Сети связи
Протоколы, сервисы и услуги в Интернет и IP-сетях

Тема 1
Введение. Общие понятия и определения.
Краткая история развития Интернет и IP-сетей.

доц. каф. СС и ПД, к.т.н. С. С. Владимиров

2020 г.

Структура дисциплины

1. Введение. Общие понятия и определения.
2. Модель ISO/OSI и стек протоколов TCP/IP.
3. Технология Ethernet.
4. Протоколы сетевого уровня — IPv4, ARP, ICMP.
5. Протоколы сетевого уровня — IPv6, ICMPv6.
6. Протоколы транспортного уровня — TCP, UDP.
7. Протоколы удалённого управления — Telnet, SSH.
8. Протоколы передачи файлов — FTP, TFTP, SFTP.
9. Система доменных имён. Протокол DNS.
10. Электронная почта. Протоколы SMTP, LMTP, POP3, IMAP.
11. Протоколы WWW — HTTP и HTTPS.
12. Протокол динамической настройки узла DHCP.

Учебно-методическое обеспечение дисциплины

Литература

1. Олифер, В. Г. Компьютерные сети. Принципы, технологии, протоколы : учебник для вузов / В. Г. Олифер, Н. А. Олифер. — СПб. : Питер, 2012. — 943 с.
2. Когновицкий, О. С. Структура и протоколы электронной почты в Интернет : учебное пособие / О. С. Когновицкий, Е. М. Доронин, Л. М. Свердлов — СПб. : СПбГУТ, 2004. — 95 с.
3. Таненбаум, Э. С. Компьютерные сети — СПб. : Питер, 2003. — 992 с.
4. Камер, Д. Э. Сети TCP/IP. Принципы, протоколы и структура — М. : «Вильямс», 2003.
5. Камышников, В. В. Основы архитектуры Internet : Уч. пособие для ВУЗов / В. В. Камышников, Ю. М. Казаченко, Н. М. Крикунов. — Самара : ПГАТИ, 2003.
6. Стивенс, У. Р. Протоколы TCP/IP. Практическое руководство — СПб: «БХВ-Петербург», 2003.
7. Столингс, В. Компьютерные сети, протоколы и технологии Интернета — СПб: «БХВ-Петербург», 2005.

Электронные курсы

1. «Интернет технологии и мультимедиа»
2. «Структура и протоколы электронной почты в INTERNET»
3. «Принципы построения компьютерных сетей»
4. «Семейство протоколов IPv6»

Основные понятия

Протокол передачи данных

Набор соглашений интерфейса логического уровня, которые определяют обмен данными между различными программами. Эти соглашения задают единообразный способ передачи сообщений и обработки ошибок при взаимодействии программного обеспечения разнесённой в пространстве аппаратуры, соединённой тем или иным интерфейсом.
Стандартизированный протокол передачи данных позволяет разрабатывать интерфейсы (на физическом уровне), не привязанные к конкретной аппаратной платформе и производителю.

Сигнальный протокол

Набор правил, использующихся для управления соединением — например, установки, переадресации, разрыва связи. Примеры протоколов: RTSP, SIP.

Сетевой протокол

Набор правил и действий (очерёдности действий), позволяющий осуществлять соединение и обмен данными между двумя и более включёнными в сеть устройствами. Новые протоколы для сети Интернет определяются IETF, а прочие протоколы — IEEE или ISO. ITU-T занимается телекоммуникационными протоколами и форматами. Сетевые протоколы строятся по многоуровневому принципу. Протокол некоторого уровня определяет одно из технических правил связи. В настоящее время для описания структуры сетевых протоколов используются модель ISO/OSI и модель TCP/IP.

Веб-служба или веб-сервис

Определяемая идентификатором ресурса программная система со стандартизированными интерфейсами. В обиходе веб-сервисами часто называют услуги, оказываемые в Интернете — поиск, электронная почта, хранение документов, передача сообщений и т. п.

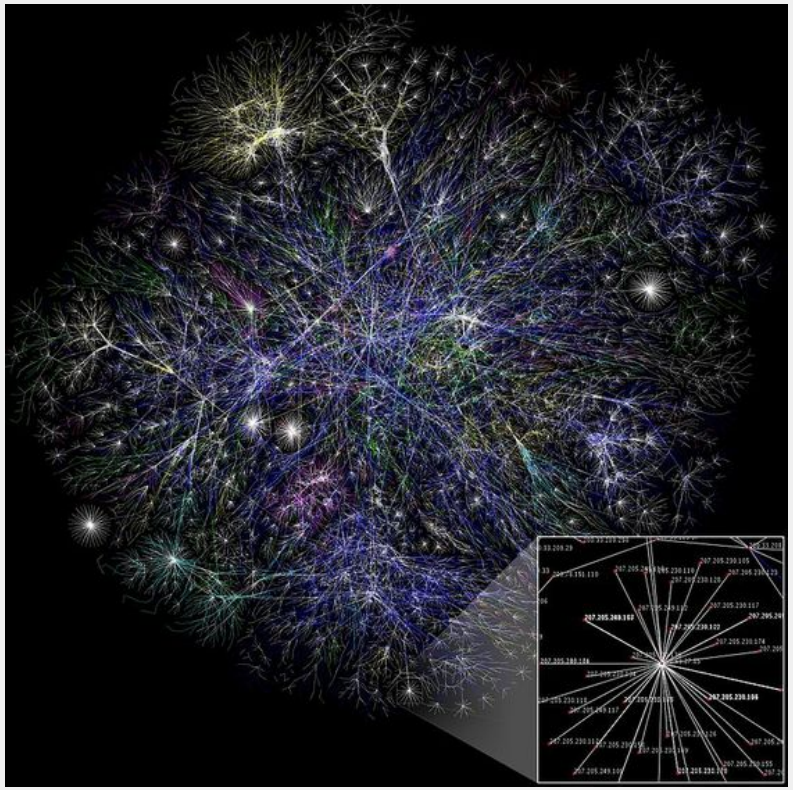
Общие сведения об Интернет

Интернет

Всемирная система объединённых компьютерных сетей для хранения и передачи информации. Построена на базе стека протоколов TCP/IP.

Интернет состоит из многих тысяч корпоративных, научных, правительственных и домашних компьютерных сетей. Объединение сетей разной архитектуры и топологии стало возможным, благодаря протоколу IP (Internet Protocol) и принципу маршрутизации пакетов данных.

На стыках сетей маршрутизаторы (программные или аппаратные) занимаются автоматической сортировкой и перенаправлением пакетов данных, исходя из IP-адресов получателей этих пакетов. Протокол IP образует единое адресное пространство в масштабах всего мира, но в каждой отдельной сети может существовать и собственное адресное подпространство, которое выбирается, исходя из размеров сети. Такая организация IP-адресов позволяет маршрутизаторам однозначно определять дальнейшее направление для каждого пакета данных. В результате между отдельными сетями Интернета не возникает конфликтов, и данные беспрепятственно и точно передаются из сети в сеть по всей планете и ближнему космосу.



Автономная система

Автономная система (Autonomous System, AS)

Система IP-сетей и маршрутизаторов, управляемых одним или несколькими операторами, имеющими единую, четко определенную политику маршрутизации с Интернетом (RFC 1930). AS имеет уникальный номер (ASN), необходимый для обмена информацией о маршрутах с другими AS, который осуществляется по протоколу междоменной маршрутизации BGP. Номера AS выделяются IANA, которая также выделяет IP-адреса региональным интернет-регистраторам (Regional Internet Registry) блоками. Локальные RIR затем присваивают организации номер AS из блока, полученного от IANA. Номер AS выдается при условии наличия адресного пространства не менее /24 (256 IP адресов). Информацию по AS удобно смотреть на сайте Hurricane Electric BGP Toolkit: https://bgp.he.net/.

Диапазоны номеров автономных систем (Autonomous System Number, ASN)

- 0–65535 — изначальный диапазон для ASN 16 бит (RFC 1930);
- 65536–4294967295 — новый диапазон для ASN 32 бита (RFC 4893).

Типы AS

- Многоинтерфейсная (multihomed) AS. Имеет соединения с более чем одним Интернет-провайдером. Это позволяет данной AS оставаться подключенной к Интернету в случае выхода из строя соединения с одним из Интернет-провайдеров. Кроме того, этот тип AS не разрешает транзитный трафик от одного Интернет-провайдера к другому.
- Ограниченная (stub) AS. Имеет единственное подключение к одной внешней автономной системе. Это расценивается как бесполезное использование номера AS, так как сеть размещается полностью под одним Интернет-провайдером и, следовательно, не нуждается в уникальной идентификации.
- Транзитная (transit) AS. Пропускает через себя транзитный трафик сетей, подключенных к ней. Таким образом, сеть A может использовать транзитную AS для связи с сетью B.

Интернет-организации: ISOC, IAB

Организации, ответственные за развитие сети Интернет

- Internet Society (ISOC) — Общество Интернета
- Internet Engineering Task Force (IETF) — Инженерный совет Интернета
- Internet Architecture Board (IAB) — Совет по архитектуре Интернета
- Internet Research Task Force (IRTF) — Исследовательская группа Интернет-технологий
- Internet Corporation for Assigned Names and Numbers (ICANN) — Корпорация по управлению доменными именами и IP-адресами
- Internet Assigned Numbers Authority (IANA) — Администрация адресного пространства Интернет
- Internet Engineering Steering Group (IESG) — Группа по выработке инженерного регламента Интернета
- World Wide Web Consortium (W3C) — Консорциум Всемирной паутины

Internet Society (ISOC) :: `http://www.internetsociety.org/`

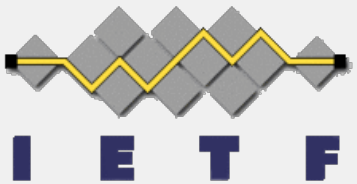
Общество Интернета — международная профессиональная организация, занимающаяся развитием и обеспечением доступности сети Интернет. Насчитывает более 20 тысяч индивидуальных членов и более 100 организаций-членов в 180 странах мира. Было основано в 1992 году, чтобы обеспечить организационную основу для консультативных и исследовательских групп, занимающихся развитием Интернета, включая IETF и IAB.

Internet Architecture Board (IAB) :: `https://www.iab.org/`

Совет по архитектуре Интернета — группа технических советников ISOC, которая осуществляет: надзор за архитектурой Интернета, включая его протоколы и связанные с ними процедуры; надзор за созданием новых стандартов Интернета; редактирование и публикацию серии документов RFC; консультации руководства ISOC по техническим, архитектурным и процедурным вопросам, связанным с Интернетом и его технологиями.

Интернет-организации: IETF

Internet Engineering Task Force (IETF) :: https://www.ietf.org/



Инженерный совет Интернета — открытое международное сообщество проектировщиков, учёных, сетевых операторов и провайдеров, созданное IAB в 1986 году и занимающееся развитием протоколов и архитектуры Интернета. Вся техническая работа осуществляется в рабочих группах IETF, занимающихся конкретной тематикой (например, вопросами маршрутизации, транспорта данных, безопасности и т. д.). Работа в основном ведётся через почтовые рассылки, но трижды в году проводятся собрания IETF.

Результаты деятельности рабочих групп оформляются в виде рабочих проектов, которые затем используются ISOC для кодификации новых стандартов.

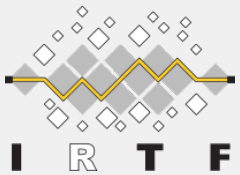
Первое собрание IETF состоялось 16 января 1986 года. На встрече присутствовал 21 спонсируемый правительством США исследователь. С 1992 года IETF вошло в состав ISOC, которое обеспечивает для IETF финансовую и правовую структуру, поскольку сам IETF не является организацией как таковой.

Задачи IETF

- Идентификация проблем и предложение решений в технических аспектах организации Интернета;
- Разработка спецификаций, стандартов и соглашений по общим архитектурным принципам протоколов Интернет;
- Вынесение рекомендаций относительно стандартизации протоколов на рассмотрение Internet Engineering Steering Group (IESG);
- Содействие широкому распространению технологий и стандартов, разрабатываемых в Internet Research Task Force (IRTF);
- Организация дискуссии для обмена информации в сообществе Интернета между учёными, разработчиками, пользователями, производителями оборудования и услуг, сетевыми администраторами и т. д.

Интернет-организации: IRTF, IESG, ICANN, IANA

Internet Research Task Force (IRTF) :: https://irtf.org/



Исследовательская группа Интернет-технологий — подразделение IAB, которое выполняет долгосрочные исследовательские программы, связанные с вопросами развития архитектуры, базовых протоколов и сетевых приложений сети Интернет. Руководящие органы IRTF назначаются IAB.

Internet Engineering Steering Group (IESG) :: https://www.ietf.org/iesg/

Группа по выработке инженерного регламента Интернета отвечает за техническое руководство деятельностью IETF и процесс стандартизации Интернета. Как подразделение ISOC, она отвечает за принятие новых спецификаций в качестве стандартов Интернета с соблюдением всех установленных процедур.

Internet Corporation for Assigned Names and Numbers (ICANN) :: https://www.icann.org/

Корпорация по управлению доменными именами и IP-адресами — международная некоммерческая организация, созданная 18 сентября 1998 года при участии правительства США для регулирования вопросов, связанных с доменными именами, IP-адресами и прочими аспектами функционирования Интернета.

Internet Assigned Numbers Authority (IANA) :: http://www.iana.org/

Администрация адресного пространства Интернет — выполняет функции управления пространствами IP-адресов, доменов верхнего уровня, а также регистрирует типы данных MIME и параметры прочих протоколов Интернета. Исполняется компанией Public Technical Identifiers, которая находится под контролем ICANN.

Интернет-организации: W3C

World Wide Web Consortium (W3C) :: `https://www.w3.org/`

Консорциум Всемирной паутины — организация, разрабатывающая и внедряющая технологические стандарты для Всемирной паутины. Консорциум возглавляет сэр Тимоти Джон Бернерс-Ли.
W3C разрабатывает для Интернета единые принципы и стандарты (W3C Recommendations), которые затем внедряются производителями программ и оборудования. Таким образом достигается совместимость между программными продуктами и аппаратурой различных компаний.

Консорциум был создан в 1994 году как консультативный орган для лидеров компьютерной индустрии. Крупнейшие мировые компании и корпорации договаривались в W3C об обеспечении совместимости своих продуктов и внедрении новых технологических стандартов. Первым крупным успехом консорциума стала стандартизация языка гипертекстовой разметки HTML в 1996 году.
Общее управление консорциумом осуществляют 3 организации:

- Массачусетский технологический институт (Massachusetts Institute of Technology, MIT) в США
- Европейский консорциум по исследованиям в области информатики и математики (European Research Consortium for Informatics and Mathematics, ERCIM) во Франции
- Университет Кейо (Keio University) в Японии

Международную координацию осуществляют так называемые «офисы W3C» (W3C Offices), которые созданы в 14 странах мира. Время от времени консорциум Всемирной паутины устраивает международные конференции.
Представительство консорциума в России было открыто W3C 16 февраля 2012 года совместно с Национальным исследовательским университетом «Высшая школа экономики» (НИУ ВШЭ).

Информационные документы Интернет: RFC и IEN

Request for Comments (RFC)

Рабочее предложение (запрос на отзыв) — документ из серии пронумерованных информационных документов Интернета, содержащих технические спецификации и стандарты, широко применяемые во всемирной сети. RFC рассматриваются как стандарты Интернета. В настоящее время первичной публикацией документов RFC занимается IETF под эгидой ISOC. Правами на RFC обладает ISOC. Почти все стандарты разрабатываются под эгидой научных или интернет-организаций (например W3C, IETF, консорциум Юникода).
Формат RFC появился в 1969 году при обсуждении проекта ARPAnet. Первые RFC распространялись в печатном виде на бумаге в виде обычных писем, с декабря 1969 г., когда заработали первые сегменты ARPAnet, документы начали распространяться в электронном виде. С 1969 по 1998 гг. бессменным и единственным редактором RFC был Джон Постел. После его смерти ISOC поручило редактирование и публикацию RFC Институту информационных наук Университета Южной Калифорнии. Запросы на отзывы официально существуют только на английском языке. Строгих требований к оформлению нет. Существует традиция выпуска первоапрельских шуточных RFC.

