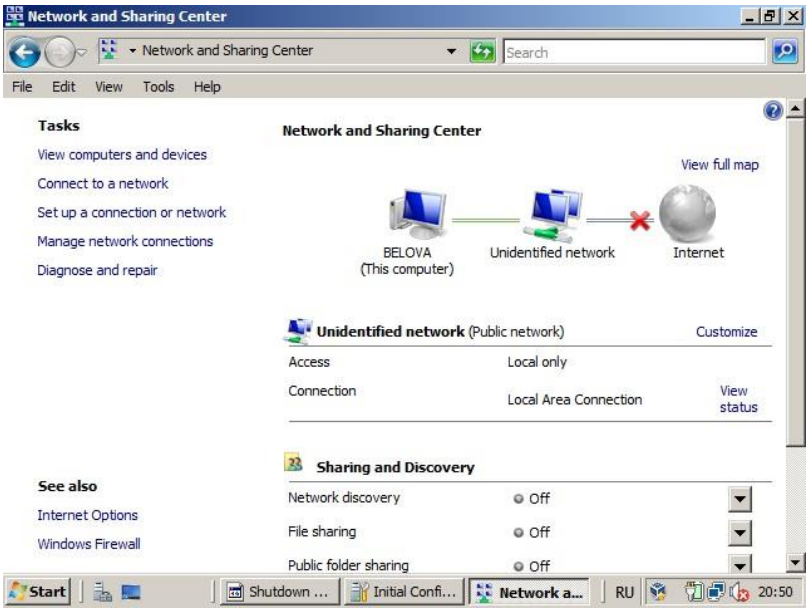
Операционная система Windows Server 2008 включает две основные области конфигурирования сетевых свойств клиента:

1) Центр управления сетями и общим доступом (Network and Sharing Center);
2) Сетевые подключения (Network Connections).

1.1 Центр управления сетями и общим доступом

Центр управления сетями и общим доступом — основное средство конфигурирования сетей с ОС Windows. Открыть его можно разными способами. Например, в меню Пуск (Start) щелкнуть правой кнопкой мыши элемент Сеть (Network) и выбрать Свойства (Properties).

Центр управления сетями и общим доступом можно также открыть из панели управления, выбрав элемент Сеть и Интернет (Network And Internet) и соответствующую ссылку.



Центр управления сетями и общим доступом можно использовать для выполнения таких операций, как выбор сетевого размещения, просмотр сетевой карты, настройка сетевого обнаружения (Network Discovery), настройка общего доступа к файлам и принтерам, а также просмотр состояния сетевых подключений. Эти многочисленные свойства описаны ниже.

■ **Сетевое размещение (Network Location).**Этот параметр задается для компьютеров Windows Vista и Windows Server 2008. Всем клиентам с этими операционными системами назначается одно из следующих сетевых размещений: **Общественное** (Public), **Частное** (Private) и **Домен** (Domain). Затем в соответствии с выбранным сетевым размещением автоматически включаются или отключаются различные сетевые свойства.

ВНИМАНИЕ! По умолчанию для всех клиентов назначается общественное сетевое размещение.

На компьютере, размещенном в общественной сети, включен брандмауэр Windows (Windows Firewall) и отключены сетевое обнаружение (Network Discovery), общий доступ к файлам и принтерам, а также сетевая карта (Network Map).

Если выбрать для компьютера частное сетевое размещение (Private), будут включены сетевое обнаружение и сетевая карта. По умолчанию общий доступ к файлам отключен, но, в отличие от общественных сетей, общий доступ к файлам на отдельном компьютере в частной сети можно включить, не изменяя параметры по умолчанию для всех компьютеров, которым назначено частное сетевое размещение.

Если компьютер присоединен к домену службы каталогов Active Directory, существующей сети будет автоматически назначен тип сетевого размещения Домен (Domain). Доменный тип сетевого размещения аналогичен частному, за исключением того, что в домене конфигурация брандмауэра Windows, сетевого обнаружения и сетевой карты определяется групповой политикой.

■ **Сетевая карта (Network Map)**. Позволяет видеть устройства в локальной сети, а также схему их подключения друг к другу и Интернету.

Сетевая карта основана на функциональности двух компонентов.

- Компонент Link Layer Topology Discovery (LLTD) Mapper запрашивает в сети устройства для включения их в карту.
- Компонент Отвечающее устройство LLTD (LLTD Responder) отвечает на запросы компонента LLTD Mapper.

При выборе профиля домена (Domain) сетевая карта по умолчанию отключается, но ее можно включить средствами групповой политики.

■ **Общий доступ к файлам (File Sharing)**. При включении этого компонента брандмауэр Windows позволяет обычным пользователям назначать общий доступ к файлам и папкам в своих профилях, то есть в папках каталога %systemroot%\Users\%username%.

Администраторы могут назначать общий доступ к любому файлу и папке на компьютере.

■ **Общий доступ к общим папкам (Public Folder Sharing)**. При включении этого компонента будет автоматически назначен общий доступ к папке %systemroot%\Users\Public и разрешен общий доступ к файлам.

■ **Использование общих принтеров (Printer Sharing)**. При включении этого компонента разрешается общий доступ к принтерам, установленным на локальном компьютере, с тем чтобы чтобы эти принтеры могли использовать другие компьютеры в сети. При выборе данного компонента автоматически включается общий доступ к файлам.

■ **Общий доступ с парольной защитой (Password Protected Sharing)**. Этот компонент доступен лишь на компьютерах, не присоединенных к домену. При его включении доступ к общим ресурсам могут получать только пользователи с действительными учетными записями на локальном компьютере.

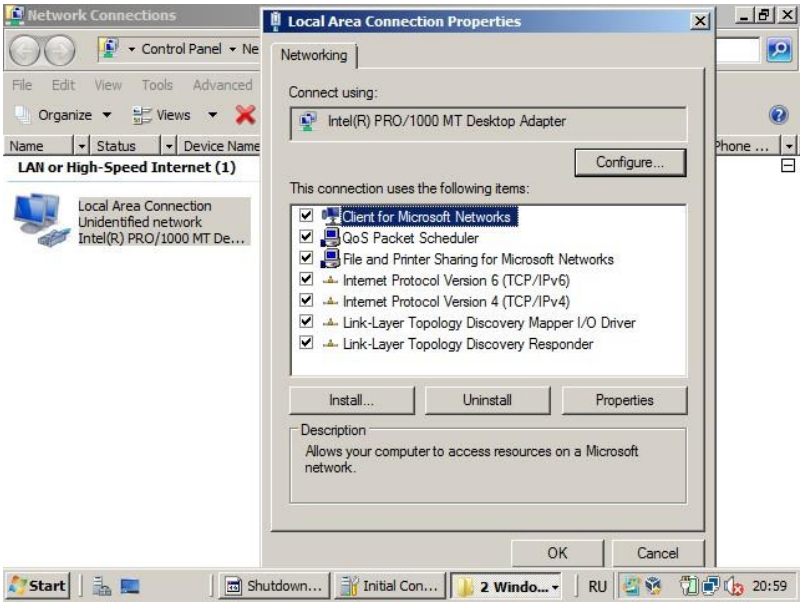
1.2 Сетевые подключения

Операционная система Windows автоматически обнаруживает и конфигурирует подключения сетевых адаптеров, установленных на локальном компьютере. Затем эти подключения отображаются в окне Сетевые подключения (Network Connections) вместе с другими подключениями, такими как коммутируемые соединения, которые добавляются вручную с помощью ссылки Установка подключения или сети (Set Up A Connection Or Network) в Центре управления сетями и общим доступом.

Окно Сетевые подключения (Network Connections) можно открыть разными способами. Например, выбрать в диспетчере сервера сам узел Диспетчер сервера (Server Manager) и щелкнуть ссылку Отобразить сетевые подключения (View Network Connections). Также можно щелкнуть опцию Настроить сеть (Configure Networking) в окне Задачи начальной настройки (Initial Configuration Tasks). В Центре управления сетями и общим доступом (Network AndSharing Center) можно щелкнуть ссылку Управление сетевыми подключениями (Manage Network Connections). И, наконец, можно ввести в командную строку, в поле Начать поиск (Start Search) или окно Выполнить (Run) команду ncpa.cpl или controlnetconnections.

Просмотр компонентов по умолчанию для сетевых подключений

Сами по себе подключения не позволяют сетевым хостам осуществлять коммуникации. Возможность осуществления коммуникаций обеспечивают сетевые клиенты, службы и протоколы, которые привязаны к подключениям. На вкладке Общие (General) окна свойств подключения показаны клиенты, службы и протоколы, привязанные к этому подключению.



На рисунке показаны компоненты, установленные по умолчанию для подключения по локальной сети. Установленные возле компонентов флажки указывают, что эти компоненты привязаны к подключению.

Просмотр дополнительных параметров подключений

Чтобы просмотреть дополнительные параметры подключений, необходимо открыть окно Сетевые подключения (Network Connections) и в меню Дополнительно (Advanced) выбрать опцию Дополнительные параметры (Advanced Settings).

В диалоговом окне Дополнительные параметры (Advanced Settings) отображается порядок (приоритет) каждого подключения. Изменяя порядок подключений на компьютере, можно определять приоритеты различных доступных подключений. Для каждого подключения можно также изменить порядок привязки используемых служб.

2. КонфигурированиеIP-подключения

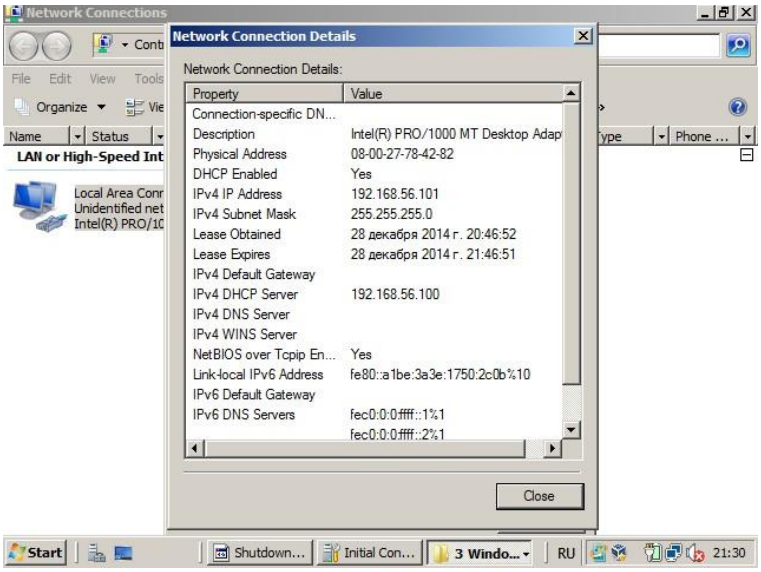
IP-конфигурация подключения состоит, как минимум, из IPv4-адреса и маски подсети, или IPv6-адреса и префикса подсети. Помимо этих минимальных данных IPконфигурация может включать такие сведения, как основной шлюз, адреса DNS-сервера, DNS-суффиксы и адреса WINS-сервера.

Если для узла не задан основной шлюз, этот узел не сможет подключаться к Интернету и другим компьютерам за пределами широковещательного домена. В некоторых ситуациях настройка основного шлюза не выполняется из соображений безопасности.

Просмотреть IP-адреса подключения можно с помощью команды ipconfig или в окне Сведения о сетевом подключении (Network Connection Details).

Чтобы открыть диалоговое окно Сведения о сетевом подключении (Network Connection Details), в окне Сетевые подключения (Network Connections) щелкните подключение правой кнопкой мыши и в контекстном меню выберите команду Состояние (Status).Затем в окне Состояние — подключение по локальной сети (Local Area Connection Status) щелкните кнопку Сведения (Details).

Откроется диалоговое окно Сведения о сетевом подключении (Network Connection Details), показанное на нижеследующем рисунке.



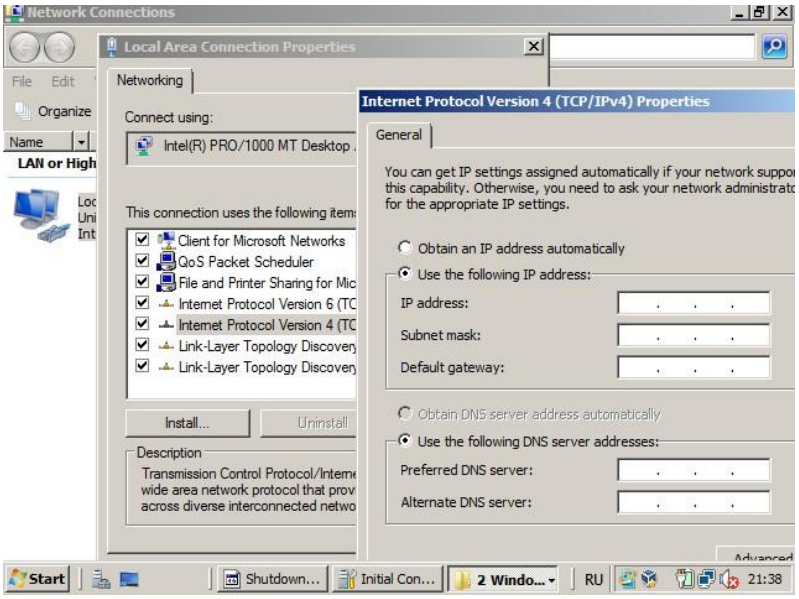
Конфигурацию IP можно настраивать вручную или автоматически. По умолчанию сетевые подключения автоматически получают IP-адрес и адрес DNS-сервера с помощью ***протокола динамического конфигурирования хоста DHCP*** (Dynamic Host Configuration Protocol).

