## Министерство образования и науки Российской Федерации

###### Новосибирский государственный технический университет

Н. Л. ДОЛОЗОВ, В.Г. КОБЫЛЯНСКИЙ

КОМПЬЮТЕРНЫЕ СЕТИ

Методические указания к выполнению лабораторных работ для студентов 3 курса направлений 01.03.02 «Прикладная математика» и 02.03.03 «Математическое обеспечение и администрирование информационных систем» факультета прикладной математики и информатики

Новосибирск

2021

Оглавление

**ВВЕДЕНИЕ**

Компьютерные сети — это одна из самых важных и захватывающих технологий нашего времени. Два десятилетия назад доступ к сетям имело лишь небольшое число людей. В настоящее время обмен данными между компьютерами стал неотъемлемой частью нашей повседневной жизни. С осознанием важности компьютерных сетей и ростом их популярности появился устойчивый спрос на специалистов разных категорий, имеющих опыт работы с сетями. Компании всё чаще привлекают специалистов, которые занимаются планированием, приобретением, установкой, эксплуатацией и управлением аппаратными и программными системами, лежащими в основе локальных и объединённых сетей. Современное компьютерное программирование не ограничивается задачами, которые решаются на отдельных компьютерах. Следовательно, программисты должны проектировать и реализовывать прикладное программное обеспечение, способное взаимодействовать с программным обеспечением, работающим на удалённых компьютерах.

В связи с этим студентам необходимо изучить и на практике освоить базовые средства для создания приложений, которые могли бы взаимодействовать в сети.

Цель настоящего учебно-методического пособия – помочь студентам теоретически и на практике, т.е. в формате лабораторных работ, освоить основные сетевые средства, начиная от самых низких уровней передачи данных, и заканчивая наивысшими уровнями прикладного программного обеспечения.

Описания всех лабораторных работ в данном пособии имеют одинаковую структуру, состоящую из трех частей. В первой части приводятся теоретические сведения и необходимые методические указания. Достаточно сложный и объёмный теоретический материал представлен в краткой и понятной форме. Во второй части сформулированы задания в различных вариантах, а третья часть содержит контрольные вопросы, ответы на которые позволят студентам выделить в работе ключевые моменты и закрепить полученные знания и практические навыки.

В пособии также приведены список источников и приложения, содержащие рисунки и примеры реализации некоторых программ, написанных на языке C++.

# Лабораторная работа № 1. Анализ структуры локальной сети ФПМИ

1. Цель работы

Изучить основные типы сетевого оборудования. Выполнить анализ структуры локальной сети факультета ФПМИ и стека протоколов INTERNET.

### Методические указания

Количество компьютеров, подключенных к глобальной Сети Интернет, измеряется миллионами (и это число постоянно растет); таким образом, Интернет создает глобальную коммуникацию, позволяя огромному числу пользователей обмениваться информацией, и использовать вычислительные ресурсы друг друга. Кроме того, в настоящее время происходит тесная интеграция Интернета с мобильными и беспроводными технологиями, что значительно расширяет круг его функций. Интернет представляет собой всемирную компьютерную сеть, то есть сеть, связывающую в единое целое миллионы вычислительных устройств, расположенных в разных уголках земного шара.

Компьютеры, подключенные к сети, маршрутизаторы и другие «компоненты» Интернета используют протоколы, осуществляющие управление приемом и передачей информации внутри Интернета. Наиболее важными протоколами в глобальной сети являются TCP (Transmission Control Protocol — протокол управления передачей) и IP (Internet Protocol — Интернет-протокол). Стек основных протоколов, использующихся в Интернете, известен под названием TCP/IP.

С точки зрения технологий и развития существование Интернета обеспечивается созданием, проверкой и внедрением Интернет - стандартов. Эти стандарты вырабатываются проблемной группой разработок для Интернета (Internet Engineering Task Force, IETF). Документы, создаваемые IETF, носят название RFC (Requests For Comments — обращения за разъяснениями). Изначально подобные документы предназначались для разрешения архитектурных проблем, возникавших в сетях-предшественницах Интернета. Со временем ситуация сложилась так, что, формально не обладая статусом стандарта, документы RFC стали стандартами де-факто. В настоящее время эти документы составляются весьма точно и детально, описывая такие протоколы, как TCP, IP, HTTP (для web) и SMTP (для электронной почты). Существует более 6000 различных документов RFC.

Протокол определяет формат и очередность сообщений, которыми обмениваются два или более устройства, а также действия, выполняемые при передаче и/или приеме сообщений либо при наступлении иных событий. Протоколы очень широко используются как в компьютерных сетях вообще, так и в сети Интернет в частности.

2.1 Общая информация о протоколах сети Интернет

Как известно, протоколы, а, следовательно, и все сетевое программное и аппаратное обеспечение организованы в виде уровней. Каждый протокол относится к определенному уровню сетевой коммуникационной модели OSI (Open System Interconnection). Многоуровневая структура позволяет детально оценивать элементы большой и сложной системы, что уже является ее значительным достоинством. Кроме того, с использованием многоуровневой структуры легче модифицировать функции системы — для этого лишь нужно внести изменения в соответствующий уровень, при этом структурно-функциональная организация системы останется прежней. Совокупность протоколов всех уровней коммуникационной модели называется стеком протоколов.

Снабжение каждого уровня коммуникационной модели собственным протоколом наряду с достоинствами имеет и несколько недостатков. Первый связан с нередко встречающейся ситуацией дублирования одних и тех же функций различными уровнями (например, контроль ошибок). Другой потенциальный недостаток заключается в том, что функциям одного уровня может понадобиться информация, хранящаяся на другом уровне (например, время). Это нарушает принцип изолированности уровней многоуровневой структуры.

Коммуникационная модель Интернета состоит из пяти уровней: физического, канального, сетевого, транспортного и прикладного. Вместо терминов «единица обмена сетевого уровня», «единица обмена канального уровня» используются специальные имена. Они приведены в таблице 1.1

Таблица 1.1

|  |  |
| --- | --- |
| *Уровень* | *Единица измерения* |
| Прикладной | Сообщение |
| Транспортный | Сегмент |
| Сетевой | Дейтаграмма/Пакет |
| Канальный | Кадр |
| Физический | Поток битов |

Поддержка протоколов может быть аппаратной, программной или смешанной. Прот околы прикладного уровня, такие как HTTP и SMTP, а также протоколы транспортного уровня практически всегда поддерживаются программно. Напротив, протоколы физического и канального уровней, тесно связанные со средой передачи данных, поддерживаются аппаратно сетевой интерфейсной картой. Сетевой уровень, находящийся в центре коммуникационной модели, может поддерживаться как аппаратно, так и программно. Далее даны характеристики каждого из пяти уровней коммуникационной модели Интернета.

**Прикладной уровень**, как следует из его названия, предназначен для поддержки сетевых приложений. Имеется множество протоколов прикладного уровня, из которых наиболее важными являются HTTP (для просмотра web-страниц), SМТР (для электронной почты) и FTP (для обмена файлами).

Главная функция **транспортного уровня** заключается в передаче сообщений прикладного уровня между клиентом и сервером. В Интернете существуют два транспортных протокола: TCP и UDP. Протокол TCP обеспечивает передачу с установлением логического соединения, то есть надежную передачу с контролем перегрузки. Протокол UDP обеспечивает передачу сообщений без установления логического соединения, то есть ненадежный вид связи, где допускаются искажения и потери данных.

**Сетевой уровень** обеспечивает передачу дейтаграмм между двумя хостами и базируется на двух основных протоколах. Первый протокол определяет поля дейтаграммы и интерпретацию их содержимого маршрутизаторами и оконечными системами. Этот протокол является единственным протоколом сетевого уровня в Интернете и имеет название IP. Вторым протоколом является один из многочисленных протоколов маршрутизации, предназначенных для определения путей дейтаграмм от отправителя до адресата. Несмотря на функциональные различия между протоколом IP и протоколами маршрутизации, а также на широкое разнообразие последних, их обычно объединяют под общим именем IP, подчеркивая этим их связующую роль в организации глобальной Сети.

Протокол транспортного уровня (TCP или UDP) передает сегмент и адрес назначения протоколу IP сетевого уровня подобно тому, как вы опускаете письмо в почтовый ящик, а протокол IP сетевого уровня доставляет сегмент конечному хосту и передает его обратно транспортному уровню.

Сетевой уровень обеспечивает передачу пакета через серию маршрутизаторов между оконечными системами. Для перемещения пакета (дейтаграммы) от одного узла к другому сетевой уровень прибегает к службам канального уровня. Таким образом, основная функция канального уровня заключается в передаче дейтаграмм между узлами на маршруте.

**Канальный уровень** использует специальный протокол, ориентированный на используемую линию связи. Иногда протоколы канального уровня обеспечивают надежную передачу между узлами. Обратите внимание на различие надежной передачи на транспортном и канальном уровне: протокол TCP обеспечивает надежность на всем пути следования сообщения, а протокол канального уровня — лишь между парой узлов. К протоколам канального уровня относятся Ethernet и РРР; иногда аналогичные функции несут технологии асинхронной передачи данных (ATM) и ретрансляции кадров. Поскольку путь от отправителя до адресата обычно состоит из цепочки разнородных линий связи, передача дейтаграммы может осуществляться различными канальными протоколами.

Если назначением канального уровня является передача кадров между соседними узлами сети, то **физический уровень** обеспечивает передачу между узлами отдельных битов информации. Протоколы физического уровня также напрямую зависят от использующейся линии связи (медной витой пары, одномодового оптоволокна и т. п.). Технология Ethernet поддерживает множество протоколов физического уровня, предназначенных для поддержки витой пары, коаксиального кабеля, оптоволоконного кабеля и некоторых других видов линий. В каждой из линий связи механизмы передачи бита различны.

2.2 Сетевое оборудование

Основными компонентами сети являются рабочие станции, серверы, передающие среды (кабели) и сетевое оборудование. **Рабочими станциями** называются компьютеры сети, на которых пользователями сети реализуются прикладные задачи. **Серверы сети** - это аппаратно-программные системы, выполняющие функции управления распределением сетевых ресурсов общего доступа. Сервером может быть любой подключенный к сети компьютер, на котором находятся ресурсы, используемые другими устройствами локальной сети. В качестве аппаратной части сервера используется достаточно мощные компьютеры.

Выделяют следующие виды сетевого оборудования.

**Сетевые карты** – это контроллеры, подключаемые в слоты расширения материнской платы компьютера, предназначенные для передачи сигналов в сеть и приема сигналов из сети.

**Терминаторы** - это резисторы номиналом 50 Ом, которые производят затухание сигнала на концах сегмента сети.

**Повторители** (репитер, от англ. repeater) — сетевое оборудование, работающее в физических сегментах сети и предназначенное для увеличения расстояния сетевого соединения путём усиления и восстановления формы входного электрического сигнала. Повторители работают на физическом уровне и имеют два порта. Они не распознают MAC - адреса и поэтому не могут использоваться для уменьшения трафика. В настоящее время эти устройства не используются.

**Концентраторы** (Hub) – это центральные устройства кабельной системы или сети физической топологии "звезда". Концентраторы работают в физических сегментах сети и при получении кадра на один порт пересылают его на все остальные. В результате получается сеть с логической структурой общей шины. Концентратор по сути является многопортовым повторителем. Основные отличия концентратора от повторителя – это возможность объединять сегменты сетей с разной физической средой (например, коаксиальный кабель и витая пара) и автоматическое отключение портов при возникновении на них ошибок.

**Мосты** (Bridge) – устройства сети, которые работают на канальном уровне в локальных сетях, передавая данные между отдельными сегментами такой сети. Это позволяет расширить максимальный размер сети, одновременно не нарушая ограничений на максимальную длину кабеля и количество подключенных к сегменту устройств. Применение мостов сокращает сетевой трафик за счет того, что пришедший кадр анализируется для выяснения MAC-адреса его получателя и передается только ему. Мосты также усиливают и конвертируют сигналы для кабеля другого типа.

**Коммутаторы** (Switch) – это мост, у которого каждый порт управляется собственным микропроцессором, за счет чего резко увеличивается производительность устройства. Коммутатор имеет встроенную память и может одновременно обрабатывать несколько кадров. Если по какой-то причине нужный порт в данный момент времени занят, то кадр помещается в память и ожидает своей очереди. Построенные с помощью коммутаторов сети могут охватывать большое число компьютеров и иметь протяженность в несколько километров.

**Маршрутизаторы** (Router) – устройства сети, работающие на сетевом уровне, и позволяющие переадресовывать и маршрутизировать пакеты из одной сети в другую, а также фильтровать широковещательные сообщения. Маршрутизация проводится на основе анализа сетевых IP-адресов, которые содержатся в каждом принятом пакете или дейтаграмме.

**Шлюзы** (Gateway) - программно-аппаратные комплексы, соединяющие разнородные сети или сетевые устройства. Шлюзы позволяет решать проблемы различия протоколов или систем адресации. Они действуют на сеансовом, представительском и прикладном уровнях модели OSI.

**Мультиплексоры** – это устройства центрального офиса, которое поддерживают несколько сотен цифровых абонентских линий. Мультиплексоры посылают и получают абонентские данные по низкоскоростным линиям связи, концентрируя весь трафик в одном высокоскоростном канале для передачи в Internet или в сеть компании.

**Межсетевые экраны** (firewall, брандмауэры) - это сетевые устройства, реализующие контроль за поступающей в локальную сеть и выходящей из нее информацией и обеспечивающие защиту локальной сети посредством фильтрации информации.

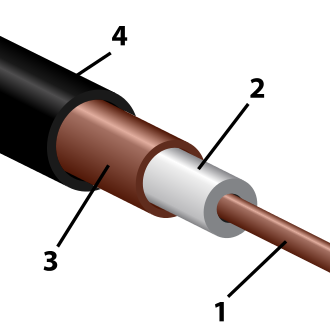
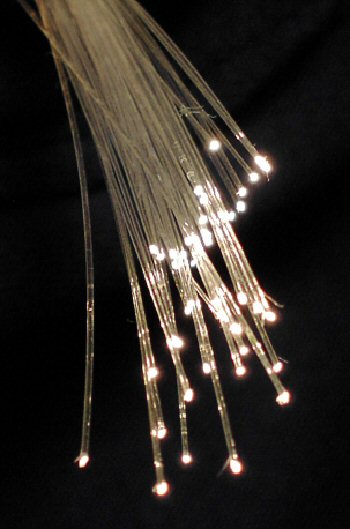
В таблице 1.2 приведены рабочие области основных коммуникационных устройств и их соответствие уровням модели OSI.

Таблица 1.2

|  |  |  |
| --- | --- | --- |
| Устройство | Рабочая область в сети | Уровень модели OSI |
| Повторитель | Физические сегменты | физический |
| Мост, коммутатор, сетевой адаптер | Логические сегменты | канальный |
| Маршрутизатор | Сети, подсети | сетевой |
| Шлюз | Интерсети | прикладной |

2.3 Физическая среда передачи

Физические среды можно разделить на два типа: проводные и беспроводные. Проводные среды передачи предполагают наличие твердотельного проводника и включают коаксиальный кабель, медную витую пару и оптоволоконный кабель (рис.1.1). В беспроводной среде передача осуществляется с применением радиоканалов и используется в беспроводных локальных сетях и в спутниковой связи.

****

а) б) в)

Рис.1.1

**Коаксиальный кабель** (рис.1.1а) состоит из двух медных проводников, расположенных вдоль оси кабеля (центральная жила и экранирующая оплетка). Такой кабель использовался в качестве разделяемой среды передачи данных в ранних виях технологии Ethernet (до 10 Мбит/с). В настоящее время в компьютерных сетях практически не используется и применяется в основном в телевизионных сетях и радиопередающих устройствах.

**Витая пара** (рис. 1.1б) является самым недорогим и наиболее популярным видом кабелей. Она представляет собой несколько пар изолированных проводников, скрученных между собой и покрытых пластиковой оболочкой. Скрутка позволяет снизить влияние электрических помех на передаваемые данные.

Качество кабеля с витой парой определяется его категорией, обозначаемой целым числом от 1 до 8, некоторые категории делятся на подкатегории, например, 5е или 6а. В кабелях выше шестой категорий все пары проводов или каждая пара в отдельности дополнительно экранируются оплетками из фольги для увеличения помехозащищенности. Кабели категории 8 могут передавать данные со скоростью до 40 Гбит/с.

В настоящее время для построения компьютерных сетей в основном применяется кабель категории 5е, обеспечивающий передачу данных со скоростью до 1 Гбит/с.

**Оптоволоконный кабель** (рис. 1.1в) представляет собой тонкий и гибкий кабель, внутри которого находится множество световых волокон (нитей), по которым передаются световые импульсы, несущие информацию о передаваемых битах. Достоинством такого кабеля является высокая пропускная способность (до 100 Гбит/с), высокая помехозащищенность, возможность передачи данных на большие расстояния (десятки километров без дополнительных усилителей, тысячи километров – с усилителями). Недостатком можно считать более высокую стоимость в сравнении с витой парой.

**Радиоканал** использует электромагнитное излучение радиодиапазона. Его основное достоинство – отсутствие кабельной системы, что дает возможность подключения к сети подвижных объектов. Характеристики радиоканала сильно зависят от состояния среды передачи радиоволн и расстояния между оконечными системами. Состояние среды передачи определяется затуханием сигнала вследствие прохождения через поглощающие предметы и взаимодействием с отраженными электромагнитными волнами, а также волнами, исходящими от других источников излучения.

Спутники связи организуют взаимодействие между двумя или более наземными станциями. Они принимают сигналы одного частотного диапазона, проводят их обработку и усиление, а затем передают сигналы в другом частотном диапазоне. Скорость обмена данными, обеспечиваемая спутниковыми каналами, составляет несколько гигабит в секунду. Существуют два типа спутников: геостационарные и низкоорбитальные.

2.4 Определение адресов и маршрутов передачи данных между узлами сети

Для определения адресов узлов сети и маршрутов передачи данных между ними необходимо использовать сетевые команды ОС, представленные в таблице 1.3

Таблица 1.3

|  |  |  |  |
| --- | --- | --- | --- |
| ОС | Команда | Пример | Назначение |
| Linux | ifconfig | ifconfig | Вывод характеристик сетевых интерфейсов текущего узла, включая IP- и MAC-адреса |
| traceroute имя\_узла | traceroute 192.168.130.5 | Вывод маршрута передачи пакетов от текущего узла сети к другому узлу. Для каждого промежуточного узла выводятся его символьное имя и IP-адрес. |
| Openscaler | ifconfig | ifconfig | Аналогично команде ifconfig |
| tracepath имя\_узла | tracepath kp.ru | Аналогично команде traceroute |
| Windows | ipconfig | ipconfig /all | Аналогично команде ifconfig |
| tracert имя\_узла | tracert 217.71.130.131 | Аналогично команде traceroute |

2.5 Локальная сеть факультета ФПМИ

Физически локальная сеть ФПМИ состоит из двух сегментов, соединенных между собой через маршрутизаторы, роль которых выполняют коммутаторы третьего уровня Cisco 3560G, каждый из которых имеет по 48 портов. Сегменты построен на коммутаторах Cisco 1800G, имеющих по 24 порта. К коммутаторам подключаются рабочие станции пользователей, расположенные в компьютерных классах, на кафедрах, в деканате и т.д.

В одном из сегментов имеется мощная вычислительная блейд-система HP c7000, состоящая из 16 блейд-серверов, а также 6 серверов HP Proliant DL350 G5, на которых реализован кластер виртуальных машин, используемых в учебном процессе (например, веб-серверы students.ami.nstu.ru, terminal.ami.nstu.ru и др.).

Структура локальной сети представлена на рис. 1.2.

1. Задание к лабораторной работе
2. Выполнить анализ структуры локальной сети факультета по следующим пунктам:
   * какие сетевые устройства используются в сети;
   * для каждого из устройств найти с Интернете и привести в отчете основные технические характеристики (количество портов, скорость передачи данных, пропускная способность, объем встроенный памяти, размер таблиц коммутации);
   * какие линии связи используются в локальной сети факультета;
   * схема соединения Вашего компьютера (ПК*i)* с сервером fpm2;
   * структура сетевого программного обеспечения согласно модели OSI на каждом узле схемы соединения ПК*i* с сервером fpm2.

2. Найти IP и MAC-адреса Вашего компьютера ПК*i* и сервера fpm2. Обратите внимание на то, что у одного компьютера могут быть несколько сетевых интерфейсов, каждый из которых имеет собственные параметры.

3. Выполнить трассировку маршрута передачи пакетов от ПК*i* до сервера fpm2 и в обратном направлении, найти количество промежуточных узлов и их IP-адреса. Поясните причину различия IP-адресов промежуточного узлов при прямой и обратной трассировках.