Список IEN размещён по адресам:

- `https://tools.ietf.org/rfc/index`
- `https://www.rfc-editor.org/rfc-index.html`

Internet Experiment Notes (IEN)

Заметки Экспериментального Интернета — серия технических заметок, создаваемая на протяжении раннего этапа развития TCP/IP и сети Интернет.
Идея IEN родилась в 1977 году, когда агенство DARPA начало развитие исследовательского проекта Internet. Существующие документы RFC относились к проекту ARPAnet, поэтому разработчики из проекта Internet решили создать собственную серию технических заметок, подобных RFC — Internet Experiment Notes. Редактором этой серии заметок также стал Джон Постел.
Всего было опубликовано 204 IEN в период с марта 1977 по сентябрь 1982. После этого, документация Internet была включена в RFC.

Список IEN размещён по адресу: `https://www.rfc-editor.org/ien/ien-index.html`

Информационные документы Интернет: RFC

Жизненный цикл стандарта (RFC 2026)

1. Выносится на всеобщее рассмотрение *интернет-проект* (Internet Draft). Проекты не имеют официального статуса и удаляются из базы через шесть месяцев после последнего изменения.
2. Удачный и непротиворечивый проект получает статус *предложенного стандарта* (Proposed Standard), и свой номер RFC. Желательно, но не обязательно, наличие программной реализации стандарта.
3. Следующая стадия — *проект стандарта* (Draft Standard) — означает, что предложенный стандарт принят сообществом, в частности, существуют две независимые по коду совместимые реализации разных команд разработчиков. В проекты стандартов ещё могут вноситься мелкие правки, но они считаются достаточно стабильными и рекомендуются для реализации.
4. Высший уровень — *стандарт Интернет*а (Internet Standard). Это спецификации с большим успешным опытом применения и зрелой формулировкой. Параллельно с нумерацией RFC они имеют свою собственную нумерацию STD. Список стандартов размещён по адресу: `https://www.rfc-editor.org/standards`
5. Многие старые RFC замещены более новыми версиями под новыми номерами или вышли из употребления. Такие документы получают статус *исторических* (Historic)

Виды RFC

1. *Экспериментальные* (Experimental) спецификации содержат информацию об экспериментальных исследованиях, интересных для интернет-сообщества. Например, прототипы, реализующие новые концепции.
2. *Информационные* (Informational) RFC предназначены для ознакомления общественности, не являются стандартами и не являются результатом консенсуса или рекомендациями. Некоторые проекты, не получившие статуса предложенного стандарта, но представляющие интерес, могут быть опубликованы как Информационные RFC.
3. *Лучший современный опыт* (Best Current Practice). Эта серия RFC содержит рекомендации по реализации стандартов, в том числе от сторонних организаций, а также внутренние документы о структуре и процедурах стандартизации.

Краткая история сети Интернет

1957 После запуска Советским Союзом первого искусственного спутника Земли, Министерство обороны США посчитало, что на случай войны США нужна надёжная система передачи информации. Агентство по перспективным оборонным научно-исследовательским разработкам США (DARPA) предложило разработать для этого компьютерную сеть.

1961 Леонард Клейнрок из MIT опубликовал первую статью «Информационный поток в больших коммуникационных сетях» по теории пакетной коммутации (июль 1961).

1962 Дж. К. Р. Ликлидер из MIT в августе 1962 г. опубликовал работу «Galactic Network», в которой обсуждалась его концепция компьютерной сети — глобального взаимосвязанного набора компьютеров, с помощью которых каждый мог бы быстро получать доступ к данным и программам с любого узла. Начиная с октября 1962 г., Ликлидер возглавил научно-исследовательскую компьютерную программу в агентстве DARPA. Работая в DARPA, он убедил своих последователей в DARPA Ивана Сазерленда, Боба Тейлора и ученого из MIT Л Дж. Робертса в важности этой концепции сети.
В этом же году Пол Бэран из RAND Corporation подготовил доклад «On Distributed Communication Networks». Он предложил использовать децентрализованную систему связанных между собой равноправных компьютеров, которая даже при разрушении её части будет работоспособна.

1964 Л Клейнрок из MIT опубликовал первую книгу «Сети связи: стохастический поток и задержка сообщений» по теории пакетной коммутации.

1965 Дж. Робертс и Т. Меррилл соединили компьютер TX-2, находящийся в штате Массачусетс, к компьютеру Q-32 в Калифорнии с использованием низкоскоростной телефонной линии.

1967 Дж. Робертс опубликовал доклад «Связь между несколькими компьютерными сетями и компьютерами» на конференции в Гатлинбурге (Теннесси), в котором описал концепцию будущей компьютерной сети ARPAnet. На той же конференции, был также доклад Д. Дэвиса и Р. Скантлбери из National Physical Laboratory (Великобритания) по концепции сети на основе передачи пакетов. Выяснилось, что работа в MIT (1961–1967), в группе RAND (1962–1965) и NPL (1964–1967) велась параллельно, при этом ученые-исследователи не знали о работе других.

Краткая история сети Интернет

1968 DARPA опубликовала заказ на разработку пакетных коммутаторов — сопрягающих процессоров сообщений (IMP). Конкурс выиграла группа Ф. Харта из компании Heart Bolt Beranek and Newman (BBN), которая совместно с Робертом Каном в 1969 г. разработала первый IMP.

1969 Сеть ARPAnet на основе коммутаторов IMP компании BBN объединила четыре университета: Калифорнийский университет в Лос-Анджелесе (UCLA, University of California, Los Angeles), Стэндфордский исследовательский институт (Stanford Research Institute), Калифорнийский университет в Санта-Барбаре (University of California, Santa Barbara) и университет штата Юта (Utah State University). Первый сервер ARPAnet был установлен 2 сентября 1969 г. в UCLA. 29 октября 1969 г. в 21:00 между двумя первыми узлами сети ARPAnet, находящимися на расстоянии в 640 км — UCLA и SRI — провели первый сеанс связи.

1970 Network Working Group (NWG) под руководством С. Крокера, завершила работу над созданием первоначального протокола связи между узлами сети ARPAnet — протокол управления сетью (NCP).

1972 Представлено первое приложение для новой сети — электронная почта.

1973 К сети ARPAnet были подключены через трансатлантический телефонный кабель первые иностранные организации из Великобритании и Норвегии, сеть стала международной.

1974 Internet Network Working Group (INWG) представила универсальный протокол передачи данных и объединения сетей — TCP/IP.

1980 В университете Дьюка (Северная Каролина) создана сеть USENET, основанная на протоколах связи UUCP, встроенных в систему UNIX.

1981 А. Фачс и Г. Фримэн основали сеть BITNET, соединив Городской университет Нью-Йорка (CUNY) и Йельский университет (YALE).

1983 ARPAnet перешла с протокола NCP на TCP/IP. Термин «Интернет» закрепился за сетью ARPAnet. Сеть разделилась на MILNET, собственно сеть для военных нужд, и ARPAnet, использовавшуюся в исследовательских целях.

Краткая история сети Интернет

1986 Создана сеть NSFNet (the National Science Foundation Network — Сеть Национального научного фонда), в создании которой приняли непосредственное участие NASA (сеть CSNET) и Министерство энергетики (сеть MFENet). USENET и BITNET вошли в NSFNET. Таким образом, NSFNET была составлена из более мелких сетей. Она имела большую пропускную способность, чем ARPAnet. К этой сети за год подключились около 10 тыс. компьютеров, название «Интернет» начало плавно переходить к NSFNet.

1988 Разработан протокол Internet Relay Chat (IRC), благодаря чему в интернете стало возможно общение в реальном времени.

1989 Британский учёный Тим Бернерс-Ли из CERN (Conseil Europeen pour la Recherche Nucleaire — Европейская организация по ядерным исследованиям) предложил концепцию Всемирной паутины — World Wide Web (WWW). В течение двух лет он разработал протокол HTTP, язык HTML и идентификаторы URI.

1990 Сеть ARPAnet прекратила своё существование, полностью проиграв конкуренцию NSFNet. Сами понятия ARPAnet, NSFNet сменились термином «Интернет». Зафиксировано первое подключение к интернету по телефонной линии (dialup access).

1991 Всемирная паутина стала общедоступна в Интернете.

1993 Появился знаменитый веб-браузер NCSA Mosaic Марка Андриссена.

1994 Образовался консорциум W3C (W3 Consortium), который объединил ученых из разных университетов и компаний (в том числе Netscape и Microsoft). С этого времени комитет стал заниматься всеми стандартами в мире Интернета. Первым шагом организации стала разработка спецификации HTML 2.0.

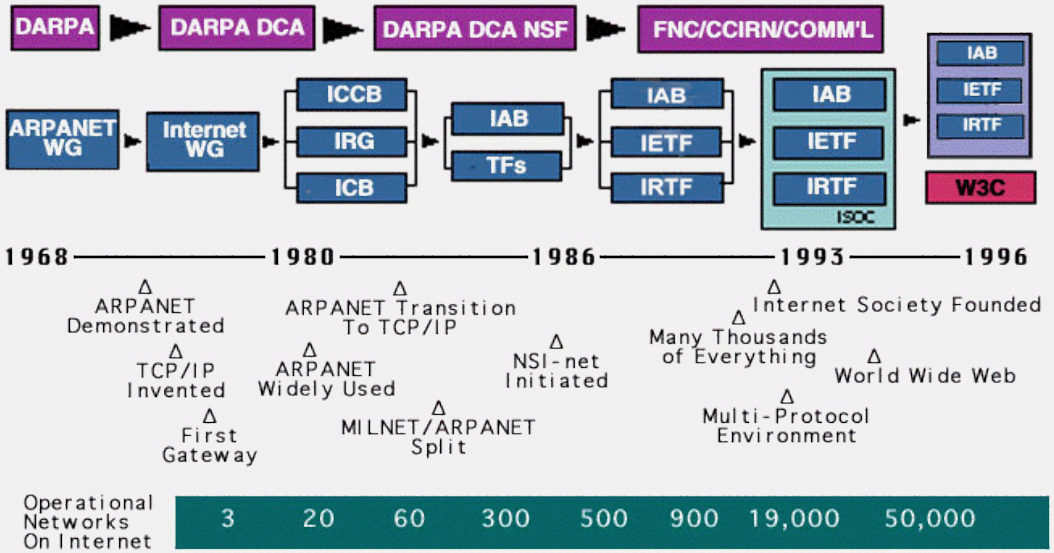
1995 Темпы роста сети Интернет показали, что регулирование вопросов подключения и финансирования не может находиться в руках одного NSF. NSFNet вернулась к роли исследовательской сети, маршрутизацией всего трафика интернета теперь занимались сетевые провайдеры, а не суперкомпьютеры Национального научного фонда.

Краткая история сети Интернет

24 октября 1995 г. Федеральная комиссия по сетям (FNC) единодушно приняла резолюцию, определяющую термин «Интернет». Это определение было разработано в ходе консультаций с членами сообществ Интернета и обладателей прав на интеллектуальную собственность.

Резолюция

Федеральная комиссия по сетям (FNC) пришла к единому мнению, что термин «Интернет» раскрывает следующее определение. Под «Интернетом» понимается глобальная информационная система, которая (1) является логически связанной с помощью глобального уникального адресного пространства на основе протокола Интернета (IP) или его последующих расширений/дополнений; (2) способна поддерживать связь с использованием пакета протоколов Transmission Control Protocol/Internet Protocol (TCP/IP) или его последующих расширений/дополнений и/или других IP-совместимых протоколов; и (3) предоставляет, использует или делает доступными, на общедоступном или частном уровне, услуги верхнего уровня, строящиеся на основе связи и связанной инфраструктуры, которая здесь описана.



Интернет-организации

Временная шкала

События

Число сетей в Интернете

Источники

- Материалы с сайта `https://wikipedia.org/`
- Материалы с сайта `http://www.internetsociety.org/`
- Материалы с сайта `https://www.ietf.org/`
- Материалы с сайта `https://www.rfc-editor.org/`
- Краткая история сети Интернет / Б. М. Лейнер, В. Дж. Серф, Д. Д. Кларк, Р. Е. Кан, Л. Клейнрок, Д. С. Линч, Дж. Постел, Л. Дж. Робертс, С. Вулф. С сайта `http://www.internetsociety.org/`
- История связи и перспективы развития телекоммуникаций : учебное пособие / Ю. Д. Украинцев, М. А. Цветов. — Ульяновск : УлГТУ, 2009. — 128 с.

Сети связи
Протоколы, сервисы и услуги в Интернет и IP-сетях

Тема 2
Модель OSI и стек протоколов TCP/IP

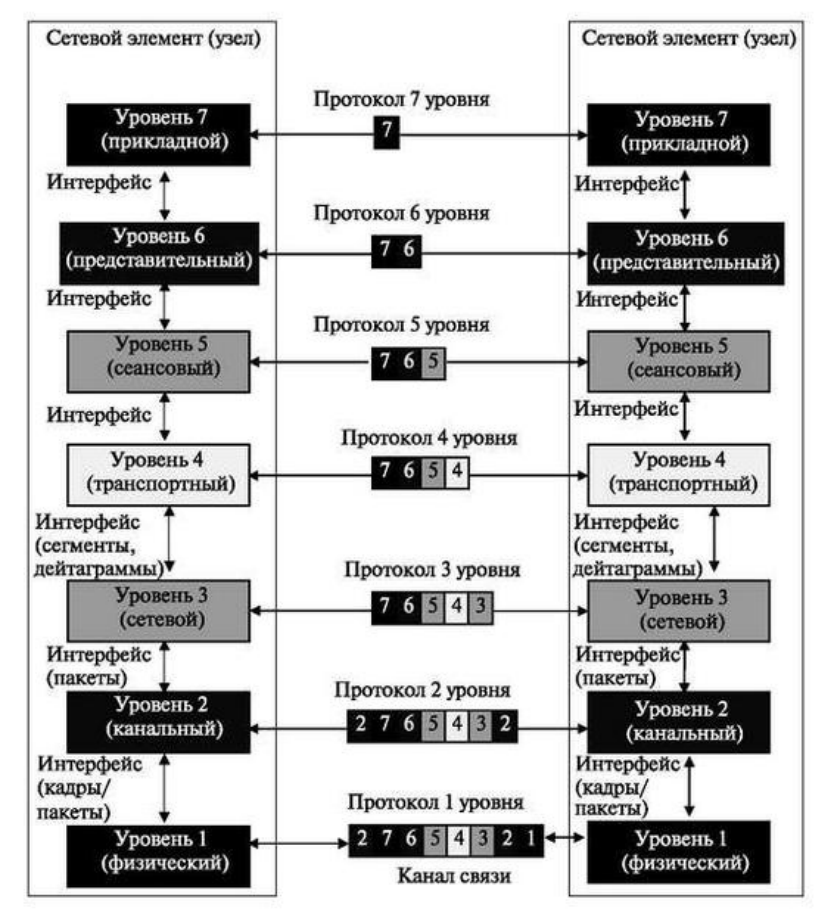
доц. каф. СС и ПД, к.т.н. С. С. Владимиров

2020 г.

