Лабораторная работа № 3

Принципы построения сетей TCP/IP

Теоретические сведения.

Введение.

Одной из основных задач, решаемых вычислительными системами, является передача данных от одной системы к другой. Для обеспечения возможности такого объединения систем в компьютерные сети разработаны и широко применяются различные сетевые технологии. Реализация взаимодействий между компьютерными системами базируется на сетевых протоколах — наборах правил, позволяющих осуществлять соединение и обмен данными между двумя и более включёнными в сеть компьютерами. Разные протоколы зачастую описывают лишь разные стороны одного типа связи; взятые вместе, они образуют стек протоколов. Названия «протокол» и «стек протоколов» также указывают на программное обеспечение, которым реализуется протокол. Практически все существующие сейчас в мире компьютерные сети объединены или, как минимум, имеют шлюзы в глобальную компьютерную сеть Internet. Понимание базовых принципов работы сети Internet и лежащего в её основе протокола TCP/IP является в настоящее время необходимым для каждого грамотного технического специалиста.

История развития Internet.

В 1969 году в Министерстве обороны США (Department of Defense, DoD) было решено построить сеть для передачи информации, которая сохраняла бы работоспособность в случае выхода из строя (уничтожения в ядерной войне) значительной части своих узлов. Разработка такой сети была поручена Агентству по перспективным оборонным научно-исследовательским разработкам (Defense Advanced Research Projects Agency, DARPA). Непосредственные работы по созданию сети были поручены Калифорнийскому университету в Лос-Анджелесе, Стэнфордскому исследовательскому центру, Университету штата Юта и Университету штата Калифорния в Санта-Барбаре. Созданная в рамках проекта компьютерная сеть, объединившая четыре вышеназванных научных учреждения, получила название ARPANET. Первый сервер ARPANET был запущен 1 сентября 1969 года в Калифорнийском университете.

В 1971 году для ARPANET была разработана первая программа для отправки электронной почты по сети, в 1973 году сеть стала международной — через трансатлантический телефонный кабель к ней были подключены организации из Великобритании и Норвегии. В 1970-х годах сеть в основном использовалась для обмена сообщениями электронной почты, включая первые появившиеся списки почтовой рассылки, новостные группы и электронные доски объявлений.

Для работы ARPANET использовался сетевой протокол NCP. Параллельно с развитием сети велись работы по развитию и усовершенствованию этого протокола. В 1973 году Роберт Эллиот (Боб) Кан (Robert Elliot Kahn, Bob

Kahn) и Винтон Грей Серф (Vinton Gray "Vint" Cerf) начали работу над созданием протокола TCP/IP (Transmission Control Protocol/Internet Protocol), предназначенного для объединения разнородных компьютерных сетей, работающих на разных сетевых протоколах и технологиях, в единое целое. В виде стандартов протокол был оформлен в 1973-1974 гг, также при содействии DARPA, и в 1975 году была запущена первая тестовая сеть TCP/IP. В марте 1982 года Министерство обороны США приняло TCP/IP как стандартный протокол в своих сетях.

1 января 1983 года сеть ARPANET перешла с протокола NCP на TCP/IP и стала называться Internet.

В 1984 году была запущена система доменных имён DNS (Domain Name Service, DNS).

В 1984 году Национальный научный фонд США (NSF) основал обширную межуниверситетскую сеть NSFNet (National Science Foundation Network). NSFNet была составлена из более мелких сетей, всего в течение 1984 года к ней было подключено около 10 тыс. компьютеров. ARPANET вошла в NSFNET в качестве национальной магистрали. С 1985 года начались подключения к NSFNET коммерческих организаций и частных сетей. К 1994 году процесс перевода Internet в коммерческое русло был завершён, в апреле 1994 года NFSNET была отключена. С тех пор глобальная сеть — это совокупность частных сетей, принадлежащих провайдерам Internet.

Стандартизация сетевых протоколов.

Так как взаимодействовать через Internet могут совершенно разные компьютерные системы, особое значение имеет стандартизация протоколов взаимодействия между ними. Своим появлением в качестве глобальной информационной сети Internet в числе прочего обязана открытым и общедоступным стандартам на использующиеся в ней протоколы. На данный момент вопросами управления и стандартизации в Internet занимаются несколько основных организаций. Из них можно отметить:

ISOC (https://isoc.org, Internet Society, общество Интернета) — международная некоммерческая образовательная организация, занимающаяся развитием и обеспечением доступности сети Интернет. Создана в 1992 году и насчитывает более 20 тысяч индивидуальных членов и более 100 организаций-членов в 180 странах мира. ISOC предоставляет организационную основу для множества других консультативных и исследовательских групп, занимающихся развитием Internet, включая IETF и IAB.

IETF (https://ietf.org, Internet Engineering Task Force, Специальная комиссия интернет-разработок) — открытое международное сообщество проектировщиков, учёных, сетевых операторов и провайдеров, созданное в 1986 году и занимающееся развитием протоколов и архитектуры Internet. Вся техническая работа осуществляется в рабочих группах IETF и открыта для публичного участия и обсуждения.

IAB (Internet Architecture Board, Совет по архитектуре Интернета) — группа технических советников ISOC, которая осуществляет надзорные функции за архитектурой Internet и созданием новых стандартов, а также консультации руководства ISOC по техническим, архитектурным и процедурным вопросам.

ICANN (Internet Corporation for Assigned Names and Numbers) — организация, занимающаяся вопросами распределения адресов IP и ведением перечней номеров портов стандартных протоколов. Непосредственно выделением адресов и номеров портов занимается IANA (Internet Assigned Numbers Authority, «Администрация адресного пространства Интернет»).

Результаты работы комитетов IETF получают своё отражение в серии документов, известных как RFC (https://www.rfc-editor.org, *англ.* Requests For Comments, Запрос комментариев). В виде RFC оформляются стандарты протоколов, предлагаемые технические изменения, информационные бюллетени, технические спецификации и точные определения по многим вопросам. Часть документов RFC возводятся IAB в статус Стандартов Интернета (*англ.* Internet Standard).

RFC оформляются в виде простых текстовых документов на английском языке, с дополнительной регламентацией использования различных оборотов речи для исключения возможности неоднозначного толкования.

Документы RFC имеют порядковые номера. По состоянию на май 2025 года было принято порядка 9778 RFC, за год с мая 2024 года добавилось 218 RFC. После принятия RFC никогда не меняются, хотя могут быть дополнены или признаны устаревшими в последующих RFC. Обновлённые документы должны содержать в себе всю актуальную информацию их предшественников. Ссылаться на документы RFC принято по их порядковому номеру, хотя каждый из них имеет и название.

Нормативы по оформлению и порядку рассмотрения и принятия RFC регламентированы в RFC 2026 «The Internet Standards Process -- Revision 3» и обновляющим и дополняющим его RFC 6410 «Reducing the Standards Track to Two Maturity Levels».

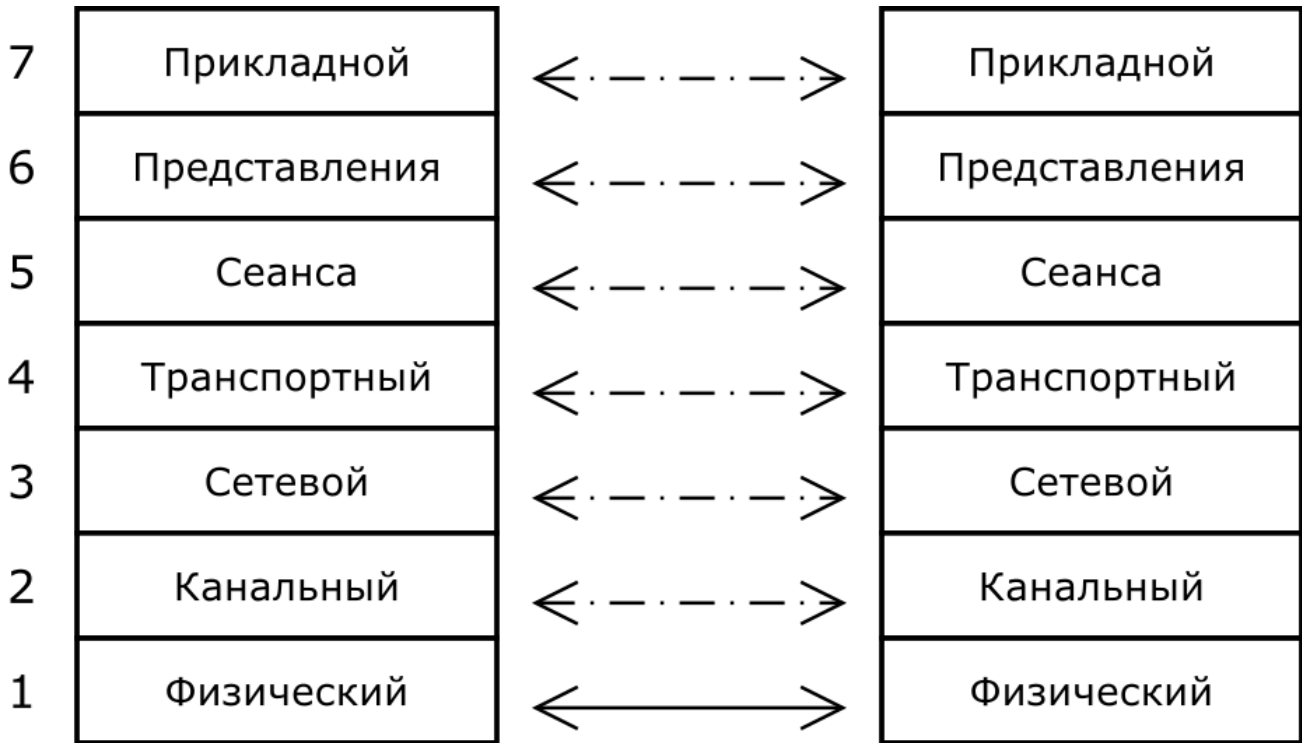
Не все RFC содержат чисто техническую информацию. Ряд RFC содержит рекомендации по порядку настройки информационных систем, организации работ и процессов (т. н. best practices, лучшие практики) – например, RFC 4086 от 08 июня 2005 года «Randomness Requirements for Security» описывает рекомендации для выбора начальных значений алгоритмов генерации псевдо-случайных чисел, которые предотвращают возникновение повторяющихся псевдо-случайных последовательностей, в т.ч. при перезагрузке компьютерных систем. Есть традиция выпуска первоапрельских RFC – например, RFC 5241 от 1 апреля 2008 года «Naming Rights in IETF Protocols» описывает порядок использования коммерческих брендов в качестве имен для описания понятий в сетевых протоколах с целью получения дополнительного источника дохода для IETF, RFC 8367 от 1 апреля 2018 года

«Wrongful Termination of Internet Protocol (IP) Packets» рекомендует воздерживаться от фильтрации пакетов IP по их возрасту, длине, стране происхождения и т. п., RFC 9564 от 1 апреля 2024 года «Faster Than Light Speed Protocol (FLIP)» описывает сетевой протокол FLIP, использующий генеративный искусственный интеллект (ИИ) типа ChatGPT для предсказания содержимого ещё не полученных сетевых пакетов и использования предсказанных пакетов для ускорения сетевого трафика, RFC 9759 от 1 апреля 2025 года «Unified Time Scaling for Temporal Coordination Frameworks» вводит новую единицу измерения времени TWP (Two-Week Principle), равную двум неделям, для использования при указании сроков выполнения задач в процессах управления и разработки программного обеспечения и приравнивает к ней все другие возможные использующиеся в этих процессах единицы. RFC 9948 от 1 апреля 2026 года «*Internet Protocol Police (IPP) - Schedule of Punishments*» регламентирует применяемые персоналом полиции протоколов (которая была создана в RFC 8962 «*Establishing the Protocol Police*» от 1 апреля 2021 г. для отслеживания нарушений RFC, заявления в IPP о нарушениях должны направляться в устройство `/dev/null`) меры воздействия на нарушителей, от «The Raised Eyebrow» до «The Head-in-Hand Gesture».

Также существуют чисто информационные RFC – например, RFC 8700 от декабря 2019 года «Fifty Years of RFCs» – заметка, отмечающая 50-летие RFC (и обновляющая RFC 5540 от 7 апреля 2009 года «40 Years of RFCs»).

Сетевая модель OSI ISO.

Существует общая модель сетевых протоколов — т.н. модель OSI (Open System Interconnection, Взаимодействие открытых систем) ISO (International Organization for Standardization, Международная организация по стандартизации). Эта модель, разработанная в начале 80-х годов прошлого века, описывает эталонную структуру взаимодействий компьютерных систем, которой желательно придерживаться при проектировании сетевых протоколов. В модели OSI проблема связи между компьютерами разделяется на семь уровней. Каждый уровень обслуживает свою часть процесса взаимодействия и реализует свои протоколы.



В модели OSI ISO выделяются следующие 7 уровней:

- 7ой - прикладной уровень (Application layer)

Верхний уровень модели, на котором выполняются взаимодействующие с пользователем приложения.

- 6ой - уровень представления (Presentation layer)

Отвечает за преобразование протоколов и кодирование/декодирование данных. На этом уровне может осуществляться сжатие и распаковка данных, их кодирование и декодирование.

- 5ый - сеансовый уровень (Session layer)

Отвечает за поддержание сеанса связи между приложениями, позволяя им взаимодействовать между собой длительное время. Также на этом уровне

выполняется проверка данных, обрабатываются ошибки передачи данных, синхронизация их потоков.

- 4ый - транспортный уровень (Transport layer)

Предназначен для доставки данных без ошибок, потерь и дублирования в той последовательности, как они были переданы. На этом уровне уже не важно, данные каких конкретно приложений передаются, откуда и куда, уровень предоставляет сам механизм передачи.

- 3ий - сетевой уровень (Network layer)

Предназначен для определения пути передачи данных, отвечает за трансляцию логических адресов и имён в физические, определение кратчайших маршрутов, коммутацию и маршрутизацию, отслеживание неполадок и заторов в сети. На этом уровне работают маршрутизаторы (*англ.* router).

- 2ой - канальный уровень (Data Link layer)

Предназначен для обеспечения взаимодействия сетей на физическом уровне и контроля за ошибками, которые могут возникнуть. Канальный уровень может взаимодействовать с несколькими физическими уровнями, контролируя и управляя этим взаимодействием. На этом уровне работают коммутаторы (*англ.* switch) и мосты (*англ.* bridge). В операционных системах работой на канальном уровне занимаются драйверы сетевых карт.