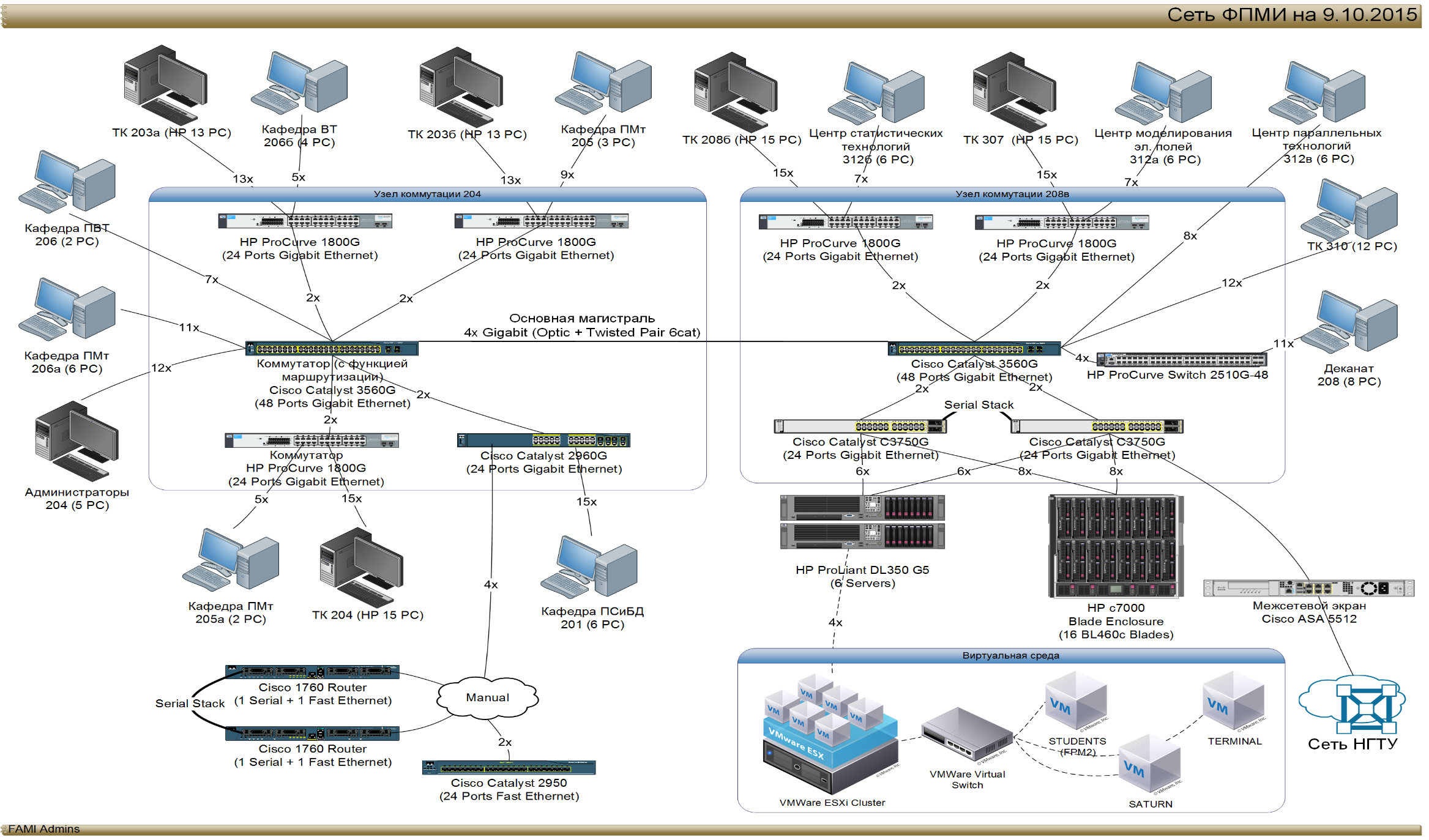


Рис. 1.2

### Контрольные вопросы

1. Дайте определения сетевого протокола. Зачем он нужен?
2. Какой стек протоколов используется в сети Internet?
3. В чем преимущества и недостатки многоуровневой организации системы?
4. Назовите виды единиц обмена информацией разных уровней в архитектуре протоколов.
5. Перечислите уровни стека протоколов Internet и назовите их основные функции.
6. Перечислите известные вам виды сетевого оборудования и объясните, для чего они используются.
7. Перечислите физические среды передачи данных и их особенности.
8. В чем состоит принцип взаимодействия прикладной программы с системным программным обеспечением.
9. На каком уровне модели OSI работают коммутатор, маршрутизатор и шлюз.
10. Назначение и принцип работы коммутатора.
11. Назначение и принцип работы маршрутизатора.

# Лабораторная работа № 2. Технология клиент-сервер: эхо-повтор.

1. Цель работы

Изучить основные принципы разработки клиент-серверных приложений на примере простейшей однопользовательской программы.

### Методические указания

2.1 Общие сведения

Модель, которая предусматривает, что прикладная программа должна пассивно ждать, пока другое приложение не инициирует связь, широко применяется в распределенных вычислениях и носит название – взаимодействие по принципу клиент/сервер. Приложение, которое активно инициирует контакт, называется клиентом, а приложение, которое пассивно ожидает контакта, называется сервером.

Большинство реализаций приложений клиент/серверной технологии характеризуются следующими общими особенностями:

а) клиентская программа:

* представляет собой произвольную прикладную программу, которая становится клиентом на время, когда ей требуется удаленный доступ, но выполняет также другие локальные действия;
* вызывается непосредственно пользователем и действует на протяжении только одного сеанса;
* функционирует локально на персональном компьютере пользователя;
* активно инициирует контакт с сервером;
* может обращаться по мере необходимости к нескольким службам, но в определенный момент времени активно контактирует только с одним удаленным сервером;
* не требует специальных аппаратных средств или сложной операционной системы.

б) серверная программа:

* представляет собой программу специального назначения, которая выделена для предоставления одной службы, но может обслуживать нескольких удаленных клиентов одновременно;
* вызывается автоматически во время начальной загрузки системы и продолжает работать, проводя один сеанс взаимодействия за другим;
* выполняется на компьютере, предоставленном в общее пользование;
* пассивно ожидает поступления запросов на установление соединения от удаленных клиентов;
* принимает запросы от клиентов, но предоставляет единственную службу;
* требует применения мощных аппаратных средств и сложной операционной системы.

В процессе обмена данными между большинством приложений Internet выполняется одна и та же последовательность операций.

* запускается на выполнение серверное приложение, которое ожидает от клиента запроса на установление соединения;
* клиент обращается к серверу, указывая его местонахождение и передавая требование приступить к обмену данными;
* клиент и сервер обмениваются сообщениями;
* после завершения передачи данных и клиент, и сервер сообщают о том, что достигнут конец файла, чтобы прекратить обмен данными.

Общая схема взаимодействия клиента и сервера приведена на рис. 2.1. Взаимодействие на транспортном уровне может происходить с использованием различных протоколов. Наиболее распространенными протоколами являются TCP и UDP.

Протокол TCP является протоколом, поддерживающим надежную передачу потока данных с предварительным установлением связи между источником информации и ее получателем. На его основе реализованы такие протоколы уровня приложений, как Telnet, FTP, HTTP. Протокол характеризуется следующими особенностями:

* перед фактической передачей данных необходимо установление связи, т.е. отправка запроса на начало сеанса передачи данных источником и подтверждение его получателем;

Ожидание запроса на соединение от клиента

Установление соединения с клиентом

Прием данных от клиента

Обработка полученных данных

Пересылка данных клиенту

Разрыв соединения с клиентом

Передача серверу запроса на соединение

Пересылка данных серверу

Прием данных от сервера

Отправка сообщения об окончании приема данных

**Сервер**

**Клиент**

Рис.2.1 Взаимодействие клиента и сервера

* после обмена данными сеанс передачи должен быть явно завершен;
* доставка информации является надежной, не допускающей дублирования или нарушения очередности получения данных;
* возможность управления потоком данных для исключения переполнения и затора.

Эти возможности позволяют протоколам верхнего уровня и приложениям, их реализующим, не заботиться о надежности и последовательности доставки данных. Поэтому протоколы приложений, использующих TCP, могут быть значительно упрощены. Но это, в свою очередь, ведет к усложнению самого протокола TCP и к возрастанию накладных расходов при передаче данных.

Протокол UDP обеспечивает логический коммуникационный канал между источником и получателем данных без предварительного установления связи. Для передачи датаграмм UDP используют протокол IP, который также не обеспечивает надежность передачи данных. Поэтому приложения, использующие данный протокол должны самостоятельно отслеживать надежность доставки, например, путем обмена подтверждениями и повторной передачей недоставленных сообщений.

Протокол UDP используют такие протоколы уровня приложений, как протокол взаимодействия с сервером доменных имен DNS, протокол удаленного копирования Trivial FTP, удаленный вызов процедур RPC и другие. Передача данных с помощью UDP вносит гораздо меньшие накладные расходы по сравнению с протоколом TCP.

Для организации взаимодействия сетевые приложения используют специальные объекты – сокеты, представляющие собой конечные точки логического соединения клиента и сервера. Сокеты работают на транспортном уровне модели OSI и позволяют скрыть от программиста все проблемы, решаемые на сетевом, канальном и физическом уровнях. Достаточно только создать сокет, правильно установить его параметры и корректно применять все методы, предоставляемые специальным API-интерфейсом (например, WinSock32).

Основными параметрами сокета являются IP-адрес и номер порта, причем для стандартных протоколов Internet зарезервированы номера портов в диапазоне (0 – 1023), а для разработчиков доступны номера, начиная с 1024. Комбинация IP адреса и номера порта однозначно определяет отдельный сетевой процесс во всей глобальной сети Internet.

Для создания соединения TCP/IP необходимы два сокета – один на локальной машине, а другой на удаленной. Таким образом, каждое сетевое соединение идентифицируется IP-адресом и портом на локальной машине, IP-адресом и портом на удаленной машине.

Работа с сокетами может проводиться на основе библиотек высокого или низкого уровней. Применение библиотек высокого уровня существенно упрощает работу программиста, но снижает уровень понимания всех этапов организации сетевого взаимодействия процессов. В лабораторной работе рекомендуется использовать API-интерфейс низкого уровня.

2.2 Функции для работы с сокетами

2.2.1 Функция **socket**

Функция socket создает объект типа сокет и возвращает его дескриптор:

s = socket(af, type, [protocol]);

Параметр *af* задает имя коммуникационного домена, используемого сокетом, и может принимать значения AF\_INET для домена Internet или AF\_UNIX для домена UNIX. Коммуникационный домен определяет систему адресации соединений: для домена Internet адрес записывается в виде IP-адреса и номера порта, а домен UNIX использует файловую систему как адресное пространство имен для организации межпроцессного обмена данными на локальном компьютере. Далее будем рассматривать только сокеты, работающие в домене Internet.

Параметр *type*, указывает тип связи, который будет использоваться в сокете. Наиболее распространенными типами являются потоковая передача с установлением логического соединения (значение SOCK\_STREAM) и блочная передача без установления логического соединения (значение SOCK\_DGRAM).

Необязательный параметр *protocol* определяет тип транспортного протокола и для Internet-домена может принимать значения IPPROTO\_TCP или IPPROTO\_UDP.

2.2.2 Функция **bind**

Сразу после создания сокет ассоциируется с некоторым семейством адресов, но не с конкретным адресом. Под адресом здесь и далее мы будем понимать адресную информацию, принятую в коммуникационном домене. Для сокетов, работающих в Internet-домене, адрес представляется парой (*host, port*), где *host* указывает IP-адрес хоста, а *port* – номер порта. Для UNIX-домена адрес указывается в виде константы SOCKET\_FILE.

Для привязки сокета к конкретному адресу применяется функция bind:

retVal = bind(s, (LPSOCKADDR)&sin, sizeof(sin));

Здесь s – дескриптор сокета, (LPSOCKADDR)&sin – адрес структуры sin, в которой задается конкретный адрес привязки, sizeof(sin) – размер структуры sin.

Структура имеет тип SOCKADDR\_IN и описывается следующим образом:

SOCKADDR\_IN sin;

sin.sin\_family = AF\_INET;

sin.sin\_port = htons(port);

sin.sin\_addr.s\_addr = host;

Имя хоста может задаваться в виде доменного имени (например, ‘localhost’) или в виде IP-адреса.

2.2.3 Функция **listen**

После привязки сокета к адресу сервер переводится в режим прослушивания порта и ожидания запросов на установление соединения от клиентов. Для этого сервер вызывает функция listen, принимающая два параметра – дескриптор сокета и максимальный размер очереди запросов на соединение:

retVal = listen(s, queuesize);

2.2.4 Функция **accept**

Обработка очереди запросов проводится функцией accept:

s\_client = accept(s, (struct sockaddr\*)&from, &fromlen)

Здесь s – дескриптор слушающего сокета, (struct sockaddr\*)&from – структура, содержащая адресную информацию о клиенте, &fromlen – размер структуры from. Функция создает новый сокет, через который будет проводиться обмен данными с клиентом и возвращает его дескриптор s\_client. При этом слушающий сокет сервера по прежнему работает в режиме прослушивания порта для получения новых запросов на соединение от клиентов.

2.2.5 Функция **connect**

В клиентских программах для установления соединения с сервером применяется функция connect, которой передаются следующие параметры: clientSock – дескриптор сокета клиента, (LPSOCKADDR)&serverInfo – структура, аналогичная описанной в разделе 2.2.2 и содержащая адрес сервера, sizeof(serverInfo) – размер этой структуры:

retVal=connect(clientSock,(LPSOCKADDR)&serverInfo, sizeof(serverInfo))

Если сокет не был связан с адресом, то connect автоматически вызовет системную функцию bind.

2.2.6 Функции **send** и **sendto**

Функции send и sendto используются для передачи данных. Если сокет подключен, для передачи данных применяется функция send, которая имеет следующие параметры: s – дескриптор сокета, через который проводится передача; pBuf и strlen(pBuf) – имя и размер буфера, содержащего данные, flags – список специальных констант, с помощью которых можно запрашивать специальные опции.

retVal = send(s, pBuf, strlen(pBuf), flags)

Функция sendto позволяет передавать сообщение с использованием неподключенного сокета. Параметры этой функции совпадают с параметрами функции send, к которым добавляется адрес получателя:

rv = sendto(s,pBuf,strlen(pBuf),flags,(LPSOCKADDR)&serverInfo, sizeof(serverInfo))

Обе функции возвращают количество переданных байтов.

2.2.7 Функции **recv** и **recvfrom**

Функции recv и recvfrom используются для приема данных. Для приема данных из подключенного сокета используется функция recv:

retVal = recv(s, pBuf, strlen(pBuf), flags)

Здесь s – дескриптор сокета, из которого читаются данные; pBuf и strlen(pBuf) – имя и размер буфера для приема данных, flags – список специальных констант, с помощью которых можно запрашивать специальные опции.

Функция recvfrom дает возможность прочитать данные из сокета с указанным адресом

rv = recvfrom (s,pBuf,strlen(pBuf),flags,(LPSOCKADDR)&serverInfo, sizeof(serverInfo))

Необязательный параметр flags аналогичен этому же параметру методов send и sendto. Обе функции возвращают количество принятых байтов.

2.2.8 Функция **close**

Функция close сообщает операционной системе, что программа завершила использование сокета. Вызов имеет форму:

closeSocket(s)

Схема взаимодействия сервера с клиентом через интерфейс сокетов приведена на рисунке 2.2.

### Задание к лабораторной работе

3.1 Общие требования

Написать простейшее приложение с одним сервером и одним клиентом, используя API-интерфейс низкого уровня. Пример реализации интерфейса приведен в приложении 1.

Запуск сервера происходит с указанием номера порта протокола TCP или UDP из диапазона возможных номеров. Каждая бригада создает свой сервер с номером порта, определяемым по правилу: номер бригады + 2000. Сервер начинает свою работу с ожидания запроса от клиента на соединение.

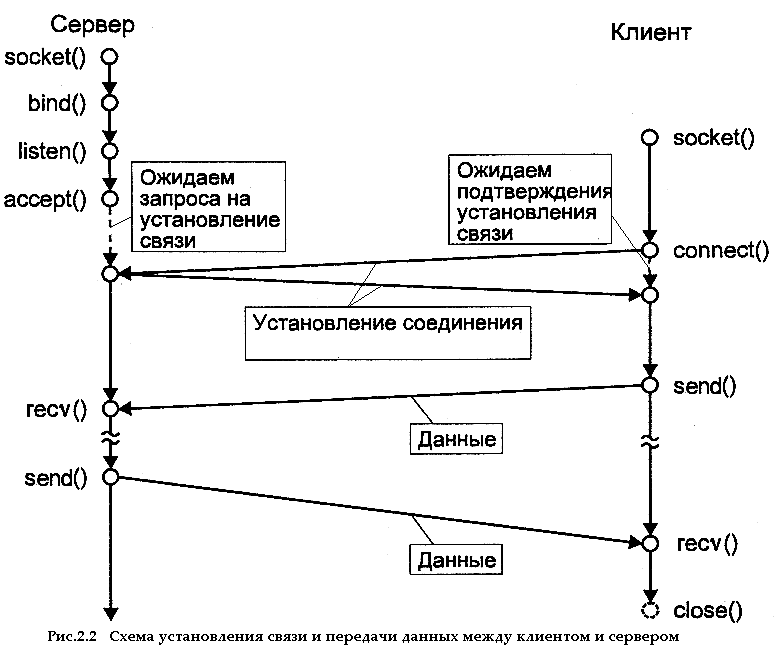
****

Рис. 2.2 Взаимодействие сервера с клиентом через интерфейс сокетов

Запуск клиента происходит с указанием ему IP-адреса и номера порта сервера. Клиент устанавливает связь с сервером и посылает набор данных, введенный пользователем, на сервер. Сервер получает набор данных, выполняет указанные в варианте действия, возвращает клиенту результат и снова переходит в состояние ожидания запроса на соединение. Клиент, получив ответ с сервера, выводит его на экран и прекращает свою работу.

Тестирование совместной работы сервера и клиента необходимо провести в режиме удаленного доступа. Если Ваш компьютер не подключен к локальной сети, то можно создать виртуальную сеть на основе бесплатной VPN-технологии Hamachi, которая является надстройкой над сетью Интернет и может использоваться для объединения в единую защищенную сеть компьютеров с серыми IP-адресами. Дистрибутив Hamachi можно скачать по адресу <http://hamachi-pc.ru/>

3.2 Варианты заданий

1. Клиент пересылает серверу данные (строки текста). Сервер возвращает клиенту полученные данные, включив в конец каждого предложения количество символов в нем.
2. Клиент пересылает серверу данные (строки текста). Сервер изменяет порядок следования букв в полученном тексте на обратный и отправляет текст в таком виде клиенту.
3. Клиент пересылает серверу данные (строки текста). Сервер в полученном тексте в конец каждого предложения вставляет свой IP-адрес и номер порта и возвращает в таком виде данные клиенту.
4. Клиент пересылает серверу данные (строки текста). Сервер создает файл с уникальным именем, записывает в него полученные от клиента данные и в качестве результата обработки данных отправляет клиенту имя созданного файла. После получения ответа с сервера клиент выводит на экран содержимое указанного сервером файла.
5. Клиент пересылает серверу данные (строку и имя директории). Сервер находит все файлы в заданной директории, содержащие указанную строку, и высылает их имена клиенту.
6. Клиент пересылает серверу имя некоторого файла. Сервер находит файл с указанным именем и пересылает его содержимое клиенту, либо сообщает клиенту, что файл с данным именем не найден.
7. Клиент пересылает серверу данные (имя директории). Сервер возвращает список файлов и поддиректорий данной директории (рекурсивно).
8. Клиент пересылает серверу два числа. Сервер возвращает сумму полученных чисел.

### Контрольные вопросы

1. Что представляет собой модель клиент-сервер?
2. Приведите общие особенности клиентских программ.
3. Приведите общие особенности серверных программ.
4. Приведите общую схему клиент-серверного взаимодействия.
5. В чем отличие взаимодействия клиента с сервером при использовании протоколов TCP и UDP? В чем преимущества и недостатки каждого из протоколов?
6. Что такое сокет? Какие виды сокетов вам известны?
7. Приведите схему взаимодействия клиента с сервером при использовании механизма сокетов
8. Основные этапы работы с сокетом на стороне сервера.
9. Основные этапы работы с сокетом на стороне клиента.

# Лабораторная работа № 3. Разработка приложения интерактивной переписки

* 1. Цель работы

Изучить основные принципы разработки многопользовательских приложений, построенных на основе технологии клиент-сервер с использованием стека протоколов TCP/IP.