Модель ISO/OSI

Сетевая модель OSI — Open Systems Interconnection Basic Reference Model

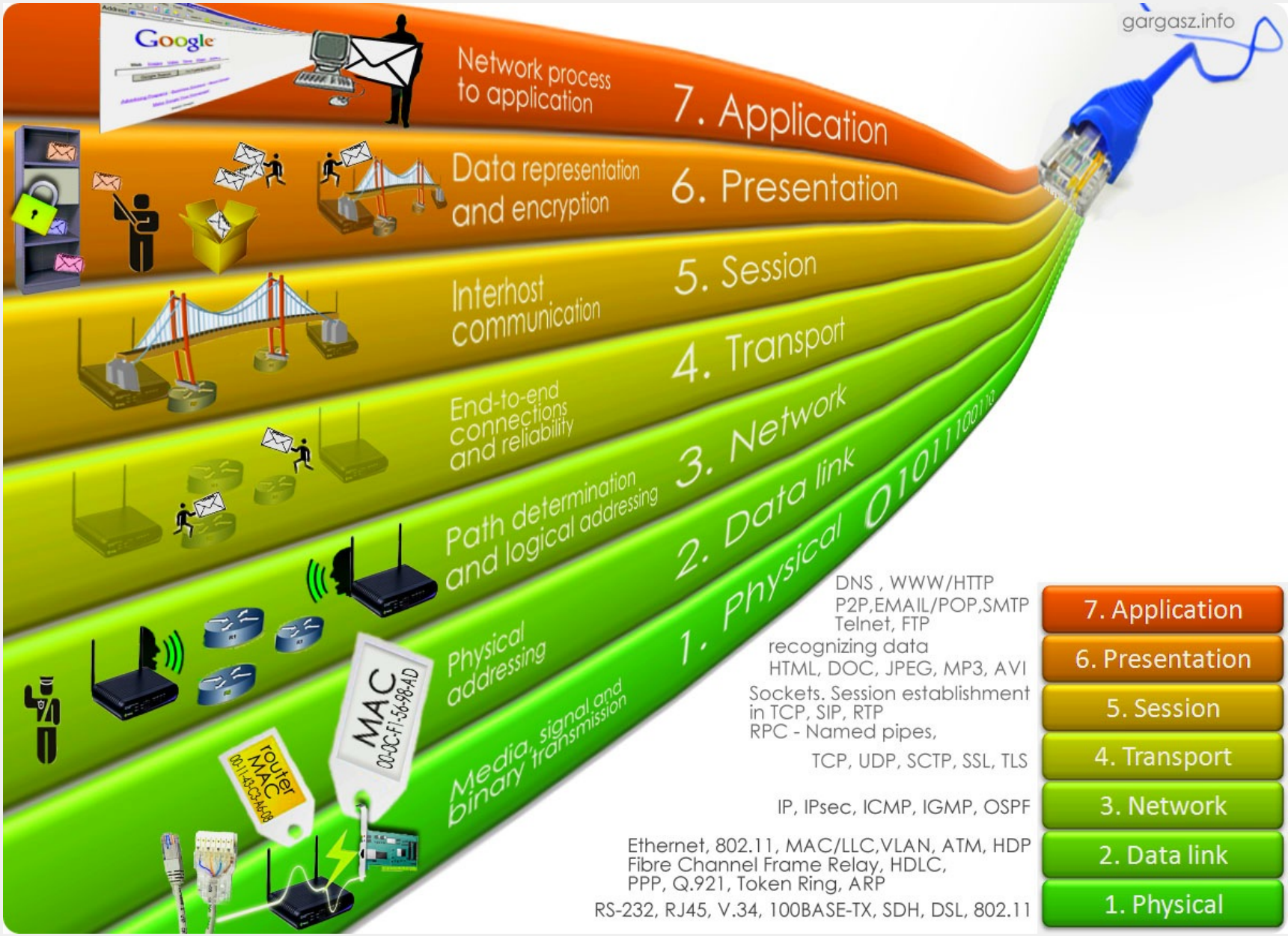
Базовая эталонная модель взаимодействия открытых систем, ЭМВОС (1978 г) — сетевая модель стека сетевых протоколов OSI/ISO (ГОСТ Р ИСО/МЭК 7498-1-99).



Любой протокол модели OSI должен взаимодействовать либо с протоколами своего уровня, либо с протоколами на единицу выше и/или ниже своего уровня. Взаимодействия с протоколами своего уровня называются горизонтальными, а с уровнями на единицу выше или ниже — вертикальными. Любой протокол модели OSI может выполнять только функции своего уровня и не может выполнять функций другого уровня, что не выполняется в протоколах альтернативных моделей.

Каждому уровню модели с некоторой долей условности соответствует свой **операнд** — *логически неделимый элемент данных, которым на отдельном уровне можно оперировать в рамках модели и используемых протоколов*. Он получил название **протокольный блок данных (Protocol Data Unit, PDU)**. На физическом уровне мельчайшая единица — бит, на канальном уровне информация объединена в кадры, на сетевом — в пакеты, на транспортном — в сегменты/датаграммы. Любой фрагмент данных, логически объединённых для передачи — кадр, пакет, датаграмма — считается сообщением. Именно сообщения в общем виде являются операндами сеансового, представительского и прикладного уровней.

Модель ISO/OSI. Схема



Модель ISO/OSI. Уровни

Прикладной уровень (уровень приложений; Application Layer)

Верхний уровень модели, обеспечивающий взаимодействие пользовательских приложений с сетью.

Уровень представления (Presentation Layer)

Обеспечивает преобразование протоколов и шифрование/дешифрование данных. Запросы приложений, полученные с прикладного уровня, на уровне представления преобразуются в формат для передачи по сети, а полученные из сети данные преобразуются в формат приложений. На этом уровне может осуществляться сжатие/распаковка или кодирование/декодирование данных, а также перенаправление запросов другому сетевому ресурсу, если они не могут быть обработаны локально.

Уровень представлений обычно представляет собой промежуточный протокол для преобразования информации из соседних уровней. Это позволяет осуществлять обмен между приложениями на разнородных компьютерных системах прозрачным для приложений образом. Уровень представлений обеспечивает форматирование и преобразование кода. Форматирование кода используется для того, чтобы гарантировать приложению поступление информации для обработки, которая имела бы для него смысл. При необходимости этот уровень может выполнять перевод из одного формата данных в другой.

Уровень представлений имеет дело не только с форматами и представлением данных, он также занимается структурами данных, которые используются программами. Таким образом, уровень 6 обеспечивает организацию данных при их пересылке.

Сеансовый уровень (Session Layer)

Обеспечивает поддержание сеанса связи, позволяя приложениям взаимодействовать между собой длительное время. Уровень управляет созданием/завершением сеанса, обменом информацией, синхронизацией задач, определением права на передачу данных и поддержанием сеанса в периоды неактивности приложений.

Модель ISO/OSI. Уровни (2)

Транспортный уровень (Transport Layer)

Предназначен для обеспечения надёжной передачи данных от отправителя к получателю. При этом уровень надёжности может варьироваться в широких пределах.

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.

Канальный уровень (Data Link Layer)

Предназначен для обеспечения взаимодействия сетей по физическому уровню и контролем над ошибками, которые могут возникнуть. Полученные с физического уровня данные, представленные в битах, он упаковывает в кадры, проверяет их на целостность и, если нужно, исправляет ошибки (формирует повторный запрос поврежденного кадра) и отправляет на сетевой уровень. Канальный уровень может взаимодействовать с одним или несколькими физическими уровнями, контролируя и управляя этим взаимодействием.
Спецификация **IEEE 802** разделяет этот уровень на два подуровня: MAC (Media Access Control) регулирует доступ к разделяемой физической среде, LLC (Logical Link Control) обеспечивает обслуживание сетевого уровня.

Физический уровень (Physical Layer)

Нижний уровень модели, который определяет метод передачи данных, представленных в двоичном виде, от одного устройства (компьютера) к другому. Функции физического уровня реализуются на всех устройствах, подключенных к сети. Со стороны компьютера функции физического уровня выполняются сетевым адаптером или последовательным портом. К физическому уровню относятся физические, электрические и механические интерфейсы между двумя системами.

Модель DOD (модель TCP/IP)

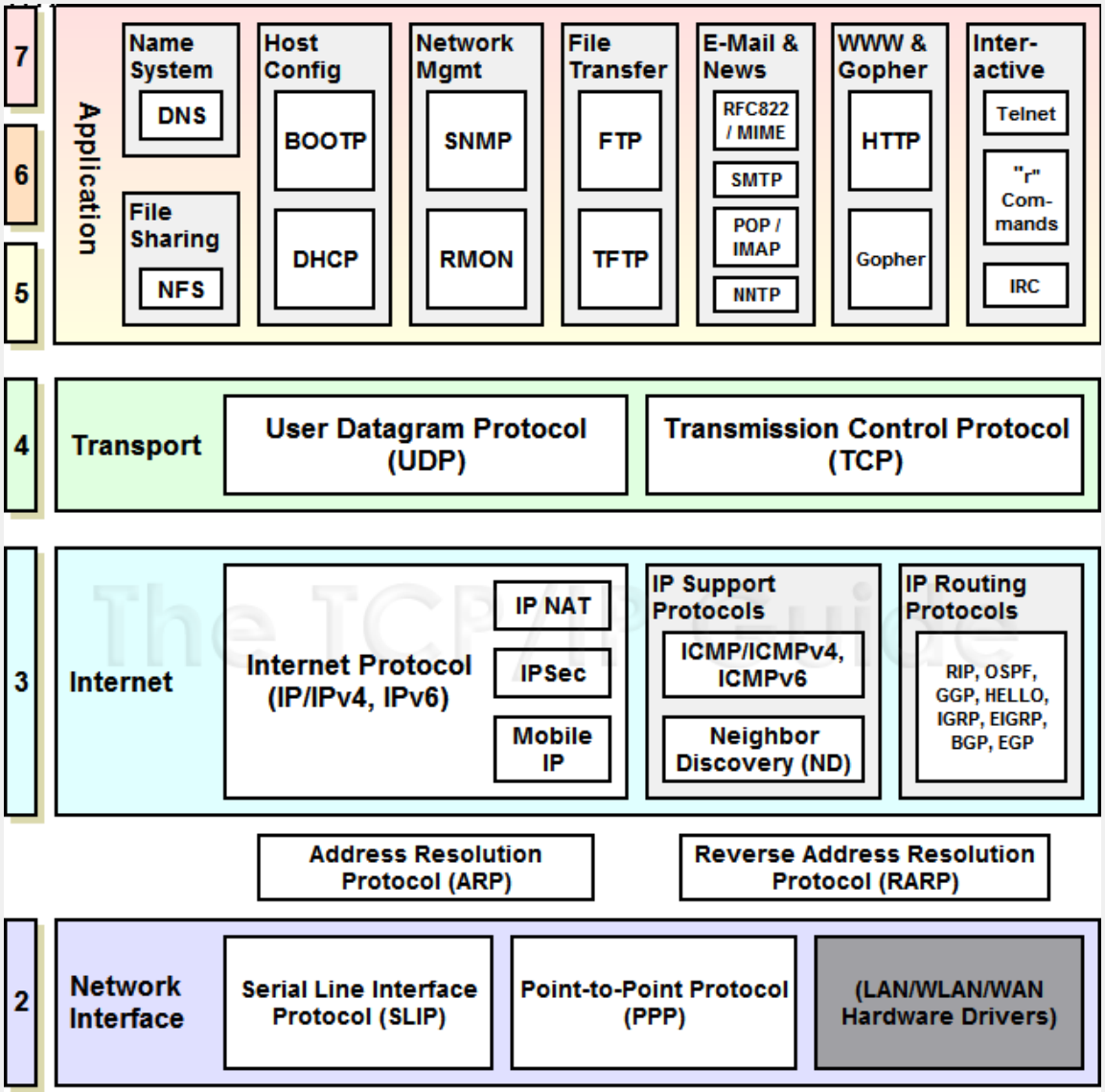
Модель DOD (Модель TCP/IP)

Модель сетевого взаимодействия, разработанная Министерством обороны США (Department of Defense), практической реализацией которой является стек протоколов TCP/IP.

Уровни модели TCP/IP

1. **Уровень приложений (Process/Application).** Верхний уровень модели, включающий протоколы, обрабатывающие данные пользователей и осуществляющие управление обменом данными между приложениями. На этом уровне стандартизируется представление данных. Этот уровень объединяет функции прикладного, представительского и сеансового уровней модели OSI.
2. **Транспортный уровень (Transport).** Содержит протоколы для обеспечения целостности данных при сквозной передаче. Обеспечивает управление инициализацией и закрытием соединений.
3. **Межсетевой уровень (Internet).** Содержит протоколы для маршрутизации сообщений в сети. Все протоколы транспортного уровня используют Internet Protocol (IP) для доставки данных от источника к получателю.
4. **Уровень сетевого доступа (Network Access).** Содержит протоколы для физической доставки данных к сетевым устройствам. Этот уровень размещает данные в фрейме.

Модель TCP/IP. Схема



Источники

1. Материалы с сайта `https://wikipedia.org/`
2. Информационно-вычислительные сети : учебное пособие / Д. А. Капустин, В. Е. Дементьев. — Ульяновск : УлГТУ, 2011.
3. Компьютерные сети. Принципы, технологии, протоколы: Учебник для вузов / В. Г. Олифер, Н. А. Олифер. — СПб. : Питер, 2010.

Сети связи
Протоколы, сервисы и услуги в Интернет и IP-сетях

Тема № 3
Технология Ethernet

доц. каф. СС и ПД, к.т.н. С. С. Владимиров

2020 г.

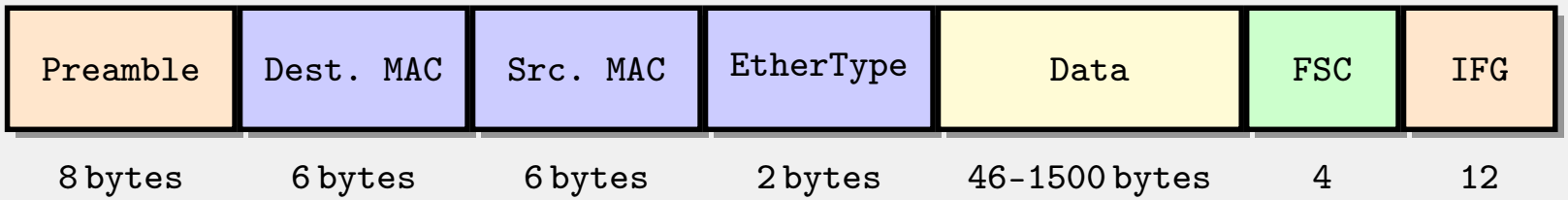
Технология Ethernet

Ethernet

Семейство технологий пакетной передачи данных для компьютерных сетей. Стандарты Ethernet определяют на физическом уровне модели OSI — проводные соединения и электрические сигналы, а на канальном уровне — формат кадров и протоколы управления доступом к среде. Ethernet в основном определяется стандартами IEEE группы 802.3.

Ethernet — это сетевой стандарт, основанный на технологиях экспериментальной сети Ethernet Network, которую фирма Xerox разработала и реализовала в 1975 году. В 1980 году фирмы DEC, Intel и Xerox совместно разработали и опубликовали стандарт Ethernet версии II для сети, построенной на основе коаксиального кабеля. Поэтому стандарт Ethernet иногда называют стандартом DIX по заглавным буквам названий фирм. Стандарт IEEE 802.3 был разработан на основе именно этого стандарта.

Структура кадра Ethernet



1. Preamble. 8 байт. Преамбула. Используется для синхронизации.
2. Destination MAC. 6 байт. MAC-адрес назначения.
3. Source MAC. 6 байт. MAC-адрес источника.
4. EtherType. 2 байта. Содержит код типа протокола верхнего уровня. Например, 0x0800 для протокола IP.
5. Data. 46–1500 байт. Поле данных. Минимальная длина поля составляет 46 байт, что требуется для корректной работы механизма обнаружения коллизий. Если данных не хватает, то добавляется поле заполнения, чтобы обеспечить минимальную длину поля данных (46 байт).
6. Frame Check Sequences (FCS). 4 байта. Контрольная сумма для выявления ошибок передачи. Используется код CRC-32.
7. Inter Frame Gap (IFG). 12 байт. Межкадровый интервал.

Метод доступа к среде

В сетях Ethernet используется метод доступа к среде передачи данных, называемый *методом коллективного доступа с опознаванием несущей и обнаружением коллизий* (carrier-sense-multiply-access with collision detection, **CSMA/CD**).
Этот метод используется исключительно в сетях с общей шиной. Все компьютеры такой сети имеют непосредственный доступ к общей шине, поэтому она может быть использована для передачи данных между любыми двумя узлами сети. Кабель, к которому подключены все станции, работает в режиме коллективного доступа (multiply-access,MA). Участок сети, в котором возможны коллизии, называют доменом (областью) коллизий (collision domain).