- 1ый - физический уровень (Physical layer)

Предназначен непосредственно для передачи потока данных, осуществляет передачу отдельных битов информации через физическую среду (т.е. отправку и приём электрических или оптических сигналов в проводных сетях, электромагнитных волн в беспроводных сетях). На этом уровне работают концентраторы (*англ.* hub), повторители (ретрансляторы) сигнала и преобразователи среды (медиаконвертеры). В персональных компьютерах физический уровень реализуется сетевым адаптером или последовательным портом.

Реализация протоколов обмена данными на каждом из уровней сетевой модели OSI ISO абстрагируются от вышестоящего уровня, которому предоставляются только определённые программные интерфейсы. Для передачи данных на вышестоящем уровне вызываются программные интерфейсы нижестоящего уровня. Уровни взаимодействуют только со своими соседями (т.е., например, интерфейсы уровня 5 могут быть вызваны только с уровня 6, а сам уровень 5 может вызвать интерфейсы только уровня 4). Благодаря такой структуре можно считать, что взаимодействие систем друг с другом на каждом из уровней идёт только по реализованным на этом уровне протоколам, без учёта специфики более низких уровней, что позволяет упростить совместную работу сетевого оборудования и программного обеспечения.

Например, при обращении браузера к веб-серверу на 7-ом уровне модели OSI ISO из браузера запрашивается документ по определённому URL, и в

ответ получаются данные от веб-сервера. Запрос данных, разбор и анализ ответа сервера в рамках синтаксиса протокола HTTP, выбор поддерживающейся и браузером, и веб-сервером версии протокола HTTP, кодирование данных веб-страницы (например, сжатие и распаковка текстовых страниц алгоритмами архивации данных, как и выбор этих алгоритмов) выполняется на 6-ом уровне модели OSI ISO. На 5-ом уровне осуществляется установление соединения браузера с сервером в зависимости от указанной в URL схемы по незащищённому протоколу HTTP или протоколу HTTPS с поддержкой шифрования трафика, а также поддержка этих соединений открытыми в течении определённого времени для использования при последовательных запросах страниц с одного веб-сервера. На 4-ом уровне выполняется установление соединения для передачи данных между компьютером, выполняющим процесс браузера, и сервером в выполняющимся на нём процессом веб-сервера, определение для этого числового адреса веб-сервера по символьному имени в URL, передача потока данных между ними, коррекция ошибок в передаваемых и получаемых данных, при необходимости повторные установления соединений и отправка данных в случае ошибок передачи. На 3-ем уровне выполняется собственно передача пакетов данных между компьютером клиента и сервером, с маршрутизацией при необходимости сетевых пакетов через глобальную компьютерную сеть. На 2ом уровне сетевые пакеты передаются драйверу сетевого карты, и на 1ом уровне идёт передача данных непосредственно через физические подключения.

При этом программные интерфейсы уровней 7, 6 и 5 реализуются в коде браузера и веб-сервера, возможно с использованием каких-либо стандартных библиотек, а уровни 4 и ниже уже являются общими для любых сетевых приложений и реализуются на уровне сетевого стека операционной системы. Ни браузеру, ни веб-серверу неважно, какие аппаратные средства используются для передачи данных между этими программами, как именно и через какие промежуточных узлы передаются данные между ними.

При ссылках на уровни сетевой модели OSI ISO вместо полных наименований уровней (физический, канальный, сетевой, транспортный и т. п.) обычно используются сокращения L1, L2, L3, L4 соответственно по номерам уровней.

Отметим, что модель OSI ISO — модель эталонная, и реальные системы, в т.ч. протоколы TCP/IP, ей не вполне соответствуют – в частности потому, что на начало их разработки сетевой модели OSI ISO ещё не существовало.

Существует также упрощённый вариант модели OSI — модель DOD, по названию использующей её организации (*англ.* Department of Defense, Министерство обороны США), в которой присутствуют 4 уровня — приложений (соответствующий 7, 6 и 5-му уровням OSI), транспортный (4-ый уровень OSI), межсетевой (3-ий уровень OSI) и сетевого интерфейса (2-ой и 1-ый уровни OSI).

Семейство протоколов TCP/IP.

Как указывалось выше, для объединения компьютеров в рамках глобальной сети Internet используется семейство протоколов TCP/IP.

На сетевом уровне, предназначенном для передачи данных из одной сети в другую, используется протокол IP (Internet Protocol). В протоколе IP данные передаются между подключёнными к сети Internet устройствами в виде форматированных блоков — сетевых пакетов. Сетевые пакеты содержат в себе заголовок со служебными данными, определяющими получателя и отправителя пакета, тип пакета, его длину, и т.д., а также произвольные данные пользователя (т.н. полезную нагрузку).

Каждое подключённое к сети Internet устройство (хост) имеет свой уникальный номер — сетевой адрес IP. При необходимости отправить данные с одного хоста на другой эти данные формируются в пакет, в заголовке которого хост-отправитель указывает адрес IP получателя, и свой адрес в качестве обратного. Протокол IP обеспечивает передачу сетевого пакета через сеть Internet от хоста-отправителя к хосту-получателю. Последний, используя адрес хоста-отправителя из заголовка полученного пакета IP, узнаёт, кто отправил сетевой пакет и может ответить на него.

Начало разработки протоколов TCP/IP относится к середине 70-х годов XX века, использование их в Internet началось в 1983 году с 4-ой версии протоколов (IPv4). Версии v1-v3, предшествовавшие v4, сейчас не используются.

IPv4 широко используется до сих пор, хотя имеет существенное ограничение — в нём используется 32-битная адресация хостов (компьютеров-узлов сети, *англ.* host), и диапазон доступных адресов стал исчерпываться ещё в конце 80-х годов прошлого столетия.

Для решения этой проблемы IETF в начале 90-х подготовила сразу несколько вариантов развития протоколов TCP/IP, четыре из которых были оформлены как v6, v7, v8 и v9 (IPv5 описан в RFC 1819, предназначен для передачи мультимедийной информации и официально называется ST2/ST2+). В рамках проводившихся обсуждений из них для дальнейшего развития и использования был выбран IPv6, со 128-битной адресацией хостов. Из остальных v7 был отвергнут из-за неполной спецификации, IPv9, на момент выбора технически самый лучший — по политическим причинам (он был основан на стандартах ISO, а не IETF), а v8 — проиграл из-за использования 64-битного адреса по сравнению со 128 битами IPv6.

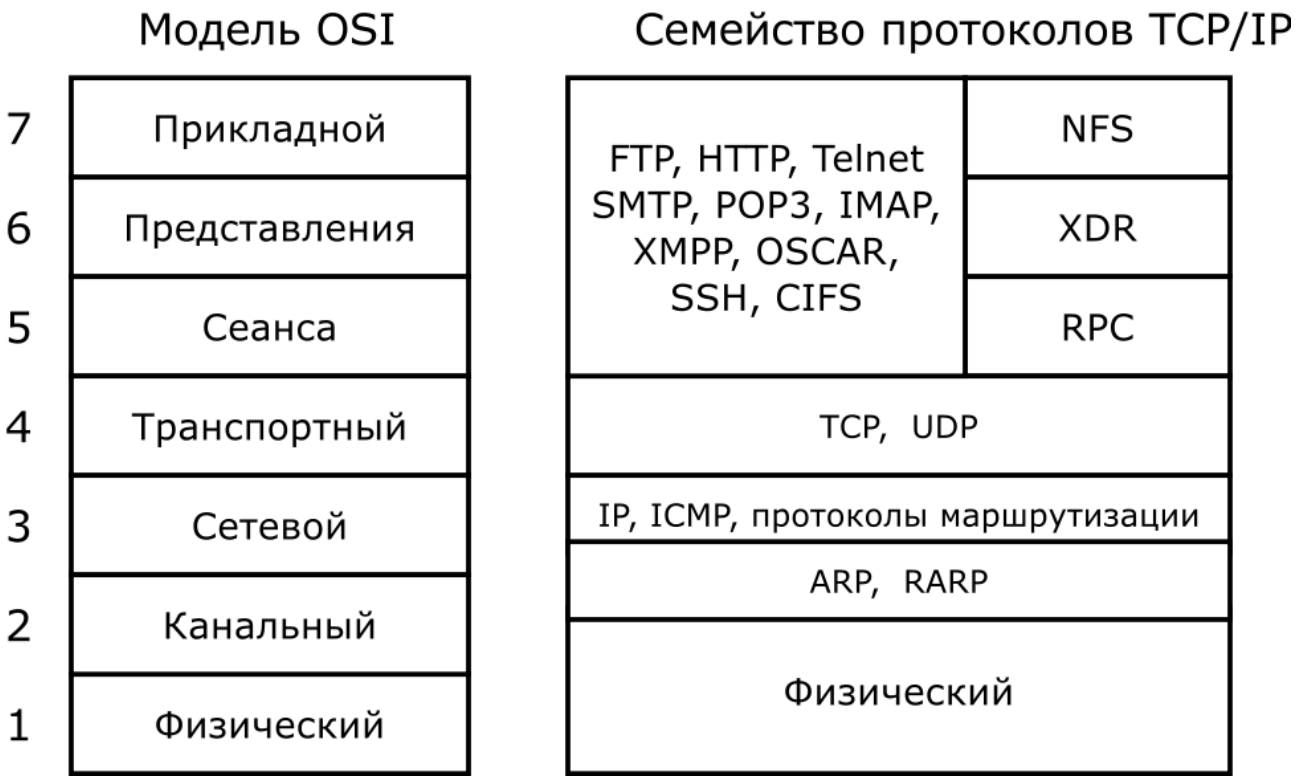
Хотя набор RFC, описывающих протокол IPv6, был подготовлен ещё в 1996 году, в силу ряда как технических, так и политических причин окончательный переход на него до сих пор не произведён. Ряд модификаций протокола IPv4 позволил более эффективно использовать его адресное пространство, и продлил существование этой версии протокола. Следует отметить, что часть этих модификаций была перенесена в IPv4 из IPv6. С другой стороны, отсрочка глобального перехода на IPv6 позволила разви-

вать протоколы и 6-ой версии, проводить их тестирование, разработать решения для совместного сосуществования сетей IPv4 и IPv6, подготовить условия для плавного и безболезненного для конечных пользователей перехода на IPv6.

В настоящий момент все доступные резервы адресного пространства IPv4 выбраны. Переход на IPv6 активно проводится; в последние годы оба семейства протоколов используются совместно с постепенным распространением поддержки IPv6 на все подключённые к Internet устройства.

Несмотря на существенно расширенное адресное пространство, в протоколах IPv6 широко используются заложенные в IPv4 принципы организации сетей, это версии одного и того же семейства протоколов и принципиальных различий между ними немного.

Поскольку семейство протоколов TCP/IP было разработано до появления модели OSI ISO, то строго в рекомендации эталонной модели оно не укладывается и обычно представляется следующим образом:



В рамках семейства протоколов TCP/IP не определяются физический и канальный уровни модели OSI. Протокол IP, разработанный специально для объединения разнородных сетей, способен функционировать поверх подавляющего большинства существующих протоколов соответствующих уровней (и даже поверх самого себя).

Физический уровень описывает среду передачи данных и лежит ниже протоколов TCP/IP. Обычно в качестве среды передачи данных используют медный кабель, оптоволокно, радио- или ИК-излучение. Протоколы физического уровня описывают физические характеристики такой среды и принцип передачи данных (разделение каналов, модуляцию, амплитуду сигналов, частоту сигналов, способ синхронизации передачи данных, время ожидания ответа и максимальное расстояние передачи).

На канальном уровне описывается, каким образом передаются пакеты данных через физический уровень, включая кодирование (т.е. специальные последовательности битов, определяющие начало и конец пакета данных).

Например, для локальных сетей широко используются протоколы Ethernet, в полях заголовка пакета которых содержится указание того, для какой машины или машин в сети предназначен этот пакет.

Примеры протоколов канального уровня: Ethernet, IEEE 802.11 Wireless Ethernet, SLIP, Token Ring, ATM, PPP.

На сетевом уровне, предназначенном для передачи данных из одной сети в другую, используется протокол IP. Для идентификации каждого подсоединённого к сети Internet устройства (хоста), ему назначается уникальный адрес: в рамках IPv4 — 32-битное число, в рамках IPv6 — 128-битное. Хосты на основе их IP-адресов объединяются в отдельные сети. Сети связываются друг с другом через маршрутизаторы — хосты, подключённые одновременно к двум или более сетям, и способные передавать пакеты данных из одной сети в другую.

Передача пакетов данных от хоста одной сети к хосту другой происходит через связывающие их сети, по цепочке маршрутизаторов. В общем случае может быть несколько разных путей, по которым пакет из одной сети может быть передан в другую. Выбор наиболее оптимального пути (и выбор критерия, по которому данный путь считается оптимальным) — задача протоколов маршрутизации.

В ходе развития глобальной сети Internet и соответствующего усложнения её топологии, для эффективной работы протоколов маршрутизации в сетевой уровень были внесены дополнительные возможности по организации передачи пакетов данных из одной сети в любую другую сеть, независимо от протоколов нижнего уровня, а также возможности запроса от удалённых хостов информации, необходимой для организации такой передачи пакетов. Подобные возможности используются, например, в протоколе ICMP (предназначенном для передачи диагностической информации IP-соединения) и IGMP (предназначенном для организации малоиспользуемой групповой передачи данных).

Вообще говоря, ICMP и IGMP расположены над IP, используют его для передачи своих данных и должны попасть на следующий, транспортный

уровень, но функционально они являются протоколами сетевого уровня, и их невозможно вписать в рамки модели OSI.

Пакеты сетевого протокола IP в заголовках содержат код, указывающий, какой именно протокол следующего уровня нужно использовать, чтобы извлечь данные из пакета. Это число — уникальный IP-номер протокола. ICMP и IGMP имеют номера, соответственно, 1 и 2.

Протоколы транспортного уровня могут решать проблему негарантированной доставки сообщений, а также гарантировать правильную последовательность прихода данных. В стеке протоколов TCP/IP транспортные протоколы определяют, для какого именно приложения предназначены эти данные.

Протокол TCP обеспечивает «гарантированный» транспортный механизм с предварительным установлением соединения. Он предоставляет приложению надёжный поток получаемых данных и даёт уверенность в их безошибочности. TCP повторно запрашивает данные в случае их потери и устраняет их дублирование, позволяет регулировать нагрузку на сеть, а также уменьшать время ожидания данных при их передаче на большие расстояния. Более того, TCP гарантирует, что данные будут получены точно в такой же последовательности, в которой были отправлены.