### Методические указания

В лабораторной работе № 2 было реализовано простое удаленное взаимодействие, в котором участвовали один клиент и один сервер. В данной лабораторной работе необходимо изменить программы таким образом, чтобы сервер мог осуществлять взаимодействие с несколькими клиентами одновременно.

2.1 Схема работы chat-сервера и chat-клиента

Работа *сервера* начинается с перехода в состояние ожидания запроса на установление соединения от клиента. Затем сервер входит в цикл, в котором он получает и отображает строки текста от клиентов.

Работа *клиентской программы* начинается с передачи серверу запроса на установление соединения. После установления соединения клиент также входит в цикл. При каждом проходе по циклу клиент выдает локальному пользователю приглашение к вводу строки текста, считывает строку, введенную с клавиатуры, отправляет ее на сервер, а затем получает и отображает строку текста, полученную с сервера. Схема взаимодействия Chat-сервера с клиентами приведена на рис.3.1.

2.2 Основные сведения о потоках выполнения

Сервер должен обслуживать одновременно несколько клиентов, создавая для каждого из них отдельный сокет. При этом возможны два варианта обработки поступающих на сервер запросов:

* применить неблокирующие сокеты и читать в цикле данные из каждого сокета по очереди,
* для каждого клиента запускать отдельный поток выполнения, который будет заниматься обменом сообщениями с клиентом.

С точки зрения производительности сервера более предпочтительным является второй вариант, потому рассмотрим основные сведения о многопоточном режиме выполнения программ.

Любой процесс, создаваемый операционной системой (ОС), представляет собой набор команд программы, запускаемой пользователем, и метаданных, представленных набором выделенных системных ресурсов и общих сведений о запускаемой программе. В качестве ресурсов выступают адресное пространство, глобальные переменные и открытые файлы. Общими сведениями являются идентификаторы текущего, родительского и дочерних процессов, а также идентификаторы владельца процесса и группы, членом которой является владелец.

Поток выполнения – это поименованный набор команд программы, который является минимальной единицей управления для ядра ОС и создается в рамках процесса. Многозадачный режим обработки программ ОС реализует путем переключения процессора между готовыми к выполнению потоками.

В однопоточном приложении есть только один поток выполнения команд и при запуске программы все команды исполняются последовательно с учетом возможных ветвлений и циклов. В этом случае поток выполнения непрерывен в течение всего времени выполнения программы и завершается только по завершении программы.

В многопоточных приложениях возможна организация нескольких потоков выполнения, которые будут исполняться процессором параллельно. Программной единицей для организации отдельного потока является функция, т.е. основной поток будет как обычно представлен функцией **main()**, а в качестве других потоков могут выступать другие функции, описанные в программе.

Любой поток выполнения, работающий в многопоточном приложении, имеет доступ ко всем ресурсам, выделенным процессу. Например, несколько потоков могут пытаться одновременно изменить содержимое одного участка памяти. Поэтому при разработке многопоточных приложений особое внимание необходимо уделять синхронизации потоков при работе с общими ресурсами, для чего наиболее удобным является применение семафоров.

Семафоры позволяют исключить выполнение одного и того же участка кода программы несколькими потоками одновременно. Самый простой двоичный семафор – замок (lock) или мутекс. Чтобы поток мог продолжить исполнение кода, он должен сначала получить замок и закрыть его. После этого он выполняет некоторый участок кода и открывает замок, чтобы другой поток (возможно, уже сделавший запрос на данный замок) мог его получить и пройти дальше к выполнению охраняемого замком участка программы.

Необходимые для работы с потоками функции содержатся в модулях **thread** и **threading**.

Передача серверу запроса на соединение

Чтение данных, введенных пользователем

Пересылка данных серверу

Прием данных от сервера

Отправка сообщения об окончании приема данных

**Клиент**

Ожидание запроса на соединение от клиента

Установление соединения с клиентом

Прием данных от клиентов

Пересылка данных клиентам

Разрыв соединения с клиентом

**Сервер**

Передача серверу запроса на соединение

Чтение данных, введенных пользователем

Пересылка данных серверу

Прием данных от сервера

Отправка сообщения об окончании приема данных

**Клиент**

Рис.3.1 Схема взаимодействия Chat-сервера с клиентами

2.3 Управление потоками

Объект **Thread** позволяет назначить действия, которые должны выполняться в отдельном потоке, и имеет следующий конструктор:

threading.Thread(group,target,name,args=(),kwargs={}) -> object

где group – группа потоков, target – функция, метод или другой объект, позволяющий вызов, именно он вызывается при запуске потока, name – имя потока, args и kwargs – соответственно позиционные и именованные аргументы для вызова потока.

Приведем некоторые методы объектов класса **threading.Thread**. Метод **start()** запускает поток, он не имеет параметров и ничего не возвращает. Метод **run()** используется для вызова в отдельном потоке. Получает свое значение из параметра **target** конструктора класса. При вызове используются **args** и **kwargs**, заданные в конструкторе.

Метод **join([time])** ожидает завершения потока. Поток, который вызывает этот метод, приостанавливается. Значение **time** задает время ожидания, после которого приостановленный поток продолжает свою работу. Метод **getName()** возвращает имя потока, метод **setName(name)** устанавливает имя потока. Метод **isAlive()** возвращает значение `истина`, если поток работает.

Класс **threading.Lock** – простейший замок, который имеет два состояния (он может быть либо открыт, либо заперт). Объект **Lock** имеет два основных метода – **acquire(),** с помощью которого поток делает запрос на запирание замка, и **release()**, используемого для снятия замка. Метод **locked()** возвращает статус замка: 0 – свободен, 1 – занят.

С помощью этого вида замка можно обеспечить выполнение некоторого участка программы одновременно только одним потоком. Еще раз запросить закрытый замок не может даже сам процесс, который его до этого запер: это приводит к бесконечному ожиданию. Однако, отпереть замок может любой процесс.

В некоторых случаях (например, в рекурсивных функциях) необходимо, чтобы один и тот же поток имел возможность запрашивать замок даже в случае, если он им уже обладает. Объект класса **threading.RLock** имеет эту возможность, и потому может сколько угодно раз запрашивать замок методом **acquire()** и столько же раз снимать методом **release().** Первоначально замок находится в открытом состоянии и не принадлежит ни одному потоку. Замок возвращается в исходное незапертое состояние только после одинакового числа запросов и освобождений. При этом все остальные потоки, которые запрашивают замок, остаются в состоянии ожидания.

Семафоры представлены классом **threading.Semaphore** и являются более общий механизмом синхронизации потоков, чем замки. С их помощью в критическую часть программы допускаются несколько потоков. Семафор ведет счетчик запросов, который при каждом запросе **acquire()** уменьшается на единицу, а при каждом **release()** – увеличивается на единицу. Счетчик не может стать меньше нуля, поэтому, если запрос поступает, когда счетчик равен нулю, потоку приходится ждать, как и в случае с замками, пока один из потоков не увеличит счетчик.

Объекты класса **threading.Event** служат для простейшей коммуникации между потоками, при которой один поток сигнализирует о событии, тогда как другие находятся в состоянии ожидания. Объекты события имеют внутренний флаг, который может быть установлен или сброшен. При своем создании флаг находится в сброшенном состоянии. Если флаг установлен, ожидания не происходит: соответствующий поток продолжает свою работу.

Модуль **thread** предоставляет низкоуровневый доступ к потокам выполнения. В этом модуле доступны следующие функции.

Создание нового потока для исполнения функции **function** с кортежем аргументов **args** и словарем **kwargs** выполняется функцией **thread.start\_new (function,args[,kwargs]).** Поток завершается по возврату из функции.

Функция **thread.exit()** возбуждает исключение **SystemExit,** которое, если оказалось неперехваченным, завершает исполняющийся поток.

Функция **get\_ident() -> id** возвращает идентификатор потока, в котором исполняется эта функция.

Вызовом функции **thread.allocate\_lock() -> lockobj** осуществляется создание объекта замка. На основе этого замка создан класс **threading.Lock**, поэтому методы для работы с этими объектами одинаковы.

### Задание к лабораторной работе

С помощью API-интерфейса реализовать простой чат**.** Каждая бригада должна написать chat-сервер и chat-клиента. Сервер должен поддерживать соединение сразу от нескольких клиентов. Обмен между клиентами осуществляется через сервер. При получении сообщения от какого-либо клиента, сервер дублирует его на своем экране и оповещает всех подсоединенных клиентов, отправляя каждому из них данное сообщение. При подсоединении нового клиента к chat-серверу, сервер оповещает каждого клиента о новом пользователе, посылая им его IP-адрес, номер порта и имя. При отключении также необходимо сформировать оповещение.

Тестирование сервера провести в режиме удаленного доступа при одновременном подключении не менее двух клиентов, загруженных на разных компьютерах. Также необходимо проверить корректность работы клиента, осуществляющего ввод данных, при одновременной отправке данных другим клиентом.

### Контрольные вопросы

1. Приведите схему взаимодействия chat-сервера и chat-клиента.
2. Что такое потоки выполнения, для чего они нужны и как они работают?
3. Какие проблемы возникают при использовании потоков и как эти проблемы решаются ?
4. Опишите классы модуля threading.
5. Какие виды замков существуют ? Расскажите о преимуществах и недостатках каждого из них.
6. Что такое семафор, в чем его отличие от замка ?
7. Как в вашей программе решается проблема учета сведений о клиентах, подключенных к серверу ?

# Лабораторная работа № 4. Анализ структуры кадра/фрейма технологии Ethernet

1. Цель работы

Спроектировать и реализовать программу, выполняющую анализ структуры кадра/фрейма технологии Ethernet.

### Методические указания

2.1 Передача фреймов по сети

При передаче фрейма по сети Ethernet электрические сигналы, несущие биты, достигают всех станций сегмента локальной сети. Сетевые интерфейсные аппаратные средства этих станции обнаруживают электрический сигнал и могут принять копию фрейма.

В локальных сетях разного типа для обеспечения непосредственной связи применяется та или иная ***схема адресации***. Каждой станции локальной сети присваивается уникальное числовое значение, называемое *физическим адресом*, *аппаратным адресом*, или *адресом доступа к передающей среде* (MAC — Media Access Control). При передаче фрейма по локальной сети отправитель включает во фрейм аппаратный адрес намеченного получателя. Аппаратное обеспечение каждой станции проверяет адрес каждого входящего фрейма, чтобы определить, должна ли станция принять этот фрейм.

Фрейм, передаваемый по разделяемой локальной сети, включает два адреса; один из них указывает получателя, а другой — отправителя. Каждый фрейм начинается с заголовка фиксированного формата, в котором отведено место для двух адресов. Участки заголовка, зарезервированные для адресов, принято называть ***полями*.** При передаче фрейма отправитель должен поместить физический адрес получателя в поле адреса назначения, а свой адрес — в поле адреса источника. Благодаря наличию адреса отправителя в каждом фрейме упрощается задача подготовки ответа получателем. Сетевые интерфейсные аппаратные средства обеспечивают проверку полей адресов в проходящих по сети фреймах и прием только тех фреймов, в которых адрес назначения совпадает с адресом станции.

2.2 Способы адресации

Способы адресации можно разбить на следующие категории:

* Статическая адресация
* Настраиваемая адресация
* Динамическая адресация

**В *статической схеме*** физические адреса назначаются изготовителем аппаратных средств. Статический физический адрес станции не меняется до тех пор, пока на ней не произойдет замена аппаратных средств.

Схема ***настраиваемой адресации*** позволяет заказчику устанавливать физический адрес. Этот механизм может предусматривать ручную настройку (т.е. перестановку переключателей при инсталляции интерфейса) или электронную настройку. Большинство аппаратных средств, требует настройки конфигурации только один раз; настройка обычно выполняется при первоначальной установке аппаратных средств.

Схема ***динамической адресации*** позволяет автоматически присваивать станции физический адрес сразу после ее загрузки. Большинство схем динамической адресации требует, чтобы станция случайным образом проверяла номера до тех пор, пока не будет найдено значение, не используемое другими компьютерами в качестве адреса. Например, станцией может быть выбрано в качестве начального значения адреса текущее время суток. После генерации случайного числа станция посылает по сети сообщение с соответствующим адресом. Если этот адрес уже используется каким-либо компьютером, то он отвечает на это сообщение. Если же никакая станция не ответит на отправленный запрос, отправитель может использовать адрес в качестве своего физического адреса. Таким образом, адрес, выбранный компьютером, зависит от того, какие адреса применяются другими компьютерами ко времени его загрузки: при каждом последующем перезапуске компьютер может получать другой адрес.

К основным ***преимуществам статической адресации*** относится удобство использования и постоянство. Эту схему легко применять, поскольку поставщики аппаратных средств назначают адреса и гарантируют, что каждое аппаратное устройство имеет физический адрес, который больше нигде в мире не повторится. Аппаратные устройства различных изготовителей можно соединять в одну физическую сеть, не опасаясь возникновения конфликтов адресов. Статическая адресация является постоянной, поскольку адрес компьютера не меняется после каждой перезагрузки.

Использование ***динамической адресации*** имеет два преимущества: для изготовителей аппаратных средств отпадает необходимость согласовывать друг с другом назначаемые адреса, а для сетевых администраторов появляется возможность уменьшить размер адресов. Адрес может стать короче потому, что он должен быть уникальным только в одной локальной сети. Динамическая схема адресации позволяет станциям в одной локальной сети выбирать такие же адреса, как и у станций другой локальной сети. Основным недостатком динамической адресации является отсутствие постоянства и возможность конфликта адресов. При каждой начальной загрузке компьютер получает новый адрес; другие компьютеры должны узнать этот новый адрес, прежде чем они смогут с ним связаться. Более того, если работа сети при загрузке компьютера временно нарушена, то два компьютера могут выбрать один и тот же физический адрес.

***Настраиваемые адреса*** могут служить компромиссом между статической и динамической схемами. Как и статические, настраиваемые адреса являются постоянными: адрес компьютера остается тем же после каждой перезагрузки. Как и динамические, настраиваемые адреса не должны быть слишком большими, поскольку адрес является уникальным только в конкретной сети. На практике большинство сетевых администраторов предпочитает присваивать последовательные значения настраиваемых адресов. Первый компьютер, установленный в сети, получает адрес 1, второму присваивается адрес 2 и т.д. Одно из преимуществ настраиваемой схемы адресации становится очевидным при замене сетевых интерфейсных аппаратных средств: в отличие от аппаратных средств, в которых используется статически присвоенный адрес, настраиваемый интерфейс может быть заменен без смены физического адреса компьютера.

2.3 Широковещательная рассылка

Во многих приложениях, использующих сеть, применяется метод, называемый ***широковещательной рассылкой***. Этот термин, который первоначально распространялся на радио- и телевизионные передачи, обозначает такие передачи, которые доступны широкому кругу пользователей. Если приложение выполняет широковещательную рассылку данных, оно предоставляет копию данных в распоряжение всех других компьютеров в сети.

Для обеспечения широковещательной рассылки в локальных сетях используется расширенная схема адресации. Проектировщики сети предусматривают не только присвоение адреса каждому компьютеру, но и назначают специальный зарезервированный адрес, называемый ***широковещательным адресом***. Аппаратный интерфейс компьютера настроен на распознавание не только физического, адреса станции, но и специального широковещательного адреса. Получив фрейм с одним из этих адресов в поле адреса назначения, интерфейс принимает фрейм и передает его копию операционной системе компьютера.

* 1. Групповая рассылка

***Групповая рассылка*** – это ограниченная форма широковещательной рассылки, преимуществом которой является то, что в ней для исследования фреймов используются сетевые интерфейсные аппаратные средства. В отличие от широковещательного, групповой фрейм не перенаправляется автоматически операционной системе (сетевым протоколам, лежащим выше канального уровня). В интерфейсных аппаратных средствах программируются критерии, в соответствии с которыми они принимают одни групповые фреймы и отбрасывают другие. Решение принимают интерфейсные аппаратные средства и передают выше по стеку только фреймы, соответствующие критериям.

Групповая рассылка предусматривает расширение схемы адресации путем резервирования адресов для групповой рассылки. Во время начальной загрузки компьютера интерфейс программируется на распознавание только адреса компьютера и широковещательного адреса. Если работающее на компьютере приложение должно получать групповые фреймы, оно обязано указать сетевому интерфейсу используемый адрес групповой рассылки. Интерфейс добавляет этот адрес к набору распознаваемых адресов и начинает принимать фреймы, отправленные по этому адресу.

* 1. Определение содержимого фрейма

Хотя рассматриваемые выше схемы адресации позволяют отправителю указать получателя кадра, используемый при этом адрес не позволяет дать описание содержимого фрейма. Поскольку многие элементы данных имеют одинаковое представление, получатель не может использовать данные кадра, чтобы определить его содержимое. Например, во всех пакетах, которые инкапсулированы во фреймы и содержат сообщения электронной почты, текстовые файлы и Web-страницы, для представления данных используется кодировка ASCII. Для передачи получателю сведения о его содержимом каждый фрейм включает дополнительную информацию, которая указывает тип его информационного наполнения. Для обозначения содержимого фрейма используются следующие методы.

• ***Явное обозначение типа фрейма*.** Проектировщики сетевого аппаратного обеспечения указывают, как должна быть включена во фрейм информация типа и какие значения должны использоваться для описания фреймов различных типов. Участок фрейма, используемый для описания его содержимого, называют полем типа фрейма, а сам фрейм — автоматически распознаваемым.

• ***Неявное обозначение типа фрейма*.** При использовании этого метода сетевые аппаратные средства не включают поле типа в каждый фрейм. Фрейм содержит только данные. Поэтому отправитель и получатель должны согласовывать между собой содержимое каждого фрейма или предусматривать использование части данных фрейма в качестве поля типа.

2.6 Заголовки фрейма и его формат

В локальной сети обычно определен точный формат применяемого в ней фрейма. Хотя отдельные детали могут отличаться, в большинстве сетевых технологий фрейм состоит из двух частей: заголовка фрейма, содержащего адрес источника и адрес назначения, за которым следует основная часть, называемая телом фрейма, или областью данных, содержащая передаваемую информацию. Общий формат фрейма показан на рис. 4.1.

Заголовок фрейма

Область данных фрейма,

или тело фрейма

Рис. 4.1. Общий формат фрейма

В большинстве сетевых технологий каждое поле заголовка фрейма имеет постоянный размер и расположение. В результате все фреймы имеют одинаковый размер заголовка. В отличие от этого, область данных фрейма не имеет постоянного размера: размер области данных определяется объемом передаваемых данных.

2.7 Формат фрейма Ethernet

Фрейм Ethernet (рис. 4.2) начинается с заголовка, который содержит три поля. 64-битовая начальная серия (преамбула), которая предшествует фрейму, содержит чередующиеся биты 1 и 0, позволяющие аппаратным средствам получателя синхронизировать работу в соответствии с входящим сигналом. Первые два поля заголовка содержат физические адреса. В сети Ethernet используется 48-битовая статическая схема адресации, в которой каждому сетевому интерфейсу (сетевой плате) изготовителем назначается уникальный адрес. Поле, обозначенное как адрес назначения, содержит физический адрес станции, на которую отправлен данный фрейм. Поле, обозначенное как адрес источника, содержит физический адрес станции — отправителя фрейма. Третье поле заголовка содержит 16-битовое обозначение ***типа фрейма Ethernet или его длину.***

8

6

6

2

46 – 1500

4

Преамбула

Адрес назначения

Адрес источника

Тип фрейма

Данные во фрейме

CRC

Рис. 4.2. Формат фрейма, используемого в сети Ethernet. Число, приведенное в каждом поле, обозначает размер поля в восьмибитовых октетах

Фрейм DIX (Digital-Intel-Xerox) сети Ethernet определяет значения, которые могут применяться в полях заголовка, и их смысл: Например, в стандарте указано, что адрес со всеми 48 битами, установленными в значение 1, зарезервирован для широковещательной рассылки, прочие адреса, которые начинаются с бита 1, используются для групповой рассылки, а шестнадцатеричное значение 8137 в поле типа фрейма указывает, что данные во фрейме соответствуют протоколу корпорации Novell, известному под названием IPX. В таблице 4.1 приведены некоторые примеры значений типов Ethernet.