Все данные, передаваемые по сети, помещаются в кадры и снабжаются адресами отправителя и получателя. Затем кадр передается по кабелю. Все станции, подключенные к кабелю, могут распознать факт передачи кадра, и станция определившая свой адрес в заголовке кадра, записывает его содержимое во внутренний буфер, обрабатывает полученные данные и посылает по кабелю кадр-ответ согласно адресу отправителя.
При описанном подходе возможна ситуация, когда две станции одновременно пытаются передать кадр данных по общему кабелю. Для уменьшения вероятности этой ситуации непосредственно перед отправкой кадра передающая станция проверяет кабель, чтобы обнаружить, не передается ли уже по кабелю кадр данных от другой станции. Если опознается несущая (carrier-sense, CS), то станция откладывает передачу своего кадра до окончания чужой передачи. Но даже при таком алгоритме две станции одновременно могут решить, что по шине в данный момент времени нет передачи, и начать одновременно передавать свои кадры — происходит коллизия, так как содержимое обоих кадров сталкивается на общем кабеле, что приводит к искажению информации.
Для корректной обработки коллизии, все станции одновременно наблюдают за возникающими на кабеле сигналами. Если передаваемые и наблюдаемые сигналы отличаются, то фиксируется обнаружение коллизии (collision detection, CD). Для увеличения вероятности немедленного обнаружения коллизии всеми станциями сети, ситуация коллизии усиливается посылкой в сеть станциями, начавшими передачу своих кадров, специальной последовательности битов, называемой jam-последовательностью.
После обнаружения коллизии передающая станция обязана прекратить передачу и ожидать в течение короткого случайного интервала времени, а затем может снова сделать попытку передачи кадра.
Из описания метода доступа видно, что он носит вероятностный характер, и вероятность успешного получения в свое распоряжение общей среды зависит от загруженности сети, то есть от интенсивности возникновения в станциях потребности передачи кадров. При разработке этого метода предполагалось, что скорость передачи данных в 10 Мб/с очень высока по сравнению с потребностями компьютеров во взаимном обмене данными, поэтому загрузка сети будет всегда небольшой.

Метод доступа к среде (2)

Основные временные и логические соотношения в CSMA/CD

1. Между двумя последовательно передаваемыми по общей шине кадрами информации должна выдерживаться пауза в 9.6 мкс; эта пауза нужна для приведения в исходное состояние сетевых адаптеров узлов, а также для предотвращения монопольного захвата среды передачи данных одной станцией.
2. При обнаружении коллизии (условия ее обнаружения зависят от применяемой физической среды) станция выдает в среду специальную 32-битную последовательность (jam-последовательность), усиливающую явление коллизии для более надежного распознавания ее всеми узлами сети.
3. После обнаружения коллизии каждый узел, который передавал кадр и столкнулся с коллизией, после некоторой задержки пытается повторно передать свой кадр. Узел делает максимально 16 попыток передачи этого кадра информации, после чего отказывается от его передачи. Величина задержки выбирается как равномерно распределенное случайное число из интервала, длина которого экспоненциально увеличивается с каждой попыткой. Такой алгоритм выбора величины задержки снижает вероятность коллизий и уменьшает интенсивность выдачи кадров в сеть при ее высокой загрузке.

Все параметры протокола Ethernet подобраны для четкого определения коллизий при нормальной работе узлов сети. Для этого минимальная длина поля данных кадра должна быть ≥ 46 байт (общая минимальная длина кадра ≥ 72 байт (576 бит)). Длина кабельной системы выбирается так, чтобы за время передачи кадра минимальной длины сигнал коллизии успел бы распространиться до самого дальнего узла сети. Для скорости 10 Мб/с максимальное расстояние между двумя любыми узлами сети ≤ 2500 м.

В случае повторных коллизий существует максимальное число попыток повторной передачи кадра (attempt limit), равное 16. При достижении предела фиксируется ошибка передачи кадра, сообщение о которой передается протоколу верхнего уровня. Для уменьшения интенсивности коллизий, каждый узел с каждой новой попыткой случайным образом увеличивает длительность паузы между попытками.

Интервал отсрочки (slot time, t_{st}) — это время, в течение которого станция гарантированно может узнать, что в сети нет коллизии. Это время тесно связано с другим важным временным параметром сети — окном коллизий (collision window, t_{cw}). Окно коллизий равно времени двукратного прохождения сигнала между самыми удаленными узлами сети — наихудшему случаю задержки, при которой станция еще может обнаружить, что произошла коллизия. Интервал отсрочки выбирается равным величине окна коллизий плюс некоторая дополнительная величина задержки Δt для гарантии: $t_{st} = t_{cw} + \Delta t$.

В стандартах 802.3 большинство временных интервалов измеряется числом межбитовых интервалов, величина которых для битовой скорости 10 Мб/с составляет 0.1 мкс и равна времени передачи одного бита (bit time, bt). В стандарте 802.3 $t_{st} = 512$ bt для максимальной длины коаксиального кабеля в 2.5 км и минимальной длины кадра 64 байта (без преамбулы), т. к. при кадрах меньшей длины станция может передать кадр и не успеть заметить факт возникновения коллизии из-за того, что искаженные коллизией сигналы дойдут до станции в наихудшем случае после завершения передачи. Такой кадр будет просто потерян.

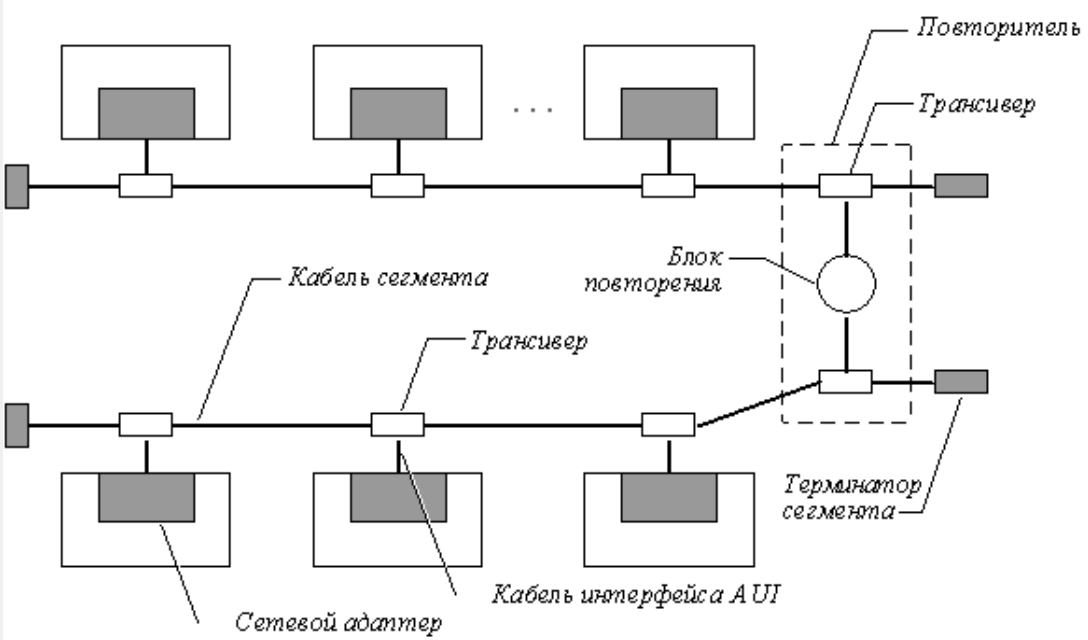
Физические спецификации Ethernet

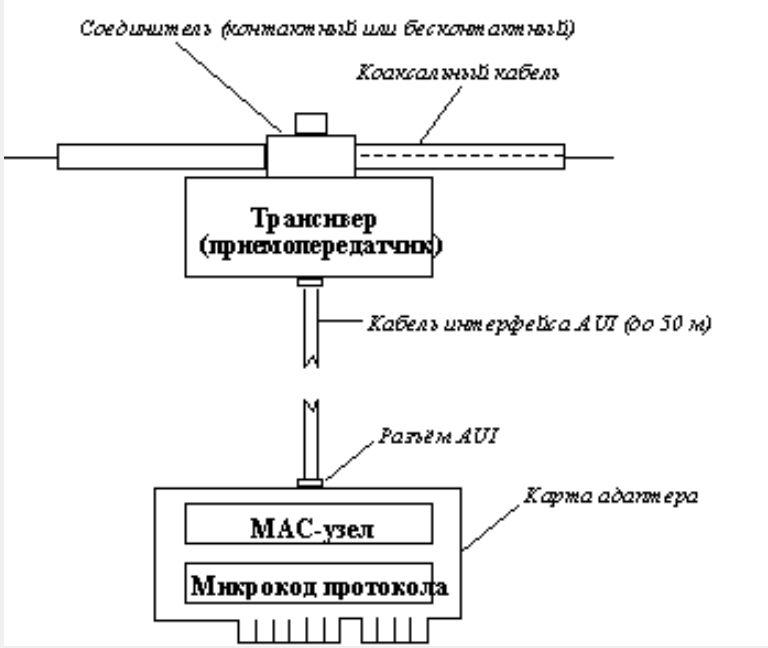
- **10Base-5** — коаксиальный кабель диаметром 0.5 дюйма («толстый» коаксиал). Имеет волновое сопротивление 50 Ом. Максимальная длина сегмента — 500 метров (без повторителей).
- **10Base-2** — коаксиальный кабель диаметром 0.25 дюйма («тонкий» коаксиал). Имеет волновое сопротивление 50 Ом. Максимальная длина сегмента — 185 метров (без повторителей).
- **10Base-T** — кабель на основе неэкранированной витой пары (Unshielded Twisted Pair, UTP). Образует звездообразную топологию с концентратором. Расстояние между концентратором и конечным узлом — $\leq$ 100 м.
- **10Base-F** — оптоволоконный кабель. Топология аналогична стандарту на витой паре. Имеется несколько вариантов этой спецификации — FOIRL, 10Base-FL, 10Base-FB.

Число 10 обозначает битовую скорость передачи данных этих стандартов — 10 Мб/с, а слово Base — метод передачи на одной базовой частоте 10 МГц.

Стандарт 10Base-5

Стандарт 10Base-5 соответствует экспериментальной сети Ethernet фирмы Xerox и может считаться классическим Ethernet. В качестве среды ПД применяется коаксиальный кабель с диаметром центрального медного провода 2,17 мм и внешним диаметром ≈ 10 мм, который используется как моноканал для всех станций. Сегмент кабеля имеет максимальную длину 500 м (без повторителей) и должен иметь на концах согласующие терминаторы (резистор, согласованная нагрузка) сопротивлением 50 Ом, поглощающие распространяющиеся по кабелю сигналы и препятствующие возникновению отраженных сигналов.
Станция подключается к кабелю при помощи приемопередатчика — трансивера, который устанавливается непосредственно на кабеле и питается от сетевого адаптера компьютера. Трансивер может подсоединяться к кабелю как методом прокалывания, обеспечивающим непосредственный физический контакт, так и бесконтактным методом.
Трансивер соединяется с сетевым адаптером интерфейсным кабелем AUI (Attachment Unit Interface) длиной до 50 м, состоящим из 4 витых пар (адаптер должен иметь разъем AUI). Допускается подключение к одному сегменту не более 100 трансиверов, причем расстояние между подключениями трансиверов не должно быть меньше 2.5 м.





Стандарт 10Base-5 (2)

Функции трансивера

- прием и передача данных с кабеля на кабель,
- определение коллизий на кабеле,
- электрическая развязка между кабелем и остальной частью адаптера,
- защита кабеля от некорректной работы адаптера.

Последнюю функцию часто называют контролем болтливости (jabber control). При возникновении неисправностей в адаптере может возникнуть ситуация, когда на кабель будет непрерывно выдаваться последовательность случайных сигналов. Так как кабель — это общая среда для всех станций, то работа сети будет заблокирована одним неисправным адаптером. Чтобы этого не случилось, на выходе передатчика ставится схема, которая проверяет количество битов, переданных в пакете. Если максимальная длина пакета превышается, то эта схема отсоединяет выход передатчика от кабеля.
Детектор коллизий определяет наличие коллизии в коаксиальном кабеле по повышенному уровню постоянной составляющей сигналов. Если постоянная составляющая превышает определенный порог, то значит на кабель работает более чем один передатчик.

Достоинства стандарта 10Base-5

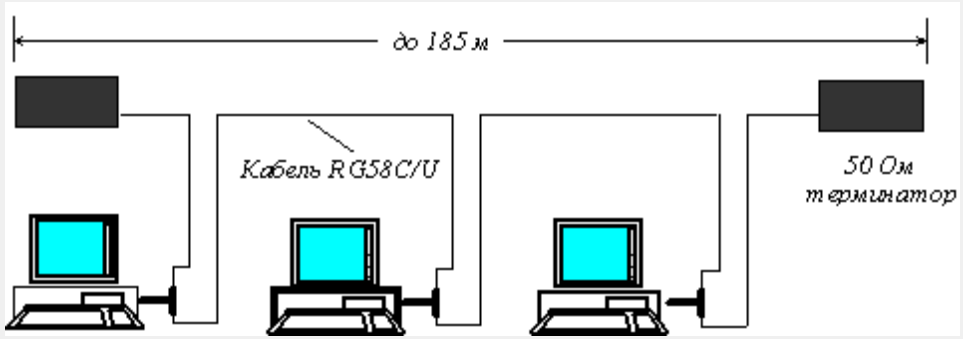
- хорошая защищенность кабеля от внешних воздействий,
- сравнительно большое расстояние между узлами,
- возможность простого перемещения рабочей станции в пределах длины кабеля AUI.

Недостатки стандарта 10Base-5

- высокая стоимость кабеля,
- сложность его прокладки из-за большой жесткости,
- наличие специального инструмента для заделки кабеля,
- при повреждении кабеля или плохом соединении происходит остановка работы всей сети,
- необходимо заранее предусмотреть подводку кабеля ко всем возможным местам установки компьютеров.

Стандарт 10Base-2

Стандарт 10Base-2 использует в качестве передающей среды коаксиальный кабель с диаметром центрального медного провода 0,89 мм и внешним диаметром ≈ 5 мм. Максимальная длина сегмента без повторителей — 185 м. Сегмент должен иметь на концах согласующие терминаторы 50 Ом.
Станции подключаются к кабелю с помощью Т-коннектора — тройника, один отвод которого соединяется с сетевым адаптером, а два других — с двумя концами разрыва кабеля. Максимальное количество станций, подключаемых к одному сегменту, 30. Минимальное расстояние между станциями — 1 м.
Трансиверы в нем объединены с сетевыми адаптерами за счет того, что более гибкий тонкий коаксиальный кабель может быть подведен непосредственно к выходному разъему платы сетевого адаптера, установленной в шасси компьютера. Кабель в данном случае «висит» на сетевом адаптере, что затрудняет физическое перемещение компьютеров.



Реализация этого стандарта на практике приводит к простому решению для кабельной сети, так как для соединения компьютеров требуются только сетевые адаптеры и Т-коннекторы. Однако этот вид кабельных соединений наиболее сильно подвержен авариям и сбоям: кабель восприимчив к помехам, в моноканале имеется большое количество механических соединений — по три на один коннектор, пользователи имеют доступ к разъемам и могут нарушить целостность моноканала. Для каждой станции требуется запас кабеля, необходимый на случай даже небольшого перемещения компьютера.
Общим недостатком стандартов 10Base-5 и 10Base-2 является отсутствие оперативной информации о состоянии моноканала. Повреждение кабеля обнаруживается сразу же (сеть перестает работать), но для поиска отказавшего отрезка кабеля необходим специальный прибор — кабельный тестер.

Стандарт 10Base-F

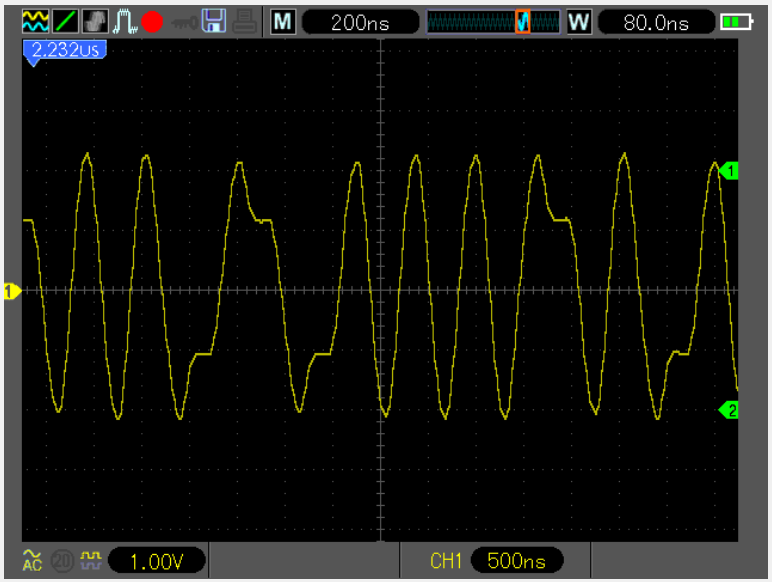
Стандарт 10Base-F использует в качестве среды передачи данных оптоволокно. Функционально сеть стандарта 10Base-F состоит из тех же элементов, что и сеть стандарта 10Base-T - сетевых адаптеров, многопортового повторителя и отрезков кабеля, соединяющих адаптер с портом повторителя. Как и при использовании витой пары, для соединения адаптера с повторителем используется два оптоволокна — одно соединяет выход Tx адаптера со входом Rx повторителя, а другое — вход Rx адаптера с выходом Tx повторителя.