2.1 Настройка IP-конфигурации вручную

Отконфигурированный вручную адрес называется ***статическим***, поскольку он остается неизменным даже после перезагрузки компьютера. Статические адреса удобно использовать для контроллеров доменов, DNS-серверов, DHCP-серверов, WINS-серверов и маршрутизаторов.

Настройка вручную конфигурации IPv4

Статический адрес и другие параметры конфигурации IPv4 можно задать для сетевого подключения вручную в диалоговом окне Свойства. Чтобы открыть это диалоговое окно, откройте свойства сетевого подключения, для которого хотите настроить конфигурацию IPv4 (через контекстное меню). В этом окне дважды щелкните компонент Протокол Интернета версии 4 (Internet. Protocol Version 4 (IPv4)).



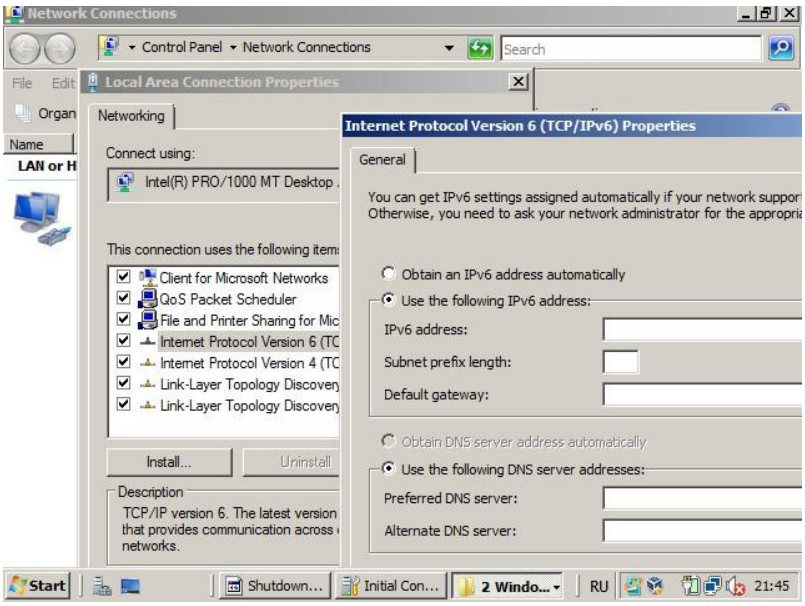
Далее нужно выбрать опцию Использовать следующий IP-адрес (Use The Following IP Address), а затем указать IP-адрес, маску подсети и (при желании) основной шлюз. Чтобы назначить подключению статическую конфигурацию DNS-сервера, надо выбрать опцию Использовать следующие адреса DNS-серверов (Use The Following DNS Server Addresses), а затем указать предпочитаемый и альтернативный адреса DNS-серверов.

Настройка конфигурации IPv6 вручную

В большинстве случаев конфигурацию IPv6 не нужно настраивать вручную, поскольку обычно статические IPv6-адреса назначаются только маршрутизаторам, а не узлам (хостам). Как правило, конфигурация IPv6 назначается для хоста автоматически.

Тем не менее, IPv6-адрес можно назначить вручную — в диалоговом окне Свойства: TCP/IPv6, показанном на следующем рисунке.

Чтобы открыть это окно, в свойствах сетевого подключения надо дважды щелкнуть компонент TCP/IPv6.



Далее выбрать опцию Использовать следующий IPv6-адрес (Use The Following IPv6 Address), а затем указатьIPv6-адрес, длину префикса подсети (как правило, 64) и (при желании) основной шлюз.

Настройка параметров IPv4 и IPv6 вручную с помощью командной строки

Для настройки IP-конфигурации сетевого подключения можно использовать утилиту командной строки Netsh. Утилита Netsh имеет много различных опций настройки IPv4 и IPv6. Более подробные сведения о синтаксисе и опциях Netsh можно найти в справке этой утилиты (netsh help).

Чтобы для подключения по локальной сети назначить IPv4-адрес 192.168.33.5 и маску подсети 255.255.255.0, можно ввести следующую команду:

netsh interface ip set address "local area connection" static 192.168.33.5 255.255.255.0

2.2 Автоматическое получение адреса для IPv4-подключения

При автоматическом назначении IP-адресов все сетевые подключения получают IPадрес от DHCP-сервера, если он доступен. Если же DHCP-сервер недоступен, подключение само автоматически назначает себе определенную альтернативную конфигурацию. Если альтернативная конфигурация не определена, подключение автоматически назначит себе в качестве IPv4-адреса ***частный адрес APIPA*** (Automatic Private IP Addressing).

Чтобы настроить автоматическое назначение IPv4-адреса для подключения, выберите соответствующую опцию в диалоговом окне Свойства: Протокол Интернета версии 4 (TCP/IPv4).

Для автоматического назначения IPv4-адреса можно также использовать утилиту Netsh.

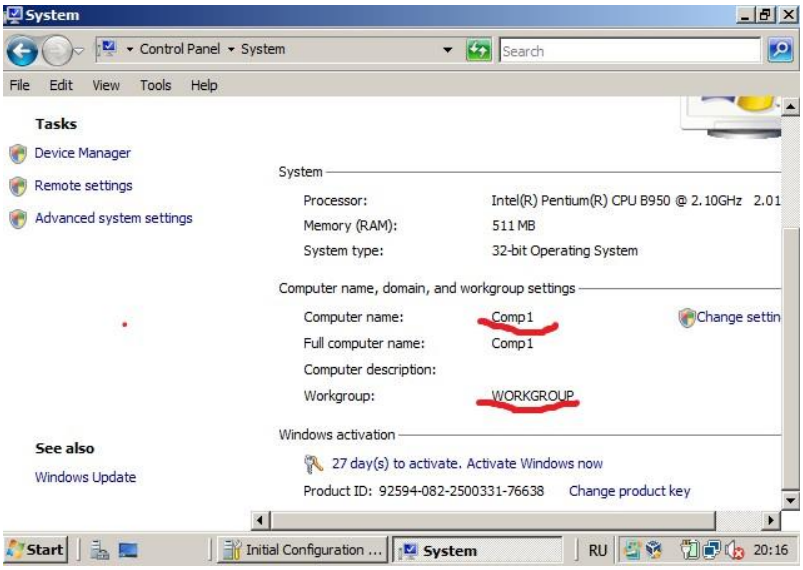
Порядок выполнения работы

Предполагается, что вы уже установили виртуальные компьютеры в соответствии с инструкциями. У вас есть две виртуальные машины Serv1 и Comp1. На машинах не должно быть установлено никаких ролей сервера.

В предложенных далее упражнениях необходимо настроить статический IPv4-адрес для подключения по локальной сети на машине Servl, альтернативный адрес для подключения по локальной сети па машине Comp1 и статический адрес на машине Comp1 с помощью командной строки. После настройки этих адресов нужно включить общий доступ к файлам на обоих компьютерах и протестировать подключения с помощью диагностических утилит, изученных в лабораторной работе № 2.

Упражнение 1. Выбор рабочей группы

Устанавливаем рабочую группу на виртуальных машинах Serv1 и Comp1 как на хосте. По умолчанию WORKGROUP.



Упражнение 2. Проверка текущей конфигурации IP

Просматриваем текущую конфигурацию IP на виртуальной машине Serv1.

1. Войдите на компьютер Serv1 как администратор.
2. Откройте окно командной строки.
3. В командную строку введите команду ipconfig/allи нажмите Enter.
4. Определите:

Имя компьютера

Количество сетевых интерфейсов

Для каждого сетевого интерфейса: MAC-адрес, IPv4-адрес, маску, адрес DHCPсервера, адрес шлюза по умолчанию, IPv6-адрес.

Выполняем те же действия на виртуальной машине Comp1.

Упражнение 3. Настройка IPv4-адреса вручную

В этом упражнении необходимо назначить статический IPv4-адрес подключению по локальной сети на машине Serv1. Статический IP-адрес нужен компьютерам, которые позже будут управлять сетевой инфраструктурой DHCP или DNS.

1. Войдите на машину Serv1 как администратор и введите в командную строку команду ncpa.cpl (с помощью этой команды можно открыть доступ к меню настроек сетевых подключений).

2. В окне Сетевые подключения (Network Connections) щелкните правой кнопкой мыши Подключение но локальной сети (Local Area Connection) и примените команду Свойства (Properties).

3. В списке Отмеченные компоненты используются этим подключением (This Connection Uses The Following Items) диалогового окна Подключение по локальной сети — свойства (Local Area Connections Properties) дважды щелкните компонент Протокол Интернета версии 4 (TCP/IPv4) (Internet Protocol Version 4 (TCP/IPv4)).

4. На вкладке Общие (General) диалогового окна Свойства: Протокол Интернета версии 4 (TCP/IPv4) выберите опцию Использовать следующий IP-адрес (Use The Following IP Address).

5. В текстовое поле IP-адрес (IP Address) введите адрес 192,168.0.1.

6. Щелкните текстовое поле Маска подсети (Subnet Mask), чтобы поместить в него курсор. Укажите маску 255.255.255.0. Щелкните ОК.

7. В диалоговом окне Подключение по локальной сети - свойства (Local Area Connections Properties) щелкните ОК.

8. В командную строку введите команду ipconfig. Будет отображен новый статический IPv4-адрес для подключения по локальной сети.

Упражнение 4. Определение альтернативной конфигурации

В этом упражнении необходимо так изменить конфигурацию IP на машине Comp1, чтобы в случае отсутствия DHCP-сервера в частной лабораторной сети подключению компьютера по локальной сети назначался адрес 192.168.0.200.

1. Войдите на машину Comp1 как администратор.

2. В Диспетчере сервера (Server Manager) щелкните ссылку Отобразить сетевые подключения (View Network Connections).

3. В диалоговом окне Сетевые подключения (Network Connections) откройте свойства подключения по локальной сети.

4. В диалоговом окне Подключение по локальной сети — свойства (Local Area Connections Properties) откройте свойства компонента Протокол Интернета версии 4 (TCP/IPv4). Обратите внимание: на вкладке Общие (General) диалогового окна Свойства: Протокол Интернета версии 4 (TCP/IPv4) (Internet Protocol Version 4 (TCP/IPv4) Properties) выбраны опции Получить IP-адрес автоматически (Obtainan IP Address Automatically) и Получить адрес DNS-сервера автоматически (Obtain DNS Server Address Automatically).

5. Перейдите на вкладку Альтернативная конфигурация (Alternate Configuration). На ней выбрана опция Автоматический частный IP-адрес (Automatic Private IP Address). Поскольку DHCP-сервер недоступен и этот параметр включен по умолчанию, компьютеру Comp1 автоматически назначается адрес APIPA

6. Выберите опцию Настраиваемый пользователем (User Configured).

7. В текстовое поле IP-адрес (IP Address) введите адрес 192.168.0.200.

8. Щелкните текстовое поле Маска подсети (Subnet Mask), чтобы поместить в него курсор. В текстовом поле будет указана маска подсети 255.255.255.0. Оставьте эту маску по умолчанию. Альтернативная конфигурация IP-адреса 192.168.0.200/24 для машины Comp1 будет использоваться до настройки DHCP-сервера в сети. Щелкните ОК.

9. В диалоговом окне Подключение по локальной сети — свойства (Local Area Connections Properties) щелкните ОК.

10. Откройте командную строку и введите команду ipconfig/all. Будет отображен новый альтернативный адрес, назначенный для машины Comp1. Кроме того, параметру Автонастройка включена (Autoconfiguration Enabled) присвоено значение Да (Yes).

Упражнение 5. Настройка статического IPv4-адреса в окне командной строки

В этом упражнении с помощью командной строки необходимо настроить для машины Comp1 статический IPv4-адрес 192.168.0.2 и маску подсети 255.255.255.0.

1. Войдите на машину Comp1 как администратор и откройте командную строку с повышенными привилегиями (это не обязательно делать при использовании учетной записи Администратор (Administrator)). Чтобы открыть командную строку с повышенными привилегиями, в меню Пуск (Start) щелкните правой кнопкой элемент Командная строка (Command Prompt) и примените команду Запуск от имени администратора (Run As Administrator).

2. В командную строку введите следующую команду: netsh interface ip set address "local area connection" static 192.168.0.2

3. В командную строку введите команду ipconfig. В результатах будет отображен новый IPv4-адрес.

Упражнение 6. Включение общего доступа к файлам

В Windows Server 2008 следует включить общий доступ к файлам, чтобы компьютер отвечал на запросы ping. Включим общий доступ к файлам в Центре управления сетями и общим доступом на обеих машинах — Serv1 и Comp1.

1. Войдите на машину Serv1 как администратор и откройте Центр управления сетями и общим доступом (Network And Sharing Center). Для этого в области уведомлений щелкните значок сети правой кнопкой мыши и выберите соответствующую команду.

2. В разделе Общий доступ и сетевое обнаружение (Sharing And Discovery) Центра управления сетями и общим доступом щелкните кнопку Выкл (Off) напротив компонента Общий доступ к файлам (File Sharing).