Таблица 4.1

|  |  |
| --- | --- |
| Значение | Описание |
| 0000-05DC | Зарезервировано для использования во фреймах LLC/SNAP IEEE |
| 0800 | Версия 4 протокола IP Internet |
| 0805 | Протокол Х.25 CCITT |
| 0806 | Протокол ARP |
| 0900 | Сетевой отладчик корпорации Ungermann-Bass |
| 0BAD | Протокол VINES корпорации Banyan Systems |
| 1000-100F | Инкапсуляция концевика пакета Berkeley UNIX |
| 6004 | Протокол LAT корпорации Digital Equipment |
| 6559 | Протокол Frame Relay |
| 8005 | Сетевой зонд корпорации Hewlett Packard |
| 8008 | Зарезервировано корпорацией AT&T |
| 8014 | Сетевые игры корпорации Silicon Graphics |
| 8035 | Реверсивный протокол ARP (RARP) |
| 8038 | Оборудование LANBridge корпорации Digital Equipment |
| 805С | Ядро версии V Станфордского университета |
| 809В | Протокол AppleTalk корпорации Apple Computer |
| 80С4-80С5 | Зарезервировано корпорацией Banyan Systems |
| 80D5 | Протокол SNA корпорации IBM |
| 80FF-8103 | Зарезервировано компанией Wellfleet Communications |
| 8137-8138 | Протокол IPX корпорации Novell |
| 818D | Зарезервировано корпорацией Motorola |
| FFFF | Зарезервировано |

Как показано в таблице, некоторые типы Ethernet предназначены для использования в системах, разработанных отдельными компаниями, а другие применяются в программном обеспечении, соответствующем международным стандартам, таким как Х.25. Наличие стандартизированных обозначений типов гарантирует использование во всех продуктах Ethernet одного и того же значения для конкретного типа фрейма. Поэтому продукты Ethernet одного и того же типа, создаваемые несколькими поставщиками, могут взаимодействовать.

2.8 Фреймы, не обеспечивающие автоматическое распознавание типа

Некоторые сетевые технологии не предусматривают в заголовке фрейма поля типа. Это значит, что фреймы не обеспечивают автоматическое распознавание типа. Для определения типа данных, содержащихся во фрейме, применяют следующий подход. Перед передачей каких-либо данных отправитель и получатель согласовывают применение первых нескольких октетов поля данных для хранения сведений о типе. Программное обеспечение компьютера-отправителя добавляет эти сведения о типе данных в начало исходящего фрейма. Программное обеспечение в компьютере-получателе извлекает информацию о типе и использует ее при обработке данных.

Для того чтобы во всем программном обеспечении для определения типов применялись одинаковые значения, смысл каждого обозначения типа был определен организациями по стандартизации. Распределением этих обозначений занималось много организаций, которые не всегда координировали свои действия. Для решения проблемы классификации типов, назначенных разными организациями по стандартизации, институт IEEE разработал стандарт, который предусматривает включение не только поля для обозначения типа, присвоенного организацией, но и поля с указанием самой организации по стандартизации. Эта спецификация, составляющая часть стандарта 802.2 IEEE, известна под названием заголовка управления логическим соединением (LLC — Logical Link Control) стандартного протокола доступа к сети (SNAP — Standard Network Access Protocol). Формат заголовка LLC/SNAP IEEE является общепринятым.

На рис. 4.3 показан пример заголовка LLC/SNAP, который содержит восемь октетов. Первые три октета представляют часть LLC, которая указывает, что далее следует поле типа данных.

AA

AA

03

00

00

00

08

00

LLC

SNAP

Поле OUI, состоящее из 3 октетов, обозначает организацию по стандартизации

Значение TYPE, представленное в 2 октетах, которое определено этой организацией

Рис. 4.3. Пример восьмиоктетного заголовка LLC/SNAP IEEE, который используется для указания типа данных. Часть SNAP обозначает организацию и тип, определенный этой организацией

Как показано на рисунке, часть заголовка SNAP разделена на два поля. Первое поле носит название уникального идентификатора организации (OUI — Organizationally Unique Identifier) и используется для обозначения организации по стандартизации. Второе поле содержит значение типа, определенное этой организацией. Например, значение OUI со всеми нулями, показанное на рис. 4.3 принадлежит организации, которая определяет типы Ethernet. Поэтому шестнадцатеричное значение 0800, показанное в этом примере в поле типа, интерпретируется в соответствии со стандартом, в котором определены типы Ethernet. Как и типы, закодированные в заголовке фрейма, поле типа LLC/SNAP даёт возможность всем компьютерам совместно использовать сеть для широковещательной рассылки фреймов. При поступлении фрейма на компьютер проверяется информация LLC/SNAP в начале области данных фрейма. Если получатель не распознает OUI или не имеет программного обеспечения для обработки данных полученного типа, фрейм отбрасывается. Поэтому широковещательный фрейм, несущий данные определенного типа, будет проигнорирован всеми компьютерами сети, кроме тех, что распознают этот тип данных.

### 3. Задание к лабораторной работе

1. Разработать и отладить программу, выполняющую анализ потока кадров. Потокикадров представлены в виде файлов двоичного формата, место нахождения которых уточняется у преподавателя. В кадрах отсутствует преамбула и контрольная сумма, для исходящего кадра длина может быть меньше минимальной. Каждая бригада выполняет обработку одного файла с именем ethersXX.bin, где ХХ – номер бригады.

При выполнении работы в дистанционном режиме в обязательном порядке выполнить анализ файлов ethers06.bin и ethers07.bin.

Требования к программе:

* предусмотреть возможность ввода имени файла с клавиатуры;
* обеспечить вывод на экран по каждому кадру: номер, размер, тип, IP- адресов (основную информацию заголовка IP-пакета), MAC- адресов (основную информацию заголовка кадра); также необходимо вывести итоговые результаты обработки: общее число обработанных кадров и число кадров каждого типа (IP, ARP, Novell, SNAP и т.д.);

Алгоритм определения формата кадров можно найти в Приложении 2. Там же находится вся необходимая информация по форматам кадров Ethernet, IP- пакетам и TCP-сегментам.

2. В отчете необходимо выполнить полный анализ кадра с номером ХХ, где ХХ номер Вашей бригады. Анализ можно провести с помощью любого 16-ричного редактора, например, свободно распространяемого редактора HxD.

3. Проверка программы преподавателем может проводиться на любом исходном файле.

### Контрольные вопросы

1. Как осуществляется передача фреймов по сети?
2. Назовите способы адресации сетевых узлов и их отличия.
3. Что называется широковещательной рассылкой? В каких случаях она используется? В чем ее достоинства и недостатки?
4. Что называется групповой рассылкой? В каких случаях она используется? Чем она отличается от широковещательной рассылки?
5. Какие методы используются для обозначения содержимого фрейма?
6. Приведите общий формат фрейма, передаваемого по сети.
7. Приведите форматы фреймов Ethernet.
8. Укажите несколько типов фреймов, используемых в сети Ethernet.
9. Как определяют тип данных, содержащихся во фрейме, не обеспечивающем автоматическое распознавание типа?

# Лабораторная работа № 5. Анализ функционирования и диагностика IP-сетей

1. Цель работы

Приобретение практических навыков работы с сетевыми командами операционных систем Windows и Linux, предназначенными для анализа и диагностики сетей TCP/IP, а также со средствами интерактивной диагностики сетей.

2. Методические указания

2.1 Общие сведения

Компьютерная сеть – это совокупность однородных или разнородных компьютеров, соединенных между собой линиями связи и имеющих специальное программное обеспечение, позволяющее организовать обмен данными. Современные сети, как правило, содержат также специальное сетевое оборудование - коммутаторы, маршрутизаторы, шлюзы и т.д. Объединение сетей разной архитектуры и топологии стало возможно благодаря протоколу IP (Internet Protocol) и принципу маршрутизации пакетов данных

В сети Интернет используется трехуровневая система адресации с использованием физических, сетевых и доменных адресов. *Физический* адрес (MAC-адрес) – это уникальный адрес узла сети внутри среды передачи данных, состоящий из шести байтов, которые принято записывать в шестнадцатеричной системе и разделять двоеточиями. Каждый сетевой интерфейс имеет уникальный MAC- адрес, который используется для идентификации отправителя и получателя в пределах одной подсети. Один узел сети может иметь несколько физических адресов, если в нем установлены несколько сетевых интерфейсов.

*Сетевой* адрес (IP-адрес), определяемый протоколом IPv4, состоит из четырех байтов, записываемых традиционно в десятичной системе счисления и разделяемых точкой. Сетевой адрес узла логически разделяется на две части, одна из которых называется адресом сети, а другая - адресом узла. Таким образом с помощью пары чисел можно адресовать любой узел, подключенный к глобальной сети на базе протокола TCP/IP.

Для того чтобы определить размер адреса сети, используется маска сети – 32-битное число, выделяющее в IP-адресе адрес сети и адрес узла: биты маски, соответствующие адресу сети, установлены в 1, а биты маски, соответствующие адресу узла в сети, установлены в 0. Возможны два способа указания маски:

- в том же виде, что и IP-адрес: IP-адрес сети – 170.160.0.0, маска подсети – 255.240.0.0;

- указанием количества начальных единичных битов в маске сети сразу после адреса сети через символ ”/”: IP-адрес сети: 170.160.0.0/12

Все IP-адреса протокола IPv4 делятся на публичные (глобальные, внешние, белые) и частные (локальные, внутренние, серые). Публичным IP-адресом называется адрес, который используется для выхода в Интернет. Доступ к устройству с публичным IP-адресом можно получить из любой точки глобальной сети, т.к. эти IP-адреса маршрутизируются в Интернете в отличие от частных IP-адресов.

Количество публичных адресов протокола IPv4 ограничено, поэтому в локальных сетях используются частные IP-адреса. Частный IP-адрес принадлежит к специальному диапазону, не используемому в сети Интернет. Следующие диапазоны определены как адреса, выделенные для локальных сетей:

10.0.0.0 – 10.255.255.255 ([маска подсети](https://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D1%81%D0%BA%D0%B0_%D0%BF%D0%BE%D0%B4%D1%81%D0%B5%D1%82%D0%B8) 255.0.0.0);

172.16.0.0 — 172.31.255.255 ([маска подсети](https://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D1%81%D0%BA%D0%B0_%D0%BF%D0%BE%D0%B4%D1%81%D0%B5%D1%82%D0%B8) 255.240.0.0);

192.168.0.0 — 192.168.255.255 ([маска подсети](https://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D1%81%D0%BA%D0%B0_%D0%BF%D0%BE%D0%B4%D1%81%D0%B5%D1%82%D0%B8) 255.255.0.0).

Частные сетевые адреса могут присваиваться узлам сети двумя способами – в ручном режиме и в автоматическом. Второй способ позволяет автоматизировать работу администратора сети, используя для этого специальный протокол DHCP.

Более современная версия протокола IPv6 использует сетевые адреса размером 16 байтов.

*Доменные* (символьные) адреса введены для удобства пользователей. Первоначально система доменных имён в Интернете была введена по административному принципу, когда домен верхнего уровня мог принимать одно из значений, определяющих характер организации: *gov* – правительственные, *edu* – образовательные, *com* – коммерческие, *org* – общественные, *net* – организации, предоставляющие сетевые услуги. Затем начал применяться территориальный принцип, в котором в качестве домена верхнего уровня используется код страны (для России - «ru», для США – «us» и т.д.).

Система доменных имен строится по иерархическому принципу, например *fpm2.ami.nstu.ru.* Здесь *fpm2*– имя узла, *ami* – имя подразделения, *nstu* – имя организации, *ru* – имя домена верхнего уровня.

Кадры, передаваемые между узлами сети на канальном уровне, несут информацию только о физических адресах получателя и отправителя. На сетевом уровне носителями данных являются пакеты (дейтаграммы), содержащие IP-адреса получателя и отправителя. На прикладном уровне для указания получателя данных пользователи чаще всего используют доменные адреса.

Однозначной связи между физическими, сетевыми и доменными адресами не существует. Поэтому единственным способом для преобразования доменного имени в сетевой адрес, а сетевого адреса в физический адрес, является применение ассоциативных таблиц, каждая запись в которых содержит непосредственное описание соответствующей связи.

Преобразование доменного имени в сетевой адрес реализуется на основе протокола DNS, программное обеспечение которого реализовано в технологии «клиент - сервер». В каждом домене Интернета имеется выделенный DNS-сервер, на котором хранятся таблицы соответствия доменных и сетевых имен всех узлов домена, а также ссылки на адреса DNS-серверов всех поддоменов и родительского домена, т.е. данные DNS могут быть представлены в виде распределенного набора таблиц.

DNS-клиенты входят в состав сетевого ПО всех операционных систем и предназначены для формирования запросов к DNS-серверам, а также получения и хранения результатов выполнения этих запросов в кэше. Записи в DNS-кэше могут быть статическими и динамическими. Просмотреть содержимое кэша можно командой **ipconfig /displaydns** (рис. 5.1),очистить кэш – командой **ipconfig /flushdns**.

Преобразование сетевого адреса в физический адрес реализуется на основе протокола ARP, программное обеспечение и таблицы соответствия которого хранятся на каждом узле сети. Записи в ARP-таблице могут быть статическими и динамическими (рис. 5.2). Статические записи создаются вручную и существуют до следующей перезагрузки устройства, динамические записи периодически обновляются.

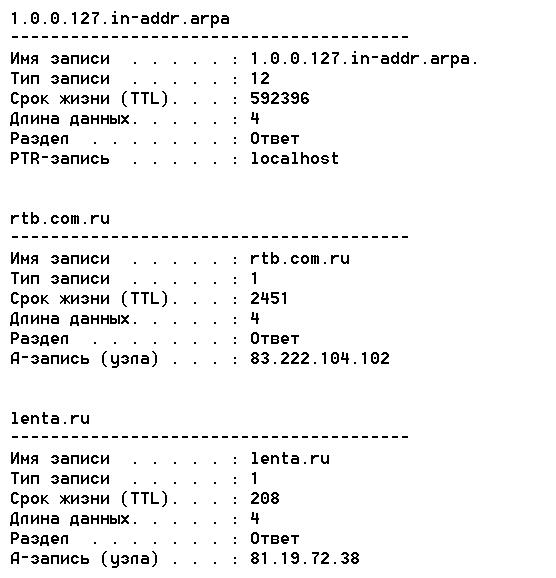


Рис. 5.1

Для работы с ARP-таблицей используется команда **arp:**

- просмотр таблицы: **arp –a;**

- добавление статических записей: **arp –s 129.35.251.23 00-AA-00-62-C6-09;**

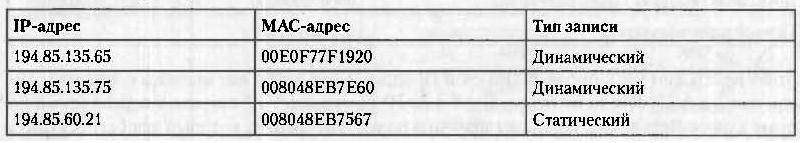


Рис. 5.2

Когда компьютер с некоторым IP-адресом решает отправить кадр с данными другому компьютеру, он выясняет, принадлежит ли адресат той же локальной сети, что и отправитель, следующим образом: на IP-адрес получателя накладывается сетевая маска и вычисляется адрес сети, которой принадлежит получатель. Если этот адрес совпадает с адресом сети отправителя, то оба находятся в одной локальной сети и MAC-адрес получателя берется из ARP-таблицы отправителя. При отсутствии в таблице IP-адреса получателя протокол ARP позволяет отправителю разослать по всему сегменту сети широковещательный запрос, который требует от компьютера, имеющего заданный IP-адрес, откликнуться и указать свой физический адрес. После получения ответа отправитель добавляет новую запись в свою ARP-таблицу и отправляет кадр узлу-адресату.

Если отправитель и получатель находятся в разных сетях, то кадр отправляется ближайшему маршрутизатору, который извлекает из него IP-пакет, находит IP-адрес получателя и в своей таблице маршрутизации находит адрес ближайшего маршрутизатора для последующей передачи пакета в заданную сеть. Маршрутизаторы могут быть аппаратными или программными.

Просмотреть таблицу маршрутизации можно командой **route print** (табл.5.1):

Таблица 5.1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Сетевой адрес** | **Маска** | **Адрес шлюза** | **Интерфейс** | **Метрика** |
| 127.0.0.0 | 255.0.0.0 | 127.0.0.1 | 127.0.0.1 | 1 |
| 0.0.0.0 | 0.0.0.0 | 198.21.17.7 | 198.21.17.5 | 1 |
| 129.13.0.0 | 255.255.0.0 | 198.21.17.6 | 198.21.17.5 | 2 |
| 198.21.17.0 | 255.255.255.0 | 198.21.17.5 | 198.21.17.5 | 1 |
| 198.21.17.5 | 255.255.255.255 | 127.0.0.1 | 127.0.0.1 | 1 |
| 198.21.17.255 | 255.255.255.255 | 198.21.17.5 | 198.21.17.5 | 1 |
| 213.34.12.0 | 255.255.255.0 | 213.34.12.3 | 213.34.12.3 | 1 |
| 213.34.12.3 | 255.255.255.255 | 127.0.0.1 | 127.0.0.1 | 1 |
| 255.255.255.255 | 255.255.255.255 | 198.21.17.6 | 198.21.17.6 | 1 |

Здесь в качестве сетевого адреса назначения может указываться IP-адрес сети, специфический IP-адрес конкретного узла или адрес, используемый по умолчанию. В приведенной таблице адрес по умолчанию задается сетевым адресом 0.0.0.0 с маской 0.0.0.0, а специфические адреса указаны в строках, имеющих значение маски. 255.255.255.255. Остальные строки в качестве адреса назначения задают известные маршрутизатору адреса сетей.

Поле «Адрес шлюза» задает адрес ближайшего маршрутизатора для продвижения IP-пакета к заданной сети, поле «Интерфейс» указывает IP-адрес выходного сетевого интерфейса текущего маршрутизатора, а поле «Метрика» хранит целое число, используемое в качестве критерия выбора маршрута передачи пакета, например, количество промежуточных маршрутизаторов, которые должен пройти пакет, чтобы достигнуть адресата.

2.2 Диагностика IP-сети

Современные сети могут иметь могут иметь сложную топологию с большим количеством компьютеров и различного коммутационного оборудования, поэтому пользователям часто приходится решать задачи, связанные с диагностикой функционирования сетевого оборудования. Например, часто возникает необходимость определения работоспособности того или иного узла сети, выяснения маршрута прохождения пакетов в случае появления больших задержек передачи данных или оценки вероятности потери пакетов в заданных сегментах сети. Кроме того, может возникнуть необходимость определения IP-адреса или географического местонахождения некоторого узла сети по его доменному имени.

Для выявления различных проблем, которые могут возникнуть при прохождении в сети IP-пакетов, в состав стека протоколов TCP/IP входит протокол ICMP.

2.2.1 Основы протокола ICMP

Протокол ICMP разрабатывался для формирования сообщений о невозможности доставки IP-дейтаграммы. Позднее были разработаны другие способы использования созданной системы управления сообщениями. Например, на ее основе были разработаны программы, предназначенные для проверки работоспособности заданных узлов сети, а также для определения маршрута передачи IP-пакетов.

ICMP-сообщение готовится в специальном формате (рис. 5.3), инкапсулируется в новый IP-пакет, который передается отправителю пакета, для которого возникла ошибка. Поэтому протоколы IP и ICMP связаны друг с другом: протокол IP использует ICMP для формирования сообщений об ошибках, а протокол ICMP использует IP для доставки этих сообщений.

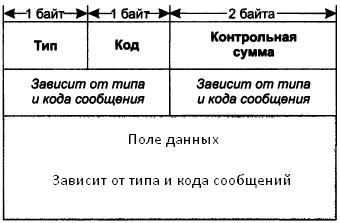


Рис. 5.3

ICMP-сообщение состоит из заголовка и поля данных. Заголовок содержит обязательную и необязательную части по 4 байта каждая. Обязательная часть имеет 3 поля: *тип* (указывает основную причину формирования сообщения), *код* (уточняет причину) и *контрольную сумму* сообщения. Поле данных имеет переменной длину и содержит данные, которые необходимо вернуть отправителю.

Интерпретация необязательной части заголовка и поля данных зависят от типа сообщения. Например, для сообщений типа 8 (эхо-запрос) и 0 (эхо-ответ) необязательная часть заголовка может содержать поля «Идентификатор» и «Номер», которые используются отправителем для проверки соответствия между запросом и ответом, а поле данных для сообщений типа 3 (адресат недоступен), 5 (перенаправление маршрута) и 11 (истечение TTL) содержит заголовок и первые 8 байтов поля данных IP-пакета, вызвавшего ошибку, чтобы отправитель мог идентифицировать этот пакет.

На рис. 5.4 приведены основные типы ICMP-сообщений:

* *Запрос/ответ эхо - повтора*. Запрос эхо - повтора может быть передан программному обеспечению протокола ICMP любого компьютера. В ответ на запрос эхо – повтора программное обеспечение ICMP должно послать ответ эхо – повтора ICMP, который содержит те же данные, что и запрос. Этот тип сообщений используется программами **ping** и **tracert.**
* *Получатель не доступен*. Обнаружив, что дейтаграмма не может быть доставлена в место назначения, маршрутизатор отправляет хосту, на котором она была создана, сообщение о том, что получатель недоступен. В сообщении указано, является ли недоступным конкретный хост назначения или недоступна сеть, к которой подключен хост назначения.
* *Истечение времени дейтаграмм*. Сообщение формируется в случае TTL=0.