UDP — протокол передачи отдельных пакетов данных без установления соединения. Он является протоколом «ненадёжной» передачи данных, в смысле невозможности удостовериться в доставке сообщения адресату, а также возможного изменения порядка получения последовательно отправленных пакетов.

В приложениях, требующих гарантированной передачи данных, используется протокол TCP. UDP обычно используется в таких приложениях, как потоковое видео и компьютерные игры, где допускается потеря пакетов, а повторный запрос затруднён или не оправдан, либо в приложениях вида запрос-ответ (например, запросы к системе DNS), где создание соединения занимает больше ресурсов и времени, чем возможное повторение отправки запроса.

И TCP, и UDP используют для определения протокола верхнего уровня число, называемое портом. Под номер порта в пакетах TCP и UDP отводится 2 байта, т.е. максимальное количество использующих стек TCP/IP приложений в системе — 65535. Реально операционными системами на использование приложениями портов накладываются дополнительные ограничения.

Существует список стандартных портов TCP и UDP, зарезервированных под использование теми или иными приложениями. Например, на порту 80/tcp по стандарту может располагаться только веб-сервер, а на порту 53/udp — DNS-сервер. Для предотвращения несанкционированного использования зарезервированных портов приложениям непривилегирован-

ного пользователя запрещается использовать порты с номерами, лежащими в диапазоне до 1000.

Список стандартных портов TCP и UDP ведёт администрация IANA.

На прикладном уровне работает большинство сетевых приложений.

Эти программы имеют свои собственные протоколы обмена информацией, например, HTTP для WWW, FTP (передача файлов), SMTP (электронная почта), SSH (безопасное соединение с удалённой системой), DNS (преобразование символьных имён в IP-адреса) и многие другие.

В массе своей эти протоколы работают поверх TCP или UDP, и привязаны к определённому порту, например, HTTP — к 80-му.

Адресация в сетях IP.

Протокол IP работает на 3-ем уровне модели OSI ISO и передаёт данные через сетевые интерфейсы 2-го уровня. В простейшем случае у компьютера имеется один сетевой интерфейс, который входит в одну сеть IP. В более сложных случаях у одного компьютера может быть несколько сетевых интерфейсов, и он может входить одновременно в несколько сетей IP. Кроме того, любой сетевой интерфейс также может входить в несколько сетей IP.

Помимо физических интерфейсов у компьютера в сетях TCP/IP есть логический интерфейс обратной петли (*англ.* loopback). Посланные на этот интерфейс пакеты сразу возвращаются с него же обратно. Наличие loopback-интерфейса позволяет использовать обмен данными между приложениями по протоколам TCP/IP в пределах одного компьютера, даже не имеющего физических интерфейсов для подключения к сети.

Каждому сетевому интерфейсу назначается один или несколько адресов IP. В рамках изначальной спецификации протокола IPv4 каждому интерфейсу мог быть назначен только один адрес. Хотя это ограничение в текущих реализациях IPv4 снято, и возможно использование нескольких адресов IPv4, в т.ч. из разных сетей, на одном сетевом интерфейсе, это требует дополнительной ручной настройки и обычно не используется. В IPv6 на сетевой интерфейс может быть назначено несколько разных адресов, и, как правило, эта возможность используется.

Для IPv4 адрес IP — целое беззнаковое число длиной в 32 бита, или 4 байта. Согласно принятым соглашениям эти адреса записываются в виде четырёх десятичных чисел по-байтно, причём отдельные числа разделяются точками, например: `192.0.2.128` . Каждая часть адреса IP может иметь значение от 0 до 255.

Для IPv6 адрес IP — целое беззнаковое число длиной в 128 битов, или 16 байтов. Адреса IPv6 записываются как 8 групп по 4 шестнадцатеричные цифры, разделённых двоеточием, например:

2001:0db8:0000:0064:0000:0000:aa72:0004

Для сокращения такой записи можно убирать ведущие нули в группах шестнадцатеричных цифр, этот же адрес можно записать как:

2001:db8:0:64:0:0:aa72:4

Кроме того, одну из последовательностей нулевых групп (как правило, самую длинную) возможно опустить, заменив на два двоеточия:

2001:db8:0:64::aa72:4

Другой вариант записи этого же IP-адреса, более длинный и, соответственно, не рекомендующийся: 2001:db8::64:0:0:aa72:4 .

Две разделённые нулевые группы опустить одновременно нельзя, т. к. при этом возникнет неоднозначность в раскрытии сокращённой формы записи адреса до полного 128-битного числа.

Адрес IP содержит адрес сети и адрес хоста в пределах этой сети. Для выделения адреса сети из адреса IP применяется битовая маска:

ip AND mask = lan; ip AND NOT mask = host

Здесь, ip — адрес IP (число), mask — маска, lan — адрес сети, host — адрес хоста в сети, AND и NOT — логические операции.

Например, маска для IPv4 192.0.2.128 — 255.255.255.0 . Адрес сети — 192.0.2.0 , адрес хоста в этой сети — 0.0.0.128 :

Адрес:	11000000	00000000	00000010	10000000	(192.0.2.128)
Маска:	11111111	11111111	11111111	00000000	(255.255.255.0)
Сеть:	11000000	00000000	00000010	00000000	(192.0.2.0)
Хост:	00000000	00000000	00000000	10000000	(0.0.0.128)

Аналогичным образом адрес сети и адрес хоста выделяются и из адреса IPv6.

Изначально сети IPv4 были разделены на классы. Каждому классу соответствует своя маска. Класс сети конкретного адреса IPv4 однозначно определяется по его первым нескольким битам.

Сети класса A:

Адреса с первым битом, равным 0
Маска: 255.0.0.0
Допустимые адреса: 1.0.0.0 — 126.255.255.255

Две подсети класса A имеет специальное значение: 0.0.0.0 — адрес текущей сети (служебное значение), 127.0.0.0 — сеть loopback-интерфейса. Адрес IPv4 loopback-интерфейса, соответственно, обычно равен 127.0.0.1 .

Сети класса B:

Адреса с первыми двумя битами, равными `10`
Маска: `255.255.0.0`
Допустимые адреса: `128.0.0.0` — `191.255.255.255`

Сети класса C:

Адреса с первыми тремя битами, равными `110`
Маска: `255.255.255.0`
Допустимые адреса: `192.0.0.0` — `223.255.255.255`

Сети класса D:

Адреса с первыми четырьмя битами, равными `1110`
Сети `224.0.0.0` — `239.255.255.255` — предназначены для групповой передачи данных (multicast).

Сети класса E:

Сети с адресами `240.0.0.0` — `254.255.255.255` — зарезервированы, на данный момент не используются и в связи с проводящейся заменой IPv4 на IPv6 использоваться скорее всего не будут.

Выделением адресов конкретным сетям, входящим в Internet, занимается организация IANA через региональные организации (для Европы — RIPE, фр. Réseaux IP Européens). Адреса IPv4 выделяются только в количестве, кратном соответствующему классу сетей. Обычно адреса выделялись региональными организациями сравнительно большими блоками провайдерам услуг Internet, которые, в свою очередь, перераспределяли их среди своих клиентов. В феврале 2011 года IANA выделила адреса последних двух оставшихся сетей IPv4 класса A, по состоянию на май 2011 года свободными были 2 сети класса B и 3 сети класса C. Точное число свободных адресов у региональных организаций неизвестно, но ограничения на выделение блоков адресов провайдерам услуг Internet до блоков размером /22 (1024 адреса) введены в настоящий момент всеми региональными организациями, кроме отвечающей за Африку AfriNIC.

Некий резерв свободных адресов IPv4 имеется у провайдеров услуг, которые продолжают выделять их по заявкам своих клиентов — с постепенно ужесточающимися требованиями по количеству и целям использования выделяемых адресов.

Основной идеей при разделении диапазонов сетей IPv4 на классы была простота определения границ сети для конкретного адреса IP. Однако при этом б**о**льшая часть адресного пространства попала в сети крупного размера, а даже самая маленькая сеть — класса C — содержит в себе 254 адреса. В ходе бурного роста Internet уже в конце 80-х годов прошлого века оказалось, что число подключающихся к сети организаций становится сопоставимо с общим числом доступных сетей. Поэтому в рамках расширения

протокола IPv4 в 1993 году было введено понятие бесклассовых сетей. При этом маска сети перестаёт быть выравненной по границе байта, и может начинаться с произвольного числа единиц — т. е., помимо рассмотренных выше масок сетей классов A, B и C вида `255.0.0.0` , `255.255.0.0` , `255.255.255.0` могут быть сети с размером, находящимся между B и C — например, `255.255.192.0` , или с размером, меньшим класса C — например, `255.255.255.240` . С другой стороны, после ввода бесклассовой адресации, стало невозможно по виду адреса IPv4 узнать маску сети, и появилась необходимость указывать эту сеть явным образом. Это делается путём записи маски сети после адреса IP, через `/` (слеш):

`192.168.0.100/255.255.255.128` , `172.31.58.20/255.255.240.0` , `127.0.0.1/255.255.255.0` .

Учитывая, что в двоичной записи маски чередоваться нули и единицы не могут, и в ней всегда сначала идёт некоторое количество единичных разрядов, а следом — оставшиеся нулевые, можно использовать более короткую форму записи, вместо маски сети указывая число единиц в ней:

`192.168.0.100/25` , `172.31.58.20/20` , `127.0.0.1/8`

Введение бесклассовой адресации сетей позволило выделять адреса IPv4 небольшими диапазонами, и, тем самым, отложить проблему исчерпания адресов.

Как правило, организация не может получить необходимое количество IPv4-адресов для назначения их на каждый из своих компьютеров (мало того, на данный момент это или невозможно, или просто не имеет смысла). В этом случае можно самостоятельно, без получения разрешений от ICANN или провайдера, использовать адреса IPv4 из диапазона, зарезервированного для частных сетей. К таким диапазонам, согласно RFC 1918 (1996 год), относятся:

- одна сеть класса A: `10.0.0.0`
- 16 сетей класса B: `172.16.0.0` — `172.31.0.0`
- 256 сетей класса C: `192.168.0.0` — `192.168.255.0`

Поскольку адреса из этих диапазонов доступны для самостоятельного назначения, эти сети не могут и не должны маршрутизироваться в Internet.

При этом для доступа к ресурсам Internet с компьютеров, находящихся в частных сетях, были разработаны технологии трансляции (подмены) адресов на сетевом уровне модели OSI ISO, а также приёма и ретрансляции запросов промежуточными серверами (т. н. прокси-серверами) на более высоких уровнях. Вместе с введением бесклассовой адресации при выделении публичных, доступных глобально, адресов, это позволило отложить переход на 128-битные адреса IPv6 до настоящего времени — ценой необходимости модификации многих сетевых протоколов, потери возможности прямых соединений значительной части хостов друг с другом, усложнения настройки серверов и т. п.

По мере исчерпания адресного пространства IPv4 вводятся новые временные меры, например, в RFC 6598 (2012 год) был выделен отдельный диапазон адресов 100.64.0.0/10 для самостоятельного назначения провайдерами, аналогичный по смыслу частным сетям RFC 1918 у клиентов и позволяющий организовать ещё один уровень трансляции адресов, уже у провайдеров доступа в Internet. Но проблему исчерпания адресного пространства IPv4 все эти меры решить не могут, а переход на использование протоколов IPv6 делает их ненужными.

В свою очередь, в сетях IPv6 изначально существует только бесклассовая адресация. Маска сети записывается аналогично IPv4, через `/` (слеш) после адреса IPv6. Полностью маску, как правило, не записывают, заменяя её диапазоном сети: т. е., используется запись вида `2001:db8::1/32` , а не `2001:db8::1/ffff:ffff::`

Проблем с числом доступных адресов в сетях IPv6 нет и в обозримом будущем не предвидится, поэтому понятия «частной сети» в сетях IPv6 не существует. Технологии трансляции адресов для IPv6 также не нужны и использоваться не должны.

Локальный интерфейс в сетях TCP/IP имеет адрес `::1/128` .

В сетях TCP/IP существует несколько видов адресации, определяющих, какие подключённые к сети хосты получат отправленный пакет данных. Это:

- направленная адресация, когда адрес получателя — это адрес определённой машины в сети;
- групповая адресация, когда адрес получателя выбирается из диапазона адресов класса D для IPv4, или `ff00::/8` для IPv6. Групповая адресация используется в специализированных приложениях типа видеоконференций;
- для сетей IPv4 — широковещательная адресация, когда пакет предназначается для всех компьютеров в сети.

В последнем случае используется т.н. широковещательный адрес сети — адрес IP, в котором все биты поля адреса хоста равны 1.

Таким образом, для IP `192.168.230.50/28` :

IP хоста — это `192.168.230.50` , маска сети — `255.255.255.240` (28 бит).

Адресом сети является `192.168.230.48` , а широковещательным адресом — `192.168.230.63` . Всего в этой подсети доступно 14 адресов.

Минимальная по размерам сеть IPv4 — `/30` . Такие сети содержат всего по 2 адреса.

В сетях IPv6 широковещательных адресов не существует, для отправки пакета IPv6 на все хосты локальной сети используется специальный адрес групповой адресации `ff02::1` .

Наиболее распространённым в сетях в настоящее время протоколом 2-го уровня модели OSI ISO является Ethernet. В сетях Ethernet каждый сетевой интерфейс имеет уникальный канальный адрес — MAC-адрес (Media Access Control), имеющий длину 6 байт. Для физических сетевых адаптеров MAC-адрес определяется производителем при изготовлении конкретного устройства. Дополнительная конфигурация сетевых интерфейсов Ethernet на канальном уровне не требуется, они могут обмениваться пакетами данных сразу после подключения к сети на 1-ом, физическом уровне модели OSI. Однако для работы протоколов IP на 3-ем, сетевом уровне модели, требуется назначить сетевому интерфейсу и адрес IP.

Существует три разных механизма назначения адреса IP интерфейсам в сетях Ethernet: статический, динамический и автоматический.

Статическая конфигурация сводится к ручному заданию адреса IP (а также маски сети и пр.) в настройках интерфейса средствами операционной системы. Сохранённые в конфигурации хоста статические адреса восстанавливаются операционной системой после перезагрузки компьютера и могут быть изменены только администратором системы. Статическая конфигурация удобна при настройке серверных систем, как правило, имеющих выделенные специально для них постоянные адреса IP и несколько подключенных к разным сетям сетевых интерфейсов. Средствами статической конфигурации можно назначить адреса как IPv4, так и IPv6. Такие адреса обычно называют статическими (*англ.* static) или глобальными (*англ.* global).