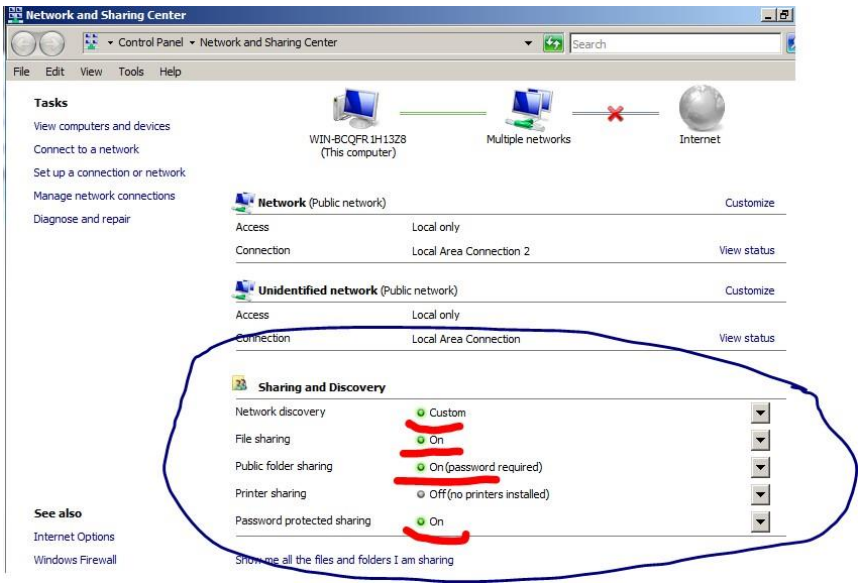
3. Включите общий доступ к файлам и щелкните кнопку Применить (Apply). В открывшемся диалоговом окне будет предложено включить общий доступ для всех публичных сетей.

4. Щелкните кнопку Да, включить общий доступ к файлам для всех публичных сетей (Yes, Turn On File Sharing For All Public Network). Эту опцию рекомендуется использовать только для тестовых сетей.

5. Повторите шаги с 1 по 4 на компьютере Comp1.

Упражнение 7. Определение машины в сети

Самостоятельно включите определение машины в сети и другие параметры, как на скриншоте ниже. Выполните необходимые действия для машин Serv1 и Comp1.



Упражнение 8. Проверка подключения

Проверьте возможность осуществления коммуникаций между двумя компьютерами в частной лабораторной сети.

С помощью диагностических утилит hostname, ipconfig определите имена, IP-адреса и локальные адреса виртуальных машин. Проверьте связь между компьютерами с помощью утилиты ping, используя имена компьютеров и IP-адреса.

1. Войдите на машину Comp1 как администратор и откройте командную строку.
2. Введите команды hostname иipconfig. Определите имя, IP-адрес, локальный адрес и другие параметры стека TCP/IP виртуальной машины.
3. В командную строку введите команду ping Serv1, а затем ping192.168.0.1. В результатах выполнения команды будет подтверждена возможность обмена данными между машинами Comp1 и Serv1через сеть IP.
4. Повторите действия для машины Serv1.

Упражнение 9. Назначение уникального локального адреса IPv6

Назначить подключению по локальной сети на компьютерах Serv1 и Comp1 уникальные локальные адреса IPv6.

1. Откройте свойства подключения по локальной сети на компьютере Serv1 и дважды щелкните компонент Протокол Интернета версии 6 (TCP/IPv6).
2. В диалоговом окне Свойства: Протокол Интернета версии 6 (TCP/IPv6) выберите опцию Использовать следующий IPv6-адрес (Use The Following IPv6 Address) и укажите такие параметры:

■ IPv6-адрес: fd00::1

■ Длина префикса подсети (Subnet Prefix Length): 64

■ Основной шлюз (Default Gateway): оставьте поле пустым.

■ Предпочитаемый DNS-сервер (Preferred DNS Server): оставьте поле пустым.

■ Альтернативный DNS-сервер (Alternate DNS Server): оставьте поле пустым.

Щелкните ОК.

3. В диалоговом окне Подключение по локальной сети -- свойства (Local Area Connection Properties) щелкните ОК.
4. Выполните шаги с 1 по 3 на машине Comp1 и укажите IPv6-адрес fd00:2.

5. На компьютере Comp1 откройте командную строку и введите команду pingfd00::1. Будет выполнен обмен четырьмя пакетами с адресом fd00::l.

Упражнение 10. Настройка сети между виртуальными машинами и хостом

Самостоятельно выполните настройку локальной сети между виртуальными машинами и хостом. Проверьте с помощью диагностических утилит (hostname, ipconfig, arp, ping) имена, IP-адреса и локальные адреса на виртуальных машинах и хосте. Проверьте связь между компьютерами в созданной локальной сети.

Поместите полученную информацию в отчет по лабораторной работе.

Контрольные вопросы

1. Какие виды сетевого размещения могут быть заданы для компьютеров с ОС MS Windows 2008? Какой вид сетевого размещения назначается по умолчанию?
2. Что надо сделать, чтобы локальный компьютер реагировал на запросы ping?
3. Какие виды сетевого ПО вы знаете?
4. Какие службы и сетевые протоколы привязаны по умолчанию ко всем сетевым подключениям? Как их просмотреть? Какой компонент сетевого ПО позволяет подключаться к общим ресурсам, расположенным на других компьютерах сети?
5. Какие параметры задаются при конфигурировании IP-подключения?
6. Какой IP-адрес называется статическим? Как вручную задать конфигурацию IPv4 и IPv6? Укажите два способа?
7. Для чего предназначен протокол DHCP? Как по умолчанию получают IP-адрес и адрес DNS-сервера сетевые подключения?
8. В каком случае сетевое подключение автоматически назначит себе частный IPv4адрес APIPA? Могут ли такие компьютеры получить доступ в Internet?
9. Какие основные признаки лежат в основе классификации компьютерных сетей?
10. Назовите разновидности логических архитектур компьютерных сетей.

3.4 Лабораторная работа № 4 Методы адресации и протоколы разрешения адресов

Цель работы

Изучить методы адресации узлов сети и протоколы разрешения адресов. Разработать программу, позволяющую получить все виды адресов узла.

Постановка задачи

4. Изучить основные теоретические вопросы, используя материалы лекций, рекомендуемую литературу и методические указания к лабораторной работе:

- методы адресации узлов сети;
- физические адреса;
- числовые адреса; - доменные имена;
- протоколы разрешения адресов.

5. Выполнить задания по лабораторной работе. При разработке программ разрешается использовать любой язык программирования и среду разработки.
6. Ответить на контрольные вопросы.
7. Подготовить отчет по лабораторной работе.
8. Выполнить контрольный тест.

Краткие теоретические сведения

1. Методы адресации узлов сети

Каждый узел сети должен иметь адрес, чтобы была возможной передача информации и функционирование сети.

К адресу узла сети и схеме его назначения можно предъявить несколько требований:

 Адрес должен уникально идентифицировать компьютер в сети любого масштаба.

 Схема назначения адресов должна сводить к минимуму ручной труд администратора и вероятность дублирования адресов.

 Адрес должен иметь иерархическую структуру, удобную для построения больших сетей. В крупных сетях отсутствие иерархии адресов может привести к большим издержкам - конечным узлам и коммуникационному оборудованию придется оперировать с таблицами адресов, состоящими из тысяч записей.

 Адрес должен быть удобен для пользователей сети, т.е. должен иметь символьное представление.

 Адрес должен иметь по возможности компактное представление, чтобы не перегружать память коммуникационной аппаратуры - сетевых адаптеров, маршрутизаторов и т. п.

Перечисленные требования трудно совместить в рамках какой-либо одной схемы адресации, поэтому на практике обычно используется сразу несколько схем, так что компьютер одновременно имеет несколько адресов. Каждый адрес используется в той ситуации, когда соответствующий вид адресации наиболее удобен.

Множество всех адресов, которые являются допустимыми в рамках некоторой схемы адресации, называется ***адресным пространством***. Адресное пространство может иметь ***плоскую (линейную)*** или ***иерархическую*** организацию. При плоской организации множество адресов никак не структурировано. При иерархической организации адресное пространство организовано в виде вложенных друг в друга подгрупп.

Наибольшее распространение получили **три схемы адресации узлов** (три типа адресов):

- локальные адреса; - числовые составные адреса;
- символьные адреса или имена.

2. Протоколы разрешения адресов. Протокол ARP

Для преобразования адресов из одного вида в другой используются специальные протоколы, которые называют ***протоколами разрешения адресов*.**

Проблема установления соответствия между адресами различных типов может решаться централизованными или распределенными средствами. В случае централизованного подхода в сети выделяется один компьютер (сервер имен), в котором хранится таблица соответствия друг другу имен различных типов, например символьных имен и числовых номеров. Все остальные компьютеры обращаются к серверу имен, чтобы по символьному имени найти числовой номер компьютера, с которым необходимо обменяться данными.

При распределенном подходе, каждый компьютер сам решает задачу установления соответствия между именами. Недостатком распределенного подхода является необходимость рассылки широковещательных сообщений - такие сообщения перегружают сеть, так как они требуют обязательной обработки всеми узлами, а не только узлом назначения. Поэтому распределенный подход используется только в небольших локальных сетях. Для крупных сетей характерен централизованный подход.

Наиболее известной службой централизованного разрешения имен является служба **DNS (Domain Name System)** сети Internet.

Пример использования распределенного подхода – протокол разрешения адресов **ARP (Address Resolution Protocol),** используемый стеком TCP/IP для преобразования IPадреса в аппаратный адрес.

Необходимость обращения к протоколу ARP возникает каждый раз, когда модуль IP передает пакет на уровень сетевых интерфейсов, например драйверу Ethernet.

Работа протокола ARP начинается с просмотра ARP-таблицы. По таблице определяется нужный MAC-адрес. Для каждой сети, подключенной к сетевому адаптеру компьютера или порту маршрутизатора, строится отдельная ARP-таблица. Пример.

IP-адрес	MAC-адрес	Тип записи
194.85.135.75	00-80-48-EB-7E-60	динамический
194.85.135.70	08-00-5A-21-A7-22	динамический
194.85.60.21	00-80-48-EB-75-67	статический

Работа с таблицей осуществляется с помощью специальной утилиты arp. Таблица выводится на экран по команде arp –a.

Статические записи создаются вручную с помощью утилиты arp. Они находятся в кэше до перезагрузки компьютера.

Динамические записи создаются протоколом ARP, добавляются и удаляются автоматически.

Если запись в течение определенного времени не обновляется, то она исключается из таблицы. Т.о. ARP-таблица содержит записи только об узлах, активно участвующих в сетевых операциях. Поэтому ее еще называют **ARP-кэш**.

Если искомого адреса в таблице нет, то протокол ARP широковещательно рассылает **ARP-запрос**, указывая IP-адрес («Чей это IP-адрес и каков ваш адрес сетевого адаптера?»). Узел, IP-адрес которого совпал с указанным в запросе, отправляет **ARP-ответ** с указанием своего локального адреса на машину, сделавшую запрос. После этого новая запись добавляется в ARP-таблицу.

Если в сети нет узла с искомым IP-адресом, ARP-ответа не будет. Протокол IP уничтожает пакеты, направленные по этому адресу.

3. Одноадресные, групповые и многоадресные типы адресов

По количеству адресуемых сетевых интерфейсов адреса можно классифицировать следующим образом:

- ***Одноадресный тип*** или ***уникальный адрес (unicast)*** используется для идентификации отдельных интерфейсов (физический интерфейс между компьютером и сетью) конечного узла или маршрутизатора; позволяет пересылать сообщения в одну точку (на один конечный узел сети).
- ***Групповой адрес (multicast)*** идентифицирует сразу несколько интерфейсов, данные доставляются каждому из интерфейсов, входящих в группу; позволяет пересылать сообщения группе произвольно расположенных в Internet узлов.
- ***Широковещательный адрес (broadcast)*** используется для доставки данных всем узлам подсети;
- ***Адрес произвольной рассылки (anycast)*** задает группу интерфейсов, но данные должны быть доставлены не всем, а одному члену группы, как правило, «ближайшему» (новый тип адреса, определен в протоколе IPv6). Назначается только интерфейсам маршрутизатора.

4. Аппаратные адреса

Аппаратные адреса называют еще ***физическими*** или ***локальными***.

Локальный адрес – такой тип адреса, который используется средствами базовой технологии для доставки данных в пределах подсети, являющейся элементом составной сети. В разных подсетях допустимы различные сетевые технологии, следовательно, различные протоколы. Поэтому существуют разные типы локальных адресов. Для ЛВС локальный адрес – это MAC-адрес сетевого адаптера (Media Access Control). Если подсетью является глобальная сеть (например, протокол IP работает над IPX или X.25), то в этом случае локальными адресами будут адреса X.25 или IPX.

Физические адреса не имеют иерархической структуры, используются аппаратурой. Компьютер может иметь несколько сетевых интерфейсов и соответственно несколько физических адресов.

Записывается адрес сетевого адаптера в ПЗУ платы сетевого адаптера на заводе изготовителе. При замене сетевого адаптера изменяется и аппаратный адрес интерфейса.

Стандарты на аппаратные адреса были разработаны IEEE. Был выбран 48-битный формат адреса для всех технологий ЛВС.

Аппаратный адрес принято записывать в 16-ричном виде, разделяя байты с помощью “-“. Например: 11-AO-17-3D-BC-01 - MAC-адрес сетевого адаптера Ethernet.

Чтобы распределить возможные диапазоны адресов между многочисленными изготовителями сетевых адаптеров, была предложена следующая структура адреса:

I/G	U/L	Идентификатор производителя (22 бита)	Уникальный адрес производителя (24 бита)

Два старших разряда определяют тип адреса и способ интерпретации остальных 46 бит.