Рис. 5.4

2.2.2 Команды диагностики IP-сетей

В состав каждой ОС входит набор утилит, предназначенный для диагностики оборудования сетей TCP/IP. Рассмотрим некоторые из этих утилит.

2.2.2.1 Команда: **ipconfig**

Команда **ipconfig** (в Linux – **ifconfig**) предназначена для управления сетевыми интерфейсами. Она позволяет просмотреть параметры настройки всех интерфейсов в кратком (по умолчанию) или в полном виде (ключ **/all**), просмотреть и очистить кэш разрешений DNS (**/displaydns** и **/flushdns**), обновить (**/renew**) или освободить (**/release**) IP-адрес любого интерфейса и т.д. Ниже приведен пример полного вывода информации по интерфейсам для ОС Windows и Linux.

**C:\> ipconfig /all**

Настройка протокола IP для Windows

Имя компьютера . . . . . . . . . : titan Основной DNS-суффикс . . . . . . : Тип узла. . . . . . . . . . . . . : гибридный IP-маршрутизация включена . . . . : нет WINS-прокси включен . . . . . . . : нет

VMware Network Adapter VMnet8 - Ethernet адаптер: DNS-суффикс этого подключения . . :

Описание . . . . . . . . . . . . : VMware Virtual Ethernet Adapter for VMnet8

Физический адрес. . . . . . . . . : 00-50-56-C0-00-08 Dhcp включен. . . . . . . . . . . : да Автонастройка включена . . . . . : да IP-адрес . . . . . . . . . . . . : 192.168.136.1 Маска подсети . . . . . . . . . . : 255.255.255.0 Основной шлюз . . . . . . . . . . :

DHCP-сервер . . . . . . . . . . . : 192.168.136.254

Основной WINS-сервер . . . . . . : 192.168.136.2

Аренда получена . . . . . . . . . : 15 марта 2017 г. 17:22:36

Аренда истекает . . . . . . . . . : 15 марта 2017 г. 17:52:36VMware Network Adapter VMnet1 - Ethernet адаптер: DNS-суффикс этого подключения . . : Описание . . . . . . . . . . . . : VMware Virtual Ethernet Adapter for VMnet1 Физический адрес. . . . . . . . . : 00-50-56-C0-00-01

Dhcp включен. . . . . . . . . . . : да

Автонастройка включена . . . . . : да

IP-адрес . . . . . . . . . . . . : 192.168.116.1

Маска подсети . . . . . . . . . . : 255.255.255.0

Основной шлюз . . . . . . . . . . :

DHCP-сервер . . . . . . . . . . . : 192.168.116.254 Аренда получена . . . . . . . . . : 15 марта 2017 г. 17:22:36

Аренда истекает . . . . . . . . . : 15 марта 2017 г. 17:52:36

GPON - Ethernet адаптер:

DNS-суффикс этого подключения . . :

Описание . . . . . . . . . . . . : Atheros AR8121/AR8113/AR8114 PCI-E Ethernet Controller

Физический адрес. . . . . . . . . : 00-24-8C-37-BB-1F

Dhcp включен. . . . . . . . . . . : да

Автонастройка включена . . . . . : да

IP-адрес . . . . . . . . . . . . : 192.168.1.102

Маска подсети . . . . . . . . . . : 255.255.255.0

Основной шлюз . . . . . . . . . . : 192.168.1.1

DHCP-сервер . . . . . . . . . . . : 192.168.1.1

DNS-серверы . . . . . . . . . . . : 192.168.1.1

Аренда получена . . . . . . . . . : 15 марта 2017 г. 16:36:00

Аренда истекает . . . . . . . . . : 15 марта 2017 г. 18:36:00

Подключение по локальной сети - Ethernet адаптер:

Состояние сети . . . . . . . . . : сеть отключена Описание . . . . . . . . . . . . : Kaspersky Security Data Escort Adapter

Физический адрес. . . . . . . . . : 00-FF-5E-89-A0-8F

Подключение по Bluetooth - Ethernet адаптер:

Состояние сети . . . . . . . . . : сеть отключена

Описание . . . . . . . . . . . . : Bluetooth PAN Network Adapter

Физический адрес. . . . . . . . . : 00-08-F4-16-53-A7

**[kvg@students ~]$ifconfig**

eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500

inet 217.71.130.131 netmask 255.255.255.128 broadcast 217.71.130.255

inet6 2001:b08:a:1040:a9ee:5fc5:451f:5cf prefixlen 128 scopeid 0x0<global>

inet6 fe80::215:5dff:fe82:8d01 prefixlen 64 scopeid 0x20<link>

inet6 2001:b08:a:1040:215:5dff:fe82:8d01 prefixlen 64 scopeid 0x0<global>

ether 00:15:5d:82:8d:01 txqueuelen 1000 (Ethernet)

RX packets 100722090 bytes 69784724650 (64.9 GiB)

RX errors 0 dropped 0 overruns 0 frame 0

TX packets 31856075 bytes 10623260236 (9.8 GiB)

TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536

inet 127.0.0.1 netmask 255.0.0.0

inet6 ::1 prefixlen 128 scopeid 0x10<host>

loop txqueuelen 0 (Local Loopback)

RX packets 11702641 bytes 7345008270 (6.8 GiB)

RX errors 0 dropped 0 overruns 0 frame 0

TX packets 11702641 bytes 7345008270 (6.8 GiB)

TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

virbr0: flags=4099<UP,BROADCAST,MULTICAST> mtu 1500

inet 192.168.122.1 netmask 255.255.255.0 broadcast 192.168.122.255

ether 52:54:00:d4:60:b6 txqueuelen 0 (Ethernet)

RX packets 0 bytes 0 (0.0 B)

RX errors 0 dropped 0 overruns 0 frame 0

TX packets 0 bytes 0 (0.0 B)

TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

В Linux название сетевого интерфейса состоит из его типа и порядкового номера(N). Все сетевые интерфейсы Ethernet в Linux называются ethN, начиная с eth0. Для каждого интерфейса указываются физические и сетевые адреса (ether, inet, inet6), маска сети (netmask), максимальный размер кадра (MTU), число принятых и переданных пакетов (RX packets, TX packets). IP-адрес сетевого интерфейса eth0 из примера – 217.71.130.131.

Интерфейс lo – это интерфейс внутреннего стека TCP/IP (петля или заглушка), предназначенный для отладки сетевого ПО и имеющий IP-адрес 127.0.0.1. Интерфейс virbr0 является виртуальным мостом, который связывает созданные пользователем виртуальные машины с основной машиной.

2.2.2.2 Команда: **nslookup**

Утилита  **nslookup** предоставляет интерфейс командной строки для  обращения к системе DNS. Она является аналогом службы DNS-клиент и позволяет диагностировать проблемы с разрешением имен в системе DNS. По умолчанию все запросы отправляются на DNS-сервер, адрес которого задан настройками сетевого подключения (см. команду **ipconfig**). В терминах утилиты такой сервер является сервером по умолчанию (default server).

При запуске **nslookup** без параметров утилита переходит в интерактивный режим, ожидая ввод команд пользователя. Ввод символа ”?” или команды **help**  (в Windows) отображает справку о внутренних командах и опциях программы **nslookup**. В Linux получение справки проводится с помощью команды **man**.

Ниже приведен результат выполнения команды **nslookup** в Windows и Linux.

**C:\> nslookup yandex.ru**

Server: UnKnown

Address: 192.168.1.1

Name: yandex.ru

Addresses: 77.88.55.70, 77.88.55.66, 5.255.255.80, 5.255.255.70

**[kvg@students ~]$nslookup yandex.ru**

Server: 217.71.130.130

Address: 217.71.130.130#53

Non-authoritative answer:

Name: yandex.ru

Address: 5.255.255.70

Name: yandex.ru

Address: 5.255.255.80

Name: yandex.ru

Address: 77.88.55.66

Name: yandex.ru

Address: 77.88.55.70

Из результатов видно, что поисковая система Яндекс использует несколько IP-адресов. Такой прием часто используется для обеспечения надежности высоконагруженных доменных имен. При этом на запрос, отправленный пользователем, отвечает наименее загруженный сервер.

2.2.2.3 Команда: **ping**

Команда посылает на указанный компьютер эхо-запрос и в течение небольшого промежутка времени ожидает ответа. Если ответ поступает, то программа сообщает пользователю IP-адрес компьютера и время двойного обращения пакета (от момента отправки сообщения до момента получения ответа), иначе выводится сообщение о том, что компьютер не отвечает. ***Ping*** используется для проверки работоспособности отдельных узлов сети и проверки правильности конфигурации сети, т.к. в выполнении этой команды участвуют система маршрутизации, схемы разрешения адресов и сетевые шлюзы. Если данная команда не работает, то и более сложные инструменты функционировать не будут.

Обязательным аргументом команды **ping** является сетевой или доменный адрес узла, ответ от которого необходимо получить. С помощью дополнительных ключей команды можно задать число отправляемых запросов, время жизни пакета и другие параметры запроса. Ниже приведены результаты работы команды для ОС Windows и ОС Linux

**C:\>ping yandex.ru**

Обмен пакетами с yandex.ru [77.88.55.66] по 32 байт:

Ответ от 77.88.55.66: число байт=32 время=6мс TTL=59Ответ от 77.88.55.66: число байт=32 время=2мс TTL=59Ответ от 77.88.55.66: число байт=32 время=4мс TTL=59Ответ от 77.88.55.66: число байт=32 время=3мс TTL=59Статистика Ping для 77.88.55.66:Пакетов: отправлено = 4, получено = 4, потеряно = 0 (0% потерь),Приблизительное время приема-передачи в мс:Минимальное = 2мсек, Максимальное = 6 мсек, Среднее = 3 мсек

**[kvg@students ~]$ping -c 4 yandex.ru**

PING yandex.ru (5.255.255.70) 56(84) bytes of data.

64 bytes from yandex.ru (5.255.255.70): icmp\_seq=1 ttl=59 time=0.878 ms

64 bytes from yandex.ru (5.255.255.70): icmp\_seq=2 ttl=59 time=0.954 ms

64 bytes from yandex.ru (5.255.255.70): icmp\_seq=3 ttl=59 time=1.25 ms

64 bytes from yandex.ru (5.255.255.70): icmp\_seq=4 ttl=59 time=1.91 ms

--- yandex.ru ping statistics ---

4 packets transmitted, 4 received, 0% packet loss, time 3004ms

rtt min/avg/max/mdev = 0.878/1.249/1.915/0.411 ms

Обратите внимание, что в Linux для указания количества отправленных пакетов используется ключ **-с**

2.2.2.4 Команда: **tracert**

Утилита **tracert** (в Linux – **traceroute**) предназначена для определения маршрута следования данных в сетях TCP/IP. Она отправляет эхо-запрос указанному узлу сети и выводит сведения о всех промежуточных маршрутизаторах, через которые прошли данные на пути к целевому узлу. В случае проблем при доставке данных до какого-либо узла программа позволяет определить, на каком именно участке сети возникли неполадки.

Для определения промежуточных маршрутизаторов **tracert** отправляет серию пакетов целевому узлу, при этом каждый раз увеличивая на 1 значение поля TTL, указывающее максимальное количество маршрутизаторов, которое может быть пройдено пакетом. Первый пакет отправляется при TTL=1 и поэтому первый же маршрутизатор возвращает обратно сообщение ICMP типа 11 (истечение времени), указывающее на невозможность доставки данных. Программа **tracert** фиксирует адрес маршрутизатора и интервал времени между отправкой пакета и получением ответа. Затем **tracert** повторяет отправку пакета при TTL=2, что позволяет первому маршрутизатору пропустить пакет дальше. Процесс повторяется до тех пор, пока пакет не достигнет целевого узла и при получении ответа от этого узла трассировка считается завершённой.

На конечном хосте IP-дейтаграмма с TTL = 1 не отбрасывается и передается программному обеспечению протокола ICMP, генерирующему сообщение типа 0.

В Linux-системах программа **traceroute** работает иначе. Она отправляет не ICMP-сообщения типа «эхо-запрос», а UDP фрагменты с таким номером UDP-порта адресата, что он заведомо не используется на адресуемом хосте (более 30000). UDP фрагменты инкапсулируются в IP- пакет, первый из которых имеет значение TTL=1, второй – равное 2 и т.д. Каждый из промежуточных маршрутизаторов выполняет декремент TTL и при значении, равным нулю, направляет отправителю ICMP-сообщение типа 11 (истечение времени). Конечный хост передает UDP фрагмент транспортному уровню, который возвращает ICMP-сообщение об ошибке «порт недоступен». Таким образом, чтобы узнать о завершении работы, программе **traceroute** достаточно обнаружить, что поступило ICMP-сообщение типа 3 с кодом ошибки, равным 3.

Утилита **traceroute** может работать в режиме передачи эхо-запросов ICMP. Для этого ее надо запустить с ключом **–I.** Ниже приведены результаты трассировки к сайту *ngs.ru* в Windows и Linux.

**C:\> tracert ngs.ru**

Трассировка маршрута к ngs.ru [195.93.187.9] с максимальным числом прыжков 30: 1 <1 мс <1 мс <1 мс 192.168.1.1 2 <1 мс <1 мс <1 мс 192.168.0.1 3 4 ms 4 ms 4 ms NVSK-BRAS3.sib.ip.rostelecom.ru [213.228.116.9] 4 4 ms 4 ms 4 ms ae9.NVSK-RGR5.sib.ip.rostelecom.ru [213.228.109.18] 5 2 ms 4 ms 4 ms ae3.NVSK-AR1.sib.ip.rostelecom.ru [213.228.109.51] 6 1 ms 4 ms 4 ms 1889.egw2.core.ngs.ru [195.93.186.2] 7 4 ms 4 ms 4 ms 195.93.186.5 8 4 ms 4 ms 4 ms webcluster.ngs.ru [195.93.187.9] Трассировка завершена.

**[kvg@students ~]$traceroute ngs.ru**

traceroute to ngs.ru (195.93.187.9), 30 hops max, 60 byte packets

1 gw-130-208v.ami.nstu.ru (217.71.130.251) 3.763 ms 4.005 ms 4.315 ms

2 gate.ami.nstu.ru (217.71.130.129) 0.379 ms 0.424 ms 0.447 ms

3 ix-i.nstu.ru (217.71.131.1) 1.086 ms 1.328 ms 1.294 ms

4 c6k-720.nstu.ru (217.71.128.67) 1.239 ms 1.161 ms 1.131 ms

5 nsk-ix.nstu.ru (217.71.128.193) 1.293 ms 1.263 ms 1.333 ms

6 egw4.core.ngs.ru (193.232.87.32) 1.108 ms 1.056 ms 1.029 ms

7 195.93.186.5 (195.93.186.5) 1.654 ms 1.729 ms 1.587 ms

8 webcluster.ngs.ru (195.93.187.9) 1.155 ms 1.129 ms 1.099 ms

2.2.2.5 Команда: **netstat**

Утилита **netstat** предназначена длявывода статистики по протоколам и подключениям TCP/IP. При вызове с ключом **–а** программа выводит информацию о состоянии всех активных сокетов с указанием локального и внешнего адресов (ESTABLISHED – соединение установлено, TIME\_WAIT - сокет закрыт, но ожидает пакеты, еще находящиеся в сети для обработки, LISTENING – сокет ожидает входящих соединений и т.д.).

Ключ **–r** указывает на необходимость вывода таблицы маршрутизации, а ключ **–s** – необходимость вывода статистики по протоколам. В Linux команда **netstat** с ключом –i выводит информацию о сетевых интерфейсах.

Ниже приведен пример вывода статистики по протоколу IPv4 в Windows и Linux.

**C:\> netstat -s**

Статистика IPv4

Получено пакетов = 18907

Получено ошибок в заголовках = 0

Получено ошибок в адресах = 58

Направлено датаграмм = 0

Получено неизвестных протоколов = 0

Отброшено полученных пакетов = 138

Доставлено полученных пакетов = 18769

Запросов на вывод = 19183

Отброшено маршрутов = 0

Отброшено выходных пакетов = 0

Выходных пакетов без маршрута = 0

Требуется сборка = 0

Успешная сборка = 0

Сбоев при сборке = 0

Успешно фрагментировано датаграмм = 0

Сбоев при фрагментации датаграмм = 0

Создано фрагментов = 0

Статистика TCP для IPv4

Активных открыто = 781

Пассивных открыто = 87

Сбоев при подключении = 3

Сброшено подключений = 121

Текущих подключений = 91

Получено сегментов = 17124

Отправлено сегментов = 12765

Повторно отправлено сегментов = 4061

Статистика UDP для IPv4

Получено датаграмм = 1529

Отсутствие портов = 116

Ошибки при получении = 0

Отправлено датаграмм = 2336

**[kvg@students ~]$netstat -s**

Ip:

76013304 total packets received

0 forwarded

0 incoming packets discarded

67453711 incoming packets delivered

39648867 requests sent out

55 dropped because of missing route

**[kvg@students ~]$netstat -i**

Kernel Interface table

Iface MTU RX-OK RX-ERR RX-DRP RX-OVR TX-OK TX-ERR TX-DRP TX-OVR Flg

eth0 1500 101285588 0 0 0 32106551 0 0 0 BMRU

lo 65536 11707980 0 0 0 11707980 0 0 0 LRU

virbr0 1500 0 0 0 0 0 0 0 0 BMU

2.2.4 Интерактивные средства диагностики IP-сетей

В настоящее время имеется достаточно много различных интерактивных сетевых сервисов, позволяющих решить проблемы пользователей, возникающие при работе с сетью. Такие сервисы условно можно разделить на две группы: предназначенные для сетевых программистов и предоставляющие услуги (хостинг или облачное хранение данных).

К первой группе относятся, например, сервисы **network-tools.com, speed-tester.info, 24whois.ru, seogadget.ru, ip-ping.ru**. Они содержат набор аналитических инструментов для веб-мастеров и позволяют проверить пинг, выполнить трассировку, определить IP-адрес по заданному доменному имени, получить информацию по указанному IP или доменному адресу, включая его местонахождение, а также выполнить другие аналогичные задачи.

К этой же группе можно отнести сервисы, выполняющие трассировку от указанного пункта в любой точке мира к компьютеру пользователя и отображающие эту трассу на географической карте. Такими сервисами являются, например,

**ping-admin.ru, monitis.com/traceroute, yougetsignal.com/tools/visual-tracert** и др.

Для сервисов второй группы средства диагностики сетей являются вспомогательными и в основном содержат функцию *whois*, с помощью которой можно получить информацию о владельце сайта, дате регистрации и дате окончания регистрации доменного имени. Примером такого сервиса является **whois.com**.

### 3. Задание к лабораторной работе

Выполнение лабораторной работы проводится в два этапа. На первом этапе необходимо изучить основные способы диагностики IP-сетей, на втором – спроектировать и реализовать программу, выполняющую основные функции утилит **ping** и **tracert**.

Задания первого этапа выполняются на рабочем компьютере (РК) с установленной ОС Windows и на удаленных серверах. Номер варианта выбирается в соответствии в номером бригады.

3.1. Этап 1

1. Подключиться с помощью клиента Putty к серверу **fpm2.ami.nstu.ru** и с помощью команды **uname** получить полную информацию об установленной операционной системе и аппаратной платформе, полученный результат включить в отчет.

2. Получить статистику по сетевым интерфейсам РК и сервера **fpm2.ami.nstu.ru**, пояснить результаты.

3. Просмотреть содержимое DNS-кэша, пояснить характеристики записей, очистить кэш.

4. Просмотреть содержимое ARP-таблицы, пояснить характеристики записей, выполнить добавление и удаление статических записей.

5. Просмотреть содержимое таблицы маршрутизации, пояснить характеристики записей

6. В командном режиме на РК и на сервере определить IP-адреса поисковых систем в соответствии с вариантом из таблицы 5.1, пояснить результаты.

Таблица 5.1

|  |  |
| --- | --- |
| Вариант | Название |
| 1 | google.com |
| 2 | google.ru |
| 3 | yandex.ru |
| 4 | yandex.com |
| 5 | rambler.ru |
| 6 | rambler.com |
| 7 | yahoo.com |
| 8 | yahoo.ru |
| 9 | ya.ru |

7. В командном режиме на РК и на сервере определить IP-адрес узлов сети в соответствии с номером варианта, указанного в таблице 5.2, выполнить его пингование и трассировку.