Напротив, автоматическая конфигурация позволяет хосту назначить себе адрес IP самостоятельно, без получения каких-либо дополнительных данных от других хостов сети и/или вмешательства администратора системы. При этом на сетевой интерфейс операционной системой назначается уникальный в пределах локальной сети адрес хоста, в сети `169.254.0.0/16` для IPv4 и в сети `fe80::/64` для IPv6. Уникальный адрес хоста при этом создаётся на основе MAC-адреса интерфейса Ethernet.

Согласно RFC 3927 для сетей IPv4 автоматический адрес хоста (последние два байта в сети `169.254.0.0/16`) выбирается как псевдослучайное число, при этом начальное значение для генератора случайных чисел хоста должно быть выбрано на базе MAC-адреса. Такой подход обеспечивает как уникальность выбранного адреса, так и его сохранение при перезагрузке хоста.

Для сетей IPv6 согласно RFC 4862 адрес хоста в сети `fe80::/64` (8 байт) задаётся непосредственно с использованием MAC-адреса канального уровня, как идентификатор EUI-64 (Extended Unique Identifier*)* — 6 байт MAC-адреса дополняются байтами `ff` и `fe`, вставляемыми в середину адреса, и

инвертируется старший 6-ой бит полученного 64-битного числа. Т. е., например, MAC-адрес `00:18:51:4e:7e:94` преобразуется в EUI-64 как `0218:51ff:fe4e:7e94` и даёт адрес IPv6 `fe80::218:51ff:fe4e:7e94/64` .

Автоматически выбранный адрес называют локальным адресом интерфейса (*англ.* local или link-local address).

Механизм автоматической конфигурации позволяет хостам назначить сетевым интерфейсам уникальные в пределах локальной сети адреса IP и связываться по TCP/IP с входящими в эту сеть другими хостами без каких-либо дополнительных настроек. С другой стороны, при автоматической настройке интерфейсов хост не имеет никакой информации о топологии сети, имеющихся в ней маршрутизаторах, серверах и пр., поэтому при автоматически назначенных на интерфейс адресах возможно пересылать пакеты IP только в пределах локальной сети.

Динамическая конфигурация лишена этого недостатка, поскольку позволяет хосту получить адрес IP и другие данные для конфигурации сетевого интерфейса с других подключенных к сети хостов, как правило — серверов и/или маршрутизаторов.

Для протоколов IPv4 динамическая конфигурация в самом протоколе не описана, и реализуется путём использования дополнительного протокола DHCP. При этом в сети на одном из хостов настраивается сервер DHCP. Сетевой интерфейс сервера DHCP конфигурируется статически, в настройках сервера DHCP указывается, какой диапазон адресов IP сервер может назначать клиентам. При инициализации интерфейса (например, при включении компьютера) на хосте запускается клиент DHCP, который через широковещательный пакет на уровне Ethernet посылает в сеть запрос о получении адреса IP. Получив такой пакет, сервер DHCP выделяет клиенту адрес IP из числа имеющихся у него свободных адресов, и дополнительно передаёт клиенту информацию об адресах маршрутизаторов, различных доступных в локальной сети серверах, времени и т. п. Получив ответ от сервера DHCP, клиент настраивает сетевой интерфейс хоста в соответствии с полученными данными.

Динамические адреса DHCP выделяются клиенту на некоторое, задаваемое сервером DHCP, время, по истечении которого клиент должен повторно запросить у сервера DHCP адрес IP и переконфигурировать интерфейс хоста.

DHCP позволяет существенно упростить настройку сетевых интерфейсов на подключенных к сети хостах: при поддержке хостом протокола DHCP все сетевые настройки получаются им при включении сетевого интерфейса с сервера DHCP. Однако имеется и ряд недостатков.

Поскольку клиент обращается к серверу DHCP путём широковещательного запроса, этот запрос получают все подключённые к сети хосты. Сервер DHCP может работать только на одном из них, иначе клиент получит несколько разных ответов от разных серверов. Если сервер DHCP в момент отправления клиентом запроса недоступен (например, выключен или пе-

резагружается), или по каким-либо причинам запрос не был получен сервером DHCP (например, из-за сбоя на физическом уровне), клиент не получит ответ на свой запрос и настроить интерфейс не сможет. Кроме того, при изменении настроек сервер DHCP не может сообщить новую конфигурацию клиентам — процесс получения настроек по DHCP инициируется самими клиентами, серверу потребуется ждать повторного запроса от клиента по истечении времени выделения ему адреса.

Возможности динамической конфигурации протоколов IPv6 реализованы в самом протоколе и существенно шире по сравнению с IPv4. На маршрутизаторах сети может быть настроена периодическая широковещательная рассылка адресов подключенных к ним сетей. Получив такой анонс от маршрутизатора, хост назначает интерфейсу адрес IPv6 из указанной в анонсе сети, выбирая адрес хоста, как и в случае автоматической конфигурации, на базе MAC-адреса интерфейса, а также конфигурирует маршруты в другие сети через этот маршрутизатор на основе полученной информации.

Если в сети существует несколько маршрутизаторов, то каждый из них может отправлять в сеть анонсы, и хосты настроят на своих сетевых интерфейсах динамические адреса для каждой из анонсированных сетей. Если хост не получает анонсов для какой-либо сети в течении некоторого заданного промежутка времени, то динамически присвоенный адрес и информация о соответствующих маршрутах удаляется. Если же в сети появляется новый маршрутизатор, то хост начинает получать новые анонсы и добавляет новый динамический адрес (*англ.* dynamic).

Маршрутизаторы IPv6, в отличие от серверов DHCPv4, не ведут учёт выделенных адресов. Они рассылают только адреса сетей, полный адрес IPv6 задаётся самим хостом. Для работы такого механизма требуется, чтобы размер сети IPv6 был не менее `/64` .

IANA выделяет провайдерам сети IPv6 размером не менее `/32` , в свою очередь, провайдеры выделяют клиентам не отдельные адреса, как в случае с IPv4, а сети IPv6. Выделяемая каждому клиенту сеть IPv6 должна быть не менее `/64` ; согласно RFC 6177 провайдерам рекомендуется выделять каждому клиенту сеть, б**о**льшую `/64` , но не более `/48` . Рекомендованный в RFC размер — `/56` .

Динамическая конфигурация в IPv6 позволяет производить быструю переконфигурацию сети в случае изменения её топологии. С другой стороны, динамическая конфигурация в IPv6 работает на 3-ем уровне сетевой модели OSI и, в отличие от DHCP, не поддерживает передачу каких-либо настроек, помимо информации о сетях и адресах маршрутизаторов. Кроме того, с её помощью нельзя назначить хосту какой-либо определённый адрес IPv6. Для решения этих задач в сетях IPv6 поддерживается и протокол DHCPv6 в дополнение к встроенным средствам динамической конфигурации.

Как отмечалось ранее, изначально сетевой интерфейс IPv4 мог иметь только один адрес IP. В современных реализациях TCP/IP такого ограничения нет, но дополнительные адреса IPv4 на интерфейсе требуют ручной настройки маршрутизации. Поэтому для назначения адреса IPv4 используется один из перечисленных выше способов конфигурации интерфейса. Как правило, наивысший приоритет имеет статическая конфигурация, если она не настроена, может быть использована динамическая, путём опроса сервера DHCP. Если сервер DHCP не доступен, может быть использована автоматическая конфигурация, или интерфейс вообще остаётся не сконфигурированным.

В протоколе IPv6 изначально предусмотрено, что каждый интерфейс может иметь несколько адресов IP из разных сетей, поэтому используются все перечисленные выше способы конфигурации. При инициализации интерфейса ему назначается автоматический адрес из сети `fe80::/64` , после получения анонсов от маршрутизаторов добавляются динамические адреса сетей, при наличии статической конфигурации — добавляются адреса, заданные вручную администратором системы, также могут быть добавлены и адреса через DHCPv6. Выбор нужного адреса, с которого будет отправлен пакет на другой хост в сети Internet, как правило, осуществляется на базе таблиц маршрутизации.

Маршрутизация в сетях TCP/IP.

Глобальная сеть Internet представляет собой множество соединённых друг с другом сетей. В точках соединения сетей размещаются маршрутизаторы — сетевые устройства, имеющие два или более интерфейса, входящие в разные сети. Через маршрутизаторы возможно передать пакет IP из одной сети в другую. Путь пакета от хоста-отправителя до хоста-получателя называется маршрутом.

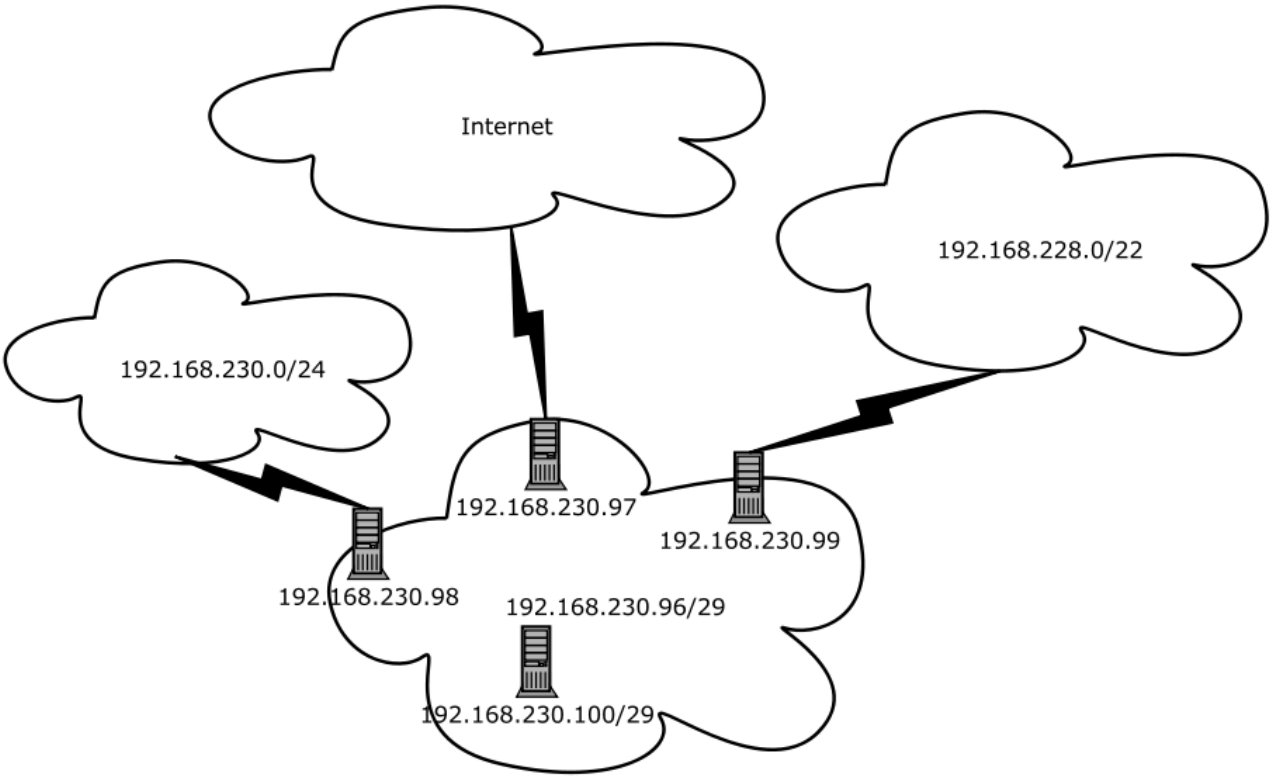
Получив пакет (или пришедший из внешней сети со 2-го уровня модели OSI, или сформированный транспортным уровнем локального компьютера), протоколы сетевого уровня определяют, предназначается ли этот сетевой пакет данному хосту, или его требуется переслать другому хосту-получателю. Для этого проверяется адрес IP получателя в заголовке сетевого пакета – если он соответствует какому-либо из назначенных на сетевые интерфейсы данного хоста адресу IP, то пакет передаётся на транспортный уровень сетевого стека. Если нет — то требуется определить, как данный пакет отправить по адресу хоста-получателя.

Для определения нужного маршрута служит таблица маршрутизации. В ней содержатся записи о том, через какой интерфейс или маршрутизатор (называемый также шлюзом, *англ.* gateway) следует оправить пакет, предназначенный для той или иной сети.

Записи в таблице маршрутизации упорядочены по размерам сетей, при наличии нескольких маршрутов для вложенных друг в друга сетей выбирается наименее общий маршрут.

В простых случаях используется статическая таблица маршрутизации, задаваемая при конфигурировании сетевого интерфейса и не меняющаяся со временем. В более сложных случаях, в магистральных маршрутизаторах применяются специальные протоколы маршрутизации, определяющие динамически изменяющийся со временем оптимальный маршрут до конечной сети.

Рассмотрим для примера следующую сеть IPv4:



В данном случае имеется подсеть `192.168.230.96/29` , в которой находятся 4 хоста. Три из них имеют соединения с другими сетями: `192.168.230.98` — с сетью `192.168.230.0/24` , `192.168.230.99` — с сетью `192.168.228.0/22` , и `192.168.230.97` — с Internet.

На четвёртом, `192.168.230.100` , таблица маршрутизации должна выглядеть следующим образом:

```
# ip route show
192.168.230.96/29 dev eth0 proto kernel scope link src 192.168.230.100
192.168.230.0/24 via 192.168.230.98 dev eth0
192.168.228.0/22 via 192.168.230.99 dev eth0
default via 192.168.230.97 dev eth0
```

Для вывода таблицы маршрутизации здесь была использована команда `ip`, предназначенная для конфигурации протокола IP на сетевых интерфейсах, `eth0` — имя сетевого интерфейса хоста.

Во-первых, пакеты для сети `192.168.230.96/29` данный хост должен отправлять просто на интерфейс `eth0` — т.к. он входит в эту сеть. Этот маршрут устанавливается автоматически при назначении интерфейсу адреса IP из диапазона адресов указанной сети.