- **Стандарт FOIRL (Fiber Optic Inter-Repeater Link)** — это первый стандарт комитета 802.3 для использования оптоволокна в сетях Ethernet. Он гарантирует длину оптоволоконной связи между повторителями до 1 км при общей длине сети не более 2500 м. Максимальное число повторителей — 4.
- **Стандарт 10Base-FL** предназначен для соединения конечных узлов с концентратором и работает с сегментами оптоволокна длиной не более 2000 м при общей длине сети не более 2500 м. Максимальное число повторителей — 4.
- **Стандарт 10Base-FB** предназначен для магистрального соединения повторителей. Он позволяет иметь в сети до 5 повторителей при максимальной длине одного сегмента 2000 м и максимальной длине сети 2740 м. Повторители, соединенные по стандарту 10Base-FB постоянно обмениваются специальными последовательностями сигналов, отличающимися от сигналов кадров данных, для обнаружения отказов своих портов. Поэтому, концентраторы стандарта 10Base-FB могут поддерживать резервные связи, переходя на резервный порт при обнаружении отказа основного с помощью тестовых специальных сигналов. Концентраторы этого стандарта передают как данные, так и сигналы простоя линии синхронно, поэтому биты синхронизации кадра не нужны и не передаются. Стандарт 10Base-FB поэтому называют также синхронный Ethernet.

Стандарты 10Base-FL и 10Base-FB не совместимы между собой.

Стандарт 10Base-T

Стандарт принят в 1991 году как дополнение к существующему набору стандартов Ethernet. Имеет обозначение 802.3i.
Использует в качестве среды двойную неэкранированную витую пару (Unshielded Twisted Pair, UTP). Соединения станций осуществляются по топологии «точка–точка» со специальным устройством — многопортовым повторителем (концентратор, hub) с помощью двух витых пар. Одна витая пара используется для передачи данных от станции к повторителю (выход Tx сетевого адаптера), а другая — для передачи данных от повторителя станции (вход Rx сетевого адаптера). Концентратор осуществляет функции повторителя сигналов на всех отрезках витых пар, подключенных к его портам, так что образуется единая среда передачи данных — моноканал (шина).
Повторитель обнаруживает коллизию в сегменте в случае одновременной передачи сигналов по нескольким своим Rx входам и посылает jam-последовательность на все свои Tx выходы. Стандарт определяет битовую скорость передачи данных 10 Мб/с и максимальное расстояние отрезка витой пары между двумя непосредственно связанными узлами (станциями и концентраторами) не более 100 м при использовании витой пары не ниже cat.3.
Возможно иерархическое соединение концентраторов в дерево. Для обеспечения синхронизации станций при реализации процедур доступа CSMA/CD и надежного распознавания станциями коллизий в стандарте определено максимально число концентраторов между любыми двумя станциями сети.
Общее количество станций в сети 10Base-T не должно превышать 1024.



10BASE-T использует разъёмы типа 8P8C, обжатые согласно таблицам T568A или T568B, определённым в стандарте TIA/EIA-568-B. Используются только вторая и третья пара (оранжевая и зелёная). Соответственно, для построения сети Ethernet 10Base-T можно использовать как 4-парные, так и 2-парные кабели UTP. Используется фазовая модуляция сигнала. На один бит отводится один период. Смена фазы колебаний означает смену логического состояния от 0 к 1 или наоборот. В дальнейшем эти состояния декодируются, как Манчестерский код. Частота несущей 10 МГц — 100 наносекунд на период. Амплитуда сигнала около 2 вольт. Ток — переменный, сама передача всегда ведётся по гальванически развязанной от устройств линии. На входе и выходе у каждого сетевого устройства, работающего по стандартам 10Base-T установлен высокочастотный развязывающий трансформатор с коэффициентом трансформации 1:1.

Манчестерское кодирование

Такой вид кодирования также называют фазовым кодированием. Каждый такт делится на две части. Информация кодируется перепадами потенциала в середине каждого такта. «1» кодируется перепадом от низкого уровня сигнала к высокому, а «0» — обратным перепадом (по стандарту IEEE 802.3).

В начале каждого такта может происходить служебный перепад сигнала, если нужно представить несколько единиц или нулей подряд. Так как сигнал изменяется по крайней мере один раз за такт передачи одного бита данных, то манчестерский код обладает хорошими самосинхронизирующими свойствами. У манчестерского кода нет постоянной составляющей (меняется каждый такт), а основная гармоника в худшем случае (при передаче последовательности единиц или нулей) имеет частоту N Гц, а в лучшем случае (при передаче чередующихся единиц и нулей) — N/2 Гц.

Методика расчета конфигурации сети Ethernet

Условия корректной работы сети Ethernet из сегментов различного типа

1. Количество станций в сети не превышает 1024 (с учетом ограничений для коаксиальных сегментов).
2. Удвоенная задержка распространения сигнала (Path Delay Value, PDV) между двумя самыми удаленными друг от друга станциями сети не превышает 575 bt.
3. Сокращение межкадрового расстояния (Interframe Gap Shrinkage или Path Variability Value, PVV) при прохождении последовательности кадров через все повторители не более, чем на 49 bt (при отправке кадров станция обеспечивает начальный IFG в 96 bt).

Требование на минимальное межкадровое расстояние связано с тем, что при прохождении кадра через повторитель это расстояние уменьшается. Каждый пакет, принимаемый повторителем, ресинхронизируется для исключения дрожания сигналов, накопленного при прохождении последовательности импульсов по кабелю и через интерфейсные схемы. Процесс ресинхронизации обычно увеличивает длину преамбулы, что уменьшает межкадровый интервал. При прохождении кадров через несколько повторителей межкадровый интервал может уменьшиться настолько, что сетевым адаптерам в последнем сегменте не хватит времени на обработку предыдущего кадра, в результате чего кадр будет просто потерян. Поэтому не допускается суммарное уменьшение межкадрового интервала более чем на 49 bt. Величину уменьшения межкадрового расстояния при переходе между соседними сегментами обычно называют в англоязычной литературе Segment Variability Value, SVV, а суммарную величину уменьшения межкадрового интервала при прохождении всех повторителей — Path Variability Value, PVV.

На практике для расчета PDV и PVV используют усредненные табличные значения для повторителей и кабелей разных типов. При этом PDV зависит как от типа среды ПД и повторителей, так и от длины кабелей. PVV зависит только от типа среды ПД в сегментах сети.

Кабель связи «витая пара»

Витая пара (twisted pair) представляет собой одну или несколько пар изолированных проводников, скрученных между собой, покрытых пластиковой оболочкой. Повив проводников производится с целью исключить влияние на пару, по которой идёт сигнал, синфазных (электромагнитные помехи одинаково влияют на оба провода пары) помех и обеспечить их вычитание за счёт применения выходных и входных дифференциальных каскадов. Для снижения связи отдельных пар кабеля (периодического сближения проводников различных пар) в кабелях UTP категории 5 и выше провода пары свиваются с различным шагом.

Экранирование

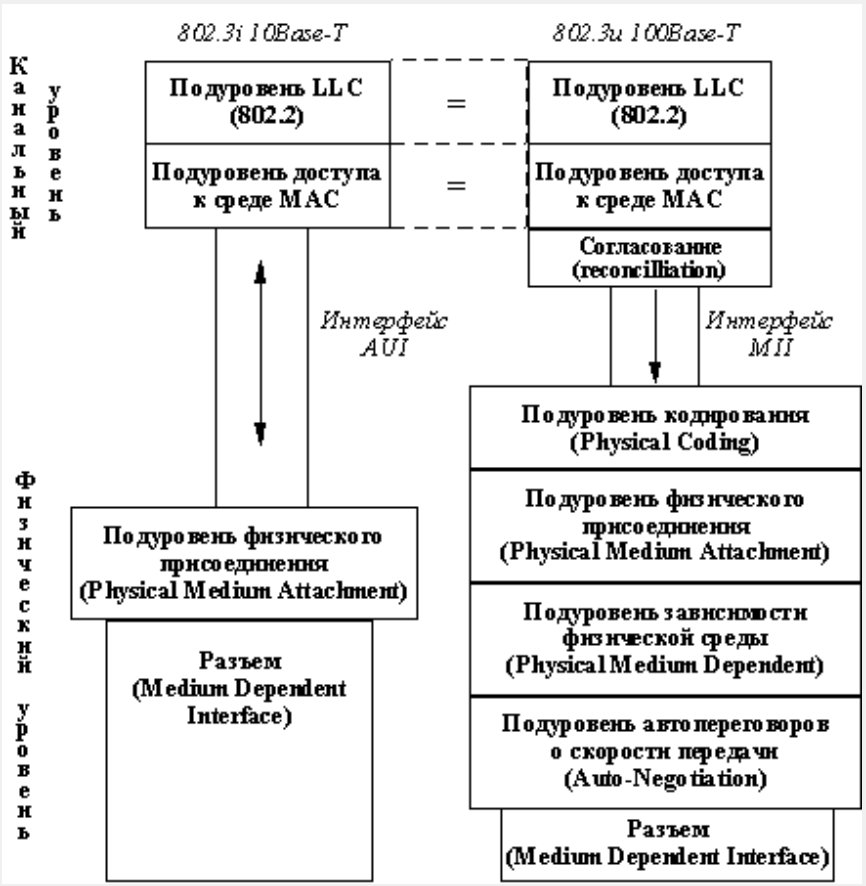
Для защиты от электрических помех при использовании высокочастотных сигналов, в кабелях cat.6a–8 используется экранирование, которое применяется как к отдельным витым парам, которые оборачиваются в алюминиевую фольгу (металлизированную алюминием полиэтиленовую ленту), так и к кабелю в целом в виде общего экрана из фольги и/или оплётки из медной проволоки. Экран всегда соединён с неизолированным общим проводом, который соединяет участки экрана в случае его разделения на секции.
Согласно международному стандарту ISO/IEC 11801, для обозначения конструкции экранированного кабеля используется комбинация из трех букв: U — неэкранированный, S — металлическая оплётка (только общий экран), F — металлизированная лента (алюминиевая фольга). Из этих букв формируется аббревиатура вида xx/xTP, обозначающая тип общего экрана и тип экрана для отдельных пар.
Типы конструкции экрана:

- **Неэкранированный кабель (U/UTP).** Экранирование отсутствует. Cat.6 и ниже.
- **Индивидуальный экран (U/FTP).** Экранирование фольгой каждых отдельных пар. Защищает от внешних помех и от перекрёстных помех между витыми парами.
- **Общий экран (F/UTP, S/UTP, SF/UTP).** Общий экран из фольги, оплётки, или фольги с оплёткой. Защищает от внешних электромагнитных помех.
- **Индивидуальный и общий экран (F/FTP, S/FTP, SF/FTP).** Индивидуальные экраны из фольги для каждой витой пары, плюс общий экран из фольги, оплётки, или фольги с оплёткой. Защищает от внешних помех и от перекрёстных помех между витыми парами.

При использовании экранированных кабелей обязательно необходимо использовать специальные разъемы 8P8C с экраном. При использовании обычных неэкранированных разъемов эффект экранирования отсутствует.

Технология Fast Ethernet

К началу 90-х годов XX века в связи с ростом быстродействия компьютерной техники пропускной способности технологии 10 Мбит/с Ethernet стало не хватать. В сетях чаще начали возникать перегрузки, вызывающие появление коллизий и падение пропускной способности. Рассмотрев ряд решений, предложенных различными производителями, комитет IEEE принял в мае 1995 года спецификацию Fast Ethernet в качестве стандарта 802.3u — дополнения к существующему стандарту 802.3. Отличия Fast Ethernet от Ethernet сосредоточены на физическом уровне.

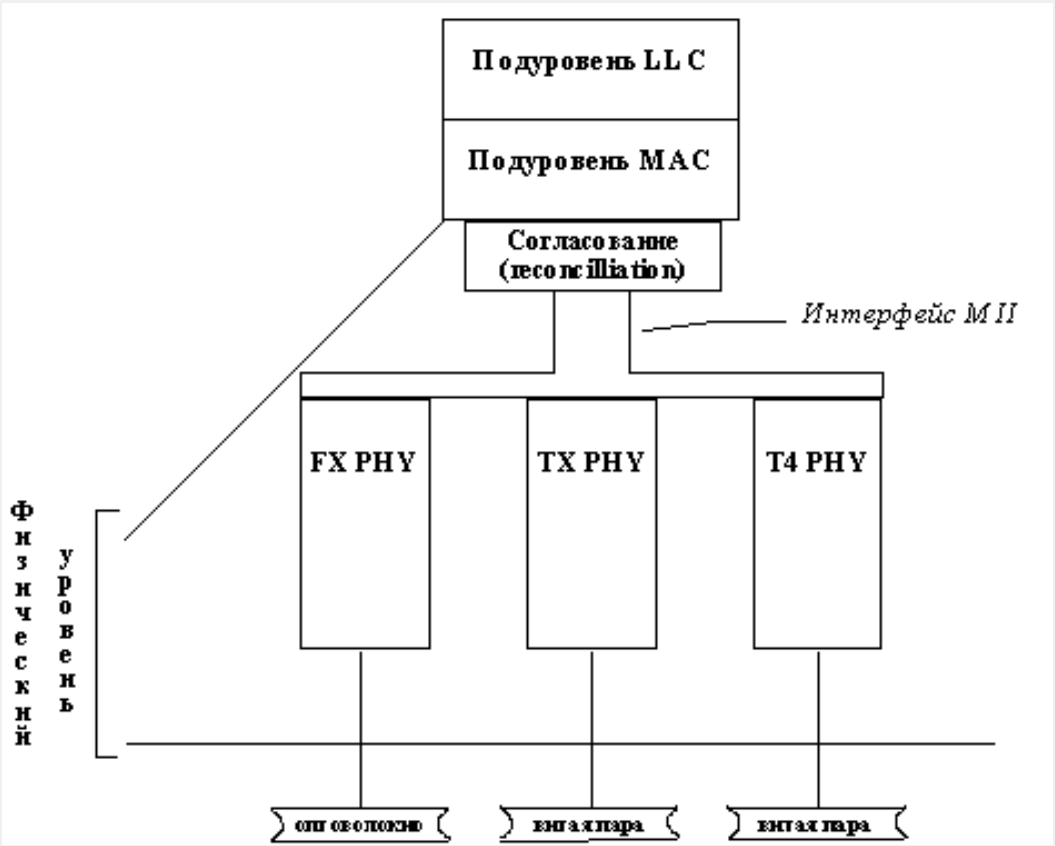


Формат кадра не изменился. В связи с десятикратным увеличением скорости все времена передачи кадров Fast Ethernet в 10 раз меньше соответствующих времен технологии 10 Мбит/с Ethernet: межбитовый интервал составляет 10 нс вместо 100 нс, а межкадровый интервал — 0.96 мкс вместо 9.6 мкс соответственно. На практике временные параметры еще меньше из-за того, что используется несущая не 100 МГц, а 125 МГц.

Физический уровень Fast Ethernet

Подуровни

- Уровень согласования (reconciliation sublayer).
- Независимый от среды интерфейс (Media Independent Interface, MII).
- Устройство физического уровня (Physical layer device, PHY).



Устройство физического уровня (PHY) обеспечивает кодирование данных, поступающих от MAC-подуровня для передачи их по кабелю определенного типа, синхронизацию передаваемых по кабелю данных, а также прием и декодирование данных в узле-приемнике.

Интерфейс MII поддерживает независимый от используемой физической среды способ обмена данными между MAC-подуровнем и подуровнем PHY. Этот интерфейс аналогичен по назначению интерфейсу AUI классического Ethernet за исключением того, что интерфейс AUI располагался между подуровнем физического кодирования сигнала (для любых вариантов кабеля использовался одинаковый метод физического кодирования — манчестерский код) и подуровнем физического присоединения к среде, а интерфейс MII располагается между MAC-подуровнем и подуровнями кодирования сигнала.