Таблица 5.2

|  |  |
| --- | --- |
| Вариант | Доменное имя |
| 1 | mit.edu, vk.com |
| 2 | stanford.edu, twitter.com |
| 3 | whitehouse.gov, msu.ru |
| 4 | caltech.edu, facebook.com |
| 5 | ox.ac.uk, drom.ru |
| 6 | ucl.ac.uk, antiplagiat.ru |
| 7 | imperial.ac.uk, youtube.com |
| 8 | ethz.ch, wikimapia.org |
| 9 | uchicago.edu, ifost.org |
| 10 | smu.edu.sg, elibrary.ru |
| 11 | curtin.edu.sg, twirpx.com |

8. С помощью интерактивных сетевых сервисов (например, **ping-admin.ru**) выполнить трассировку, определить местонахождение и владельца узла сети в соответствии с номером варианта, указанного в таблице 5.2. Результат трассировки в виде скриншота географической карты представить в отчете и выполнить его анализ. Начальный пункт трассировки – г. Новосибирск.

3.2. Этап 2

Реализовать Windows- или Linux-приложение, которое будет выполнять основные функции одной из утилит мониторинга сети в соответствии с таблицей 5.3. Результат работы функций должен быть идентичен результату работы программ **ping** и **tracert**.

Таблица 5.3

|  |  |  |  |
| --- | --- | --- | --- |
| Вариант | Аналог | ОС | Функция |
| 1 | ping | Windows | запрос эхо - повтора |
| 2 | tracert | Windows | трассировка маршрута передачи данных |
| 3 | ping | Windows | запрос отметки времени |
| 4 | ping | Windows | запрос маски сети |
| 5 | ping | Linux | запрос эхо - повтора |
| 6 | traceroute | Linux | трассировка маршрута передачи данных |
| 7 | ping | Linux | запрос отметки времени |
| 8 | ping | Linux | запрос маски сети |

### 4. Контрольные вопросы

1. Как определить доступность рабочей станции в сети Интернет?
2. Как определить количество маршрутизаторов на пути от вашего компьютера до требуемого вам узла сети ?
3. На каком уровне модели OSI работают программы **tracert** и **traceroute** ?
4. Назовите основные типы ICMP – сообщений.
5. Назначение поля «Код» в ICMP – сообщении.
6. Основные возможности стандартной утилиты **ping**.
7. Алгоритм работы утилиты **traceroute**.
8. Структура ICMP пакета.
9. Схема инкапсуляции ICMP – пакета в Ethernet – кадр.

# Лабораторная работа № 6. Организация сетевого взаимодействия в облачной платформе НГТУ

1. Цель работы

Изучить основные принципы использования облачных технологий для отладки и тестирования сетевых приложений на примере корпоративного Облака НГТУ.

### Методические указания

2.1 Общие сведения

В настоящее время для хранения и обработки информации часто используются облачные технологии. Облачные службы предлагают большое количество услуг и функций для государственных учреждений и коммерческих структур, а также для физических лиц (интернет-магазины, социальные сети, защита важных цифровых документов и т.д.). В отличие от систем удаленного хранения данных, предназначенных для обмена файлами (Яндекс Диск, Google Диск, DropMeFiles и т.д.), облачные системы предоставляют пользователям полноценную вычислительную систему, включающую процессоры, оперативную и внешнюю память, а также разнообразное программное обеспечение (рис.2.1). Ключевым признаком облачных технологий является обработка данных на удаленных серверах.

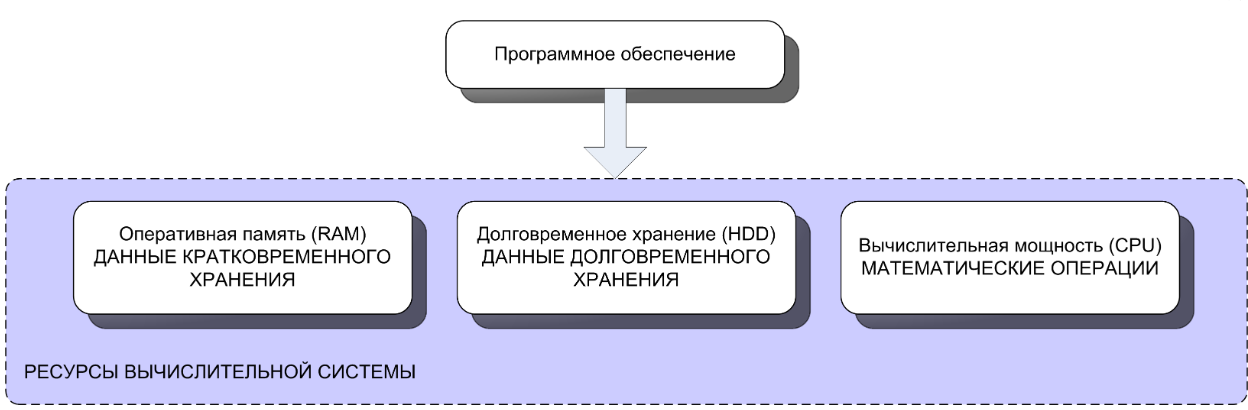


Рис. 2.1 Структура ресурсов Облака

Пользователям Облака НГТУ выделяются следующие ресурсы:

* виртуальные серверы пользователей;
* терминальные серверы общего пользования;
* размещенное в Облаке НГТУ программное обеспечение;
* размещенные в Облаке СУБД MariaDB и PostgreSQL.

При работе с Облаком пользователь должен выполнять следующие правила:

* использовать ресурсы Облака только для учебной и исследовательской деятельности;
* предпринимать необходимые меры предосторожности для защиты своих виртуальных машин и других ресурсов от несанкционированного доступа;
* не допускать попадания своих регистрационных данных к посторонним лицам;
* соблюдать требования законодательства РФ, в том числе исключить использование нелегального программного обеспечения и не осуществлять действия в сети Интернет и сети НГТУ, которые могут быть расценены как правонарушение;
* не занимать вычислительные ресурсы, если на это нет необходимости в учебной или исследовательской деятельности.

Нарушение положений настоящих правил, в том числе нецелевое использование сетевых и вычислительных ресурсов НГТУ, может привести к материальному и/или моральному ущербу, а также создать угрозу для компьютерной безопасности НГТУ.

Справку по облачной платформе можно получить по адресу <https://cloud.nstu.ru/wiki/>

## 2.2 Создание виртуального сервера

Вход в облачную платформу НГТУ проводится по адресу <https://cloud.nstu.ru/> с использованием учетной записи личного кабинета студента. Управление виртуальными серверами (ВС) проводиться на панели управления, изображенной на рис. 2.2.

Параметры процессора, памяти и хранилища указываются при создании ВС и ограничены вашей [квотой](https://cloud.nstu.ru/wiki/%D0%BA%D0%B2%D0%BE%D1%82%D0%B0) на ресурсы кластера. Базовые значения квот для студентов следующие:

* число процессорных ядер (vCPU) – 4;
* память (оперативная) – 2 Гб;
* хранилище (внешняя память) – 100 Гб.

Значения vCPU и Память учитываются в квоте только для включенных ВС, т.е. при стандартной квоте в 4 vCPU и 4 Гб памяти Вы можете создать несколько серверов по 4 vCPU и 2 Гб ОЗУ, но запустить единовременно сможете только один. Дисковая квота учитывает фактически израсходованное дисковое пространство, поэтому не зависит от того включен сервер или нет.

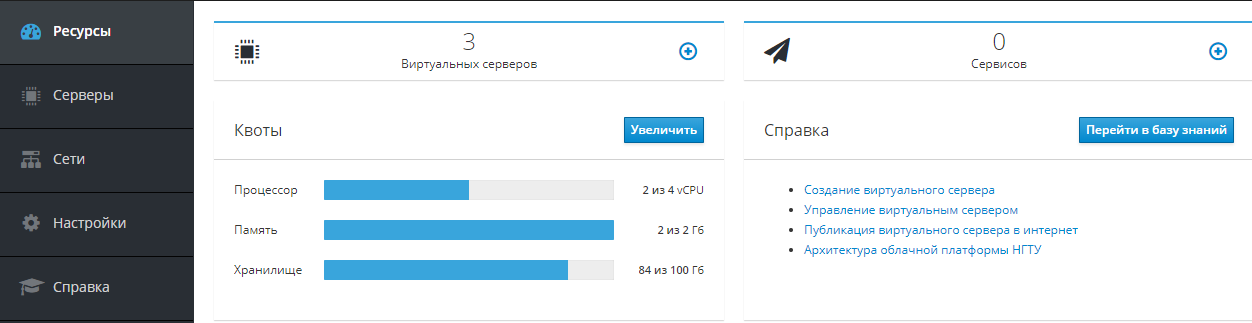


Рис. 2.2 Панель управления

Для того чтобы создать ВС необходимо выполнить следующие этапы. Перейти в панель управления во вкладку «*Серверы*», нажать на кнопку «Создать виртуальный сервер», заполнить поля в появившейся форме (рис.2.3) и нажать на кнопку «*Создать*».

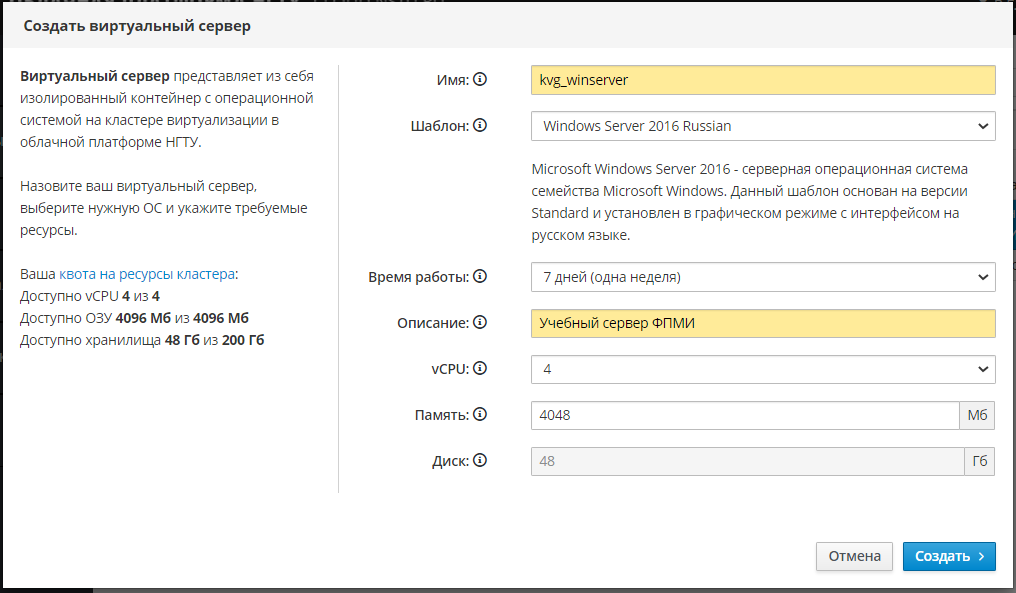


Рис. 2.3 Создание виртуального сервера

Список параметров:

1. *имя* – краткое уникальное имя сервера;
2. *шаблон* – базовая операционная система, от выбора которой зависят минимально допустимые настройки параметров процессора, памяти и диска. Вам будут доступны следующие предустановленные образы операционных систем:

* Windows Server 2016 Russian;
* Windows Server 2016 English;
* CentOS 7.8 Linux;
* CentOS 8 Linux;
* Fedora Server 28;
* Debian Linux 9;
* Ubuntu Server 18.04 LTS.

В зависимости от образа операционной системы установлены различные пароли встроенных администраторов: для Centos, Fedora, Debian и Ubuntu используется логин **root,** пароль **Pa$$w0rd!** , адля всех версий Windows пароль задается пользователем самостоятельно через консоль веб-интерфейса;

1. время работы – разрешенное время непрерывной работы сервера без перезагрузки, по истечении времени работы сервер выключается автоматически.
2. *описание* - произвольное описание, используется для удобства управления сервером в дальнейшем;
3. *vCPU* – количество процессорных ядер, можно указать любое значение в пределах квоты;
4. *память* – количество оперативной памяти в мегабайтах, можно указать любое значение в пределах квоты;
5. *диск* – объем системного диска в гигабайтах, для разных шаблонов заранее определен его минимальный размер.

# 2.3 Управление виртуальным сервером

Для выполнения любых действий с сервером необходимо выделить конкретную строку в списке всех созданных ВС, отображаемых на вкладке «*Серверы*» (рис. 2.4), и нажать соответствующую кнопку.

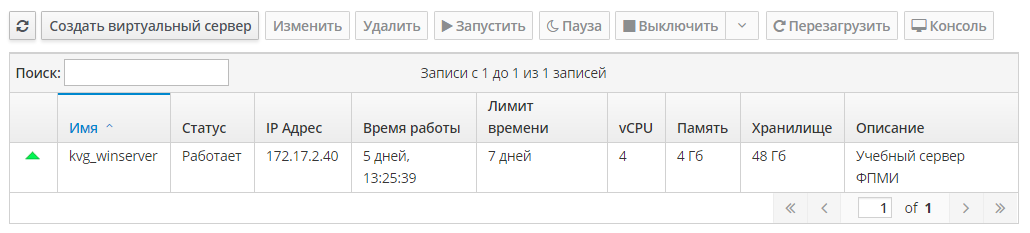


Рис. 2.4 Вкладка «Серверы»

Возможны следующие действия:

* создание и удаление;
* включение, выключение и перезагрузка операционной системы;
* изменение параметров;
* работа с виртуальным монитором (консолью).

После первого запуска ВС получает от DHCP-сервера Облака внутренний IP-адрес в диапазоне от 172.17.2.1 – 172.17.3.254.

У каждого сервера есть два состояния – выключен и включен, по аналогии с обычным компьютером. Во включенном состоянием он потребляет все выделенные ресурсы (память и процессорное время), а в выключенном состоянии операционная система не загружена и сервер потребляет только ресурс хранения.

Для включения ВС используйте кнопку ***«****Запустить****»***, для выключения – кнопку ­***«****Выключить****»***. Кнопка ***«****Пауза****»*** используется для сохранения текущего состояния оперативной памяти на жесткий диск, операция аналогична гибернации на персональных компьютерах. Если ВС более не требуется, то его можно удалить из Облака кнопкой ***«****Удалить****»***, данное действие доступно только в выключенном состоянии. Для первоначальной настройки или задач восстановления может потребоваться прямой доступ к экрану ВС, для этого во включенном состоянии используется кнопка ***«****Консоль****»***.

*Обратите внимание: для генерации кода нажатия клавиш Ctrl+Alt-Del в консоли необходимо нажать кнопку CtrlAltDel, которая находится в правом верхнем углу экрана.*

Вы можете отредактировать некоторые параметры сервера (рис. 2.5). Например, можно увеличить или уменьшить объем оперативной памяти, системного диска или изменить количество процессорных ядер. Кнопка ***«****Изменить****»*** доступна только в выключенном состоянии.

Для удаленной работы с ресурсами ВС необходимо опубликовать его в сети Интернет на вкладке «Сети». Возможны следующие варианты публикации (рис.2.6):

* для организации взаимодействия сетевых приложений, использующих протоколы HTTP или HTTPS;
* для удаленного управления сервером на основе протоколов RDP (Remote Desktop Protocol) или SSH (Secure Shell);
* для организации взаимодействия сетевых приложений, использующих стек протоколов TCP/IP.

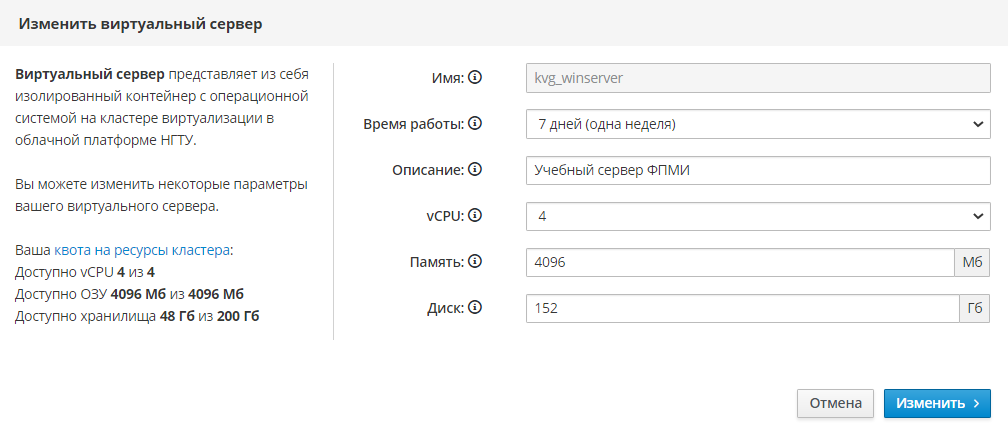


Рис. 2.5 Панель изменения параметров сервера

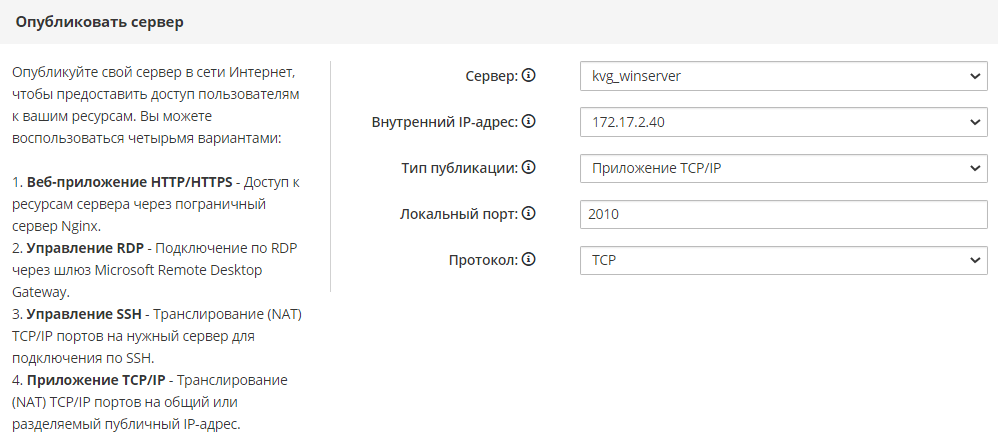


Рис. 2.6 Настройка публикации для сетевых приложений

Публикация для доступа по протоколам HTTP или HTTPS, использующим стандартный порт 80, необходима в том случае, если на ВС будет устанавливаться программное обеспечение какого-либо web-сервера. Для такой публикации необходимо задать только доменное имя web-ресурса, доступ к которому будет проводится через внутренний прокси-сервер, имя которого будет сгенерировано автоматически. Для корректной работы к web-ресурсу требуется создать DNS запись, указывающую на сервер *vbr1.cloud.nstu.ru* в качестве псевдонима; например, если адрес ресурса (сайта) *kws.cloud.nstu.ru*, то DNS запись должна иметь вид:

*kws.cloud.nstu.ru. CNAME vbr1.cloud.nstu.ru.*

Протокол RDP (Remote Desktop Protocol) позволяет получить доступ к программным и информационным ресурсам ВС c установленной ОС Windows через стандартный порт 3379. При этом управление ВС обеспечивается промежуточным сервером Microsoft Remote Desktop Gateway, клиент RDP имеется во всех дистрибутивах Windows. Достоинствами такого доступа по сравнению с консолью являются возможность копирования файлов между локальным компьютером и ВС с помощью стандартной комбинации клавиш Ctrl+C и Ctrl+V, а также меньшее время реакции.

Для подключения по RDP необходимо по ссылке «Скачать файл» скачать и запустить исполняемый файл настройки удаленного доступа с именем *cloud-rdp-имя\_сервера.rdp*, после чего внутренний IP-адрес и номер порта выставляются по умолчанию. Файл настройки можно сохранить на локальном компьютере и использовать для подключения к ВС без браузера, при этом потребуется дважды ввести пароли: для подключения к шлюзу удаленного доступа вводятся данные учетной записи студента от личного кабинета НГТУ, а для доступа к операционной системе ВС вводится пароль, заданный при создании ВС.

Протокол SSH (Secure Shell) предназначен для защищенного удаленного доступа к компьютерам с установленной ОС GNU/Linux через стандартный порт 22. Безопасность обеспечивается путем использования различных алгоритмов шифрования данных. При публикации по протоколу SSH для ВС будет создано внешнее имя *ssh.cloud.nstu.ru:порт* и ссылка для подключения, внутренний IP-адрес и номера портов выставляются по умолчанию Чтобы подключиться по протоколу SSH  к ОС GNU\Linux можно использовать программу PuTTY для Windows или встроенный SSH  клиент для Linux.

Стек TCP/IP дает возможность организовать работу различных сетевых приложений, когда серверная часть приложения размещается на ВС, а клиентская часть – на любом узле локальной или глобальной сети. Для такой публикации указывается номер локального порта TCP/IP, который необходимо использовать для публичного доступа, и транспортный протокол. После нажатия кнопки «Добавить» ВС будет доступен для публичного доступа с указанием внешнего IP-адреса и внешнего номера порта.

На рис.2.7 приведены результаты публикации ВС по всем перечисленным вариантам.