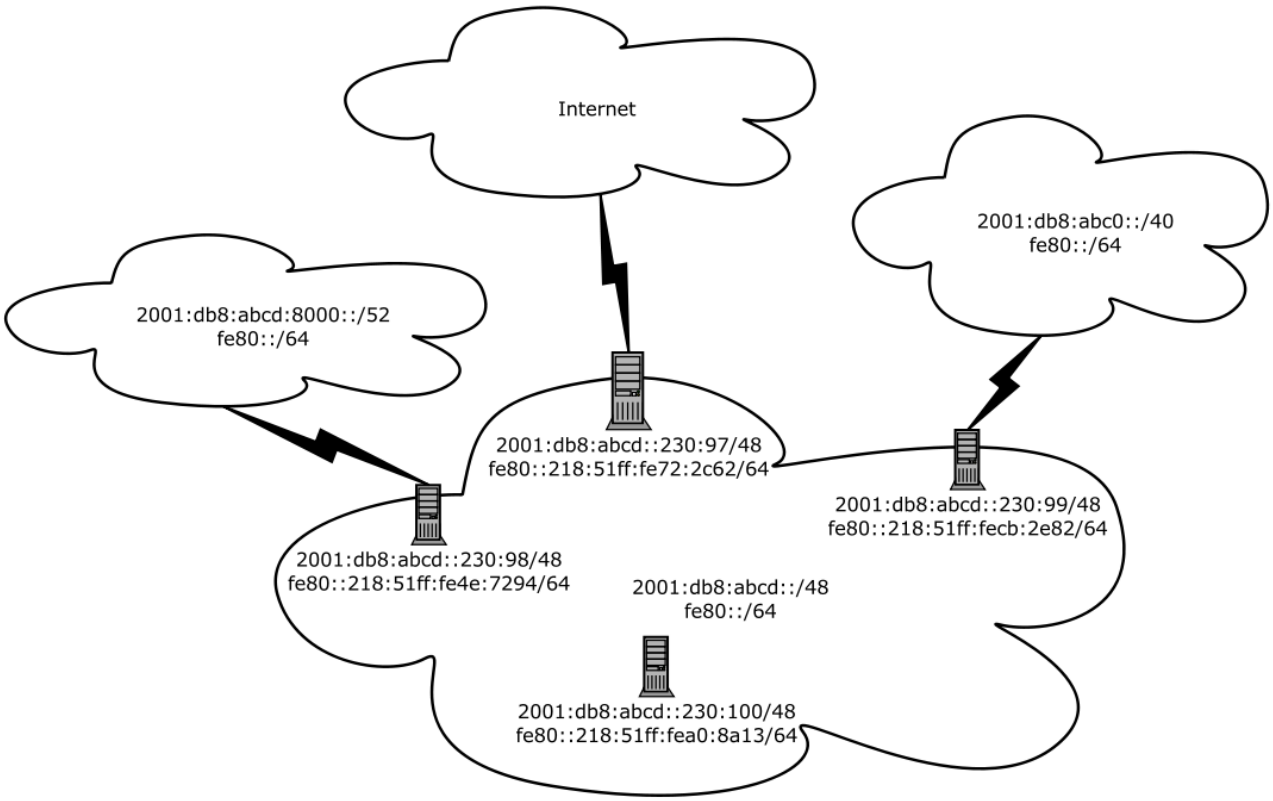
Далее, пакеты для сети `192.168.230.0/24` должны отсылаться через шлюз `192.168.230.98` (вторая запись в таблице, англ. *via* — через, и также с интерфейса `eth0`).

С сетью `192.168.228.0/22` граничит шлюз `192.168.230.99` — третья запись в таблице.

Все остальные пакеты должны отсылаться в сеть Internet через шлюз `192.168.230.97`. Ключевое слово `default` соответствует сети `0.0.0.0/0.0.0.0`, т.е. всем возможным адресам IPv4.

При этом перебор адреса идёт последовательно, от меньшей подсети к большей. Поэтому пакет для хоста, например, `192.168.230.150` будет отослан через шлюз `192.168.230.98` — этот адрес не входит в самую маленькую указанную подсеть, `192.168.230.96/29`, но входит в `192.168.230.0/24`. Отметим, что этот хост входит и в подсеть `192.168.228.0/22`, но она крупнее `192.168.230.0/24`, и выбрана не будет.

Рассмотрим сеть той же топологии, работающую по протоколам IPv6:



Имеется сеть Ethernet, в которой находятся 4 хоста. Помимо работающей поверх канального уровня сети IPv6 `2001:db8:abcd::/48` , данные хосты будут объединены и через подсеть автоматической конфигурации `fe80::/64` . Через первый маршрутизатор, соответствующий `192.168.230.98` для сети IPv4, доступна сеть `2001:db8:abcd:8000::/52` . На его сетевой интерфейс, подключённый к рассматриваемой сети, назначен статический адрес `2001:db8:abcd::230:98/48` и автоматический — `fe80::218:51ff:fe4e:7294/64` . Аналогично, маршрутизатор `192.168.230.97` имеет адреса `2001:db8:abcd::230:97/48` и `fe80::218:51ff:fe72:2c62/64` и подключён к Internet, а `192.168.230.99` — `2001:db8:abcd::230:99/48` и `fe80::218:51ff:fecb:2e82/64` , через него доступна сеть `2001:db8:abc0::/40` .

На четвёртом хосте, `192.168.230.100` , назначены статический адрес IPv6 `2001:db8:abcd::230:100/48` и автоматический — `fe80::218:51ff:fea0:8a13/64` .

Таблица маршрутизации на нём должна выглядеть следующим образом:

```
# ip -6 route show | sed -e 's/metric.*//'
2001:db8:abcd:8000::/52 via 2001:db8:abcd::230:98 dev eth0
2001:db8:abc0::/40 via 2001:db8:abcd::230:99 dev eth0
2001:db8:abcd::/48 dev eth0  proto kernel
fe80::/64 dev eth0  proto kernel
default via 2001:db8:abcd::230:97 dev eth0
```

С помощью команды `sed` были отрезаны концы выведенных строк, содержащие дополнительную, выходящую за рамки данной лабораторной работы информацию по маршрутам.

Первые две записи описывают маршруты в сетях `2001:db8:abcd:8000::/52` и `2001:db8:abc0::/40` , и предписывают отправлять пакеты данных для этих сетей через соответствующие маршрутизаторы. Две следующие записи предписывают отправлять пакеты для сетей `2001:db8:abcd::/48` и `fe80::/64` непосредственно через интерфейс `eth0`, как и в случае IPv4. Последняя запись устанавливает маршрут по умолчанию через шлюз `2001:db8:abcd::230:97` .

Следует отметить, что для указания адресов маршрутизаторов можно было использовать и их автоматические адреса, т. е., например, установить маршрут по умолчанию через `fe80::218:51ff:fe72:2c62` .

В данном случае маршруты и адреса IPv6 статически сконфигурированы на сетевом интерфейсе. Включив динамическую маршрутизацию, можно было бы не задавать статические маршруты вообще — всю необходимую информацию хост получил бы с маршрутизаторов. При этом на его сетевой интерфейс был бы назначен динамический адрес `2001:db8:abcd::218:51ff:fea0:8a13/48` , а динамически настроенная таблица маршрутизации имела бы вид:

```
2001:db8:abcd:8000::/52 via fe80::218:51ff:fe4e:7294 dev eth0
2001:db8:abc0::/40 via fe80::218:51ff:fecb:2e82 dev eth0
2001:db8:abcd::/48 dev eth0  proto kernel
fe80::/64 dev eth0  proto kernel
default via fe80::218:51ff:fe72:2c62 dev eth0
```

Таким образом, как для протокола IPv4, так и для протокола IPv6 в таблице маршрутизации указывается перечень сетей IP, доступных или непосредственно через сетевые интерфейсы данного хоста (т. е. те локальные сети, в которые подключен данный хост), или через хосты-маршрутизаторы. Для пересылки пакета маршрутизатору тот должен быть доступен с данного хоста, т. е. входить в одну из локальных сетей. Переслать пакет через какой-либо произвольный шлюз, не входящий в локальную сеть, невозможно – т. к. в заголовке пакета есть только адрес получателя пакета, и решение о дальнейшем пути маршрутизации пакета принимается на каждом из промежуточных шлюзов независимо.

В заголовке пакета IP предусмотрено поле TTL (Time To Live) - байтовый счётчик, определяющий время жизни пакета при его передаче в глобальной сети. При отправлении пакета хост-источник указывает в поле TTL начальное значение (обычно 64). При прохождении пакета через каждый промежуточный шлюз значение поля уменьшается на 1, при достижении значением TTL нуля очередной шлюз считает, что пакет не может достигнуть хоста-получателя, и дальше передавать его не имеет смысла. Такой пакет отбрасывается, и хосту-источнику отправляется пакет ICMP с сообщением о недоступности хоста-получателя. Такой механизм делает невозможным бесконечную передачу пакетов между маршрутизаторами из-за образования, например, петель в сетевых маршрутах в следствии каких-либо ошибок маршрутизации.

Так сообщения ICMP об ошибках отправляются отбросившим сетевой пакет шлюзом от своего адреса IP, в теории есть возможность проследить путь пакета от хоста-источника до хоста-получателя. В этом случает на хост-получатель оправляется сетевой пакет с установленным TTL=1, его отбрасывает первый шлюз - возвращая свой IP в сообщении об ошибке. Далее отправляется пакет с TTL=2, и т.п. На практике могут быть проблемы с получением ответов как из-за использования на промежуточных шлюзах адресов из диапазона частных сетей, так и из-за фильтрации трафика ICMP на межсетевых экранах. Кроме того, маршруты следования пакетов к хосту-получателю, и в обратном направлении от хоста-получателя к хосту-источнику могут быть разными. Соответствующие утилиты, позволяющие таким образом отследить маршрут до хоста-получателя - `traceroute` / `tracepath6` и более новые `tracepath` / `tracepath6`.

Система доменных имён DNS.

Каждому хосту в Internet соответствует свой адрес. Для обращения одного хоста к другому, т.е., например, для запроса браузером на одной машине документа с веб-сервера на другой, необходимо знать адрес этого сервера. Однако использовать числовое значение адреса IP удобно только компьютерам, в отличие от пользователей. Поэтому существует возможность сопоставления адресу IP символьного имени, которое может иметь осмысленный для человека вид. Такое сопоставление можно организовать несколькими методами.

Самый простой из них — записать необходимые пары адрес-имя в файле конфигурации. В UNIX-системах таким файлом является `/etc/hosts`, содержащий записи в формате:

```
IP  name [ name] [...]
```

Например:

```
$ cat /etc/hosts
127.0.0.1       localhost.localdomain localhost
::1             localhost6.localdomain6 localhost6
```

В начале существования Internet все имена входящих в сеть систем содержались в `/etc/hosts`. Существовал эталонный и централизованно обновлявшийся файл `hosts`, по которому периодически администраторами систем обновлялись локальные `/etc/hosts`. Однако когда число объединённых сетью компьютеров перевалило за несколько сотен, появилась потребность в масштабируемой системе.

Такой системой стала разработанная в 1983 году Полом Мокапетрисом (Paul V. Mockapetris) система DNS (Domain Name System, Система доменных имён, RFC 1034 и RFC 1035), которая позволила гибко и децентрализовано задавать имена хостов, делегируя полномочия по изменению имён компьютеров в конкретных информационных системах администраторам этих систем.

Имена DNS структурированы и состоят из отдельных частей, доменов, разделённых точками. Домены имеют уровни, отсчитывающиеся с конца имени, и образуют дерево имён. Логический узел в дереве имён носит название «зоны». В зонах хранятся записи о соответствии имён входящих в зону хостов их IP-адресам, а также ссылки на серверы DNS, обслуживающие зоны доменов более высокого уровня.

Домен нулевого уровня (корневой домен) содержит адреса серверов DNS доменов первого уровня, его зона хранится на корневых серверах DNS. Таких серверов на данный момент 13, их адреса известны и не меняются со временем.

Домены первого уровня (top-level domain, TLD) образуют верхний уровень иерархии имён в системе DNS. Они разделяются на национальные домены первого уровня, с двухбуквенными именами по стандарту ISO 3166 («`us`»,

«ru», «be», «uk», «tv» и т. п.), интернационализованные домены первого уровня, с именами из не-латинских букв («рф», «срб», «бг» и т. п.), общие домены первого уровня («com», «org», «net» и т. п.), домены, спонсируемые заинтересованными в них организациями («aero», «edu», «mil», «travel» и т. п.), а также служебные, зарезервированные и тестовые домены. Вопросы регистрации доменов первого уровня относятся к ведению IANA.

Для каждого из доменов первого уровня существует организация, выдающая всем абонентам имена, заканчивающиеся на соответствующий «.домен» – домены второго уровня. Эта организация устанавливает правила регистрации доменов второго уровня внутри своего TLD, а также обязана организовать и поддерживать серверы DNS него.

Серверы DNS для делегированных имён второго уровня поддерживаются получившими эти имена абонентами, которые, в свою очередь, могут создавать доменные имена третьего уровня и т. д.

Получающаяся иерархическая система имён позволяет, пройдя по цепочке начиная от корневых серверов, узнать адрес сервера DNS для конкретного имени и соответствующий адрес IP.

Клиенты системы DNS (приложения, работающие на компьютере) непосредственно ко всей цепочке серверов DNS не обращаются. Вместо этого в настройках хоста указывается один или несколько серверов DNS, к которым он должен обращаться с запросами. Для UNIX-систем список серверов DNS хранится в файле /etc/resolv.conf, который имеет вид:

```
$ cat /etc/resolv.conf
search ddns.ossg.ru
nameserver 192.168.250.1
```

В одной или нескольких строках nameserver указываются адреса серверов DNS для данной системы, а в параметре search — список доменов по умолчанию. Перед тем, как обратиться к серверам DNS, сначала клиент проверяет файл /etc/hosts. Если нужного имени там не оказывается, выполняется запрос к одному из серверов DNS. Если этот сервер не отвечает, производится запрос к следующему серверу. Если получить IP для имени не удаётся, возвращается ошибка.

Длина названия домена в системе DNS не может превышать 63 символов ASCII, общая длина имени не может превышать 253 символов, включая разделяющие части имени точки. В имени домена DNS допускается использование только букв латинского алфавита (a-z), цифр (0-9) и дефиса (-). Регистр букв в именах DNS не учитывается, т.е. yandex.ru, Yandex.RU, YANDEX.RU — одно и то же имя.