- I/G (Individual /Group) - определяет, индивидуальный это адрес или групповой:

  0 - индивидуальный;

  1 - групповой (такие пакеты получают все сетевые адаптеры с этим адресом).

- U/L (Universal /Local) – флаг определяет, как был присвоен адрес данному адаптеру:

  0 - производителем;

  1 - организацией, использующей данную сеть (редко).

- 22 разряда - Идентификатор производителя. IEEE присваивает один или несколько уникальных идентификаторов каждому производителю. Это позволяет исключить совпадения адресов адаптеров от разных производителей (~ 4 млн. вариантов).

- Младшие 24 разряда присваивает производитель сетевого адаптера (возможно 16 млн. комбинаций).

5. Числовые составные адреса

Во многих случаях для работы в больших сетях в качестве адресов узлов используют числовые составные адреса. Типичным представителями адресов этого типа являются IPv4 и IPv6-адреса.

IPv4 - это 32-битный адрес (4 байта).

Для удобства чтения в технической литературе и прикладных программах IPv4-адреса представляются в виде 4-х десятичных чисел, разделенных точками. Каждое из чисел соответствует одному октету (8 битам) и может иметь значения от 0 до 255. Этот формат называется **точечно-десятичным** (Decimal-Pant Notation). Например: 10010001.00001010.00100010.00000011

145.10.34.3

IPv6-адреса занимают 128-бит или 16 байт.

IPv6-адреса принято записывать в 16-ричном виде. Их делят на 8 блоков по четыре шестнадцатиричных цифры в каждом. Каждый блок отделяется двоеточием. Пример полного IPv6-адреса:

2001:00B8:3FA9:0000:0000:0000:0003:9C5A

IPv6-адрес можно сократить, исключив все незначащие нули в блоках и заменив все смежные нулевые блоки двойным двоеточием (::). Таким образом, предыдущий адрес можно сократить до такого:

2001:DB8:3FA9::D3:9C5A

6. Символьные адреса.

Символьные адреса **или** ***имена*** предназначены для запоминания людьми и поэтому обычно несут смысловую нагрузку. Символьные адреса легко использовать как в небольших, так и крупных сетях. Для работы в больших сетях символьное имя может иметь сложную иерархическую структуру.

Примеры символьных адресов – DNS и URI-адреса, имена NetBIOS.

Компьютерные имена преобразуются в сетевые адреса в процессе разрешения имен.

Например, сети Windows включают три системы разрешения имен: DNS, LLMNR (Link Local Multicast Name Resolution) и NetBIOS. Основной является DNS, поскольку этот метод разрешения имен используется для поддержки служб доменов Active Directory (Active Directory Domain Services) и разрешения всех имен Интернета. Инфраструктура DNS требует настройки сетевой конфигурации на серверах и клиентах.

Протокол NetBIOS обеспечивает разрешение имен в IPv4-сетях Windows без DNS.

Протокол LLMNR преобразует имя узла в IPv6-адрес.

DNS (Domain Name System) – Доменная система имен, используемая в Internet и корпоративных сетях.

DNS – служебный протокол прикладного уровня, имеющий архитектуру «клиентсервер» и предназначенный для разрешения доменного имени в IP-адрес.

Доменное имя представляет собой перечень имен доменов, разделенных точками. Начинается доменное имя с имени конечного узла или типа ресурса. Последним указывается домен верхнего уровня.

Например: www.google.com или 216-5.povt.fitr.bntu.by

Существует два основных типа доменов верхнего уровня.

■ Организационные (родовые) домены. Имя такого домена указывает основную функцию или род деятельности организаций в DNS-домене.

■ Географические домены. Эти домены именуются с использованием кодов страны и региона из двух символов согласно стандарту 3166 Международной организации по стандартизации ISO, например .uk (Великобритания) или .it (Италия).

Задания на лабораторную работу Задание 1.

Определение адресов локального узла с помощью утилиты ipconfig

Определить типы адресов локального узла и их значения, используя утилиту ipconfig с ключом /all.

Задание 2. Разработка программы для определения адресов локального узла

Разработать консольное приложение (аналог утилиты ipconfig), в котором определить и вывести на экран:

1) имя локального хоста, имя домена, полное доменное имя хоста;
2) все сетевые интерфейсы локального хоста (тип, описание, имя); 3) состояние интерфейса (подключен или нет в настоящее время); 4) для каждого интерфейса:
   - физический адрес и размер физического адреса;

- IPv4-адрес, маску, размер IPv4-адреса; размер сетевого префикса; - IPv6-адрес, размер IPv6-адреса, размер сетевого префикса.

Адреса выводить на экран в общепринятой форме записи.

Для получения адресов использовать классы пространства имен System.Net.NetworkInformation (IPGlobalProperties, NetworkInterface, IPInterfaceProperties (свойство UnicastAddresses), PhysicalAddress, DNS и др.).

Сравнить полученные значения адресов с адресами из Задания 1.

Задание 3. Изучение специальных IP-адресов

Используя специальные поля класса IPAddress, вывести на экран для адресов IPv4 и IPv6:

1) адрес петли обратной связи;
2) широковещательный IP-адрес;
3) адрес, обозначающий все сетевые интерфейсы данного узла.

Задание 4. Определение IP-адреса по доменному имени

Разработать консольное приложение для получения IP адреса по доменному имени (аналог утилиты nslookup).

Получить и вывести на экран для заданного пользователем произвольного DNSимени:

1) IPv4-адреса;
2) IPv6-адреса;
3) Имена-псевдонимы узла (Alias-имена).

Для получения адресов использовать методы класса Dns пространства имен System.Net.

Задание 5. Изучение протокола ARP

1. С помощью утилиты arp просмотреть ARP-таблицу локального узла. Сохранить полученную информацию в файле.
2. Организовать сетевую активность (ping, tracert).

Просмотреть таблицу преобразования адресов и сравнить ее с ранее полученными результатами. Пояснить причины изменений.

3. Выполнить ping локального DNS-сервера. Определить по таблице arp mac-адрес DNS-сервера.
4. Сделать перерыв в сетевой активности на несколько минут, после которого опять просмотреть arp-таблицу. Пояснить причины изменений (или отсутствия таковых) в таблице arp за время перерыва.

Контрольные вопросы

1. Какие требования предъявляются к адресу узла сети?
2. Что такое адресное пространство? Приведите пример плоского и иерархического адресного пространства.
3. Какие методы адресации используются в компьютерных сетях? Приведите примеры адресов каждого типа.

4. Как можно классифицировать адреса по количеству адресуемых сетевых интерфейсов?

5. Приведите пример протокола разрешения адресов, использующего централизованный подход, распределенный подход.

6. Что такое локальный адрес? Какая форма записи используется для MAC-адресов? Какой аппаратный адрес используется для широковещательной передачи?

7. Какая форма записи используется для IPv4, IPv6? Сколько места в памяти они занимают?

8. Приведите примеры символьных адресов.

9. Какие протоколы разрешения имен могут использоваться в сетях Windows?

10. Какова структура доменного имени? Какие типы доменов верхнего уровня вы знаете?

Содержание отчета

1. Титульный лист
2. Цель работы
3. Для заданий 1 – 5: скриншоты + листинг (для заданий с программой)
4. Выводы

3.5 Лабораторная работа № 5 Основы технологии сокетов

Цель работы

Изучить:

- понятие сокета, типы сокетов TCP/IP;
- принципы реализации архитектуры клиент-сервер на основе интерфейса сокетов;
- основные классы и методы .NET Framework для разработки сетевых приложений с использованием Windows Sockets.

Постановка задачи

1.Изучить методические указания к лабораторной работе, материалы лекций и рекомендуемую литературу.

2.Разработать консольное клиент-серверное приложение, демонстрирующее взаимодействие на основе потоковых сокетов.

3.Разработать консольное клиент-серверное приложение, демонстрирующее взаимодействие на основе дейтаграммных сокетов.

4.Ответить на контрольные вопросы.

Методические указания

Понятие сокета

Сокет (Socket - гнездо, разъем) - абстрактное программное понятие, используемое для обозначения в прикладной программе конечной точки сетевого соединения.

Технология (интерфейс) сокетов – название программного интерфейса для обеспечения обмена данными между процессами. Процессы при таком обмене могут выполняться как на одном компьютере, так и на разных, связанных между собой сетью.

Интерфейс сокетов расположен над транспортным уровнем стека TCP/IP, но в некоторых случаях может взаимодействовать напрямую с сетевым уровнем в обход транспортного.

Типы сокетов

Существуют три основных типа сокетов: потоковые, дейтаграммые и сырые.

Потоковые сокеты – это сокеты с установлением соединения, состоящие из потока байтов, который может быть двунаправленным. Т.е. через такую конечную точку приложение может и передавать, и получать данные. Потоковый сокет гарантирует обнаружение и исправление ошибок, обрабатывает доставку и сохраняет последовательность данных. Качество передачи достигается за счет использования протокола TCP.

Дейтаграммные сокеты – это сокеты без установления соединения, не обеспечивающие надежность при передаче. Применяются для приложений, когда неприемлемы затраты времени, связанные с установлением явного соединения. Для передачи данных используется протокол UDP.

Сырые сокеты (raw sockets - необрабатываемые, простые) – это сокеты, которые взаимодействуют с протоколами сетевого уровня в обход протоколов транспортного

уровня. Используются для непосредственного доступа приложения к IP-пакетам сетевого уровня.

Адресация сокетов. Номера портов

Мы рассмотрим адресацию сокетов только для стека протоколов TCP/IP, как самого распространенного на сегодняшний день.

Адрес сокета при использовании протоколов TCP/IP – это IP-адрес и номер порта прикладной службы.

IP-адрес уникален в Internet. Номер порта уникален на отдельном компьютере. Следовательно, адрес сокета будет уникален в Internet. Это позволяет удаленным процессам общаться исключительно на основе адреса сокета.

Порт задается, чтобы решить задачу одновременного сетевого взаимодействия с несколькими приложениями на одном узле. Если на компьютере выполняется несколько приложений, то получая пакет из сети, можно идентифицировать приложение по уникальному номеру порта, который задан при установлении связи.

Программисты должны быть внимательны при выборе номера порта, поскольку некоторые доступные порты зарезервированы для использования популярными службами, такими как FTP, HTTP и т.д. Эти порты обслуживаются и распределяются центром Internet Assigned Numbers Authority (IANA), их описание содержится в RFC 1700.

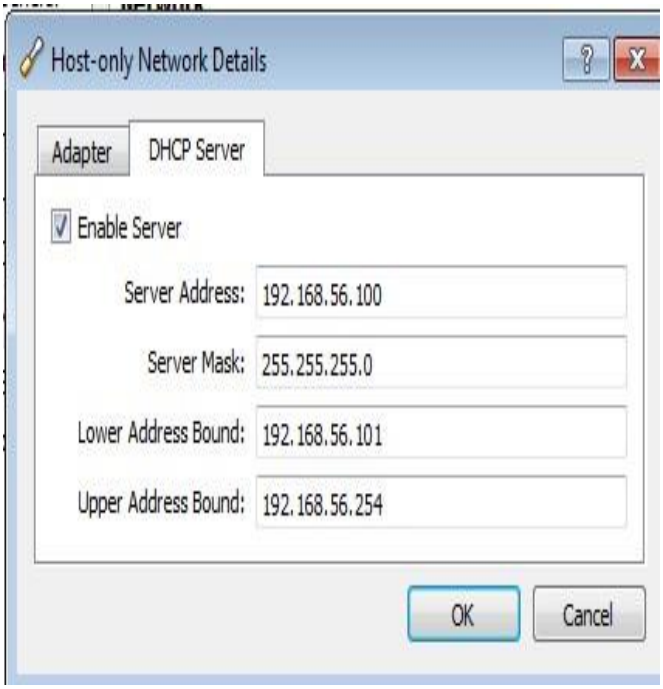
Номера портов разделяются на 3 категории (стандартные, зарегистрированные и динамические и/или частные):

- Номера от 0 до 1023 зарезервированы для стандартных служб.
- Порты с номерами от 1024 до 49151 являются регистрируемыми.
- Порты с номерами от 49152 до 65535 - динамические и частные порты.

Во избежание накладок с портами, уже занятыми системой или другим приложением, ваша программа должна выбирать порты, начиная с 1024. Можно вместо конкретного номера порта задать 0, тогда система сама выберет произвольный неиспользуемый в данный момент номер.

Коммуникационная модель клиент-сервер

Приложение, использующее сокеты, состоит из распределенной программы, исполняемой на обоих концах канала связи. Программу, инициирующую передачу, называют ***клиентом***. ***Сервер*** представляет собой модуль, пассивно ожидающий входящих запросов на установку соединений от удаленных клиентов. Как правило, серверное приложение загружается при запуске системы и прослушивает свой порт, ожидая входящих соединений. Клиентские приложения пытаются установить соединение с сервером, после чего начинается обмен данными. По завершении сеанса связи клиент, как правило, разрывает соединение. На рисунке представлена базовая модель взаимодействия для потоковых сокетов.



Классы для работы с адресами в .NET

Пространство имен System.Net содержит классы для работы с доменными именами и IP-адресами: Dns, IPHostEntry, IPAddress, IPEndPoint и др..