Подуровень согласования нужен для того, чтобы согласовать работу подуровня MAC с интерфейсом MII.

Физический уровень Fast Ethernet (2)

Спецификации физического уровня

- **100Base-TX** — задействована витая пара UTP или S/UTP cat.5 (две пары), поддерживается дуплексная передача данных, расстояние до 100 м.
- **100Base-T**4 — витая пара UTP cat.3 или выше (четыре пары), передача данных идёт в полудуплексе. Практически не используется.
- **100Base-T2** — витая пара UTP cat.3 (две пары). Поддерживается полный дуплекс, когда сигналы распространяются в противоположных направлениях по каждой паре. Скорость передачи в одном направлении — 50 Мбит/с. Практически не используется.
- **100Base-FX** — многомодовое волокно. Максимальная длина сегмента 400 м в полудуплексе (для гарантированного обнаружения коллизий) или 2 км в полном дуплексе.
- **100Base-SX** — многомодовое волокно. Максимальная длина ограничена только величиной затухания в оптическом кабеле и мощностью передатчиков, по разным материалам от 2 до 10 км.
- **100Base-FX WDM** — одномодовое волокно. Максимальная длина ограничена только величиной затухания в ВОЛС и мощностью передатчиков. Интерфейсы бывают двух видов, отличаются длиной волны передатчика и маркируются либо цифрами (длина волны), либо одной латинской буквой A(1310) или B(1550).
- **100Base-LX** — обеспечивает передачу данных со скоростью до 100 Мбит/с через оптический кабель по одному одномодовому волокну на длине волны 1310 нм. Максимальная длина сегмента — 15 км в режиме полного дуплекса. Существует вариант 100Base-LX10 с максимальной длиной сегмента 10 км.

Физический уровень 100Base-FX

Спецификация определяет работу протокола Fast Ethernet по многомодовому оптоволокну в полудуплексном и полнодуплексном режимах на основе схемы кодирования и передачи оптических сигналов, использующейся в стандарте FDDI. Как и в стандарте FDDI, каждый узел соединяется с сетью двумя оптическими волокнами, идущими от приемника (Rx) и от передатчика (Tx).

Метод кодирования 4B/5B

При этом методе каждые 4 бита данных MAC-подуровня (называемых символами) представляются 5 битами. Использование избыточного бита позволяет применить потенциальные коды при представлении каждого из пяти бит в виде электрических или оптических импульсов. Потенциальные коды обладают по сравнению с манчестерскими кодами более узкой полосой спектра сигнала, а, следовательно, предъявляют меньшие требования к полосе пропускания кабеля. Однако, прямое использование потенциальных кодов для передачи исходных данных без избыточного бита невозможно из-за плохой самосинхронизации приемника и источника данных: при передаче длинной последовательности единиц или нулей в течение долгого времени сигнал не изменяется и приемник не может определить момент чтения очередного бита.

При использовании пяти бит для кодирования шестнадцати исходных 4-битовых комбинаций, можно построить такую таблицу кодирования, в которой любой исходный 4-битовый код представляется 5-битовым кодом с чередующимися нулями и единицами. Тем самым обеспечивается синхронизация приемника с передатчиком. Так как исходные биты MAC-подуровня должны передаваться со скоростью 100 Мбит/с, то наличие одного избыточного бита вынуждает передавать биты результирующего кода 4B/5B со скоростью 125 Мбит/с, то есть межбитовое расстояние в устройстве PHY составляет 8 нс.

Так как из 32 возможных комбинаций 5-битовых порций для кодирования порций исходных данных нужно только 16, то остальные 16 комбинаций в коде 4B/5B используются в служебных целях или являются запрещенными. Существование запрещенных комбинаций символов позволяет отбраковывать ошибочные символы, что повышает устойчивость работы сетей с PHY FX/TX.

Наличие служебных символов позволило использовать в спецификациях FX/TX схему непрерывного обмена сигналами между передатчиком и приемником и при свободном состоянии среды, что отличает их от спецификации 10Base-T, когда незанятое состояние среды обозначается полным отсутствием на ней импульсов информации. Для обозначения незанятого состояния среды используется служебный символ Idle (11111), который постоянно циркулирует между передатчиком и приемником, поддерживая их синхронизм и в периодах между передачами информации, а также позволяя контролировать физическое состояние линии.

Физический уровень 100Base-FX (2)

Коды 4B/5B

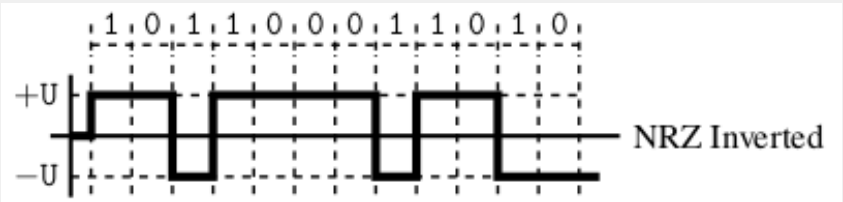
Коды Данных

4B Код	5B Символ
0000	11110
0001	01001
0010	10100
0011	10101
0100	01010
0101	01011
0110	01110
0111	01111
1000	10010
1001	10011
1010	10110
1011	10111
1100	11010
1101	11011
1110	11100
1111	11101

Управляющие и Недопустимые Коды

4B Код	5B Символ
Ожидание	11111
Начало потока	11000
Начало потока	10001
Конец потока	01101
Конец потока	00111
Ошибка передачи	00100
Недопустимый	00000
Недопустимый	00001
Недопустимый	00010
Недопустимый	00011
Недопустимый	00100
Недопустимый	00101
Недопустимый	00110
Недопустимый	01000
Недопустимый	10000
Недопустимый	11001

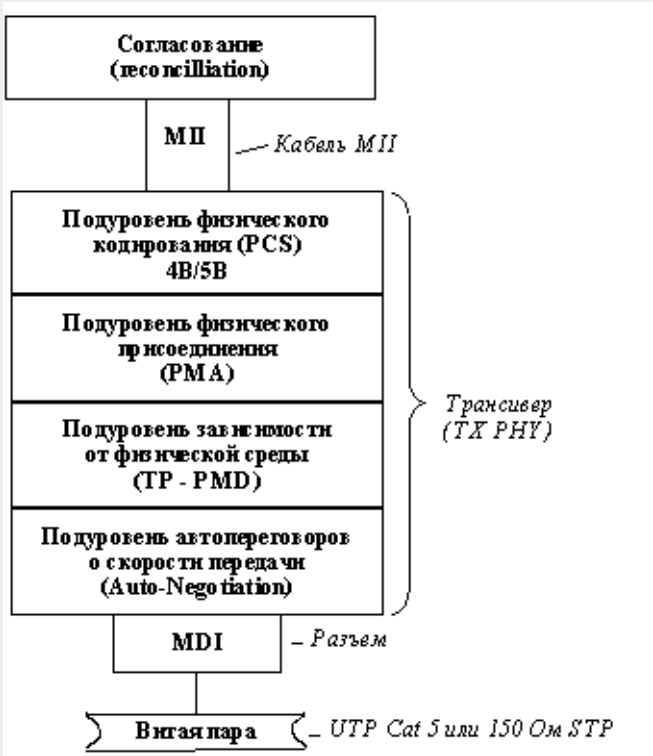
Код NRZI



После преобразования 4-битовых порций MAC-кодов в 5-битовые порции PHY их необходимо представить в виде оптических или электрических сигналов в кабеле, соединяющем узлы сети. В спецификации PHY FX используется метод NRZI — Non Return to Zero Invert to ones — метод без возврата к нулю с инвертированием для единиц. Этот метод представляет собой модификацию простого потенциального метода кодирования, называемого Non Return to Zero (NRZ), когда для представления 1 и 0 используются потенциалы двух уровней. В методе NRZI также используется два уровня потенциала сигнала, но потенциал, используемый для кодирования текущего бита зависит от потенциала, который использовался для кодирования предыдущего бита (так называемое, дифференциальное кодирование). Если текущий бит имеет значение 1, то текущий потенциал представляет собой инверсию потенциала предыдущего бита, независимо от его значения. Если же текущий бит имеет значение 0, то текущий потенциал повторяет предыдущий.

Из описания метода NRZI видно, что для обеспечения частых изменений сигнала, а значит и для поддержания самосинхронизации приемника, нужно исключить из кодов слишком длинные последовательности нулей. Коды 4B/5B построены так, что гарантируют не более трех нулей подряд при любом сочетании бит в исходной информации.

Физический уровень 100Base-TX



Основные отличия PHY TX от спецификации PHY FX — использование метода MLT-3 для передачи сигналов 5-битовых кодов 4B/5B по витой паре, а также наличие функции автопереговоров (Auto-negotiation) для выбора режима работы порта.

Также в спецификации PHY TX используется пара скремблер/дескремблер (scrambler/descrambler). Скремблер принимает 5-битовые блоки данных от подуровня PCS (кодирование 4B/5B) и зашифровывает сигналы перед передачей на подуровень MLT-3 для равномерного распределения энергии сигнала по всему частотному спектру — это уменьшает электромагнитное излучение кабеля.

Автопереговорный процесс (Auto-negotiation)

Функция согласно спецификациям PHY TX и PHY T4, с помощью которой два взаимодействующих устройства PHY могут автоматически выбрать наиболее эффективный режим работы. Является стандартом технологии 100Base-T.

5 различных режимов работы устройств PHY TX или PHY T4 на витых парах:

1. 10Base-T — 2 пары cat.3 (самый низкий приоритет);
2. 10Base-T full-duplex — 2 пары cat.3;
3. 100Base-TX — 2 пары cat.5 (или Type 1A STP);
4. 100Base-TX full-duplex — 2 пары cat.5 (или Type 1A STP);
5. 100Base-T4 — 4 пары cat.3 (самый высокий приоритет).

Переговорный процесс происходит при включении питания устройства, а также может быть инициирован и в любой момент модулем управления. Для организации переговорного процесса используются служебные сигналы проверки целостности линии технологии 10Base-T — link test pulses, если узел-партнер поддерживает только стандарт 10Base-T. Узлы, поддерживающие функцию Auto-negotiation, также используют существующую технологию сигналов проверки целостности линии, посылая пачки (Fast Link Pulse burst, FLP) таких импульсов, инкапсулирующие информацию переговорного процесса Auto-negotiation.

Устройство, начавшее процесс auto-negotiation, посылает своему партнеру пачку импульсов FLP, в котором содержится 8-битный код самого приоритетного режима, поддерживаемого данным узлом. Если узел-партнер поддерживает функцию Auto-negotuiation и также может поддерживать предложенный режим, то он отвечает пачкой импульсов FLP, в которой подтверждает данный режим и на этом переговоры заканчиваются. Если же узел-партнер может поддерживать менее приоритетный режим, то он указывает его в ответе и этот режим выбирается в качестве рабочего. Таким образом, всегда выбирается наиболее приоритетный общий режим узлов.

Узел, который поддерживает только технологию 10Base-T, каждые 16 миллисекунд посылает импульсы для проверки целостности линии, связывающей его с соседним узлом. Такой узел не понимает запрос FLP, который делает ему узел с функцией Auto-negotiation, и продолжает посылать свои импульсы. Такой ответ на запрос FLP является признаком необходимости использования стандарта 10Base-T.

Физический уровень 100Base-TX (2)

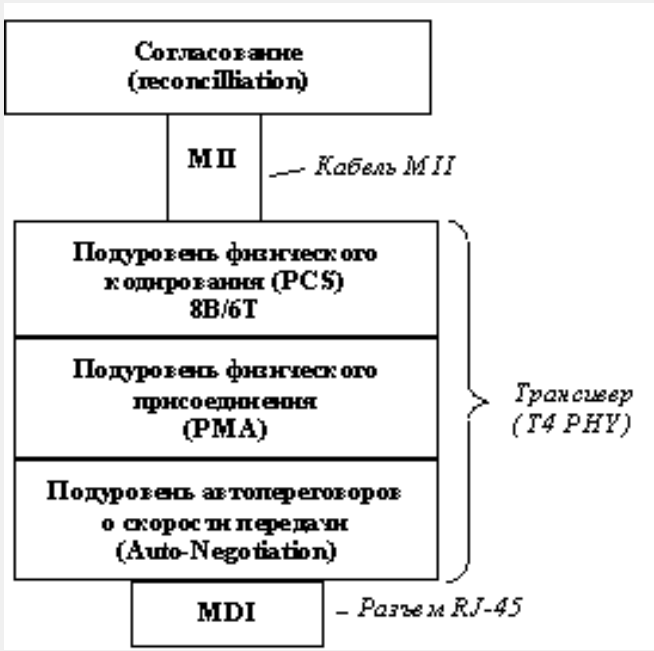
Кодирование MLT-3 (Multi Level Transmission 3)

Код MLT-3 схож с кодом NRZ-I, но в отличие от него имеет три уровня сигнала. В MLT-3 циклично перебираются уровни напряжений $-U$, 0, $+U$. Смена уровня соответствует передаче сигнала 1 бит, при передаче 0 бит уровень не изменяется. Как и в случае NRZ-I, код MLT-3 имеет кодовую эффективность, равную 1 бит/бод, при этом для возврата в предыдущее состояние требуется четыре перехода (бода): $-U \to 0$, $0 \to +U$, $+U \to 0$, $0 \to -U$.
В связи с этим реальная частота уменьшается до четверти бода. Передача таким сигналом больше подходит для медных линий. Впервые код MLT-3 был предложен компанией Crescendo Communications для использования в технологии CDDI (FDDI по медному кабелю).

Полнодуплексный режим работы 100Base-TX

Узлы, поддерживающие спецификации PHY FX и PHY TX, могут работать в полнодуплексном режиме (full-duplex mode). В этом режиме не используется метод доступа к среде CSMA/CD и отсутствует понятие коллизий — каждый узел одновременно передает и принимает кадры данных по каналам Tx и Rx.
Полнодуплексная работа возможна только при соединения сетевого адаптера с коммутатором или же при непосредственном соединении коммутаторов.
При полнодуплексной работе стандарты 100Base-TX и 100Base-FX обеспечивают скорость обмена данными между узлами 200 Мбит/с.
В полнодуплексном режиме необходимо определить процедуры управления потоком кадров, так как без этого механизма возможны ситуации, когда буферы коммутатора переполнятся и он начнет терять кадры Ethernet, что всегда крайне нежелательно, так как восстановление информации будет осуществляться более медленными протоколами транспортного или прикладного уровней.
Ввиду отсутствия стандартов на полнодуплексные варианты Fast Ethernet каждый производитель сам определяет способы управления потоком кадров в коммутаторах и сетевых адаптерах. Обычно, при заполнении буфера устройства до определенного предела, это устройство посылает передающему устройству сообщение о временном прекращении передачи (XOFF). При освобождении буфера посылается сообщение о возможности возобновить передачу (XON).

Физический уровень 100Base-T4

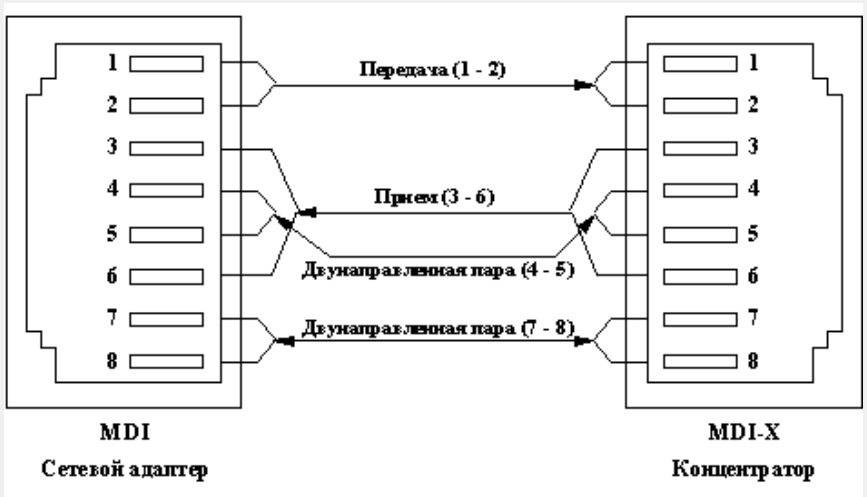


Спецификация PHY T4 была разработана для того, чтобы можно было использовать для Fast Ethernet имеющуюся проводку на витой паре cat.3. Эта спецификация использует все 4 пары кабеля для того, чтобы можно было повысить общую пропускную способность за счет одновременной передачи потоков бит по нескольким витым парам.