Интернационализованные домены, содержащие имена из не-латинских букв, преобразуются из кодировки Unicode в набор символов ASCII с помощью кодировки Punycode (RFC 3492). Отображение таких имён DNS в символах национальных алфавитов является задачей программ пользовательского уровня, внутри системы DNS эти имена хранятся и используются только в кодировки ASCII. Например, имя `яндекс.рф` в кодировке Punycode записывается как `XN—D1ACPJX3F.xn--p1ai`, и в DNS размещён именно домен второго уровня `XN—D1ACPJX3F.xn--p1ai`, в домене первого уровня `xn--p1ai.`.

Разработанная более 30 лет назад, система DNS оказалась крайне удобной, быстро и надёжно работающей и отлично масштабируемой. Помимо хранения записей о соответствии символьных имён числовым адресам IP, DNS используется также для размещения информации об обслуживающих почтовые домены серверах электронной почты, записей для автоматического определения клиентским программным обеспечением адресов серверов различных служб, и пр.

Рассмотрим работу системы DNS на примере. Для работы с серверами DNS, в т.ч. для получения адреса IP по доменному имени, можно использовать команду `host` из пакета `bind-utils`. Предположим, была вызвана команда `host` с параметром в виде имени хоста `cbias.mpei.ac.ru` . Если в файле `/etc/hosts` такой записи не обнаруживается, то клиентская библиотека DNS обращается с запросом к одному из указанных в `/etc/resolv.conf` серверов.

В свою очередь, сервер DNS, получив запрос, разбирает доменное имя на составные части. Доменом первого уровня в данном случае является «`ru`». Следует запрос к одному из корневых серверов (его адрес известен заранее) о сервере имён, содержащем зону «`ru`». Результаты этого запроса можно посмотреть на клиенте, указав команде `host` возвращать адрес сервера DNS, а не адрес хоста:

```
$ host -t NS ru.
ru name server ns9.ripn.net.
ru name server e.dns.ripn.net.
ru name server f.dns.ripn.net.
ru name server ns.ripn.net.
ru name server ns2.nic.fr.
ru name server ns5.msk-ix.net.
```

Далее следует обращение к одному из полученных серверов с целью узнать сервер DNS для зоны «`ac.ru`»:

```
$ host -t NS ac.ru
ac.ru name server ns-soa.darenet.dk.
ac.ru name server dns.princeton.edu.
ac.ru name server ns1.free.net.
ac.ru name server ns2.free.net.
ac.ru name server sunic.sunet.se.
```

Продолжая запросы, сервер DNS находит, где хранится зона для «mpei.ac.ru»:

```
$ host -t NS mpei.ac.ru
mpei.ac.ru name server ns3.mpei.ac.ru.
mpei.ac.ru name server ns4.nic.ru.
mpei.ac.ru name server ns1.mpei.ac.ru.
mpei.ac.ru name server ns2.mpei.ac.ru.
```

Наконец, с одного из этих серверов возвращается адрес IP для нужного имени:

```
$ host  cbias.mpei.ac.ru
cbias.mpei.ac.ru has address 193.233.68.98
```

Проводя поиск имени, сервер DNS запоминает полученные промежуточные результаты. Поэтому при повторном запросе от клиента адреса cbias.mpei.ac.ru, сервер DNS сразу выдаст искомое значение из своего кэша (*англ.* cache), а при поиске адреса для записи itf.mpei.ac.ru — сразу обратится к DNS-серверам домена «mpei.ac.ru». Всё это позволяет как ускорить процесс разрешения имён, так и снизить нагрузку на серверы DNS в Internet.

Время жизни записи в кэше сервера DNS ограничено, и определяется записями в соответствующей зоне. По истечении времени жизни запись из кэша удаляется, и при повторном запросе процесс разрешения имён повторится с начала. Тем самым появляется возможность обновления записей в DNS при необходимости изменения адреса IP для доменного имени.

Преобразование имени DNS в IP не обязательно однозначное: одному символьному имени может соответствовать несколько адресов IP:

```
$ host google.ru
google.ru has address 216.239.59.104
google.ru has address 72.14.221.104
google.ru has address 66.249.93.104
google.ru mail is handled by 10 smtp4.google.com.
google.ru mail is handled by 10 smtp1.google.com.
google.ru mail is handled by 10 smtp2.google.com.
google.ru mail is handled by 10 smtp3.google.com.
```

Для части хостов в DNS указаны только адреса IPv4, для других — как IPv4, так и IPv6, некоторые имеют адреса только в сетях IPv6:

```
$ host sixy.ch
sixy.ch has address 80.254.71.229
sixy.ch has IPv6 address 2a02:200:3:1::103
$ host ipv6.google.com
ipv6.google.com is an alias for ipv6.l.google.com.
ipv6.l.google.com has IPv6 address 2a00:1450:8004::68
```

Получив в ответ несколько адресов, клиент упорядочивает адреса в соответствии с версией протокола. По стандарту, при наличии настроенной на клиенте сети IPv6 и доступности хоста как по IPv4, так и по IPv6, для рабо-

ты желательно выбирать именно IPv6, хотя конечное решение остаётся за клиентом. При наличии нескольких адресов для выбранной версии протокола, обычно выбирается первый из них. Если установить соединение с удалённым хостом по этому адресу не получается, выбирается следующий, и т. д.

DNS возвращает адреса в случайном порядке; задав для одного имени несколько разных адресов IP, можно получить примерно равномерное распределение запросов клиентов между соответствующими серверами, т. е. осуществить балансировку нагрузки на сервер.

С другой стороны, и одному адресу может соответствовать несколько символьных имён, в т.ч. и из разных доменных зон:

```
$ host edu.cbias.ru
edu.cbias.ru has address 87.242.75.168
$ host edu.ossg.ru
edu.ossg.ru has address 87.242.75.168
```

Имена DNS никак не связаны с распределением адресов IP по сетям, что позволяет объединять в рамках одного домена (или одного доменного имени) компьютеры, расположенные в разных физических сетях.

Помимо рассмотренного прямого преобразования имени DNS в адрес также существует и обратное — определение символьного имени DNS по адресу IP. Из-за неоднозначности прямого соответствия доменного имени адресу IP, для обратного преобразования используются специальные зоны и дерево доменов в домене «`.in-addr.arpa.`» для адресов IPv4 и «`.ip6.arpa.`» для IPv6, а из-за существенно более редкой надобности в обратном преобразовании для значительного числа хостов определить доменное имя по их адресу IP нельзя.

Серверы DNS, отвечающие на запросы пользователей и выполняющие рекурсивный поиск по серверам DNS, называются рекурсивными. Серверы DNS, хранящие записи об именах в определённых зонах - авторитетными. В сетевых настройках клиента указывается один или несколько адресов IP рекурсивных серверов DNS - как правило, находящихся в локальной сети или в сети провайдера. Для поиска имени в DNS клиенты обращаются к одному из рекурсивных серверов DNS, который в свою очередь выполняется поиск имени на авторитетных серверах DNS, и возвращает найденное значение. Для работы в сети в рамках стека протокола IPv4 требуется указание адресов IPv4 серверов DNS, для стека протокола IPv6 - адресов IPv6. Адреса серверов DNS указываются или в настройках операционной системы для сетевых интерфейсов при использовании статической настройки сети, или получаются с серверов DHCP/DHCPv6 при использовании динамической настройки интерфейсов.

Рекурсивные серверы могут быть авторитетными для каких-либо зон, что даёт возможность создания локальных зон DNS в рамках локальной сети, без необходимости публичной их регистрации в глобальных TLD. В настройках публичных авторитетных серверов DNS возможность обработки рекурсивных запросов от произвольных клиентов отключается, иначе появляется возможность их использования для организации DDoS-атак.

Рекурсивные серверы DNS локальной сети настраиваются сетевыми администраторами и доступны в рамках соответствующих локальных сетей. Рекурсивные серверы DNS провайдеров услуг обычно доступны только клиентам соответствующих провайдеров. Также существуют публичные общедоступные рекурсивные серверы DNS, к которым возможны запросы от произвольных клиентов. К таким серверам относятся, например, сервера Google (`dns.google`, IPv4 `8.8.4.4` и `8.8.8.8`, IPv6 `2001:4860:4860::8888` и `2001:4860:4860::8844`), Cloudflare (`one.one.one.one`, IPv4 `1.1.1.1` и `1.0.0.1`, IPv6 `2606:4700:4700::1111` и `2606:4700:4700::1001`), Yandex (IPv4 `77.88.8.8` и `77.88.8.1`, IPv6 `2a02:6b8::feed:0ff` и `2a02:6b8:0:1::feed:0ff`).

Наряду с магистральными каналами связи и магистральными маршрутизаторами, работа системы DNS является критически важной для работы глобальной компьютерной сети Internet в целом и для устойчивости национального сегмента глобальной сети в частности. Отказоустойчивость системы DNS обеспечивается дублированием информации о зонах по независимым авторитетным серверам, размещённым в разных физических сетях.

Зоны доменов в DNS согласно требованиям RFC должны размещаться минимум на двух серверах DNS с адресами из разных сетей IP. Записи об этих авторитетных серверах DNS, относящиеся к зонам первого уровня, размещаются на авторитетных серверах DNS TLD, которые также многократно дублируются. Всего существует тринадцать авторитетных серверов TLD с фиксированными IP-адресами, которые задаются в настройках рекурсивных серверов DNS, обычно на этапе компиляции их кода. За счёт использования anycast – адресации эти тринадцать корневых серверов TLD DNS представлены в глобальной компьютерной сети Internet большим количеством распределённых по разным континентам реплик.

Ряд реплик корневых серверов размещены в том числе и в национальном сегменте глобальной сети. Дополнительно к этому организована и развивается Национальная Система Доменных Имён (НСДИ), дублирующая работу корневых серверов DNS и обеспечивающая независимый источник информации о серверах в национальных доменных зонах `ru.` и `рф.` . В рамках НСДИ имеются и публичные рекурсивные серверы DNS, позволяющие получить информацию из DNS произвольным клиентам – это `a.res-nsdi.ru` (`195.208.4.1`; `2a0c:a9c7:8::1`) и `b.res-nsdi.ru` (`195.208.5.1`; `2a0c:a9c7:9::1`).

Конфигурация сетей TCP/IP в Linux.

Для работы компьютерной системы в сетях TCP/IP необходимо настроить подключение к сети, в которой используется протокол TCP/IP, задать для этого сетевого интерфейса адрес IP, и настроить таблицу маршрутизации.

Обмен данными в сети происходит через сетевые интерфейсы. Сетевые интерфейсы могут как соответствовать физическим устройствам компьютера (например, сетевым картам Ethernet), так представлять логические устройства. К последним относятся, например, интерфейсы PPP (Point-to-Point Protocol), работающие поверх последовательных модемных соединений, и виртуальные сетевые интерфейсы, организующие передачу данных поверх других сетей.

Для передачи информации в сети помимо самих передаваемых данных требуется также знание того, с какого хоста и на какой хост передаются эти данные. Поэтому для сетевых интерфейсов стандартное для UNIX представление устройств как файлов не имеет смысла, и в каталог `/dev/` соответствующие сетевым интерфейсам файлы устройств не отображаются. Работа с сетевыми интерфейсами осуществляется только через вызовы ядра операционной системы, а на прикладном уровне — через наборы соответствующих утилит.

В любом случае, для конфигурации сетевого интерфейса требуется сначала настроить соответствующее устройство на уровне ядра (т. е., для физических устройств — загрузить соответствующий драйвер устройства, для логических — сконфигурировать подключение через существующие сетевые интерфейсы или инициировать его через физические устройства).

Сетевые интерфейсы представляют собой абстрактные устройства, работающие на 2-ом уровне модели OSI ISO. Для работы поверх них протоколов TCP/IP требуется дальнейшая настройка интерфейсов: назначение им адресов IP и создание записей в таблице маршрутизации.

В ряде случаев адрес IP назначается интерфейсу внешними программами при регистрации интерфейса в системе (например, для интерфейсов PPP), в других случаях (чаще всего, для физических интерфейсов) его требуется назначать отдельно после создания интерфейса на уровне ядра. Как отмечалось выше, в этом случае назначение адреса IP и внесение записей в таблицу маршрутизации может осуществляться статически, с указанием необходимых адресов и маршрутов в конфигурации системы вручную, либо динамическими и автоматическими средствами конфигурации.

Для статической конфигурации сетевых интерфейсов в Linux существуют два отдельных набора утилит — `net-scripts` и `iproute2`. Первый из них считается морально устаревшим, не позволяет полностью использовать текущие возможности стека TCP/IP, однако ещё применяется в ряде дистрибутивов и во встраиваемых системах. Второй — `iproute2` — более новый, обеспечивает больше возможностей, и рекомендован к использованию.

Рассмотрим возможности основной утилиты пакета iproute2 — команды ip.

Просмотр состояний сетевых интерфейсов осуществляется командой

```
 # ip link show
 1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode
DEFAULT group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
314: eth0@if120: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state
    link/ether 00:16:3e:60:6b:03 brd ff:ff:ff:ff:ff:ff link-netnsid 0
315: ext@if9: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP
mode DEFAULT group default qlen 10000
    link/ether 00:16:3e:1a:12:61 brd ff:ff:ff:ff:ff:ff link-netnsid 0
```

Здесь lo — локальный интерфейс обратной петли (loopback), ext и eth0 — два сетевых интерфейса. Перед именем интерфейса указывается его номер в системе (в данном случае - 1, 314, 315). Локальный интерфейс обратной петли всегда имеет номер 1. Номера остальных интерфейсов могут меняться в зависимости от порядка их настройки в операционной системе и приводятся только для справок.

В зависимости от сетевых настроек операционной системы часть интерфейсов могут быть подчинёнными (дочерними) по отношению к другим, например, сетевые интерфейсы могут включаться в виртуальные мосты. В этом случае после имени сетевого интерфейса в выводе ip link указывается имя главного (родительского) интерфейса (в данном случае – @if120, @if9). Имена родительских интерфейсов, как и номера сетевых интерфейсов в системе, в выводе ip link приводятся только для справки – в используемых далее командах настройки сетевых интерфейсов их указывать не нужно, и далее в выводе команд они будут опускаться.