Чтобы получить IP-адрес из DNS-имени хоста можно использовать статический метод Resolve класса Dns. У одного хоста может быть несколько IP-адресов и альтернативных имен. При получении IP-адреса метод Resolve возвращает объект класса IPHostEntry, который содержит массив адресов, имя хоста и альтернативные имена. Например, получим адресную информацию для хоста www.microsoft.com:

string hostname = “www.microsoft.com”; IPHostEntry

host = Dns.Resolve(hostname);

В классе Dns есть различные статические методы, возвращающие объекты IPHostEntry. Они приведены в таблице 1.

Таблица 1 – методы класса Dns

Метод Dns	Описание
Resolve()	По Dns-имени или IP-адресу в десятично-точечной нотации получает IPv4-адреса и альтернативные имена хоста
GetHostEntry()	По Dns-имени или IP-адресу в десятично-точечной нотации получает IPv6-адреса и альтернативные имена хоста
GetHostByName()	По заданному Dns-имени хоста возвращает список IPадресов
GetHostByAddress()	Получает строку IP-адреса в десятично-точесной нотации или объект IPAddress, возвращает объект IPHostEntry
GetHostName()	Возвращает имя локального хоста

IP-адреса обрабатываются в классе IPAddress. Используя свойство AddressList выберем из полученного с помощью класса Dns списка IP-адресов хоста первый:

IPAddress ipAddr = host.AddressList[0];

Используя класс IPEndPoint, создадим локальную конечную точку для сервера. Конечная точка задает адрес сокета – это комбинация IP-адреса и номера порта. Выбираем свободный номер порта из соответствующего диапазона:

```
int i = 11000;
IPEndPoint ipEndPoint = new IPEndPoint(ipAddr, i);
```

Для создания объекта IPAddress можно использовать статический метод Parse(). Например:

```
IPAddress address = IPAddress.Parse(“172.16.56.2”);
IPAddress address = IPAddress.Parse(defaultGateway);
```

Этот метод преобразует строку с IP-адресом в десятично-точечной нотации в целое 4хбайтовое число с сетевым порядком следования байт. К полю с IP-адресом можно обратиться с помощью свойства IPAddress.Address.

В классе IPAddress есть несколько доступных только для чтения полей, которые возвращают специальные IP-адреса:

- IPAddress.Loopback возвращает адрес петли обратной связи 127.0.0.1;
- IPAddress.Broadcast возвращает широковещательный адрес 255.255.255.255;
- IPAddress.Any возвращает адрес 0.0.0.0, который можно использовать для получения сообщений на всех сетевых интерфейсах данного узла.

Класс IPAddress также содержит методы для преобразования порядка следования байт: IPAddress.NetworkToHostOrder(), IPAddress.HostToNetworkOrder().

Поддержка сокетов в .NET

Поддержку сокетов в .NET обеспечивают классы в пространстве имен System.Net.Sockets. Основные классы приведены в таблице 2.

Таблица 2 - Классы в пространстве имен System.Net.Sockets для работы с сокетами

Класс	Описание
NetworkStream	Реализует базовый класс потока, из которого данные отправляются и в котором они получаются. Это абстракция высокого уровня, представляющая соединение с каналом связи TCP/IP.
Socket	Класс, обеспечивающий базовую функциональность приложения на основе сокетов.
TcpClient	Данный класс строится на классе Socket, чтобы обеспечить TCP-обслуживание на более высоком уровне. TcpClient предоставляет несколько методов для отправки и получения данных через сеть.
TcpListener	Построен на низкоуровневом классе Socket. Предназначен для создания серверных приложений. Он ожидает входящие запросы на соединения от клиентов и уведомляет приложение о любых соединениях.
UdpClient	Предназначен для реализации обслуживания по протоколу UDP.
SocketException	Это исключение порождается, когда в сокете возникает ошибка.

Класс Socket.

Класс Socket является базовым. Методы этого класса выполняют различные проверки, связанные с безопасностью, а затем переправляются к соответствующим аналогам из Windows Sockets API.

Важнейшие свойства класса Socket описаны в таблице 3.

Таблица 3 - Свойства класса Socket

Свойство	Описание
AddressFamily	Задает семейство адресов сокета – значение из перечисления SocketAddressFamily.
Available	Возвращает объем доступных для чтения данных.
Blocking	Дает или устанавливает значение, показывающее, находится ли сокет в блокирующем режиме.
Connected	Возвращает значение, информирующее, соединен ли сокет.
LocalEndPoint	Дает локальную конечную точку.
ProtocolType	Дает тип протокола сокета.
RemoteEndPoint	Дает удаленную конечную точку сокета.
SocketType	Дает тип сокета.

Основные методы класса Socket представлены в таблице 4.

Таблица 4 – Методы класса Socket

Метод	Описание
Accept()	Создает новый сокет для обработки входящего запроса на соединение.
Bind()	Связывает сокет с локальной конечной точкой для ожидания входящих запросов на соединение.
Close()	Закрывает сокет.
Connect()	Устанавливает соединение с удаленным хостом.
GetSocketOption()	Возвращает значение SocketOption.
IOControl()	Устанавливает для сокета низкоуровневые режимы работы. Этот метод обеспечивает низкоуровневый доступ к лежащему в основе экземпляру класса Socket.
Listen()	Помещает сокет в режим прослушивания. Этот метод предназначен только для серверных приложений.
Receive()	Получает данные от соединенного сокета.
ReceiveFrom()	Принимает дейтаграмму в буфер данных и сохраняет конечную точку.
Poll()	Определяет статус сокета.
Select()	Проверяет статус одного или нескольких сокетов.
Send()	Отправляет данные соединенному сокету.
SendTo()	Отправляет дейтаграмму на указанную конечную точку.
SetSocketOption()	Устанавливает опцию сокета.
Shutdown()	Запрещает операции отправки и получения данных на сокете.

Чтобы настроить серверный TCP-сокет в управляемом коде, прежде всего нужно создать экземпляр класса Socket. Его конструктор принимает три параметра: AddressFamily,

SocketType и ProtocolType. Параметр AddressFamily определяет используемую сокетом схему адресации. Чаще всего в качестве этого параметра используются значения InterNetwork (для адресов IPv4) и InterNetworkV6 (для адресов IPv6). Параметр SocketType определяет тип сокета. Например, Stream для потоковых сокетов, и Dgram для дейтаграммных сокетов. Параметр ProtocolType определяет применяемый сокетом протокол и принимает такие значения, как Tcp, Udp, Idp, Ggp и т. д.

Например, создать сокет для взаимодействия по протоколу TCP можно следующим образом:

Socket s = new Socket(IPEndPoint.Address.AddressFamily.InterNetwork, SocketType.Stream, ProtocolType.Tcp);

Как только потоковый сокет на сервере создан, его необходимо привязать к адресу. Привязка клиентского сокета к адресу не обязательна. Чтобы привязать сокет к адресу, используется метод Bind объекта Socket. Этому методу нужен адрес и порт, которые будут сопоставлены с сокетом, поэтому в качестве параметра он принимает экземпляр класса, производного от EndPoint. Как правило, это объект класса IPEndPoint (кроме него в .NET Framework входит только один класс, производный от EndPoint, — IrDAEndPoint, который служит для взаимодействия посредством инфракрасного порта).

После вызова метода Bind метод Socket.Listen переводит сокет в режим прослушивания и конфигурирует для сокета внутренние очереди. Когда клиент пытается подключиться к серверу, в очередь помещается запрос на установление соединения. Метод Listen принимает один аргумент — максимальное число запросов соединений, которые могут находиться в очереди.

Метод Socket.Accept извлекает из очереди первый запрос, устанавливает соединение с клиентом и возвращает новый объект Socket, который можно использовать для коммуникационного взаимодействия с данным клиентом.

В клиентской части приложения после создания объекта Socket обычно вызывается метод Connect. Можно также сначала вызвать метод Bind, если нужно, чтобы клиент использовал конкретный порт для связи с сервером. Если вы не связали клиентский сокет с портом, метод Connect выберет порт клиента автоматически. При вызове Connect клиентское приложение пытается установить соединение с сервером. Метод Connect также принимает объект EndPoint, через который определяется целевой удаленный хост. Как только соединение установлено, клиент и сервер могут передавать данные методами Send и Receive.

Когда приложения завершат обмен данными, необходимо закрыть сокет. Чтобы корректно инициировать закрытие сокета, вызывается метод Shutdown. Это позволяет передать все неотправленные данные и получить еще не принятые данные из буфера. Если вы синхронно читаете данные из сокета и получаете 0 байтов, значит, вторая сторона закрыла сокет.

Если при выполнении какого-либо метода сокета возникает сетевая ошибка, генерируется исключение SocketException. Этот объект является оболочкой кода ошибки, полученного от Win32. SocketException.ErrorCode — тот же код ошибки, который вы получили бы, вызвав Winsock-функцию WSAGetLastError. Основные коды ошибок Winsock приведены в таблице 5.

Таблица 5 – Основные коды ошибок

Номер ошибки	Значение Winsock	Значение SocketError	Описание
10004	WSAEINTR	Interrupted	Системный вызов прерван. Это может произойти, если выполняется вызов сокета, асокет закрыт.
10048	WSAEADDRINUSE	AddressAlreadyInUse	Адрес, к которому вы пытаетесь привязать сокет или который вы хотите прослушивать, уже занят.
10053	WSACONNABORTED	ConnectionAborted	Соединение было прервано локальным компьютером.
10054	WSAECONNRESET	ConnectionReset	Соединение было прервано удаленным компьютером.
10061	WSAECONNREFUSED	ConnectionRefused	Соединение с конечной точкой отклонено. Это может произойти, если хост отключен или если порт занят, а очередь запросов заполнена.

Дейтаграммные сокеты.

Создать детаграммный сокет можно следующим образом:

Socket s = new Socket(AddressFamily.InterNetwork, SocketType.Dgram, ProtocolType.Udp);

При использовании дейтаграммных сокетов сначала создают сокет. Затем выполняют привязку сокета к интерфейсу, на котором будут принимать данные, методом Bind (как и в случае протоколов, ориентированных на соединения). Разница в том, что вместо использования методов Listen или Accept нужно просто ожидать приема входящих данных. Поскольку в этом случае соединения нет, принимающий сокет может получать дейтаграммы от любой машины в сети.

Простейший метод приема — ReceiveFrom:

Другой способ приема (отправки) данных на сокетах, не требующих соединения, — установление соединения (хоть это и звучит странно). После создания сокета можно вызвать метод Connect. Фактически никакого соединения не происходит. Адрес сокета, переданный в функцию соединения, ассоциируется с сокетом, чтобы было можно использовать метод Receive вместо ReceiveFrom (поскольку источник данных известен).

Есть два способа отправки данных через сокет, не требующий соединения. Первый и самый простой — создать сокет и вызвать метод SendTo.

Как и при получении данных, сокет, не требующий соединения, можно подключать к адресу конечной точки и отправлять данные методом Send. После создания этой привязки использовать для обмена данными метод SendTo с другим адресом нельзя — будет выдана ошибка. Отменить привязку сокета можно, лишь вызвав метод Close с описателем этого сокета, после чего следует создать новый сокет.

Поскольку соединение не устанавливается, его формального разрыва или корректного закрытия не требуется. После прекращения отправки или получения данных отправителем

или получателем просто вызывается метод Close с описателем требуемого сокета, в результате чего освобождаются все выделенные ему ресурсы.

Задания на лабораторную работу

Основные задания (обязательные для выполнения)

Задание 1.

Используя класс Socket пространства имен System.Net.Sockets .Net Framework, разработать синхронное консольное клиент-серверное приложение. Клиент и сервер должны осуществлять взаимодействие по протоколу TCP.

В серверной части вывести на экран дескриптор, IP-адрес и порт слушающего сокета. При получении от клиента запроса на установление соединения вывести на экран новый дескриптор сокета, а также IP-адрес и номер порта подключившегося клиентского сокета. Клиент и сервер должны запросить и получить друг от друга текущие дату и время.

Выводить на экран все отправленные и полученные по сокету данные в клиентской и северной части приложения.

Выполнять проверку и обработку ошибок.

В отчете привести блок-схему, листинг клиентской и серверной части приложения, а также копии экранов.

Задание 2.

Разработать клиент-серверное приложение, аналогичное предыдущему заданию, но для взаимодействия клиента и севера использовать протокол UDP и дейтаграммные сокеты.

В отчете привести блок-схему, листинг клиентской и серверной части приложения, а также копии экранов.

ВНИМАНИЕ:

При защите лабораторной работы демонстрацию разработанных клиент-серверных приложений проводить в локальной сети кафедры, либо в виртуальной локальной сети.

Дополнительные задания по вариантам Вариант 1

Разработать клиент-серверное приложение для получения списка открытых TCPпортов удаленного хоста. Взаимодействие клиента и сервера осуществлять на основе потоковых сокетов.

Для получения портов использовать класс SocketException. Сканирование портов выполнять в цикле по номерам портов.

Вариант 2

Разработать клиент-серверное приложение для получения списка открытых UDPпортов удаленного хоста. Взаимодействие клиента и сервера осуществлять на основе потоковых сокетов.

Для получения портов использовать класс SocketException. Сканирование портов выполнять в цикле по номерам портов.

Вариант 3

Разработать клиент-серверное приложение для получения списка IPv4-адресов и масок сетевых интерфейсов удаленного узла. Взаимодействие клиента и сервера осуществлять на основе потоковых сокетов. Для получения адресов использовать пространство имен System.Net.NetworkInformation, классы NetworkInterface, IPInterfaceProperties (свойство UnicastAddresses).