Вместо кодирования 4B/5B в этом методе используется кодирование 8B/6T. Каждые 8 бит информации MAC-уровня кодируются 6 троичными цифрами (ternary symbols). Избыточность кода 8B/6T выше, чем кода 4B/5B, так как на $2^8 = 256$ исходных комбинаций приходится $3^6 = 729$ результирующих комбинаций. Каждая троичная цифра имеет длительность 40 нс. Группа из 6 троичных цифр затем передается на одну из трех передающих витых пар, независимо и последовательно. Четвертая пара всегда используется для прослушивания несущей частоты в целях обнаружения коллизии. Скорость передачи данных по каждой из трех передающих пар равна 33.3 Мбит/с, поэтому общая скорость протокола 100Base-T4 составляет 100 Мбит/с. В то же время из-за принятого способа кодирования скорость изменения сигнала на каждой паре равна всего 25 Мбод, что и позволяет использовать витую пару cat.3.

В 100Base-T4 используется трёхуровневая амплитудно-импульсная модуляция (PAM-3).

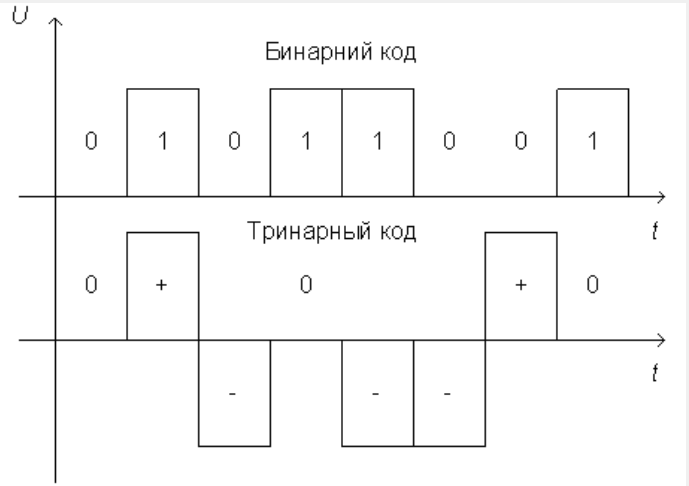
На рисунке показано соединение порта MDI сетевого адаптера 100Base-T4 с портом MDI-X повторителя. Пара 1-2 всегда используется для передачи данных от порта MDI к порту MDI-X, пара 3-6 всегда используется для приема данных портом MDI от порта MDI-X, а пары 4-5 и 7-8 являются двунаправленными и используются и для приема, и для передачи, в зависимости от потребности. Однонаправленные пары 1-2 и 3-6 используются для обнаружения коллизий при передаче данных.



Физический уровень 100Base-T4 (2)

Алгоритм кодирования 8B6T

Data octet	6T code group	Data octet	6T code group	Data octet	6T code group	Data octet	6T code group
00	+-00+-	10	+0+--0	20	00-++-	30	+-00-+
01	0+-+-0	11	++0-0-	21	--+00+	31	0+--+0
02	+-0+-0	12	+0+-0-	22	++-0+-	32	+-0-+0
03	-0++-0	13	0++-0-	23	++-0-+	33	-0+-+0
04	-0+0+-	14	0++--0	24	00+0-+	34	-0+0-+
05	0+--0+	15	++00--	25	00+0+-	35	0+-+0-
06	+-0-0+	16	+0+0--	26	00-00+	36	+-0+0-
07	-0+-0+	17	0++0--	27	--+++-	37	-0++0-
08	-+00+-	18	0+-0+-	28	-0-++0	38	-+00-+
09	0-++-0	19	0+-0-+	29	--0+0+	39	0-+-+0
0A	-+0+-0	1A	0+-++-	2A	-0-+0+	3A	-+0-+0
0B	+0-+-0	1B	0+-00+	2B	0--+0+	3B	+0--+0
0C	+0-0+-	1C	0-+00+	2C	0--++0	3C	+0-0-+
0D	0-+-0+	1D	0-+++-	2D	--00++	3D	0-++0-
0E	-+0-0+	1E	0-+0-+	2E	-0-0++	3E	-+0+0-
0F	+0--0+	1F	0-+0+-	2F	0--0++	3F	+0-+0-



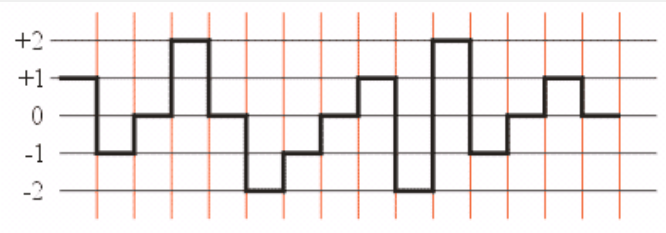
Технология Gigabit Ethernet

Технологии передачи Ethernet-кадров со скоростью 1 Гбит/с впервые появились в 1998, когда был опубликован стандарт IEEE 802.3z, определяющий передачу по оптоволоконному кабелю — т. н. технология 1000Base-X, где X определяет тип технологии. Вскоре, в 1999 появился стандарт IEEE 802.3ab, определивший стандарт гигабитной передачи данных по неэкранированной витой паре (UTP) cat.5, 5e и 6, и известный как 1000Base-T. Этот стандарт позволил перейти на более высокие скорости при сохранении возможности использования существующей кабельной инфраструктуры. В 2004 был ратифицирован стандарт IEEE 802.3ah, добавивший еще два стандарта передачи по оптоволокну — 1000Base-LX10 и 1000Base-BX10.
Как и в случае Fast Ethernet, основные отличия G.Ethernet лежат на физическом уровне. Временные характеристики также уменьшились пропорционально увеличению скорости.
По стандарту 1Gbit Ethernet поддерживает как полудуплексную, так и полнодуплексную передачу. На практике полудуплексная передача не используется.

Основные стандарты

- **1000Base-CX** — Экранированный сбалансированный 2-парный медный кабель (витая пара с коннектором DE-9 или 8P8C) — до 25 м. Не совместима с 1000Base-T/TX. Используется в платформе IBM BladeCenter для связи блэйд-серверов с коммутатором.
- **1000Base-SX** — Многомодовое оптоволокно — до 220–550 м.
- **1000Base-LX** — Многомодовое (до 550 м) или одномодовое (до 5 км) оптоволокно.
- **1000Base-LX10** — Одномодовое оптоволокно (1310 нм) — до 10 км.
- **1000Base-BX10** — Одномодовое оптоволокно (WDM: 1490 нм на приём, 1310 нм на передачу) — 10 км.
- **1000Base-T** — Витая пара (cat.5, 5e, 6, 7) — до 100 м.
- **1000Base-TX** — Витая пара (cat.6, 7) — до 100 м.

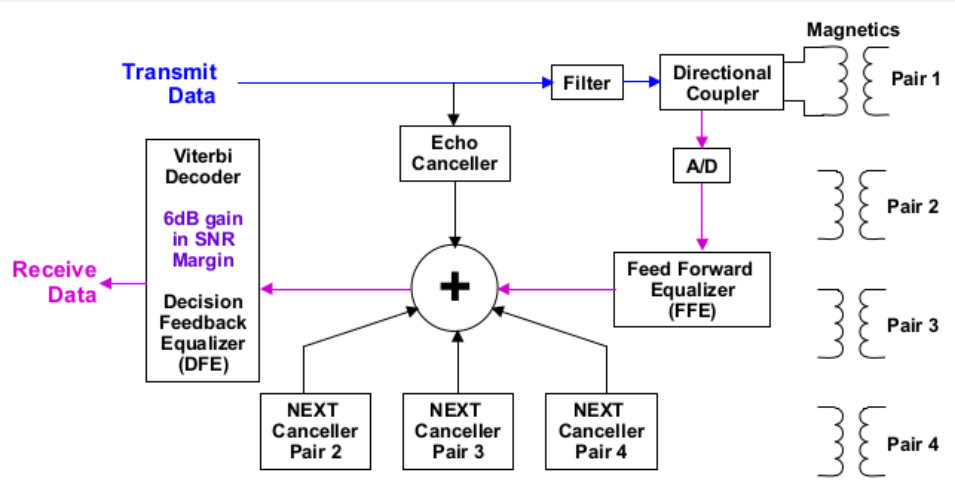
Технология 1000Base-T



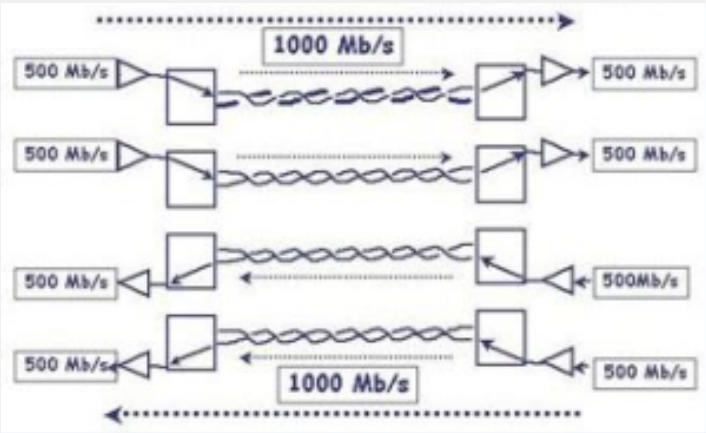
Основной гигабитный стандарт, который использует витую пару cat.5e и лучше. В передаче данных участвуют все 4 пары. Расстояние — до 100 метров.
Передача производится дибитами. Данные в линии представляются в так называемом коде PAM-5. В нем передаваемый сигнал имеет набор из пяти фиксированных уровней -2, -1, 0, $+1$, $+2$. Четыре из них используются для кодирования информационных битов, а пятый предназначен для коррекции ошибок.

Поскольку кабель cat.5 рассчитан на частоту 125 МГц (т. е. способен работать с бодовой скоростью 125 МБод), то информационная скорость по одной витой паре составит 250 Мбит/с, что при задействовании всех четырех пар обеспечивает искомые 1000 Мбит/с.
Пятый избыточный уровень кода PAM-5 используется для механизма построения коррекции ошибок (Forward Error Correction, FEC). Он реализуется треллис-кодированием и декодером Витерби. Применение механизма коррекции ошибок позволяет увеличить помехоустойчивость приемника на 6 дБ.

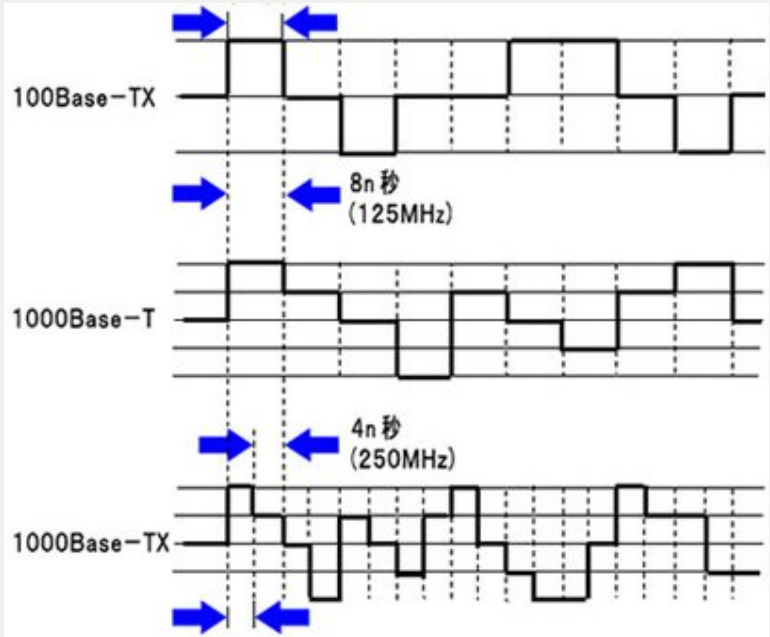
Для обеспечения двунаправленной передачи данных по каждой витой паре в трансиверах приходится использовать гибридные устройства, развязывающие передающий и приемный тракты, и мощные процессоры цифровой обработки сигналов (DSP), которые подавляют эхосигналы, генерируемые гибридными устройствами. Кроме того, в оборудовании 1000Base-T осуществляется подавление перекрестных наводок.



Технология 1000Base-TX



Стандарт разработан создан Ассоциацией Телекоммуникационной Промышленности (Telecommunications Industry Association, TIA) и опубликован в марте 2001 как «Спецификация физического уровня дуплексного Ethernet 1000 Мбит/с (1000Base-TX) симметричных кабельных систем категории 6 (ANSI/TIA/EIA-854-2001)». Стандарт разделяет принимаемые и посылаемые сигналы по парам (две пары передают данные, каждая на 500 Мбит/с и две пары принимают), что упрощает бы конструкцию приёмопередающих устройств за счет отсутствия схемы цифровой компенсации наводок и возвратных помех по сравнению с 1000Base-T. Однако, поскольку для работы технологии требуется кабель cat.6 или лучше, а к 2001 году оборудование 1000Base-T, работающее по линиям cat.5e уже широко использовалось, оборудование 1000Base-TX распространения не получило.



Технология 1000Base-X (IEEE 802.3z)

Стандарт IEEE 802.3z включает в себя 1000BASE-SX для передачи сигала по многомодовому оптоволокну, 1000BASE-LX — по одномодовому оптоволокну, и почти вышедший из употребления 1000BASE-CX — по экранированному сбалансированному медному кабелю. Эти стандарты используют кодирование 8b/10b, которое повышает скорость передачи линии на 25%, с 1000 Мбит/с до 1250 Мбит/с. Символы затем отправляются с использованием кода NRZ.

Кодирование 8b/10b

Table A-1. 8b/10b Data Symbol Codes

Data Byte Name	Data Byte Value (hex)	Bits HGF EDCBA (binary)	Current RD- abcdei fghj(binary)	Current RD+ abcdei fghj (binary)
D0.0	00	000 00000	100111 0100	011000 1011
D1.0	01	000 00001	011101 0100	100010 1011
D2.0	02	000 00010	101101 0100	010010 1011
D3.0	03	000 00011	110001 1011	110001 0100
D4.0	04	000 00100	110101 0100	001010 1011
D5.0	05	000 00101	101001 1011	101001 0100
D6.0	06	000 00110	011001 1011	011001 0100

При этом виде кодирования 8 бит кодируются 10-битным символом. При этом, код фактически представляет собой два кода: 5b/6b, за которым следует код 3b/4b. Коды подобраны так, чтобы количество 1 и 0 в каждой кодовой комбинации колебалось в пределах ± 2. Для того, чтобы обеспечить равномерность нулей и единиц при длительной передаче, кодовые комбинации зависят от соотношения числа 1 и 0 в предыдущих кодовых комбинациях.

10 Gigabit Ethernet

Технология впервые определена в 2002 в спецификации IEEE 802.3ae-2002. 10G Ethernet поддерживает только полнодуплексный режим работы.

Основные стандарты

- 10GBase-SR ("short range"). Использует многомодовое волокно и лазеры 850 нм. Используется схема кодирования 64b/66b. Дальность 25–400 м. Используются модули SFP+ и XFP.
- 10GBase-LR ("long reach"). Использует одномодовое волокно и лазеры 1310 нм. До 10 км.
- 10GBase-LX4. Использует технологию WDM. 4 трансивера на 3.125 Gbit/s и кодирование 8B10B. До 300 м по многомоде и до 10 км по одномоде.
- 10GBase-T (IEEE 802.3an-2006). Использует витую пару cat.6 (до 55 м) или cat.6a (до 100 м). Применяется модуляция PAM-16 и помехоустойчивые LDPC-коды. Передача по четырем парам дуплексом с эхоподавлением, как и в 1000Base-T.

Стандарт 10GBase-T послужил основой для стандартов 2.5GBase-T (100 м по cat.5e) and 5GBASEase-T (100 м по cat.6) (IEEE 802.3bz-2016).