*В использующихся в лабораторной работе виртуальных серверах сетевой интерфейс ext используется для подключения виртуального сервера к внешней сети, через него осуществляется связь с VPS. Пытаться настроить интерфейс ext в ходе лабораторной работы не ст**о**ит, если его отключить или неправильно сконфигурировать, связь по SSH оборвётся.*

Видно, что, в отличии от ext, eth0 не включён, т. е. находится в состоянии DOWN (поскольку рассматриваются виртуальные сетевые интерфейсы, то во включённом состоянии они помечены как UNKNOWN. Для включённых интерфейсов, соответствующих реальным физическим устройствам, должно отображаться состояние UP). Включить интерфейс можно командой

```
 # ip link set eth0 up
 # ip link show eth0
 3: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state
UNKNOWN
     link/ether 00:18:51:a0:8a:13 brd ff:ff:ff:ff:ff:ff
```

Посмотреть назначенные адреса на интерфейсе можно командой

```
 # ip addr show
 1: lo: <LOOPBACK,UP,LOWER_UP> mtu 16436 qdisc noqueue state UNKNOWN
     link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
     inet 127.0.0.1/8 scope host lo
     inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
 2: ext: <BROADCAST,POINTOPOINT,NOARP,UP,LOWER_UP> mtu 1500 qdisc noqueue state
UNKNOWN
     link/void
     inet 192.168.230.100/32 scope global ext:1
 3: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state
UNKNOWN
     link/ether 00:18:51:a0:8a:13 brd ff:ff:ff:ff:ff:ff
     inet6 fe80::218:51ff:fea0:8a13/64 scope link
        valid_lft forever preferred_lft forever
```

Хотя мы и включили интерфейс `eth0`, адрес IPv4 ему ещё не назначен. Зато для IPv6 автоматическая конфигурация уже назначила ему адрес из сети `fe80::/64` , создав его на основе MAC-адреса. Т. е. в рамках локальной сети IPv6 уже есть возможность доступа к другим подключённым к ней хостам.

Также указывается версия протокола для адреса (`inet` соответствует IPv4, `inet6` — для IPv6), его тип (`scope link` — для локальных адресов link-local, `scope dynamic` и `scope global` для динамических и статических адресов соответственно), а также оставшееся время использования данного адреса (`valid_lft` и `preferred_lft`). Статические и автоматические адреса ограничений по времени жизни не имеют (могут использоваться вечно, *англ.* `forever`), для динамически полученных — будет указываться оставшееся время до обновления настроек.

Назначим интерфейсу адрес IPv4 `192.168.230.100` из сети `/29`:

```
 # ip addr add 192.168.230.100/29 dev eth0
 # ip addr show
 ....
 3: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state
UNKNOWN
     link/ether 00:18:51:a0:8a:13 brd ff:ff:ff:ff:ff:ff
     inet 192.168.230.100/29 scope global eth0
     inet6 fe80::218:51ff:fea0:8a13/64 scope link
        valid_lft forever preferred_lft forever
```

После этого можно послать ICMP-запрос на назначенный адрес и, разумеется, получить ответ — т. к. в данном случае отправка и получение пакетов по протоколу выполняется в рамках сетевого стека одной системы, не покидая её:

```
# ping 192.168.230.100
PING 192.168.230.100 (192.168.230.100) 56(84) bytes of data.
64 bytes from 192.168.230.100: icmp_seq=1 ttl=64 time=0.081 ms
64 bytes from 192.168.230.100: icmp_seq=2 ttl=64 time=0.144 ms
^C
--- 192.168.230.100 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 0.081/0.112/0.144/0.033 ms
```

Команда `ping` посылает пакеты ICMP 1 раз в секунду, прервать его выполнения можно клавишами `<Ctrl>+<C>`. Можно указать как другие интервалы отправки пакетов, так и задать определённого количество отправляемых пакетов – соответствующие ключи запуска команды `ping` доступны в её справочном руководстве.

Таким же образом можно назначить и другие адреса на тот же или на другой сетевой интерфейс, например:

```
# ip addr add 192.168.210.100/25 dev eth0
# ip addr show eth0
3: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state
UNKNOWN
    link/ether 00:18:51:a0:8a:13 brd ff:ff:ff:ff:ff:ff
    inet 192.168.230.100/29 scope global eth0
    inet 192.168.210.100/25 scope global eth0
    inet6 fe80::218:51ff:fea0:8a13/64 scope link
       valid_lft forever preferred_lft forever
```

Удалить ненужный адрес можно командой

```
# ip addr del 192.168.210.100/25 dev eth0
# ip addr show eth0
3: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state
UNKNOWN
    link/ether 00:18:51:a0:8a:13 brd ff:ff:ff:ff:ff:ff
    inet 192.168.230.100/29 scope global eth0
    inet6 fe80::218:51ff:fea0:8a13/64 scope link
       valid_lft forever preferred_lft forever
```

Адреса IPv6 назначаются аналогичным образом, но команде `ip` требуется указать, что это именно адрес IPv6. Для этого используется ключ `-6`. Также этот ключ можно использовать при просмотре адресов, для отображения только адресов IPv6. Симметричный ключ `-4` используется для указания команде `ip` рассматривать только адреса IPv4, и применяется по умолчанию.

Например, назначим интерфейсу адрес из сети `2001:db8:abcd::/48` :

```
# ip -6 addr add 2001:db8:abcd::230:100/48 dev eth0
# ip -6 addr show dev eth0
3: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500
    inet6 2001:db8:abcd::230:100/48 scope global
       valid_lft forever preferred_lft forever
    inet6 fe80::218:51ff:fea0:8a13/64 scope link
       valid_lft forever preferred_lft forever
```

Для проверки доступности назначенного адреса IPv6 используется команда `ping6`, имеющая тот же формат вызова, что и `ping` для IPv4:

```
# ping6 2001:db8:abcd::230:100
PING 2001:db8:abcd::230:100(2001:db8:abcd::230:100) 56 data bytes
64 bytes from 2001:db8:abcd::230:100: icmp_seq=1 ttl=64 time=0.433 ms
64 bytes from 2001:db8:abcd::230:100: icmp_seq=2 ttl=64 time=0.105 ms
^C
--- 2001:db8:abcd::230:100 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1000ms
rtt min/avg/max/mdev = 0.105/0.269/0.433/0.164 ms
```

Посмотреть таблицу маршрутизации позволяет команда

```
# ip route show
192.168.230.96/29 dev eth0  proto kernel  scope link  src 192.168.230.100
```

Мы видим только маршрут на сеть IPv4, в которую входит назначенный нами адрес IPv4, и в которую, очевидно, входит сетевой интерфейс. Эта запись была добавлена при выполнении '`ip addr add`' автоматически.

В отличии от '`ip addr show`', '`ip route show`' выводит информацию только из таблицы маршрутизации IPv4. Для просмотра таблицы IPv6 требуется указывать ключ '`-6`' команды `ip`.

Добавим маршрут на сеть `192.168.230.0/24` через шлюз `192.168.230.98`:

```
# ip route add 192.168.230.0/24 via 192.168.230.98
# ip route show
192.168.230.96/29 dev eth0  proto kernel  scope link  src 192.168.230.100
192.168.230.0/24 via 192.168.230.98 dev eth0
```

Аналогично можно добавлять и удалять другие маршруты:

```
# ip route add 192.168.238.0/24 via 192.168.230.98
# ip route add 192.168.228.0/22 via 192.168.230.99
# ip route show
192.168.230.96/29 dev eth0 proto kernel  scope link  src 192.168.230.100
192.168.230.0/24 via 192.168.230.98 dev eth0
192.168.238.0/24 via 192.168.230.98 dev eth0
192.168.228.0/22 via 192.168.230.99 dev eth0
# ip route del 192.168.238.0/24 via 192.168.230.98
# ip route show
192.168.230.96/29 dev eth0 proto kernel  scope link  src 192.168.230.100
192.168.230.0/24 via 192.168.230.98 dev eth0
192.168.228.0/22 via 192.168.230.99 dev eth0
```

Маршрут по-умолчанию добавляется с указанием ключевого слова 'default':

```
# ip route add default via 192.168.230.97
# ip route show
192.168.230.96/29 dev eth0 proto kernel  scope link  src 192.168.230.100
192.168.230.0/24 via 192.168.230.98 dev eth0
192.168.228.0/22 via 192.168.230.99 dev eth0
default via 192.168.230.97 dev eth0
```

После выполнения этих настроек таблица статической маршрутизации IPv4 будет полностью настроена в соответствии с приведённой выше схемой и будут доступны всех хосты из подсетей 192.168.230.0/24, 192.168.228.0/22.

Для работы с таблицей маршрутизации протокола IPv6 необходимо указывать ключ '-6' команды 'ip'. Просмотреть таблицу маршрутизации протокола IPv6 позволяет команда

```
# ip -6 route show
2001:db8:abcd::/48 dev eth0  proto kernel  metric 256  mtu 1500 advmss 1440
hoplimit 4294967295
fe80::/64 dev eth0  proto kernel  metric 256  mtu 1500 advmss 1440 hoplimit
4294967295
```

Так же, как и в случае IPv4, видны автоматически добавленные при назначении адресов IP маршруты на доступные через локальный интерфейс сети.

Добавляем маршруты на сети IPv6:

```
# ip -6 route add 2001:db8:abcd:8000::/52 via 2001:db8:abcd::230:98
# ip -6 route add 2001:db8:abc0::/40 via 2001:db8:abcd::230:99
# ip -6 route add default via 2001:db8:abcd::230:97
# ip -6 route show | sed -e 's/metric.*//'
2001:db8:abcd:8000::/52 via 2001:db8:abcd::230:98 dev eth0
2001:db8:abc0::/40 via 2001:db8:abcd::230:99 dev eth0
2001:db8:abcd::/48 dev eth0  proto kernel
fe80::/64 dev eth0  proto kernel
default via 2001:db8:abcd::230:97 dev eth0
```

После этого таблица статической маршрутизации IPv6 будет полностью настроена в соответствии с приведённой выше схемой, будут доступны хосты из подсетей 2001:db8:abcd:8000::/52 и 2001:db8:abc0::/40 , а запросы для остальных хостов будут направляться в Internet через маршрутизатор 2001:db8:abcd::230:97 .

Для настройки системы DNS достаточно внести IP-адреса серверов DNS в файл `/etc/resolv.conf`. Они указываются после ключевого слова `nameserver`, по одному адресу в строке:

```
# cat /etc/resolv.conf
nameserver 192.168.230.97
nameserver 2001:db8:abcd::230:97
```

Настройки сетевых интерфейсов, осуществляющиеся запусками команды `ip`, применяются на сетевом интерфейсе сразу после выполнения команды. Но полученное состояние не запоминается, и после выключения сетевого интерфейса все сделанные в таком, ручном режиме настройки пропадают.

Для сохранения конфигурации сетевых интерфейсов, в т.ч. между перезагрузками операционной системы, используются зависящие от конкретного дистрибутива Linux способы. Нужные параметры (способ конфигурирования сетевого интерфейса, адреса IP и маски сетей, маршруты и т. п.) записываются в файлы внутри каталога `/etc`; формат записи этих параметров и точное расположение нужных файлов конфигурации можно узнать в документации используемого дистрибутива. При загрузке операционной системы соответствующими утилитами используемой в дистрибутиве системы инициализации эти параметры считываются и применяются к сетевым интерфейсам или через вызовы команды `ip` из скриптов инициализации, или непосредственными вызовами методов API ядра Linux.

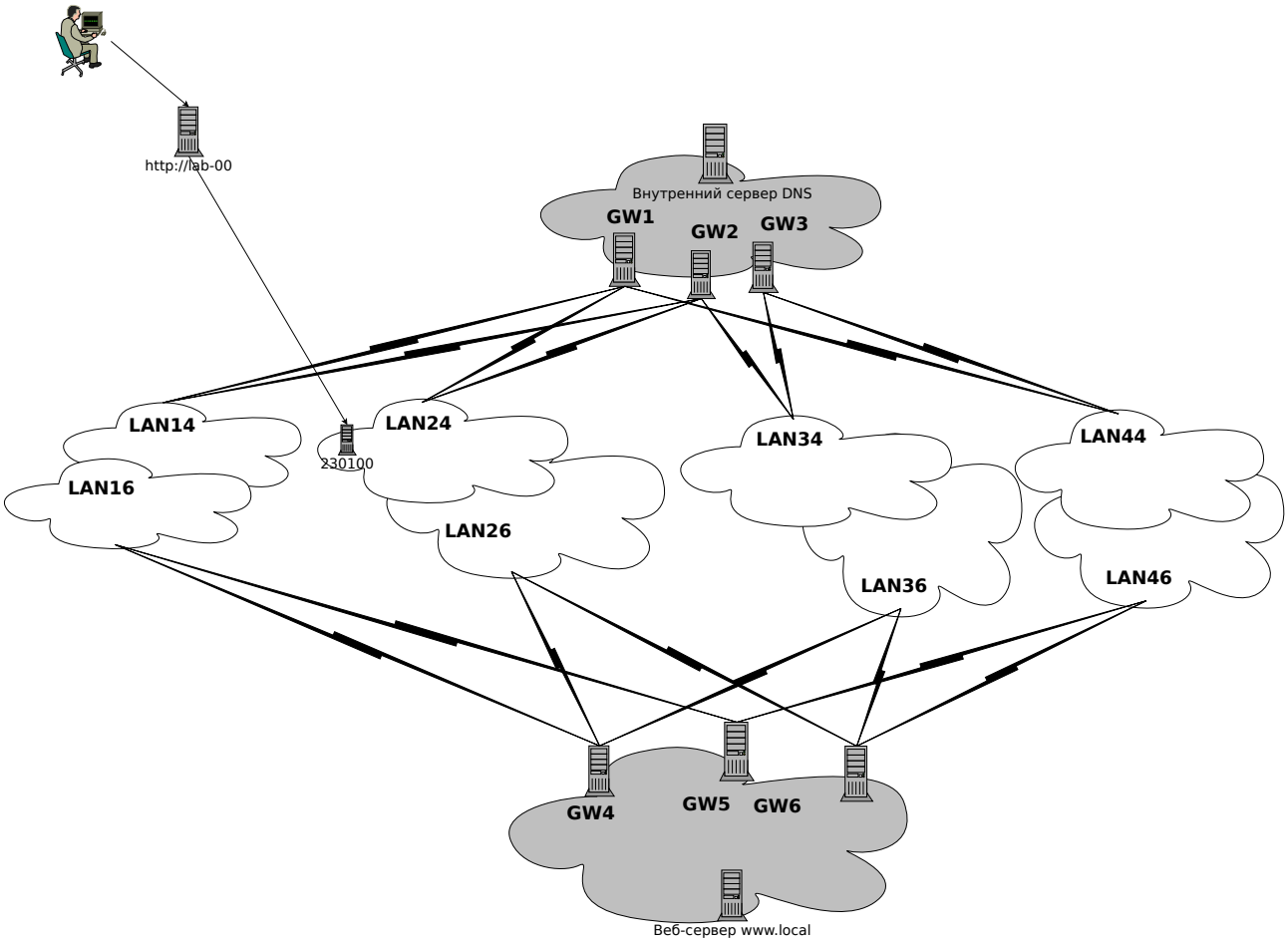
Выполнение лабораторной работы.