Вариант 4

Разработать клиент-серверное приложение для получения списка IPv6-адресов сетевых интерфейсов удаленного узла. Взаимодействие клиента и сервера осуществлять на основе потоковых сокетов. Для получения адресов использовать классы NetworkInterface, IPInterfaceProperties (свойство UnicastAddresses).

Вариант 5

Разработать клиент-серверное приложение для получения DNS-имени удаленного узла (имя узла + имя домена), а также IP-адреса DNS-сервера удаленного узла. Взаимодействие клиента и сервера осуществлять на основе потоковых сокетов.

Вариант 6

Разработать клиент-серверное приложение для получения списка физических адресов сетевых интерфейсов удаленного узла. Взаимодействие клиента и сервера осуществлять на основе потоковых сокетов. Для получения адресов использовать классы пространства имен System.Net.NetworkInformation (класс PhysicalAddress и др).

Вариант 7

Разработать клиент-серверное приложение для получения списка активных TCPпортов удаленного хоста. Взаимодействие клиента и сервера осуществлять на основе потоковых сокетов.

Для получения портов использовать класс IPGlobalProperties.

Варианты 8-14

Задания аналогичные вариантам 1-7, только с использованием дейтаграммных сокетов для взаимодействия клиента и сервера.

Контрольные вопросы

1. Что такое сокет? Как по отношению к уровням стека TCP/IP расположен интерфейс сокетов?
2. Сколько сокетов необходимо для взаимодействия клиента и сервера? Что представляет собой адрес сокета?
3. Назовите типы сокетов. В каком случае предпочтительнее использовать тот или иной тип сокетов? Какой тип сокетов обеспечивает надежность и порядок доставки?
4. Чем различаются потоковые протоколы и протоколы, ориентированные на передачу сообщений? Как это отражается на коде при написании программ?
5. Чем отличается процесс получения и отправки данных на сокете, не требующем соединения?

6. Какой адрес можно использовать, чтобы прослушивать все сетевые интерфейсы локального узла?

7. Какой порядок байтов используется в Intel-совместимых процессорах? Какой порядок байтов применяется для работы с сокетами?

8. Какие номера портов могут задействовать прикладные программы при работе с сокетами?

9. Как осуществляется корректное завершение сеанса работы с потоковым сокетом, с дейтаграммным сокетом?

10. Как можно узнать номера портов, занятых стандартными службами? В каком файле хранится эта информация?

3.6 Лабораторная работа № 6 Протоколы транспортного уровня

Цель работы

Изучить транспортные протоколы Интернет.

Приобрести практические навыки создания клиент-серверных приложений, взаимодействующих по протоколам TCP и UDP на основе применения классов среды .NET Framework.

Постановка задачи

1. Изучить методические указания к лабораторной работе, материалы лекций и рекомендуемую литературу.
2. Разработать приложения клиент-сервер для демонстрации работы с TCP и UDP протоколами согласно заданию.
3. Ответить на контрольные вопросы.

Методические указания

1. Протокол TCP

TCP (Transmission Control Protocol) – протокол управления передачей, один из самых широко распространенных протоколов транспортного уровня. Это ориентированный на соединение протокол, предназначенный для обеспечения надежной передачи данных между процессами. Описан в документах RFC 793, 1122, 1323.

TCP выполняет функции контроля ошибок и управления потоком данных.

TCP не поддерживает широковещание и многоадресную рассылку. Он может использоваться только для соединений «один-к-одному». При этом протокол устанавливает дуплексный виртуальный канал передачи между конечными узлами.

Каждый взаимодействующий процесс идентифицируется сокетом – парой IP-адрес и номер порта. Логическое соединение по протоколу TCP между двумя прикладными процессами идентифицируется парой сокетов. Каждый процесс одновременно может участвовать в нескольких соединениях. Единицей данных TCP является сегмент.

2. Протокол UDP

UDP (User Datagram Protocol) - протокол пользовательских дейтаграмм.

Быстрый дейтаграммный протокол. Разработан для предоставления прикладным программам транспортных услуг.

Работает поверх IP. Не гарантирует доставку, не устанавливает виртуальных соединений, не осуществляет повторных передач, не выполняет переупорядочивания пакетов, не управляет потоком данных. Все эти функции при использовании UDP возложены на приложения (прикладные протоколы). Т.о. приложения должны сами обеспечивать надежность передачи.

По этим причинам UDP в основном служит для передачи мультимедийных данных, где важнее своевременность, а не надежность доставки.

Протокол UDP может работать по схеме «один-ко-многим», поэтому широко применяется для рассылки группового и широковещательного трафика.

3. Реализация протокола TCP на платформе .NET Класс TcpListener

Для работы с TCP протоколом среда .NET Framework предоставляет высокоуровневые классы TcpListener и TcpClient, относящиеся к пространству имен System.Net.Sockets. В отличие от более низкоуровневого класса Socket, в котором при получении/отправке данных применяется побайтовый подход, классы TcpListener и TcpClient используют потоковую модель. В них взаимодействие между клиентом и сервером основывается на потоке с применением класса NetworkStream. Они не поддерживают некоторые возможности, предлагаемые классом Socket, тем не менее, полезны во многих ситуациях.

Класс TcpListener обеспечивает работу с TCP-протоколом на стороне сервера. Он слушает запросы клиентов, принимает запрос и создает новый экземпляр класса Socket или TcpClient, который можно использовать для взаимодействия с клиентом.

Класс TcpListener инкапсулирует закрытый объект Socket, m_ServerSocket, доступный только для производных классов.

Основные методы класса TcpListener представлены в таблице.

Возвращаемый результат	Название метода	Описание
Socket	AcceptSocket()	Принимает соединение клиента и возвращает объект Socket, используемый для взаимодействия с клиентом
TcpClient	AcceptTcpClient()	Принимает соединение клиента и возвращает объект TcpClient, используемый для взаимодействия с клиентом
bool	Pending()	Показывает, есть ли запросы, ожидающие соединения
void	Start()/Start(int)	Переводит данный экземпляр TcpListener в режим прослушивания запросов соединения
void	Stop()	Закрывает данный экземпляр TcpListener, находящийся в режиме прослушивания

Основные свойства класса TcpListener представлены в таблице.

Тип	Имя свойства	Описание
IPEndPoint	LocalEndPoint	Открытое свойство LocalEndpoint возвращает объект IPEndpoint, который содержит информацию о локальном IP-адресе и номере порта, используемых для ожидания входящих запросов от клиентов.
bool	Active	Защищенное свойство, указывает, слушает ли в настоящий момент TcpListener запросы соединения.
Socket	Server	Защищенное свойство, возвращает базовый объект Socket, используемый объектом TcpListener, чтобы слушать запросы соединения.

Рассмотрим последовательность действий на стороне TCP-сервера.

1) Создание экземпляра класса TcpListener

Для создания экземпляра класса TcpListener (т.е. создания сокета) существуют три перегруженных конструктора:

1) public TcpListener(int point);
2) public TcpListener(IPEndPoint endPoint);
3) public TcpListener(IPAddress ipAddr, int port).

Первый конструктор указывает используемый для прослушивания запросов порт. При этом IP адрес равен IPAddress.Any, что эквивалентно значению 0.0.0.0. Т.е. сервер должен принимать сообщения клиентов на всех сетевых интерфейсах.

Во второй конструктор передается объект IPEndPoint, определяющий IP адрес и порт, на котором выполняется прослушивание.

Последний перегруженный конструктор принимает объект IPAddress и номер порта.

2) Прослушивание запросов клиентов

На следующем шаге после создания сокета можно приступать к прослушиванию запросов клиентов. Для этого в классе TcpListener есть метод Start(), выполняющий следующую последовательность действий:

1) Связывает сокет с IP адресом и портом, переданными в параметрах конструктору класса TcpListener (аналогично методу Bind() из класса Socket);

2) Вызывает метод Listen() базового класса Socket и начинает прослушивать запросы соединения от клиентов.

```
TcpListener tcpListener = new TcpListener(ipAddr, port); tcpListener.Start();
```

Приступив к прослушиванию сокета, можно вызывать метод Pending(), чтобы проверять наличие ожидающих запросов соединения в очереди. Этот метод позволяет проверять наличие ожидающих клиентов до вызова синхронного метода Accept(), который блокирует выполняющийся поток.

```
if (tcpListener.Pending())
{    … В очереди есть запросы на соединение от клиентов    }
```

3) Прием соединений от клиентов

Для принятия запроса на соединение от клиента используются методы AcceptSocket() и AcceptTcpClient(). Они возвращают соответственно объекты Socket или TcpClient:

```
Socket sock = tcpListener.AcceptSocket();
TcpClient sock = tcpListener. AcceptTcpClient();
```

4) Отправка и получение сообщений

В зависимости от типа сокета, созданного при установлении соединения, обмен данными выполняется методами Send() и Receive() объекта Socket или с помощью чтениязаписи объекта NetworkStream.

5) Остановка сервера

После завершения взаимодействия с клиентом нужно остановить слушающий сокет. Для этого используется метод tcpListener.Stop().

4. Класс TcpClient

Обеспечивает работу с TCP-протоколом на стороне клиента. Он построен на классе Socket и обеспечивает TCP-сервисы на более высоком уровне. В классе TcpClient есть закрытый объект данных m_ClientSocket, используемый для взаимодействия с сервером TCP.

Основные методы класса TcpClient представлены в таблице.

Название метода	Описание
Close()	Закрывает TCP-соединение
Connect()	Соединяется с удаленным хостом TCP
GetStream()	Возвращает объект NetworkStream, используемый для передачи данных между клиентом и удаленным хостом

Основные открытые свойства класса TcpClient представлены в таблице.

Тип	Имя свойства	Описание
LingerOption	LingerState	Устанавливает или возвращает объект LingerOption, содержащий информацию о том, будет ли соединение оставаться открытым после закрытия сокета и как долго
bool	NoDelay	Указывает, будет ли сокет задерживать отправку и получение данных, если буфер, назначенный для отправки или получения данных, не заполнен. Если свойство имеет значение false, TCP задержит отправку пакета, пока не будет накоплен достаточный объем данных. Это средство помогает избежать неэффективной отправки через сеть слишком маленьких пакетов.
int	ReceiveBufferSize	Задает размер в байтах буфера для входящих данных. Используется при считывании данных из сокета.
int	ReceiveTimeout	Задает время в миллисекундах, которое TcpClient будет ждать получения данных после инициирования этой операции. Если это время истечет, а данные не будут получены, возникнет исключение SocketException.
int	SendBufferSize	Задает размер в байтах буфера для исходящих данных.
int	SendTimeout	Задает время в миллисекундах, которое TcpClient будет ждать подтверждения числа байтов, отправленных удаленному хосту от базового сокета. При истечении времени SendTimeout порождается исключение SocketException.

Основные защищенные свойства класса TcpClient представлены в таблице.

Тип	Имя свойства	Описание
bool	Active	Указывает, есть ли активное соединение с удаленным хостом
Socket	Client	Задает базовый объект Socket, используемый объектом TcpClient. Поскольку это защищенное свойство, к базовому сокету можно обращаться, если класс произведен от TcpClient.

Создание сокета

Первый шаг при разработке TCP-клиента – создание экземпляра класса TcpClient. В классе TcpClient определено четыре перегруженных конструктора:

1) public TcpClient();
2) public TcpClient(IPEndPoint ipEnd); 3) public TcpClient(string hostname, int port); 4) public TcpClient(AddressFamily family).

Если используется первый или второй конструктор, то для установления соединения с удаленным хостом надо вызывать метод Connect(), в котором указывать удаленную конечную точку, с которой надо соединиться.

Перегруженный конструктор TcpClient(IPEndPoint ipEnd) инициализирует новый экземпляр класса TcpClient, связанный с указанной локальной конечной точкой.

Перегруженный конструктор TcpClient(string hostname, int port) принимает в качестве параметров имя и порт удаленного узла. Он создает новый экземпляр класса TcpClient, разрешает указанное доменное имя в IP-адрес и устанавливает удаленное соединение с указанным хостом и портом. Однако с помощью этого конструктора нельзя задать локальный адрес и порт, с которым желательно связаться.

Еще один конструктор TcpClient(AddressFamily family) позволяет передать в качестве параметра значение перечисления AddressFamily.

Установление соединения с сервером

Следующий шаг – установление соединения с удаленным хостом. Для этого предназначен метод Connect().

Существуют несколько перегруженных вариантов метода Connect().

Имя	Описание
Connect(IPEndPoint)	Устанавливает соединение клиента с удаленным хостом, используя конечную точку IPEndPoint
Connect(IPAddress, int)	Устанавливает соединение клиента с удаленным хостом, используя указанный IP-адрес и номер порта
Connect(IPAddress[], int)	Устанавливает соединение клиента с удаленным хостом, используя несколько указанных в массиве IP-адресов и номер порта
Connect(String, int)	Устанавливает соединение клиента с удаленным хостом, используя имя хоста и номер порта

Так как установка соединения зависит от многих факторов: качества работы сети, наличия работающего приложения-сервера и т.д., рекомендуется всегда вызывать этот метод в блоке try/catch для обработки исключений.

Отправка и получение сообщений.

Данные передаются по сети между двумя узлами в форме непрерывного потока. Для обработки таких потоков в .NET имеется специальный класс NetworkStream.

Класс NetworkStream входит в пространство имен System.Net.Sockets и используется для отправки и получения данных через сокеты.

Объект NetworkStream – это небуферизованный поток, который не поддерживает произвольный доступ к данным. Невозможно изменить позицию внутри потока, и, следовательно, использование метода Seek() и свойства Position порождает исключение.