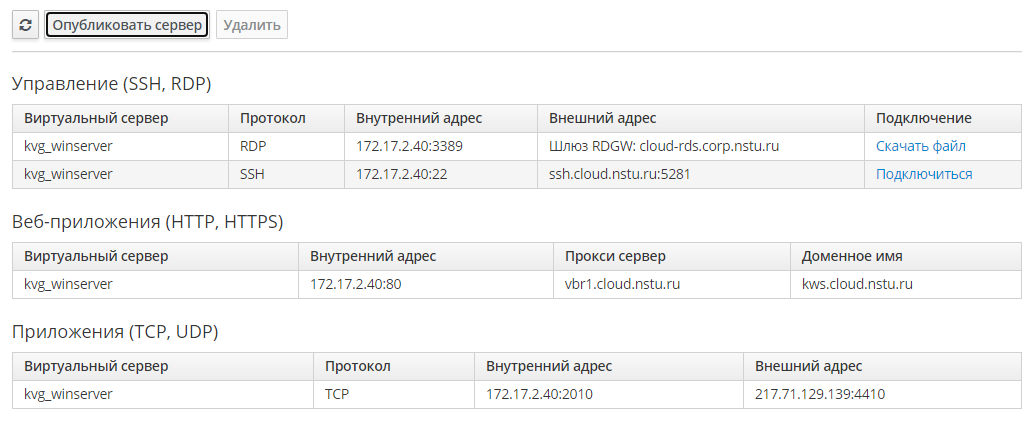


Рис. 2.7 Результаты публикации сервера

* 1. Настройка системы безопасности ОС

В составе любой современной ОС всегда присутствует система безопасности, которая используется для обнаружения и предотвращения различных сетевых атак на web-приложения. Она анализирует весь входящий и исходящий трафик и в режиме реального времени принимает решение о разрешении или запрете доступа. Политика безопасности может использовать позитивную или негативную модели безопасности, а также их комбинацию, например позволяет разрешить доступ только к определенным приложениям или открыть определенные порты.

В ОС Windows такая программа называется брандмауэр, в GNU/Linux – файрвол. При выполнении лабораторной работы на ВС рекомендуется устанавливать Windows Server 2016, поэтому далее в основном будем говорить о брандмауэре, управление которым проводится через программу Брандмауэр Windows.

Запуск программы проводится в режиме *Настройки/Сети и Интернет* с помощью ссылки *Брандмауэр Windows*. Включить и отключить брандмауэр можно в меню *Включение и отключение брандмауэра* (рис. 2.8), по умолчанию брандмауэр включен для локальной и глобальной сети (рис. 2.9). Отключение брандмауэра нежелательно, т.к. в этом случае резко снижается уровень безопасности.

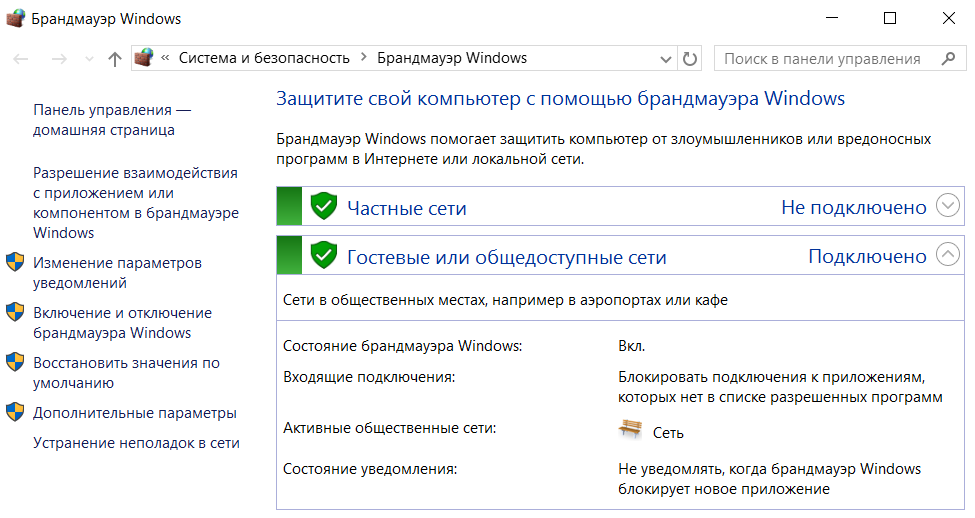


Рис. 2.8 Окно управления брандмауэра

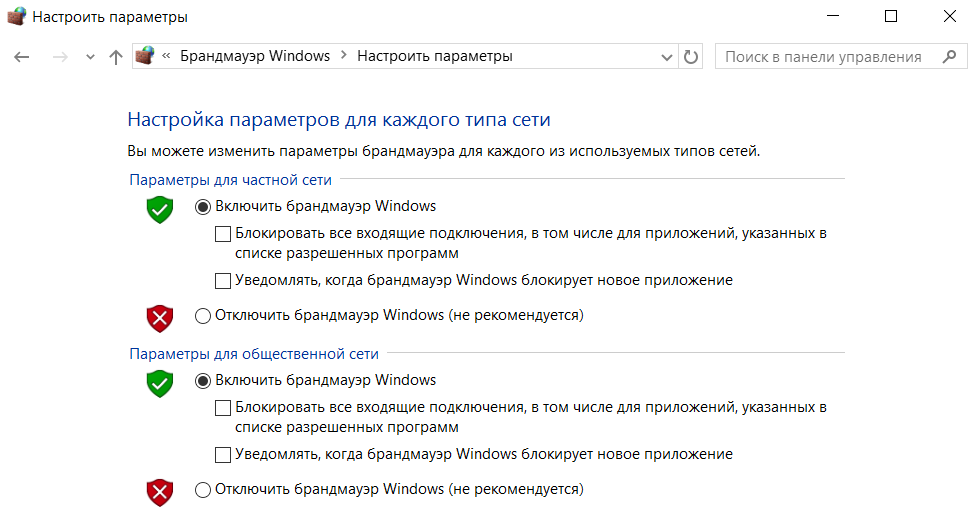


Рис. 2.9 Настройка параметров брандмауэра для локальной и глобальной сети

Более тонкую настройку брандмауэра можно выполнить через меню «Дополнительные параметры», в котором можно создавать новые правила фильтрации входящего или исходящего трафика (рис.2.10). Правила могут быть следующих типов:

* предопределенными (выбираются из стандартного списка правил);
* для конкретного объекта (определяют доступ для конкретного приложения или порта);
* настраиваемыми (объекты и режимы доступа задаются пользователем).

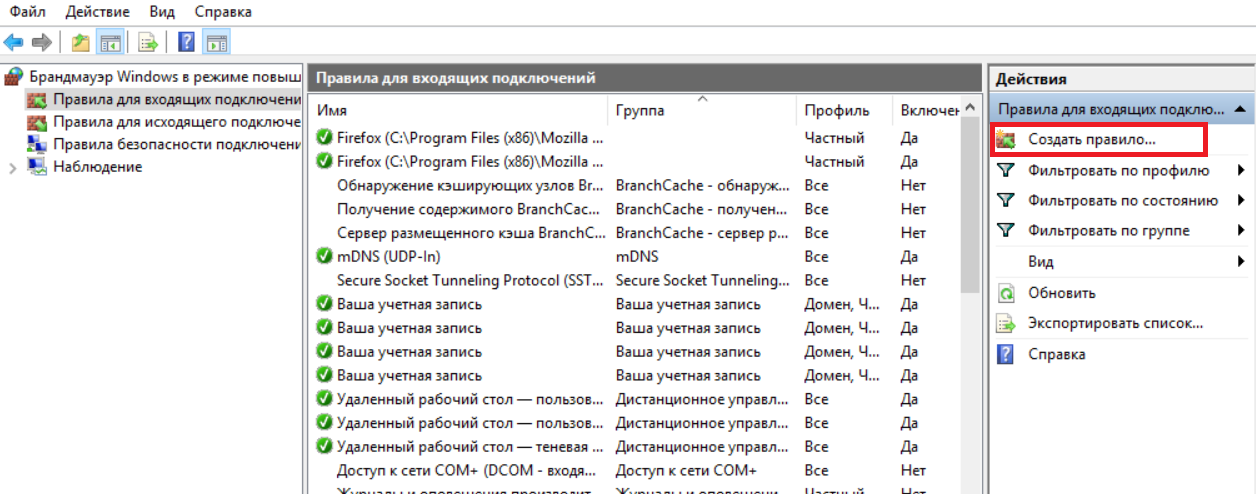


Рис.2.10 Окно дополнительных параметров брандмауэра

Например, если серверная часть сетевого приложения настроена на прослушивание локального порта 2010/TCP, то надо создать правила, которые разрешают брандмауэру открыть этот порт для входящего и исходящего подключений. В этом случае указываются тип правила (для порта), номер порта (2010), транспортный протокол (TCP), разрешение на подключение для всех профилей (доменный, частный, публичный) и имя правила. Последовательность выполнения действий показана на рис. 2.11 – 2.14.

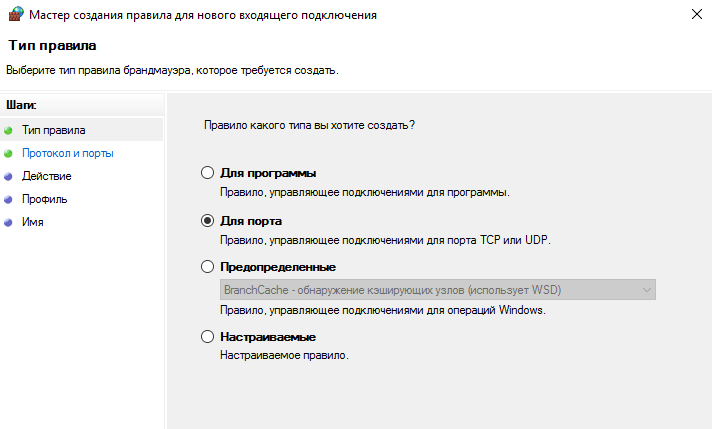


Рис.2.11 Выбор типа правила

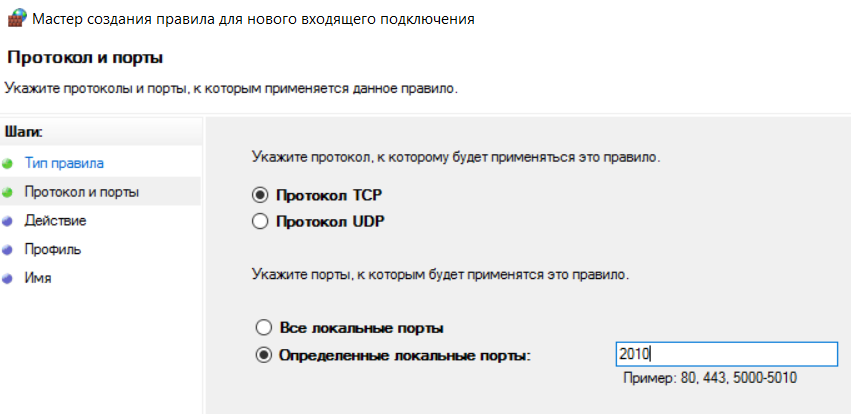


Рис.2.12 Выбор протокола и порта

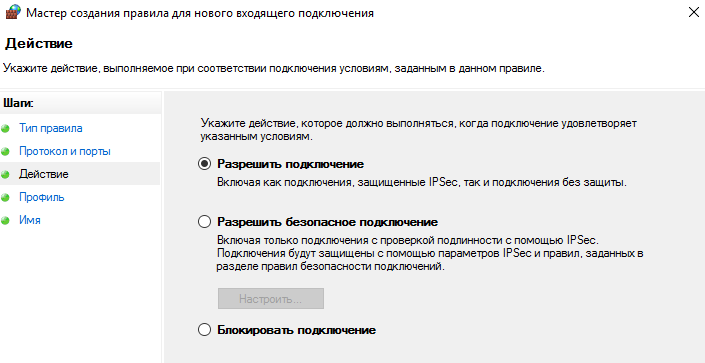


Рис.2.13 Выбор действия

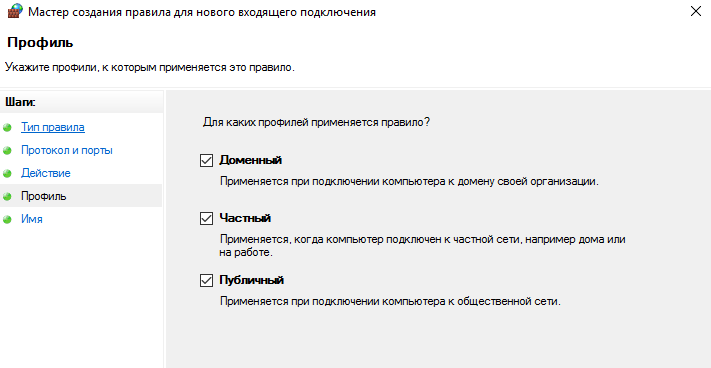
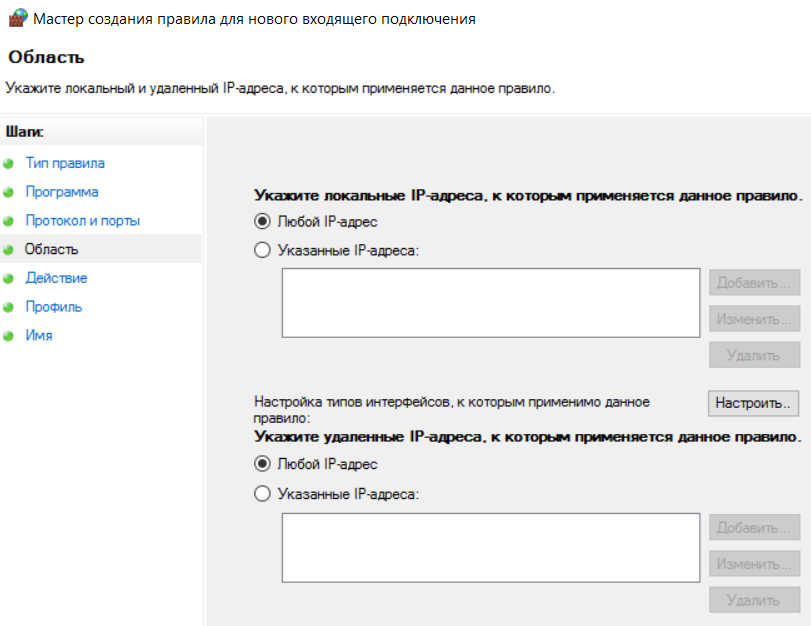


Рис.2.14 Выбор профиля

На рис.2.15 показана настройка локальных и удаленных IP-адресов, к которым применяется настраиваемое правило.



Для GNU/Linux разработаны несколько файрволов, наиболее известными из которых являются iptable, ufw, FERM, firewalld. Управление файрволом обычно проводится в командном режиме. Например, для разрешения доступа к порту 2010 в CentOS 7, где используется firewalld, надо ввести последовательность команд:

firewall-cmd --permanent --add-port=2010/tcp;

firewall-cmd --reload.

Здесь опции команд имеют следующие значения:

add-port – добавить порт в конфигурацию;

permanent –постоянная конфигурация, т.е. будет действовать до и после перезагрузки ОС;

reload – перезагрузить правила из постоянной конфигурации

# 2.5. Организация сетевого взаимодействия в Облаке

Сетевое программное обеспечение (ПО) обычно имеет два компонента – серверную и клиентскую. Серверная часть после загрузки в память компьютера переводится в режим ожидания запросов. Удаленные клиенты устанавливают соединение с сервером, обращаются к нему с запросами, получают ответы, после чего соединение разрывается. Такая технология сетевого взаимодействия называется «клиент-серверной».

Одной из основных проблем, встающих перед разработчиками сетевого ПО, является проблема отладки и тестирования, т.к. для этого необходимо иметь несколько компьютеров, подключенных к локальной или глобальной сети, что не всегда представляется возможным. Облачная платформа, обладающая мощными вычислительными ресурсами, предоставляет пользователю широкие возможности для отладки и тестирования сетевого многопоточного ПО с одновременным подключением к серверу нескольких клиентов.

Вариант 1. Студент создает ВС, загружает на него серверную и клиентскую части своего приложения и использует для тестирования локальную петлю (127.0.0.1). Этот вариант является самым простым и дает возможность изучить технологию создания и управления сервером, а также проверить работоспособность приложения. Однако с точки зрения организации сетевого взаимодействия он эквивалентен запуску сетевого приложения на локальном компьютере.

Вариант 2. Студент создает в пределах своей квоты два ВС и публикует их для работы по протоколам стека TCP/IP, получая для каждого ВС собственный внешний IP-адрес и внешний номер порта. Для организации обмена данными на один ВС загружается серверная часть приложения, на второй ВС – клиентскую часть. В этом случае подключение клиента к серверу можно проводить по локальным IP-адресам.

Вариант 3. Студент создает ВС, публикует его для работы по протоколам стека TCP/IP, получая для него внешний IP-адрес и внешний номер порта, и загружает на него серверную часть сетевого приложения. Клиентская часть устанавливается на локальный компьютер, подключение к серверу проводится по внешнему IP-адресу и внешнему номеру порта ВС.

Вариант 4. В состав Облачной платформы входят несколько терминальных серверов, которые могут быть использованы для отладки и тестирования сетевых приложений. На сервере Cloud Dev установлено ПО, предназначенное в основном для разработчиков программного обеспечения, на сервере Cloud Math – с уклоном на математические и проектные дисциплины, а сервер Ansys содержит базовый комплект офисного ПО и систему конечно-элементного анализа Ansys 2020 R2. Доступ к серверам проводится с помощью единой учетной записи НГТУ по протоколу RDP, файл настройки можно скачать по адресу <https://cloud.nstu.ru/wiki/терминальные_сервера>.

Для выполнения лабораторной работы рекомендуется использовать сервер CloudDev. По умолчанию брандмауэр этого сервера запрещает доступ внешним программам к портам даже в пределах локальной сети Облака, а изменение его параметров доступно только с правами администратора. В этих условиях возможно использовать следующую технологию:

* копировать на сервер CloudDev клиентскую часть сетевого приложения, а на ВС – серверную часть;
* настроить брандмауэр, открыв нужный порт для входящего и исходящего трафика сервера;
* запустить сервер и подключить к нему клиента, размещенного на CloudDev.

1. Задание к лабораторной работе
   1. Общее задание
2. Ознакомиться с правилами работы в Облаке НГТУ.
3. На основе программ, разработанных при выполнении лабораторных работ № 2 и 3 подготовить тестовые программы, которые будут использоваться при дальнейшем выполнении работы. Функциональность программ должна соответствовать требованиям заданий, изложенных в описании работ № 2 и 3. Дополнительные требования к серверным программам:

* возможность обработки запросов, поступающих из любых типов сетей (локальной, глобальной или внутренней петли);
* номер прослушивающего порта вводить с клавиатуры;
* выводить на экран IP-адрес компьютера, на котором загружается серверная программа;
* при подключении каждого клиента выводить на экран его IP-адрес и номер порта.

1. Создать облачный виртуальный сервер (ВС), имя которого должно содержать номер группы, номер бригады и номер студента в бригаде, например pmi-8105-1. Параметры сервера можно указать любые в пределах своей квоты, операционная система – Windows Server 2016.
2. Выполнить публикацию ВС, указав тип публикации «Приложение TCP/IP». Укажите тип транспортного протокола и номер порта, связанные с сокетом Вашего серверного приложения. После публикации ВС получит внешний IP-адрес и внешний номер порта.
3. Выполнить повторную публикацию ВС, указав тип публикации «Управление RDP». Скачать на локальный компьютер исполняемый файл настройки удаленного соединения.
4. Включить ВС, подключиться к нему с локального компьютера по протоколу RDP и установить распространяемый компонент Microsoft Visual C++ 2015, который устанавливает компоненты среды выполнения библиотек Visual C++, C Runtime (CRT), Standard C++, MFC, C++ AMP и OpenMP, что  позволит запускать приложения, разработанные для среды .NET без установки громоздкого пакета Visual Studio 2015. Скачать распространяемый компонент можно с официального сайта Microsoft: <https://www.microsoft.com/ru-ru/download/details.aspx?id=53840>
5. Настроить брандмауэр для работы серверной части приложения. Открыть порт сервера для входящего и исходящего трафика.
6. Загрузить на ВС серверную и клиентскую части своего приложения «Эхо-повтор» из лабораторной работы № 2, проверить их работоспособность с использованием локальной петли (127.0.0.1).
7. Загрузить на ВС одного студента серверную часть приложения «Эхо-повтор», а на ВС другого студента – клиентскую часть приложения «Эхо-повтор». Организовать обмен данными, подключив клиента к серверу по его внутреннему IP-адресу.
8. Загрузить клиентскую часть приложения «Эхо-повтор» на локальный компьютер. Организовать обмен данными между этим клиентом и сервером «Эхо-повтор», используя для подключения внешний IP-адрес ВС.
9. Подключиться к терминальному серверу CloudDev и загрузить на него клиентскую часть приложения «Эхо-повтор». Организовать обмен данными между этим клиентом и сервером «Эхо-повтор», используя для подключения внутренний IP-адрес ВС.
10. Выполнить пункты 1-3 с приложением интерактивной переписки из лабораторной работы № 3.