Лабораторная работа посвящена изучению основ семейства протоколов TCP/IP и маршрутизации сетях на основе этих протоколов.

В каждый виртуальный сервер добавлен интерфейс `eth0`.

Интерфейс `eth0` входит в одну из 4-х рабочих сетей Ethernet (2-ой уровень модели OSI ISO), в каждой из которых присутствуют по 2 сети TCP/IP (3-ий уровень модели OSI ISO).

Также имеется две сети, в которых находятся внутренний сервер DNS и внутренний веб-сервер `www.local` соответственно. Сети соединяются друг с другом через 6 маршрутизаторов GW1...GW6 . Известно, что для каждой из рабочих сетей 3-го уровня доступны два маршрутизатора. Общий вид сетей показан на рис:



Сети LAN14, LAN24, LAN34 и LAN44 (и соответствующие им маршрутизаторы GW1, GW2, GW3) работают только по протоколам IPv4.

Один из маршрутизаторов сетей LAN14 — LAN44 имеет адрес из списка, указанного в задании на лабораторную работу. Адрес второго маршрутизатора сети — первый или последний адрес IP соответствующей локальной сети.

Через маршрутизаторы GW1 — GW3 доступен внутренний сервер DNS, адрес которого также указан в задании на лабораторную работу.

Сети LAN16, LAN26, LAN36 и LAN46 (и соответствующие им маршрутизаторы GW4, GW5 и GW6) работают только по протоколам IPv6. Динамическое назначение адресов в этих сетях не поддерживается. Размер каждой из сетей — `/64` . Адреса IPv6 маршрутизаторов GW4 — GW6 можно получить с внутреннего сервера DNS по символьным именам `gw4.local`, `gw5.local` и `gw6.local` соответственно.

Через маршрутизаторы GW4 — GW6 доступен внутренний веб-сервер `www.local` . Узел веб-сервера `www.local` также является шлюзом в глобальную компьютерную сеть Internet, работающим по протоколам IPv6.

Задание на лабораторную работу размещено в репозитории Git с именем `lab-NN`, соответствующем номеру виртуальной машины (см. лабораторную работу №1), в ветке репозитория `task/lab3`. Для его получения требуется обновить репозиторий с сервера Git и перейти на указанную ветку. Перечень адресов коммутаторов находится в файле `gw.txt`; адрес сервера DNS – `ns.txt` . Обновление состояния репозитория с сервера Git и переключение на ветку `task/labs` выполняется командами вида

```
$ git pull --all
$ git checkout task/lab3
```

Требуется:

1) Перед началом работы следует проверить наличие обновлений пакетов, установленных в виртуальном сервере. В случае наличия обновлений — установить новые версии пакетов.

2) Определить, в какую сеть IPv4 входит интерфейс `eth0`.

Для этого следует, рассчитав адреса сетей и диапазоны адресов в этих сетях, последовательно выбирать произвольный IP из каждой сети, назначать его интерфейсу, и пробовать получить ответ, посылая запрос из-

вестному маршрутизатору из этой сети. Для проверки доступности маршрутизатора следует использовать команду `ping` (формат вызова: `ping [<IP>|<host>]`). Выбирая адрес IP, следует учитывать, что он должен быть уникальным, т.е. не должен совпадать ни с адресами каждого из маршрутизаторов, ни с адресами других VPS в данной сети.

3) Назначить интерфейсу адрес IPv4 соответствующей сети, определить адреса двух видимых из данной сети IPv4 маршрутизаторов.

4) Определить и задать статические маршруты до каждой из рабочих сетей LAN14 — LAN44, а также до сети внутреннего сервера DNS.

Для этого следует, последовательно прописывая маршруты для каждой из трёх оставшихся сетей на один из двух известных маршрутизаторов, проверять их доступность командой `ping` до получения положительного результата.

5) Настроить службу распознавания имён DNS.

В качестве сервера DNS следует выбрать адрес внутреннего сервера DNS. Сервер DNS указывается в записи `nameserver` в файле `/etc/resolv.conf` . После изменения файла `/etc/resolv.conf` в ALT Linux для обновления сетевых настроек следует выполнить команду

```
# update_chrooted all
```

6) Определить, в какую сеть IPv6 входит интерфейс `eth0`.

Для этого следует получить с внутреннего сервера DNS адреса IPv6 маршрутизаторов GW4 — GW5, используя команду `host`. Зная адреса маршрутизаторов и размер сетей LAN16 — LAN46 — `/64` , провести поиск сети аналогично п.1.

7) Назначить интерфейсу `eth0` адрес IPv6 соответствующей сети, определить адреса двух видимых из данной сети IPv6 маршрутизаторов.

8) Определить и задать статические маршруты до каждой из рабочих сетей LAN16 — LAN46, а также до сети внутреннего сервера `www.local` , аналогично п. 3.

9) Задать маршрут по умолчанию для сетей IPv6, проходящий через сервер `www.local` .

Для проверки доступности внешних серверов IPv6 можно использовать сервер `ipv6.google.com` .

10) Используя текстовый браузер elinks , просмотреть содержимое страниц внутреннего веб-сервера www.local и каких-либо внешних веб-серверов, доступных по IPv6 (например, ipv6.google.com , version6.ru , test-ipv6.com , …).

Выйти из браузера elinks можно или через меню, доступное по клавише <F10>, или с помощью «горячей клавиши» <q>.

11) Настроить проксирование запросов к внутреннему веб-серверу www.local.

Приняв от браузера запрос к какой-либо странице, веб-сервер может как сам обработать данный запрос, так и переслать его другому веб-серверу. Полученный ответ далее веб-сервер возвращает браузеру точно так же, как если бы страница была создана им самим. Такой механизм называется проксированием (от англ. proxy - полномочие; передача голоса; доверенность).

В ходе лабораторной работы №1 в виртуальный сервер был установлен веб-сервер lighttpd. Чтобы настроить проксирование запросов через него на внутренний веб-сервер www.local, нужно:

- подключить модуль mod_proxy веб-сервера, раскомментировав строку «include "conf.d/proxy.conf"» в файле modules.conf;
- задать в подключенном файле конфигурации модуля mod_proxy (conf.d/proxy.conf) секцию параметров вида:

```
proxy.server = ( ".htm" =>
                 ( "www.local" =>
                   (
                     "host" => "<www.local IP address>",
                     "port" => 80
                   )
                 )
               )
```

Согласно данным настройкам, при поступлении запросов к файлам *.htm веб-сервер lighttpd будет соединяться с другим веб-серверу по адресу, указанному в параметре "host", по протоколу TCP на порт, указанный в параметре "port", и передавать этому веб-серверу поступивший запрос. Запросы к другим файлам, в т.ч. к размещённым в рамках лабораторной работы №2 скриптам и индексной странице index.html, продолжат обрабатываться веб-сервером как раньше.

В параметре "host" требуется указать адрес IP сервера www.local.

Изменения настроек вступают в силу после перезапуска lighttpd.

После применения настроек проверьте работу веб-сервера lighttpd и получите со своего рабочего места через него страницу /index.htm от внутреннего веб-сервера www.local.

12) Обеспечить сохранение выполненных настроек при перезагрузке виртуального сервера.

Для этого предлагается сохранить выполненную ранее последовательность команд настройки интерфейса в виде скрипта командного интерпретатора в файле /root/bin/net.sh . Историю выполненных команд можно посмотреть через команду командного интерпретатора history. В файле требуется предусмотреть команды, необходимые как для настройки интерфейса, так и для удаления этих настроек.

Примерный вид файла /root/bin/net.sh –

```
#!/bin/sh
if   [[ "$1" == "start" ]]; then
   ## команды настройки сетевого интерфейса
elif [[ "$1" == "stop" ]]; then
   ## команды удаления настроек сетевого интерфейса
else
   echo "Usage: $0 [start|stop]"
fi
```

Вместо комментариев размещаются команды настройки и удаления настроек сетевого интерфейса.

Автоматический запуск данного скрипта при загрузке системы можно выполнить с помощью сервера crond, задав время выполнения скрипта в момент запуска сервера crond сокращением @reboot :

```
@reboot   /root/bin/net.sh start
```

Также по желанию, возможно вместо сервера crond использовать для автоматического запуска скрипта средства, предоставляемые системами инициализации systemd или sysvinit .

При задании последовательности команд в /root/bin/net.sh требуется обратить внимание на то, что файл /etc/resolv.conf при перезагрузке системы виртуальной машины восстанавливается в состояние по-умолчанию. Таким образом, в скрипте /root/bin/net.sh при настройке сетевого интерфейса требуется выполнить задание настроек сервера DNS (п. 5), как и восстановить старые значения при удалении настроек с него.

13) Перезагрузить виртуальный сервер, проверить восстановление сетевых настроек и доступность серверов сети после перезагрузки.

14) Созданный скрипт `/root/bin/net.sh` требуется разместить в репозитории Git `lab-NN`, в отдельной ветке с именем «`lab3`». Изменения скрипта в ходе доработки должны отражаться в коммитах репозитория. По завершению работы требуется опубликовать изменения репозитория `lab-NN` на сервере Git.

Задания на лабораторную работу.

1. Определить, в какие сети IPv4 и IPv6 входит интерфейс `eth0` .
2. Назначить интерфейсу `eth0` два адреса IP из соответствующих сетей.
3. Определить и задать статические маршруты до каждой из рабочих сетей, а также до сетей внешнего сервера DNS и внутреннего веб-сервера.
4. Настроить службу распознавания имён DNS.
5. Задать маршрут по-умолчанию для протокола IPv6.
6. Настроить проксирование запросов к внутреннему веб-серверу.
7. Обеспечить восстановление сделанных настроек интерфейса `eth0` при перезагрузке виртуального сервера.

Цель работы: настроить сетевой интерфейс таким образом, чтобы можно было достичь каждой из сетей, а также хостов IPv6 в Internet, как по адресу, так и по имени DNS; обеспечить сохранение данных настроек между перезагрузками виртуального сервера.

Контрольные вопросы.

1. Какие основные идеи заложены в сетевой модели OSI ISO?
2. Какие основные протоколы относятся к стеку TCP/IP?
3. Что такое адрес IPv4, форма его записи?
4. Что такое адрес IPv6, форма его записи?
5. Что такое сетевая и машинная части адреса IP?
6. На какие классы разделяются сети IPv4?
7. Что такое сетевая маска, какие они бывают?
8. Какие диапазоны адресов IPv4 выделены для частных сетей?
9. Какие способы конфигурирования сетевых интерфейсов используются в сетях TCP/IP поверх сетей Ethernet?
10. Почему в сетях IPv6, в отличие от сетей IPv4, широко используется автоматическая конфигурация сетевых интерфейсов?
11. Зачем нужны протоколы UDP и TCP? Чем они отличаются?
12. Что такое номер порта в протоколах TCP, UDP?
13. Как производится маршрутизатизация в сетях TCP/IP?
14. Что такое статический маршрут? Как он выглядит?
15. Каков порядок выбора маршрута для IP-пакета? Что такое маршрут по умолчанию?
16. Зачем нужна система DNS?
17. Каковы принципы работы системы DNS?

Содержание отчёта о выполнении лабораторной работы

В состав отчёта о выполнении лабораторной работы должны входить:

0. Титульная страница отчёта, с правильным полным названием учебного заведения, подразделения и кафедры, учебного курса, названия лабораторной работы, данных о выполнившем лабораторную работу студенте, данных о проверяющем лабораторную работу преподавателе. Титульная страница отчёта не должна содержать фактических, грамматических и синтаксических ошибок, стиль оформления должен соответствовать общеинститутским требованиям и не содержать явных ошибок форматирования.
1. Снимок экрана с итоговой конфигурацией сетевого интерфейса `eth0`. На снимке экрана должна быть видна только конфигурация уровня L3 (как IPv4, так и IPv6) сетевого интерфейса `eth0` и использовавшаяся для её получения команда.
2. Снимок экрана с результатами проверки доступности с использованием команды `ping` любого из маршрутизаторов IPv4 внешних по отношению к сети интерфейса `eth0` сетей LAN14...LAN44 и внутреннего сервера DNS. На снимке экрана должны быть видны использованные для проверки доступности команды `ping` и 3–5 строк полученных ответов от выбранного маршрутизатора и от внутреннего сервера DNS соответственно.
3. Снимок экрана с итоговой конфигурацией маршрутов IPv6. На снимке экрана должны быть видны только записи таблицы маршрутизации протокола IPv6 и использовавшаяся для их получения команда.
4. Снимок экрана с запущенным в виртуальном сервере текстовым браузером `elinks` и выводом веб-страницы любого внешнего веб-сервера, доступного по IPv6.
5. Часть системного журнала `journald`, содержащую начальные строки журнала после перезапуска виртуальной машины и строки, показывающие автоматический запуск скрипта настройки сетевого интерфейса. Записи журнала начиная с момента последней загрузки системы можно получить, указав в команде `journalctl` ключ `--boot` . Количество представленных на снимке экрана начальных строк должно соответствовать последней десятичной цифре выбранного для сетевого интерфейса `eth0` адреса IPv4. На снимке экрана должны быть видны использованные для получения данной выборки информации из системного журнала команды.

Формат отчёта о выполнении лабораторной работы – PDF, требуемое имя файла отчёта — `report-3.pdf` . Отчёт требуется разместить в репозитории Git в ветке `master` и опубликовать изменения на сервере репозиториев `git.edu.cbias.ru` .

Литература

1. Георгий Курячий, Кирилл Маслинский
   «Введение в ОС Linux» - учебное пособие по работе с операционной системой Linux, распространяется на условиях лицензии GNU FDL: http://heap.altlinux.org/issues/textbooks/LinuxIntro.george/index.html

2. Робачевский А.М., Немнюгин С.А., Стесик О.Л. Операционная система UNIX. – 2 изд., СПб.: BHV – Санкт-Петербург, 2005. – 636 с.

3. Документы RFC: http://www.rfc-editor.org

4. Эви Немет, Гарт Снайдер, Скотт Сибасс, Трент Р. Хейн. UNIX: Руководство системного администратора. 3-е изд. - СПб.: BHV - Санкт-Петербург, 2005 — 925 с.