Прежде чем отправлять и получать данные, нужно определить базовый поток. Для этих целей в классе TcpClient есть метод GetStream(). Он создает экземпляр класса NetworkStream и возвращает его вызывающей программе Следующий код создает экземпляр класса TcpClient, устанавливает соединение с сервером и затем создает экземпляр класса NetworkStream: TcpClient client = new TcpClient(); client.Connect(“192.168.10.1”, 80);

NetworkStream tcpStream = client.GetStream();

Далее можно использовать методы Read() и Write() класса NetworkStream для чтения из потока и записи в поток.

Метод Write() используется для выполнения синхронной записи в поток. Он записывает в поток последовательность байтов и продвигает текущую позицию в потоке вперед на число записанных байтов.

Метод Write() принимает три параметра: массив байтов, содержащий отправляемые данные, позицию в потоке, с которой нужно начать запись, и длину данных.

tcpStream.Write(sendBytes, 0, sendBytes.Length); sendBytes

– отправляемый массив байтов.

Метод Read() используется для выполнения синхронного чтения из потока. Он считывает указанное число байтов и продвигает позицию в потоке вперед на число считанных байтов.

Метод Read() имеет такой же набор параметров – массив байтов для сохранения данных, которые считываются из потока, позицию начала считывания и число считываемых байтов.

byte[] rcvBytes = new byte[client.ReceiveBufferSize];

int length_rcvBytes = tcpStream.Read(rcvBytes, 0, client.ReceiveBufferSize);

Свойство ReceiveBufferSize класса TcpClient задает размер в байтах буфера для чтения, поэтому его можно использовать как размер массива байтов.

После взаимодействия с сервером, чтобы освободить все ресурсы и закрыть клиентский сокет, следует вызвать метод Close():

client.Close();

5. Реализация протокола UDP на платформе .NET Класс UdpClient

Класс UdpClient предназначен для реализации в сети протокола UDP. Он построен на классе Socket.

Стандартная схема применения класса UdpClient такова. Необходимо создать экземпляр класса. Далее через вызов метода Connect() соединиться с удаленным хостом. В действительности метод Connect() в данном случае не устанавливает соединение, а служит для указания пункта назначения дейтаграммы. Оба шага можно сделать и в одной строке (без вызова метода Connect()), если сразу указать в конструкторе UdpClient удаленный адрес и порт.

Третий шаг – отправка и получение данных с помощью методов Send() и Receive(). Затем методом Close() закрывают сокет.

Одна из возможных схем отправки и получения данных по протоколу UDP с использованием класса UdpClient представлена на рисунке.

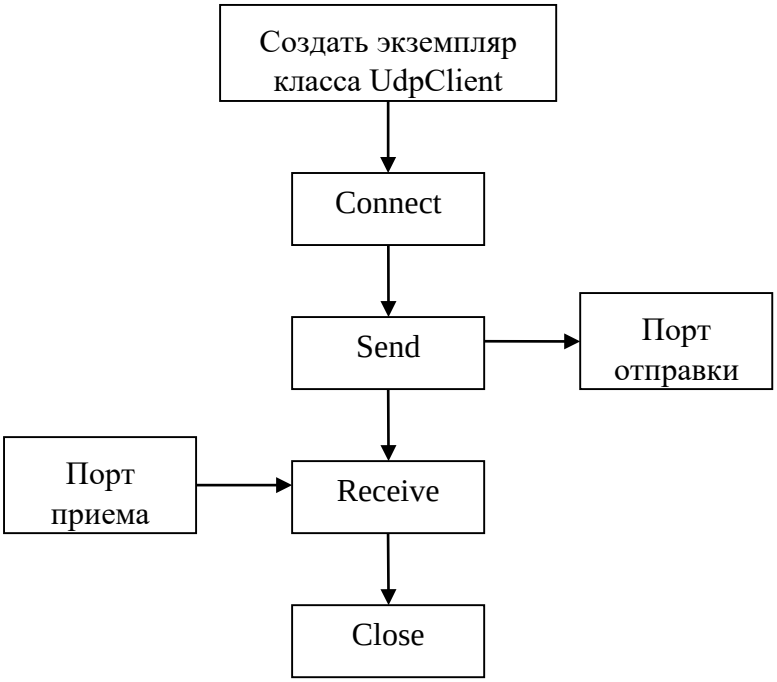


Рисунок - Схема отправки и получения данных по протоколу UDP

Экземпляр класса UdpClient можно создать несколькими способами, которые отличаются передаваемыми параметрами. Дальнейшее использование объекта UdpClient зависит от того, как он создавался.

Простейший способ состоит в вызове конструктора по умолчанию (без передачи параметров): UdpClient(). В этом случае далее нужно или вызвать метод Connect() или задать информацию о получателе при отправке данных в методе Send(). Если выбран конструктор по умолчанию, то при получении данных будут использоваться произвольный свободный порт и IP-адрес 0.0.0.0 .

Можно создать объект UdpClient, указав в качестве параметра только номер порта. В этом случае UdpClient будет слушать все локальные интерфейсы, т.е. опять использовать IP-адрес 0.0.0.0 .

Следующий способ создания объекта UdpClient заключается в использовании объекта IPEndPoint, который содержит локальный IP-адрес и номер порта. IP-адрес должен принадлежать одному из интерфейсов локальной машины, иначе произойдет ошибка.

Последний способ состоит в передаче конструктору UdpClient имени хоста и номера порта удаленного хоста. Это позволяет исключить шаг с вызовом метода Connect().

6. Многопоточное приложение клиент-сервер

В многопоточном сервере приложение работает в основном потоке, а для каждого подключившегося клиента создается новый поток. Поток существует все время взаимодействия с клиентом. При отключении клиента поток уничтожается.

В случае небольшого сервера и нескольких клиентов такой подход работает хорошо, к тому же его легко реализовать. К сожалению, многопоточный сервер плохо масштабируется. Главный его недостаток — большое число создаваемых и уничтожаемых потоков. Такой сервер может принять около 1000 соединений, после чего начинают генерироваться исключения, извещающие о нехватке памяти (память быстро заканчивается потому, что каждый поток имеет собственный стек, объем которого по умолчанию равен 1 МБ).

Избежать создания отдельного потока для каждого соединения можно создав заранее некоторое количество потоков - пул. При подключении клиента берется поток из пула потоков. В нем происходит взаимодействие с данным клиентом. Задачи выполняются параллельно. Если потоки не имеют общих данных, то не будет накладных расходов на синхронизацию, что делает работу достаточно быстрой. После завершения работы поток не убивается, а лежит в пуле, ожидая следующей задачи. Это убирает накладные расходы на создание и удаление потоков. Этот подход вполне приемлем, если соединения будут недолговечными. Однако соединения с серверами (например, с сервером-чатом) обычно остаются открытыми длительное время, поэтому ограничивать число соединений числом потоков в пуле зачастую неприемлемо.

Преимущества многопоточной модели: высокая скорость взаимодействия с каждым отдельным клиентом.

Недостатки многопоточной модели:

1) Каждый поток забирает часть ресурсов сервера. Следовательно, количество потоков ограничено мощностью сервера.

2) Большие накладные расходы на обработку потоков (создание, удаление, переключение между потоками).

3) Значительное время простоя процессора из-за невозможности перераспределения нагрузки между потоками.

Задания на лабораторную работу

Задание 1

1. Используя класс TcpClient, разработать консольное приложение TCP-клиент, устанавливающее удаленное соединение с сервером с использованием DNS-имени и номера порта. Клиент должен отправлять запрос серверу, получать ответ и отображать его для пользователя.

2. Используя класс TcpListener, разработать консольное приложение эхо-сервер для получения сообщений и отправки их назад TCP-клиенту. Сервер должен принимать сообщения клиентов по заданному локальному IP-адресу и номеру порта.

Задание 2

Разработать консольное приложение, осуществляющее взаимодействие по протоколу UDP. Использовать класс UdpClient. Приложение должно представлять собой простейший чат, позволяющий двум узлам, на которых оно запущено, обмениваться текстовыми сообщениями.

При запуске приложение должно запрашивать адреса локального и удаленного сокета.

Использовать отдельные потоки для получения и передачи данных, чтобы синхронные методы не заблокировали основной поток.

Задание 3

Доработать TCP-сервер из задания 1, превратив его в синхронный многопоточный сервер. Для каждого подключающегося клиента создавать отдельный поток. Выводить на экран (или в файл журнала подключений) адреса подключившихся клиентов, время подключения, идентификатор обрабатывающего потока.

Продемонстрировать работу сервера, подключая большое количество клиентов в автоматическом режиме.

Контрольные вопросы

1. К какому уровню стека TCP/IP относится TCP-протокол и каковы его основные функции?
2. Как называется единица данных протокола TCP? Каков ее максимальный размер?
3. Как выполняется адресация приложений на транспортном уровне?
4. Как TCP устанавливает и разрывает соединение?
5. Какой алгоритм используется для обнаружения и исправления ошибок в протоколе TCP? Как он работает?
6. Перечислить достоинства и недостатки протокола UDP? В каких случаях применяется протокол UDP?
7. Какие классы в пространстве имен System.Net.Sockets .NET Framework обеспечивают поддержку сокетов и могут использоваться для работы с транспортными протоколами?
8. Какой конструктор класса TcpClient сам устанавливает удаленное соединение с сервером и не нуждается в вызове метода Connect? Какой конструктор класса TcpClient позволяет задать локальный адрес и порт, с которыми желательно связаться серверу?
9. Какими способами можно создать экземпляр класса UdpClient? Каким образом можно задать информацию о получателе дейтаграмм?
10. Каковы преимущества и недостатки многопоточной модели?

4 Экзаменационные вопросы

1. Эволюция и современные тенденции развития сетевых технологий.
2. Понятие и основные компоненты компьютерной сети.
3. Сетевое программное обеспечение.
4. Требования к адресации узлов сети. Понятие адресного пространства. Основные схемы адресации.
5. Классификация адресов по количеству адресуемых сетевых интерфейсов. Групповая адресация.
6. Протоколы разрешения адресов. Централизованный и распределенный подход. Протокол ARP.
7. Локальные адреса.
8. Числовые-составные адреса. IPv4-адреса. Назначение IP-адресов. Протокол DHCP.
9. Диапазоны IPv4-адресов (публичные, частные, APIPA, специальные IP-адреса).
10. Адреса IPv6. Форма записи. Типы (глобальные, канальные, уникальные).
11. Символьные адреса. Методы разрешения имен в ОС Windows.
12. Система DNS.
13. Иерархическое доменное пространство имен. Полное доменное имя FQDN.
14. DNS-серверы и распознаватели. Методы разрешения DNS-имен.
15. Основы классификации компьютерных сетей.
16. Локальные и глобальные сети.
17. Сети операторов связи и корпоративные сети.
18. Понятие логической архитектуры компьютерной сети. Одноранговая архитектура.
19. Архитектура клиент-сервер. Web-архитектура.
20. Основы технологии сокетов (понятие сокета, адресация, порядок байт, типы сокетов).

21. Клиент и сервер на базе сокетов.
22. Потоковые сокеты. Классы .NET для реализации потоковых сокетов.
23. Дейтаграммные сокеты. Классы .NET для реализации дейтаграммных сокетов.
24. Реализация многопользовательского сервера.
25. Многоуровневый подход к стандартизации в компьютерных сетях. Понятия «протокол», «интерфейс», «стек протоколов».
26. Сетезависимые уровни OSI.
27. Канальный уровень модели OSI.
28. Сетенезависимые уровни эталонной модели взаимодействия открытых систем.
29. Понятие стека коммуникационных протоколов. Стек TCP/IP. Преимущества и недостатки.
30. Сеть Интернет. Стандарты TCP/IP.
31. Архитектура стека TCP/IP.
32. Уровень межсетевого взаимодействия. Функции. Протоколы.
33. Протокол IP. Структура IP-пакета.
34. Понятие маршрутизации. Виды маршрутизации. Табличная маршрутизация.
35. Групповая адресация. Протоколы групповой передачи.
36. Автономные системы. Протоколы маршрутизации.
37. Понятие маски. Использование масок в IP-адресации.
38. Общая характеристика протокола TCP. Структура заголовка TCP-сегмента.
39. Протокол UDP.
40. Общая характеристика протокола IPv6.
41. Дефицит IP-адресов. Технологии NAT и CIDR.
42. Протоколы электронной почты.
43. Универсальный идентификатор ресурсов URI.

44. Сервис WWW и его составляющие.
45. Протокол HTTP.
46. Протоколы прикладного уровня. Протокол FTP.
47. Классификация и характеристики линий связи.
48. Типы кабелей. Структурированная кабельная система.
49. Витая пара и коаксиальный кабель.
50. Волоконно-оптический кабель.
51. Базовые топологии компьютерных сетей.
52. Понятие коммутации. Коммутация пакетов.
53. Общая характеристика технологии Ethernet. Форматы кадров.
54. Метод доступа CSMA/CD.
55. Технология Fast Ethernet.
56. Технология Gigabit Ethernet.
57. Виды коммуникационного оборудования.
58. Алгоритм работы прозрачного моста.
59. Беспроводные локальные сети IEEE 802.11
60. Персональные сети и технология Bluetooth.