3.2 Дополнительное задание

1. Создать второй ВС, работающий под управлением GNU/Linux в соответствии с вариантом из таблицы 6.1. Вариант выбирается в соответствии с номером бригады с использованием циклической нумерации: бригада № 6 выбирает вариант № 1, бригада № 7 – вариант № 2 и т.д.

Таблица 6.1

|  |  |
| --- | --- |
| Вариант | Операционная система |
| 1 | CentOS 7.5 Linux Minimal |
| 2 | CentOS 8 Linux Minimal |
| 3 | Fedora Server 28 |
| 4 | Debian Linux 9 |
| 5 | Ubuntu Server 18.04 LTS |

1. При необходимости внести изменения в серверную часть приложения «Эхо-повтор» для работы в среде GNU/Linux. Отладку программы можно выполнить на сервере *fpm2.ami.nstu.ru*.
2. Разместить серверную часть приложения на втором ВС, настроить файрвол для работы с портом сервера.
3. Выполнить пункты 9-11 общего задания, используя в качестве сервера второй ВС.

Результаты выполнения всех пунктов задания заносить в отчет в виде скриншотов и обязательных пояснений. На скриншотах должны быть четко видны IP-адреса, номера портов и результаты обмена данными между клиентской и серверной частями приложений.

### 3. Контрольные вопросы.

1. Назначение облачной платформы и ее отличие от файлообменных сервисов.

2. Порядок создания облачного сервера.

3. Что такое публикация облачного сервера, типы публикации.

4. Поясните различия между терминальным сервером и виртуальным сервером.

5. Варианты отладки сетевых приложений в облачной платформе.

6. Можно ли организовать обмен данными между облачными серверами, работающими под управлением ОС Windows и GNU/Linux, поясните причину.

7. Брандмауэр: назначение и возможности управления.

8. Поясните различия между удаленным управлением по RDP и по SSH.

# СПИСОК ИСТОЧНИКОВ

1. Олифер В.Г., Олифер Н.А. Компьютерные сети . Принципы, технологии, протоколы: учебник для вузов, 4-е изд.– СПб: Питер, 2010. - 944 c.
2. Камер Дуглас. Компьютерные сети и Internet. Разработка приложений для Internet: Пер. с англ. –М.: Издательский дом «Вильямс», 2002.-640с.
3. 2.Куроуз Дж., Росс К. Компьютерные сети. Многоуровневая архитектура Интернета. 2-е изд. –СПб.: Питер, 2004. -765с.

# Поисов Д. А. Начинаем работать с VirtualBox, <http://all-ht.ru/inf/vpc/VirtualBox.html>

1. Семёнов Ю.А. (ГНЦ ИТЭФ), Телекоммуникационные технологии. <http://citforum.ru/nets/semenov/>
2. Моск Д. Настройка Firewalld в CentOS. <https://www.dmosk.ru/miniinstruktions.php?mini=firewalld-centos>
3. Мартин urban.prankster Пранкевич. Изучаем firewalld – простой и мощный файрвол для Linux на замену iptables. <https://xakep.ru/2017/02/15/firewalld/>
4. Как настроить Хамачи. http://hamachi-pc.ru/instructions/kak-nastroit-hamachi.html

# Приложение 1

## Пример API-интерфейса прикладного программирования для обмена данными по сети.

### Пример кода эхо-сервера

**server.cpp**#pragma comment (lib,"Ws2\_32.lib")

#include <winsock2.h>

#include <stdio.h>

#include <iostream>

#include <sstream>

#include <string>

using namespace std;

int main(void)

{

WORD sockVer;

WSADATA wsaData;

int retVal;

sockVer = MAKEWORD(2,2);

WSAStartup(sockVer, &wsaData);

//Создаем сокет

SOCKET servSock = socket(PF\_INET, SOCK\_STREAM, IPPROTO\_TCP);

if(servSock == INVALID\_SOCKET)

{

printf("Unable to create socket\n");

WSACleanup();

system("pause");

return SOCKET\_ERROR;

}

SOCKADDR\_IN sin;

sin.sin\_family = PF\_INET;

sin.sin\_port = htons(3114);

sin.sin\_addr.s\_addr = INADDR\_ANY;

retVal = bind(servSock, (LPSOCKADDR)&sin, sizeof(sin));

if(retVal == SOCKET\_ERROR)

{

printf("Unable to bind\n");

WSACleanup();

system("pause");

return SOCKET\_ERROR;

}

printf("Server started at %s, port %d\n", inet\_ntoa(sin.sin\_addr), htons(sin.sin\_port));

while(true)

{

//Пытаемся начать слушать сокет

retVal = listen(servSock, 10);

if(retVal == SOCKET\_ERROR)

{

printf("Unable to listen\n");

WSACleanup();

system("pause");

return SOCKET\_ERROR;

}

//Ждем клиента

SOCKET clientSock;

SOCKADDR\_IN from;

int fromlen=sizeof(from);

clientSock = accept(servSock, (struct sockaddr\*)&from, &fromlen);

if(clientSock == INVALID\_SOCKET)

{

printf("Unable to accept\n");

WSACleanup();

system("pause");

return SOCKET\_ERROR;

}

printf("New connection accepted from %s, port %d\n", inet\_ntoa(from.sin\_addr), htons(from.sin\_port)) ;

char szReq[256];

retVal = recv(clientSock, szReq, 256, 0);

//Пытаемся получить данные от клиента

if(retVal == SOCKET\_ERROR)

{

printf("Unable to recv\n");

system("pause");

return SOCKET\_ERROR;

}

printf("Data received\n");

string s = (const char\*) szReq;

if(s[0]=='s')// Команда на выключение сервера

{

char \*szResp = "Server shutdown";

retVal = send(clientSock, szResp, 256, 0);

closesocket(clientSock);

break;

}

else

{

istringstream ss(s);

int a, b;

ss >> a;

ss >> b;

char szResp[256];

sprintf(szResp,"%d", a+b);

//Пытаемся отослать данные клиенту

printf("Sending response from server\n");

retVal = send(clientSock, szResp, 256, 0);

if(retVal == SOCKET\_ERROR)

{

printf("Unable to send\n");

system("pause");

return SOCKET\_ERROR;

}

closesocket(clientSock);

printf("Connection closed\n");

}

}

//Закрываем сокет

closesocket(servSock);

WSACleanup();

return 0;

}

Пример кода клиента службы эхо - повтора

**client.cpp**

#pragma comment (lib,"Ws2\_32.lib")

#include <stdio.h>

#include <winsock2.h>

#include <string>

#include <iostream>

using namespace std;

int main()

{

WORD ver = MAKEWORD(2,2);

WSADATA wsaData;

int retVal=0;

WSAStartup(ver,(LPWSADATA)&wsaData);

LPHOSTENT hostEnt;

hostEnt = gethostbyname("localhost");

if(!hostEnt)

{

printf("Unable to collect gethostbyname\n");

WSACleanup();

system("pause");

return 1;

}

//Создаем сокет

SOCKET clientSock = socket(PF\_INET, SOCK\_STREAM, IPPROTO\_TCP);

if(clientSock == SOCKET\_ERROR)

{

printf("Unable to create socket\n");

WSACleanup();

system("pause");

return 1;

}

string ip;

cout << "ip>";

cin >> ip;

cin.ignore();

SOCKADDR\_IN serverInfo;

serverInfo.sin\_family = PF\_INET;

serverInfo.sin\_addr.S\_un.S\_addr = inet\_addr(ip.c\_str());

serverInfo.sin\_port = htons(3114);

//Пытаемся присоединится к серверу по ip и port

retVal=connect(clientSock,(LPSOCKADDR)&serverInfo, sizeof(serverInfo));

if(retVal==SOCKET\_ERROR)

{

printf("Unable to connect\n");

WSACleanup();

system("pause");

return 1;

}

printf("Connection made sucessfully\n");

printf("Enter a and b or 'stop' to shutdown server\n");

char pBuf[256];

gets(pBuf);

printf("Sending request from client\n");

//Отсылаем данные на сервер

retVal = send(clientSock, pBuf, strlen(pBuf), 0);

if(retVal == SOCKET\_ERROR)

{

printf("Unable to send\n");

WSACleanup();

system("pause");

return 1;

}

char szResponse[256];

//Пытаемся получить ответ от сервера

retVal = recv(clientSock, szResponse, 256, 0);

if(retVal == SOCKET\_ERROR)

{

printf("Unable to recv\n");

WSACleanup();

system("pause");

return 1;

}

char \*Resp;

Resp = szResponse;

if(pBuf[0] != 's')

printf("a + b = %s\n",Resp);

else

printf("%s\n",Resp);

closesocket(clientSock);

WSACleanup();

system("pause");

return 0;

}

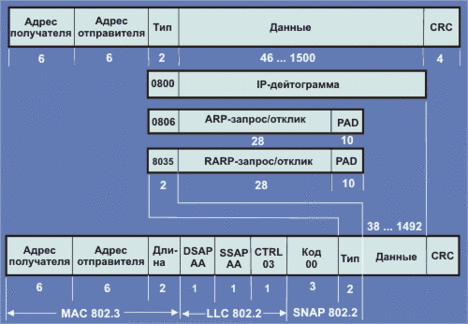
# Приложение 2

## Основные единицы обмена для различных уровней стека TCP/IP

### 2.1. Типы кадров технологии Ethernet (размеры полей указаны в байтах)



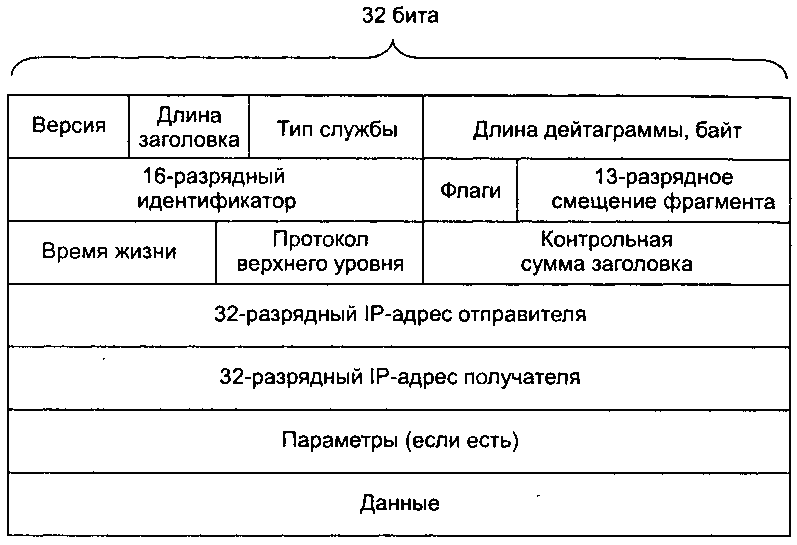
### 2.2. Ethernet инкапсуляция (RFC 894, размеры полей указаны в байтах.)



### 2.3. Алгоритм определения формата кадра

Отличить один формат кадра Ethernet от другого не представляет большого труда, и сделать это можно с помощью следующего простого алгоритма. Сначала программа должна проверить значение поля типа протокола/длины кадра (13-й и 14-й байты в заголовке). Если записанное там значение превышает 0x05DC (максимально возможная длина поля данных), то это кадр Ethernet\_II, иначе следует продолжить проверку. Если первые два байта поля данных равны 0xFFFF, то это формат Ethernet\_802.3 для NetWare 3.х. В противном случае это стандартный формат кадра 802.2, и остается только выяснить, какой из двух - обычный (Ethernet\_802.2) или расширенный (Ethernet\_SNAP). В случае Ethernet\_SNAP значение первого и второго байтов в поле данных равняется 0xAA.

2.4. Ключевые поля IPv4-дейтаграммы.



***□ Версия.*** Четыре бита в этом поле определяют номер версии протокола IP. По этому номеру маршрутизатор может определить, как интерпретировать остальные поля IP-дейтаграммы. В различных версиях протокола IP применяются различные форматы IP-дейтаграмм. На рисунке показан формат дейтаграммы текущей версии протокола IP **(IPv4)**.

***□ Длина заголовка.*** Поскольку IPv4-дейтаграмма может содержать разное количество необязательных полей параметров (включаемых в заголовок IPv4-дейтаграммы), эти четыре бита необходимы для того, чтобы определить, где заканчивается заголовок и начинаются данные. В большинстве IP-дейтаграмм не содержатся поля параметров, поэтому обычно заголовок IP-дейтаграммы 20-байтный.

***□ Тип службы.*** Поле типа службы (Type Of Service, TOS) было включено в заголовок IPv4-дейтаграммы, чтобы была возможность разделять IP-дейтаграммы на типы (например, выделять дейтаграммы, для которых требуется низкая задержка, или высокая пропускная способность, или высокая надежность). Так, может оказаться полезным отличать дейтаграммы реального времени (например, используемые в IP-телефонии) от прочего трафика (например, FTP).

***□ Длина дейтаграммы.*** Это полная длина IP-дейтаграммы (*заголовок плюс данные*) в байтах. Поскольку размер этого поля равен 16 бит, теоретически максимальный размер IP-дейтаграммы может составлять 65 535 байт. Однако размер дейтаграмм редко превосходит 1500 байт и обычно ограничивается значением 576 байт.

***□ Идентификатор, флаги, смещение фрагмента.*** Эти три поля имеют отношение к так называемой IP-фрагментации. Этот вопрос мы подробно рассмотрим чуть позже. Интересно отметить, что новая версия протокола IP (IPv6) запрещает фрагментацию в маршрутизаторах.

***□ Время жизни.*** Поле времени жизни (Time То Live, TTL) позволяет гарантировать, что дейтаграммы не будут вечно циркулировать в сети (например, из-за существующей в течение долгого времени маршрутной петли). Значение этого поля уменьшается на единицу на каждом маршрутизаторе. Когда значение поля TTL достигает нуля, маршрутизатор отбрасывает дейтаграмму.

***□ Протокол.*** Это поле используется только тогда, когда IP-дейтаграмма достигает конечного адресата. Значение поля определяет протокол транспортного уровня, которому следует передать данные из IP-дейтаграммы. Например, значение 6 означает, что порция данных должна быть передана протоколу TCP, а значение 17 — протоколу UDP. Список всех возможных номеров имеется в RFC 1700, RFC 3232. Обратите внимание, что роль номера протокола в IP-дейтаграмме полностью аналогична роли номера порта в сегменте транспортного уровня. Номер протокола представляет собой «клей», связывающий вместе сетевой и транспортный уровни, тогда как номер порта связывает транспортный уровень с прикладным ( в кадре канального уровня также есть специальное поле, связывающее канальный уровень с сетевым уровнем).

***□ Контрольная сумма заголовка.*** Контрольная сумма заголовка помогает маршрутизатору обнаруживать ошибки в полученных IP-дейтаграммах. Контрольная сумма заголовка вычисляется путем суммирования всех двухбайтовых слов заголовка в дополнительном коде. Маршрутизатор вычисляет контрольную сумму заголовка для каждой полученной дейтаграммы и таким образом проверяет ошибки в заголовке. Как правило, маршрутизаторы отбрасывают дейтаграммы, в которых обнаруживают ошибки. Обратите внимание, что контрольную сумму нужно вычислять заново и снова сохранять в поле заголовка на каждом маршрутизаторе, так как на единицу уменьшается поле времени жизни, могут также измениться поля параметров (описание быстрых алгоритмов для вычисления контрольной суммы заголовка IP-дейтаграммы содержится в RFC 1071).

***□ IP-адреса отправителя и получателя.*** Эти поля содержат 32-разрядные IP-адреса отправителя и конечного получателя IP-дейтаграммы.

***□ Параметры.*** Поле параметров позволяет расширить IP-заголовок. Параметры заголовка представляют собой редко используемые необязательные поля IP-дейтаграммы.

***□ Данные (полезная нагрузка).*** Наконец, мы добрались до последнего, самого важного поля, ради которого и существует дейтаграмма! В большинстве случаев поле данных IP-дейтаграммы содержит сегмент транспортного уровня (TCP или UDP), который необходимо доставить адресату. Однако поле данных может содержать и другие типы данных, например сообщения протокола ICMP.

### 2.5. Формат UDP-сообщений



Формат UDP-дейтаграммы.

***Основные поля:***

***Длина UDP-сообщения*** равна числу байт в UDP-дейтаграмме, включая заголовок.

***Контрольная сумма*** содержит код, полученный в результате контрольного суммирования UDP-заголовка и поля данные. (Не трудно видеть, что этот протокол использует заголовок минимального размера (***8 байт***)).

Номера портов от 0 до 1023 стандартизованы - использовать их в прикладных задачах не рекомендуется, поэтому прежде, чем использовать какой-то порт в своей программе, следует заглянуть в RFC-1700.

### 2.6. Формат TCP - сегмента

Протокол **TCP (transmission control protocol**, RFC-793, -1323, -1644[T/TCP], -2018, -2581, -2582[RENO], -2861, -2873, -2883[SACK], -2923[MTU], -2988[RTO], -3293[GSMP], -3448[TFRC], -3465, -3481) в отличии от UDP осуществляет доставку дейтограмм, называемых сегментами, в виде байтовых потоков с установлением соединения. Протокол TCP применяется в тех случаях, когда требуется гарантированная доставка сообщений. Он использует контрольные суммы пакетов для проверки их целостности, и освобождает прикладные процессы от необходимости таймаутов и повторных передач для обеспечения надежности. Для отслеживания подтверждения доставки в TCP реализуется алгоритм "скользящего" окна. Внутренняя структура модуля TCP гораздо сложнее структуры UDP. Подобно UDP прикладные процессы взаимодействуют с модулем TCP через порты. Под байтовыми потоками здесь подразумевается то, что один примитив, например, **read** или **write** может вызвать посылку адресату последовательности сегментов, которые образуют некоторый блок данных (сообщение).

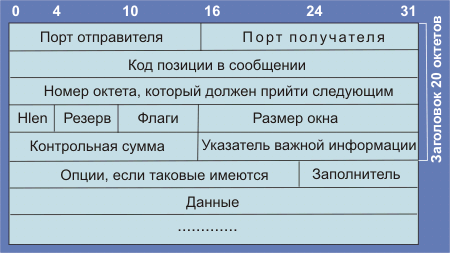
Хотя протоколы UDP и TCP могли бы для сходных задач использовать разные номера портов, обычно этого не происходит. Модули TCP и UDP выполняют функции мультиплексоров/демультиплексоров между прикладными процессами и IP-модулем. При поступлении пакета в модуль IP он будет передан в TCP- или UDP-модуль согласно коду, записанному в поле протокола данного IP-пакета.

Если IP-протокол работает с адресами, то TCP, также как и UDP, с портами. Именно с ***номеров портов*** отправителя и получателя начинается заголовок TCP-сегмента.

Поле ***код позиции в сообщении*** определяет порядковый номер первого октета в поле данных пользователя.

Поле ***Hlen*** – определяет длину заголовка сегмента, которая измеряется в 32-разрядных словах.

Далее следует поле ***резерв***, предназначенное для будущего использования, в настоящее время должно обнуляться.



Формат TCP сегмента

Поле ***размер окна*** сообщает, сколько октетов готов принять получатель. Окно имеет принципиальное значение, оно определяет число сегментов, которые могут быть посланы без получения подтверждения. Значение ширины окна может варьироваться во время сессии. Значение этого поля равное нулю также допустимо и указывает, что байты вплоть до указанного в поле *номер октета, который должен прийти следующим*, получены, но адресат временно не может принимать данные. Разрешение на посылку новой информации может быть дано с помощью посылки сегмента с тем же значением поля *номер октета, который должен прийти следующим*, но ненулевым значением поля ширины окна.

Поле ***контрольная сумма*** предназначено для обеспечения целостности сообщения.

Поле ***указатель важной информации*** представляет собой указатель последнего байта, содержащий информацию, которая требует немедленного реагирования.

Поле ***опции*** зарезервировано на будущее и в заголовке может отсутствовать, его размер переменен и дополняется до кратного 32-бит с помощью поля *заполнитель*.

Поле ***данные*** в TCP-сегменте может и отсутствовать, характер и формат передаваемой информации задается исключительно прикладной программой, максимальный размер этого поля составляет в отсутствии опций 65495 байт. TCP является протоколом, который ориентируется на согласованную работу хостов и программного обеспечения партнеров, участвующих в обмене информацией.