5 Тесты

5.1 Тест №1

1. Какой адрес называется адресом "петли обратной связи"?
2. Как определить есть ли связь с хостом www.microsoft.com?
3. С помощью какой утилиты можно получить всю адресную информацию о стеке TCP/IP? (Введите командную строку)
4. Какая утилита позволяет вывести маршрут до удаленного узла?
5. С помощью какой утилиты можно получить информацию о портах, занятых на данном компьютере?
6. Какая утилита предназначена для просмотра и редактирования таблицы маршрутизации?
7. Как определить имя компьютера?
8. С помощью какой утилиты можно узнать соответствие между IP-адресами и физическими адресами узлов сети?
9. С помощью какой утилиты можно определить количество сетевых интерфейсов узла?
10. Протокол ARP предназначен для:
    a) преобразования IP-адреса в физический
    b) преобразования имени узла в IP-адрес
    c) передачи информации по сети
    d) получения IP-адреса узла
11. С помощью какой утилиты можно определить потери данных на промежуточных узлах?
12. Как получить справку по утилите ping? (Напишите нужную строку)
13. Как проверить, что TCP/IP установлен и правильно сконфигурирован на локальном компьютере?
14. Какие утилиты для работы используют эхо-запросы протокола ICMP?

15. С помощью какой команды можно вывести символьные имена всех доступных в данный момент компьютеров локальной сети или домена?

    a) netstat

    b) net view

    c) net use

    d) hostname

5.2 Тест № 2

1. Какие параметры являются обязательными при настройке сетевого подключения для TCP/IP?
2. Какой протокол используется для динамического назначения IP-адреса?
3. Программа для запроса доступа к удаленным ресурсам и услугам – это
4. В каком случае сетевое подключение автоматически назначит себе частный IPv4адрес APIPA?

    a) если задано динамическое получение IP-адреса, но DHCP-сервер недоступен и не задан вручную адрес альтернативной конфигурации

    b) если назначен статический IP-адрес и не задан адрес альтернативной конфигурации

    c) если задано динамическое получение IP-адреса и не задан вручную адрес альтернативной конфигурации

    d) если задан адрес альтернативной конфигурации и недоступен DHCP-сервер

5. Могут ли компьютеры с адресами APIPA получить доступ в Internet?
6. В каком диапазоне назначаются адреса APIPA? (Укажите минимальное и максимальное значение IP-адреса)
7. Основной шлюз – это

    a) IP-адрес ближайшего маршрутизатора

    b) IP-адрес ближайшего коммутатора

    c) IP-адрес сервера

d) IP-адрес DNS сервера

e) DNS-имя сервера

8. Узел, предоставляющий свои ресурсы другим узлам сети – это

9. Пара модулей «клиент - сервер», обеспечивающих совместный доступ пользователей к определенному типу ресурсов через сеть – это

10. В сети какой логической архитектуры все узлы равноправны?

11. Сеть, главным назначением которой является поддержание работы конкретного предприятия, владеющего данной сетью – это

12. PAN – это_____

13. Сеть кампуса – это

14. С какого значения начинаются канальные IPv6 адреса, соответствующие конфигурируемым с помощью протокола APIPA IPv4 адресам?

15. Какие сети исторически появились первыми?

5.3 Тест № 3

1. Набор формализованных правил, по которым осуществляется взаимодействие одинаковых уровней в разных узлах сети – это

2. Единицей данных какого уровня является кадр?

3. Интерфейс – это

   a) согласованный набор стандартов, достаточный для построения компьютерной сети

   b) набор формализованных правил, по которым осуществляется взаимодействие одинаковых уровней в разных узлах сети

   c) набор формализованных правил, по которым осуществляется взаимодействие соседних уровней одного узла

4. Стек коммуникационных протоколов – это

   a) набор протоколов, достаточный для организации сетевого взаимодействия

b) набор формализованных правил, по которым осуществляется взаимодействие одинаковых уровней в разных узлах сети

c) набор формализованных правил, по которым осуществляется взаимодействие соседних уровней одного узла

5. Какой тип адреса используется на канальном уровне?
6. На каком уровне используется IP - адрес?
7. На каком уровне используются символьные адреса?
8. На каком уровне осуществляется маршрутизация IP - пакетов?
9. Какой уровень обеспечивает передачу данных с той степенью надежности, которая для них требуется?
10. С помощью протоколов какого уровня пользователь получает доступ к разделяемым сетевым ресурсам?
11. Выберите, какие из протоколов работают на сетевом уровне?

a) HTTP
b) FTP
c) Fast Ethernet
d) IP
e) Frame Relay
f) ARP
g) RIP
h) ICMP

12. На каком уровне работают маршрутизаторы?
13. Выберите протоколы прикладного уровня:

a) HTTP
b) SMTP
c) IP
d) Ethernet
e) DNS
f) POP3
g) TCP

14. Транспортные протоколы работают:

a) только на конечных узлах сети
b) на маршрутизаторах
c) на всех коммуникационных устройствах
d) на конечных узлах и маршрутизаторах
e) на конечных узлах и маршрутизаторах

15. Как называется единица данных сетевого уровня?
16. Перечислите уровни эталонной модели взаимодействия открытых систем по порядку сверху вниз, указывая номера уровней.
17. Перечислите уровни, которые относятся к сетезависимым
18. Какая единица данных используется на физическом уровне?
19. Какой уровень отвечает за управление потоком данных, установление и разрыв соединения?
20. Для какого уровня важны характеристики среды передачи?

5.4 Тест № 4

1. Из предложенных адресов выберите DNS - адрес
   a) 01-AB-CD-12-17-02
   b) 127.0.0.1
   c) rot1.hf.bgu.com
   d) http://www.microsoft.com/rts/
   e) 2001:DB8:21DA::D167:37AB
2. Какой протокол преобразует IP адрес в MAC адрес сетевого адаптера?
3. Выберите преимущества LLMNR
   a) не требует конфигурирования
   b) поддерживается всеми версиями Windows
   c) может использоваться с IPv6
   d) включен по умолчанию
   e) поддерживает IPv4
   f) можно использовать для разрешения имен компьютеров за пределами локальной подсети
4. Основной административный элемент, посредством которого серверы DNS осуществляют управление процессом разрешения имен – это

5. Какую архитектуру имеет система DNS?
6. Какому домену принадлежат корневые серверы имен?
7. Какой тип адреса используется аппаратурой?
   a) символьный
   b) сетевой
   c) физический
8. Какой тип адреса предназначен для пользователей?
   a) DNS b) MAC
   c) IP
9. Выберите из списка IPv4 – адрес
   a) 20-AB-1C-FF-13-03 b) 172.16.04.27
   c) www.tut.by d) fe80::125A:0340:FFE8:3450%8
10. Multicast-адрес – это
    a) адрес произвольной рассылки
    b) групповой адрес
    c) уникальный адрес
    d) широковещательный адрес
11. На каком уровне используется IP - адрес?
12. Составные части в DNS - имени отделяются друг от друга
13. Укажите размер физического адреса в байтах
14. Какой протокол преобразует символьное имя в IP-адрес?
15. Укажите размер IPv6-адреса в байтах
16. Выберите из списка IPv6 адреса
    a) www.bntu.by
    b) 172.16.43.1
    c) ::1
    d) http://[2000::125A:0340:FFE8:3450]

e) fe80::125A:0340:FFE8:3450%8

17. Выберите из предложенных адреса, которые имеют иерархическую структуру: a) DNS
    b) IP
    c) MAC
18. Какой из протоколов является централизованным?
    a) DNS
    b) ARP
19. Какой из адресов используется для доставки информации одному члену группы? a) unicast
    b) multicast
    c) broadcast
    d) anycast
20. Какой аппаратный адрес используется для широковещательной передачи?
    a) 00-00-00-00-00-00
    b) 255.255.255.255
    c) 0.0.0.0
    d) FF-FF-FF-FF-FF-FF

5.5 Тест № 5

1. Мост строит адресную таблицу:
    a. с помощью обмена служебными сообщениями с другими мостами;
    b. с помощью ПО стека протоколов;
    c. на основании пассивного наблюдения за трафиком, циркулирующим в подключенных к его портам сегментах.
2. Основной функцией концентратора является

a. локализация трафика;

b. отключение неверно работающих портов;

c. использование резервных связей;

d. повторение кадра.

3. На каком уровне модели OSI работает технология Ethernet?

4. Технология FastEthernet описана в стандарте
   a. 802.3u

   b. 802.3k

   c. 802.3ab

5. Все отличия FastEthernet от Ethernet сосредоточены на

   a. канальном уровне

   b. сетевом уровне

   c. физическом уровне

6. Какова скорость передачи Fast Ethernet?

7. Физическая топология, используемая в сетях FastEthernet

8. Какой тип кабеля является самым гибким, дешевым и простым в монтаже?

9. У какого типа кабеля наибольшая сложность и стоимость монтажных работ?

10. В качестве источника сигнала для одномодового оптоволокна применяются

    a. вакуумные лампы

    b. светодиоды

    c. полупроводниковые лазеры

11. Какие кабели в зависимости от электрических и механических характеристик разделяется на категории?

    a. экранированная витая пара

    b. неэкранированная витая пара

    c. толстый коаксиал

12. Какой кабель больше подвержен помехам?
    a. неэкранированная витая пара
    b. толстый коаксиал
    c. волоконно-оптический
13. Какой термин определяет схему сети.
14. В сетях с топологией звезда отсоединение одного компьютера приводит к
    a. остановке всей сети
    b. нарушению работы одного сегмента
    c. широковещательному шторму
15. Терминаторы используются в сетях с топологией _____

5.6 Тест № 6

1. Адрес сокета – это _____
2. Какая из характеристик не соответствует архитектуре клиент – сервер?
    a. централизованное администрирование
    b. централизованная защита ресурсов
    c. любое количество пользователей
    d. использование специального сетевого по
    e. все компьютеры равноправны
3. В каком из вариантов допущена ошибка при создании сокета?
    a. Socket soc = new Socket(AddressFamily.InterNetwork, SocketType.Dgram, ProtocolType.Udp);
    b. Socket soc = new Socket(AddressFamily.InterNetwork, SocketType.Stream, ProtocolType.Tcp);

c. Socket soc = new Socket(AddressFamily.InterNetwork, SocketType.Raw, ProtocolType.Ip);

d. Socket soc = new Socket(AddressFamily.InterNetwork, SocketType.Stream, ProtocolType.Udp);

e. во всех перечисленных

4. Какая из служебных утилит TCP/TP позволяет получить занятые порты TCP и UDP?

   a. Arp

   b. Ipconfig

   c. Netstat

   d. Ping

   e. Tracer

5. Какой из адресов может заменить в программе все сетевые интерфейсы данного узла?

   a. 0.0.0.0

   b. 127.0.0.1

   c. 172.16.12.34

   d. 10.0.0.0

   e. 245.15.173.1

6. Адрес, который используется для тестирования клиент-серверных приложений в пределах одного узла, называется

   a. ограниченное широковещательное
   b. направленное широковещательное

   c. "петля обратной связи"

   d. шлюз по умолчанию

   e. пул адресов

7. Сокет –это?

   a. гнездо, разъём

   b. конечная точка сетевого соединения

c. абстрактное программное понятие

d. IP - адрес и номер порта

e. все перечисленное

8. Какой из перечисленных классов позволяет работать с сокетами в .NET

   a. Socket

   b. TCPListener

   c. TCPClient

   d. UDPClient

   e. Все перечисленные

9. Как называется программный компонент, посылающий запросы на доступ к удаленным ресурсам?

10. Над протоколами какого уровня стека протоколов находится интерфейс сокетов?

11. Перечислите типы сокетов.

12. Какой тип сокета обеспечивает высокую надежность при передаче данных?

13. Какой тип сокета позволяет работать с протоколами сетевого уровня?

14. Какой порядок байтов используется при передаче данных по сети?

15. Какие номера портов можно использовать в прикладных программах?

6 Список литературы

Основная литература

1. Олифер, В.Г. Компьютерные сети. Принципы, технологии, протоколы: Учебник для вузов / В.Г. Олифер, Н.А. Олифер. - 6-е изд. - СПб: Питер, 2020. - 1008с.

2. Таненбаум, Э. Компьютерные сети/ Э. Таненбаум, Д. Уззеролл. - 5-е изд. - СПб: Питер, 2016. - 960с.

3. Куроуз, Дж. Компьютерные сети: Нисходящий подход/ Джеймс Куро-уз, Кит Росс.- 6е изд.- М: «Эксмо», 2016.- 912с.

Дополнительная литература

1. Одом, У. Официальное руководство Cisco по подготовке к сертификационным экзаменам CCENT/CCNA ICND1 100-105/ Уэнделл Одом.- М.: Изд-во Вильямс, 2018. – 1088с. 2. Глейзер, Дж. Многопользовательские игры. Разработка сетевых приложений/ Дж. Глейзер, С. Мадхав. - СПб: Питер, 2017. – 368с.

3. Леонтьев, В. Windows 10/ В. Леонтьев. – 4-е изд. - М: «Эксмо», 2019.- 384с.

4. Куроуз, Дж. Компьютерные сети. Настольная книга системного администратора/ Джеймс Куроуз, Кит Росс.- 6-е изд.- М: «Эксмо», 2016.- 912с.

5. Робачевский, А. Интернет изнутри. Экосистема глобальной сети/ А. Робачевский.- 2-е изд. – М.: Изд-во Альпина Паблишер, 2017. – 224с.

6. Сергеев, А. Основы локальных компьютерных сетей. Учебное пособие/ А. Сергеев. – СПб: Изд-во Лань, 2016. – 184с.

7. Стандарты Интернета (RFC) [Электронный ресурс].- Режим доступа: http://rfc.com.ru/