Адресация на канальном уровне модели OSI

MAC-адрес (Media Access Control — управление доступом к среде, Hardware Address)

Уникальный идентификатор, присваиваемый каждой единице активного оборудования компьютерных сетей.

Основные виды адресов канального уровня

- MAC-48/EUI-48
- EUI-64

Адреса в каждом из пространств должны быть глобально уникальными.

Структура адреса MAC-48/EUI-48

MAC-48/EUI-48

- EUI-48 (Extended Unique Identifier) — идентификатор, созданный путём соединения 24-битного уникального идентификатора организации (OUI) с 24-битным дополнительным идентификатором, который назначается организацией, получившей OUI.
- MAC-48 — частный случай использования идентификатора EUI-48 в качестве аппаратного адреса сетевого интерфейса.

Уникальный идентификатор организации (OUI)

Organizationally Unique Identifier (OUI)

Уникальный идентификатор организации — 24-битный номер, который присваивается регистрационной администрацией IEEE.
OUI имеет три октета, или 24 бита. Значащими являются только 22 бита. Старший бит первого октета OUI присваивается сразу в двух значениях — 0 и 1, которые используются, в частности, в MAC-адресах для обозначения одиночного или группового адресата. Следующий за ним бит всегда имеет значение 0. В шестнадцатеричном (каноническом) формате (отображается такими программами как `ipconfig` и `ifconfig`) значащими являются соответственно 2 младших бита в первом октете. Таким образом, у всех одиночных адресов в шестнадцатеричном формате второй символ всегда является четным, у групповых адресов — нечетным.

Полный список OUI размещен по адресу `http://standards.ieee.org/regauth/oui/oui.txt`

Примеры OUI

```
OUI/MA-L                    Organization
company_id                  Organization

E0-43-DB    (hex)           Shenzhen ViewAt Technology Co.,Ltd.
E043DB      (base 16)           Shenzhen ViewAt Technology Co.,Ltd.
CN

24-05-F5    (hex)           Integrated Device Technology (Malaysia) Sdn. Bhd.
2405F5      (base 16)           Integrated Device Technology (Malaysia) Sdn. Bhd.
MY

2C-30-33    (hex)           NETGEAR
2C3033      (base 16)           NETGEAR
US
```

Формат MAC-адреса

Шестнадцатеричный (канонический) формат

- AC-DE-48-01-02-03 (Windows)
- AC:DE:48:01:02:03 (Unix)
- ACDE.4801.0203 (Cisco)

```
|                               MAC-48                                  |
|            OUI                |  дополнительный идентификатор   |
| Октет 0   | Октет 1   | Октет 2   | Октет 3   | Октет 4   | Октет 5   |
|  A     C |  D     E |  4     8 |  0     1 |  0     2 |  0     3 |
|1010  1100|1101  1110|0100  1000|0000  0001|0000  0010|0000  0011|
         ||
         |одиночный (0) или групповой (1) адресат
         |
         всегда 0 при использовании OUI
```

Бит-реверсный (неканонический) формат

- 35:7B:12:80:40:C0

```
|                               MAC-48                                  |
|            OUI                |  дополнительный идентификатор   |
| Октет 0   | Октет 1   | Октет 2   | Октет 3   | Октет 4   | Октет 5   |
|  3     5 |  7     B |  1     2 |  8     0 |  4     0 |  C     0 |
|0011  0101|0111  1011|0001  0010|1000  0000|0100  0000|1100  0000|
 ||
 |всегда 0 при использовании OUI
 |
 одиночный (0) или групповой (1) адресат
```

64-битный расширенный уникальный идентификатор EUI-64

Extended Unique Identifier EUI-64

EUI-64 — это 64-битный идентификатор, созданный путём соединения 24-битного OUI с 40-битным дополнительным идентификатором, который назначается организацией, получившей OUI. Используются в ZigBee, FireWire, а также в IPv6 в качестве младших 64 бит сетевого адреса узла.
В соответствии с рекомендациями IEEE, первые 4 цифры дополнительного 40-битного идентификатора не должны быть $\texttt{FFFE}_{16}$ или $\texttt{FFFF}_{16}$ (то есть EUI-64 идентификаторы вида $\texttt{ccccccFFFEeeeeee}_{16}$ и $\texttt{ccccccFFFFeeeeee}_{16}$ недопустимы) — они используются для поддержки инкапсуляции значений идентификаторов MAC-48 и EUI-48 в EUI-64.

Отображение EUI-48 и MAC-48 в EUI-64

Тип адреса	Исходный вид	Отображение в EUI-64
EUI-48	`xx-yy-zz-ww-vv-tt`	`xx-yy-zz-FF-FE-ww-vv-tt`
MAC-48	`xx-yy-zz-ww-vv-tt`	`xx-yy-zz-FF-FF-ww-vv-tt`

Список использованных источников

1. Материалы с сайта `https://wikipedia.org/`
2. Информационно-вычислительные сети : учебное пособие / Д. А. Капустин, В. Е. Дементьев. — Ульяновск : УлГТУ, 2011.
3. Компьютерные сети. Принципы, технологии, протоколы: Учебник для вузов / В. Г. Олифер, Н. А. Олифер. — СПб. : Питер, 2010.
4. Базовые технологии локальных сетей / В. Г. Олифер, Н. А. Олифер // CIT Forum. 1999. URL: `http://citforum.ru/nets/protocols2/index.shtml`
5. Семенов, Ю. А. Телекоммуникационные технологии. URL: `http://book.itep.ru/`
6. Кабельные системы категории 6 и оборудование Gigabit Ethernet / М. Фаррант, Т. Уалднер // Сети и системы связи.
7. Gigabit Ethernet over Category 5 / F. Mlinarsky. 1998.
8. Технология 1000Base-T на физическом уровне / С. Пахомов // КомпьютерПресс. №2. 2002.
9. RFC-7042. IANA Considerations and IETF Protocol and Documentation Usage for IEEE 802 Parameters.

Сети связи
Протоколы, сервисы и услуги в Интернет и IP-сетях

Тема № 4
Протокол IPv4

доц. каф. СС и ПД, к.т.н. С. С. Владимиров

2020 г.

Протокол IP

Протокол IP

Маршрутизируемый протокол сетевого уровня стека TCP/IP. Создан в 1981 г. Назначение протокола — передача пакетов между хостами сетей TCP/IP. Основной стандарт для протокола IPv4 — RFC 791 (STD 5) с обновлениями в RFC 1349, 2474, 6864. Являясь одним из базовых протоколов современных компьютерных сетей, протокол IP затрагивается и в многих других стандартах RFC и STD. Неотъемлемой частью протокола IP является адресация сетевых устройств.

IP объединяет сегменты сети в единую сеть, обеспечивая доставку пакетов данных между любыми узлами сети через произвольное число промежуточных узлов (маршрутизаторов). IP не гарантирует надёжной доставки пакета до адресата — в частности, пакеты могут прийти не в том порядке, в котором были отправлены, продублироваться (приходят две копии одного пакета), оказаться повреждёнными (обычно повреждённые пакеты уничтожаются) или не прийти вовсе. Гарантию безошибочной доставки пакетов дают некоторые протоколы более высокого уровня — транспортного уровня сетевой модели OSI, — например, TCP, которые используют IP в качестве транспорта.

Версии протокола IP

1. IPv4 — 32-битная адресация. Основной сетевой протокол в настоящее время.
2. IPv6 — 128-битная адресация. Перспективный протокол. Доля использования IPv6 постепенно увеличивается. Широко используется в сетях Интернета вещей и сенсорных сетях.

Адресация IPv4

Адрес IPv4

Сетевой адрес устройства. В версии IPv4 имеет длину 4 байта (32 бита). IP-адрес можно разделить на две части — адрес сети и адрес узла в сети. Традиционной формой записи IPv4 адреса является запись в виде четырёх десятичных чисел (от 0 до 255), разделённых точками.

Структура адреса IPv4

В рамках одной сети биты адреса сети неизменны.

Число сетевых узлов в одной сети

Если в IP-адресе под адрес узла отведено n бит, то такая сеть содержит

$$N = 2^n - 2 \text{ узлов.}$$

Оставшиеся два адреса отведены под

- адрес сети — все биты адреса узла равны `0`
- широковещательный адрес — все биты адреса узла равны `1`

Виды адресации

- Классовая адресация (Classful network)
- Бесклассовая адресация (Classless Inter-Domain Routing, CIDR)

Классовая адресация IP сетей

Классовая адресация (Classful network)

Принцип сетевой адресации, использовавшийся в Интернете в период с 1981 по 1993 годы. Адресное пространство протокола IPv4 делится на пять классов адресов: A, B, C, D и E. Принадлежность адреса к конкретному классу задаётся первыми битами адреса. Каждый класс определяет либо соответствующий размер сети, то есть количество возможных адресов хостов внутри данной сети (классы A, B, C), либо сеть многоадресной передачи (класс D). Диапазон адресов пятого класса (E) был зарезервирован для будущих или экспериментальных целей.

Классы адресов

Класс	Первые биты	Распр. байт Сеть, Хост	Число сетей	Хостов в сети	Начальный адрес	Конечный адрес
A	0	C.X.X.X	128	16777214	0.0.0.0	127.255.255.255
B	10	C.C.X.X	16384	65534	128.0.0.0	191.255.255.255
C	110	C.C.C.X	2097152	254	192.0.0.0	223.255.255.255
D	1110	Групповой адрес			224.0.0.0	239.255.255.255
E	1111	Зарезервировано			240.0.0.0	255.255.255.255

Бесклассовая адресация IP сетей

Бесклассовая адресация (Classless Inter-Domain Routing, CIDR)

Метод IP-адресации, позволяющий гибко управлять пространством IP-адресов, не используя жёсткие рамки классовой адресации. Использование этого метода позволяет экономно использовать ограниченный ресурс IP-адресов, поскольку возможно применение различных *масок подсетей* к различным подсетям.

Маска подсети (Variable length subnet mask, VLSM)

Битовая маска, определяющая, какая часть IP-адреса узла сети относится к адресу сети (эти биты в маске равны `1`), а какая — к адресу самого узла в этой сети (биты маски, равные `0`). Маска подсети не является частью IP-пакета. Она указывается в сетевых настройках узла сети.
Зная IP-адрес и маску подсети, можно определить, к какой сети относится данный IP-адрес. Для этого необходимо применить адресу и маске операцию поразрядной конъюнкции (логическое И).
По маске хост-отправитель определяет, куда необходимо направлять пакет. Если при передаче пакета IP-адрес узла назначения относится к той же сети, что и отправитель, пакет передается в рамках сети. Если же получатель принадлежит другой сети, то отправитель передает пакет на маршрутизатор для дальнейшей передачи.

Пример использования маски подсети

```
IP-адрес:       11000000.10101000.00000001.00000010 (192.168.1.2)
Маска подсети:  11111111.11111111.11111110.00000000 (255.255.254.0)
Адрес сети:     11000000.10101000.00000000.00000000 (192.168.0.0)
```

Способы записи маски подсети

- Десятичный: `255.255.254.0`
- Двоичный: `11111111.11111111.11111110.00000000`
- Постфиксный: `/23`

Специальные IP-адреса

Часть адресов IPv4 зарезервированы IANA для конкретных целей. Полный список этих адресов приведен в RFC 3330 и RFC 6890. Некоторые зарезервированные адреса имеют свои RFC, в которых приведено их подробное описание.

Некоторые важные специальные адреса IPv4

- `0.0.0.0/8` — адреса из этого блока относятся к адресам *этой же* сети (RFC 1700).
- `127.0.0.0/8` — диапазон адресов для петлевого (localhost, loopback) интерфейса (RFC 1700). Как правило используется адрес `127.0.0.1/32`. Использование loopback-адреса позволяет устанавливать соединение и передавать информацию для программ-серверов, работающих на том же компьютере, что и программа-клиент, независимо от конфигурации аппаратных сетевых средств компьютера (не требуется сетевая карта, модем, и прочее коммуникационное оборудование, интерфейс реализуется при помощи драйвера псевдоустройства в ядре операционной системы). Таким образом, для работы клиент-серверных приложений на одном компьютере не требуется изобретать дополнительные протоколы и дописывать программные модули.
- `169.254.0.0/16` — диапазон «link-local» адресов, предназначенных для использования в одной физической сети. Эти адреса автоматически присваиваются интерфейсу в тех случаях, когда другие способы автоматического присвоения адресов (DHCP) недоступны.
- `255.255.255.255/32` — широковещательный адрес («limited broadcast»). Используется только внутри одной физической сети.

Частные IP-адреса

Частный IP-адрес (Private IP address)

IP-адрес, принадлежащий к специальному диапазону, выделенному для локальных IP-сетей, т. е. не используемому в сети Интернет. Также его называют внутренним, внутрисетевым, локальным или «серым», в отличие от «белых» адресов, применяемых в сети Интернет. Распределение таких адресов никем не контролируется. В связи с дефицитом свободных IP-адресов, провайдеры как правило раздают своим абонентам именно внутрисетевые адреса — а не внешние.
Иногда частные адреса называют неанонсированными, внешние (так называемые «белые IP») — анонсированными.
Диапазоны частных адресов определены IANA и указаны в RFC 1918.

Диапазоны частных адресов IPv4

Начальный адрес диапазона	Конечный адрес диапазона	Маска подсети (десятичный вид)	Маска подсети (постфикс)
`10.0.0.0`	`10.255.255.255`	`255.0.0.0`	`/8`
`172.16.0.0`	`172.31.255.255`	`255.240.0.0`	`/12`
`192.168.0.0`	`192.168.255.255`	`255.255.0.0`	`/16`

Пакеты, идущие с внутренних IP-адресов или на них, магистральные маршрутизаторы не пропускают. То есть, внутрисетевые машины, если не предпринимать никаких мер, изолированы от Интернета. Тем не менее, есть несколько технологий, которые позволяют выходить таким машинам в Интернет.

Сервер-посредник (прокси-сервер). Сетевой туннель

Сервер-посредник (прокси-сервер)

Многие из старых интернет-служб (электронная почта, IRC, Usenet) специально спроектированы для машин, которые не имеют прямого выхода в Интернет. Для этого в самих протоколах предусмотрена эстафетная передача информации через сервер-посредник.

На примере электронной почты. Корпоративный почтовый сервер имеет два IP-адреса: внутренний и внешний. Для отправки почты пользователь по протоколу SMTP связывается с сервером. Сервер от своего имени выходит в интернет и переправляет почту дальше по цепочке. На этот же сервер по протоколу SMTP поступает входящая корреспонденция. Чтобы проверить ящик, пользователи соединяются с сервером по протоколу POP3.

Для Всемирной паутины была придумана технология «прокси-сервер». Машина с частным адресом обращается к прокси-серверу и посылает на него команды HTTP. Прокси-сервер связывается с веб-сервером от своего имени.

Такая конструкция удовлетворила важнейшие нужды внутрисетевых пользователей. Минусом является сложная архитектура сервера-посредника — он должен поддерживать множество разных протоколов, либо необходимо использовать отдельный сервер-посредник для каждого протокола. А по протоколам, которые посредник не поддерживает или которые не рассчитаны на эстафетную передачу, выход в Интернет невозможен. Одни программы (ICQ, Skype, P2P-часть протокола BitTorrent) проходят сквозь прокси-серверы, «заворачивая» свой протокол в HTTP-пакеты, другие (Subversion, связь с трекером в протоколе BitTorrent) — изначально реализуют свой протокол поверх HTTP. Следующая технология, NAT, позволила внутрисетевым машинам выходить в интернет по любому прикладному протоколу.

Прокси-серверы работают на прикладном уровне и потому могут налаживать цензуру сайтов и кэшировать страницы для экономии трафика — поэтому прокси-серверы широко применяются в корпоративных сетях (даже если другие протоколы работают через NAT). Кроме того, прокси-серверы применяются для особых задач, на которые NAT не способен (например, для передачи файлов в мессенджерах, когда обе машины за NAT).

Сетевой туннель

Технология, когда пакеты сетевого уровня «заворачиваются» в пакеты более высоких уровней (например, транспортного). Это позволяет наладить виртуальную локальную сеть поверх сети совсем другого устройства. Существует много технологий туннелирования (PPPoE, VPN, Hamachi и другие), со своими областями применения.

Трансляция сетевых адресов (NAT)

Технология была задокументирована в 1994 г. (RFC 1631, заменен на RFC 3022 в 2001 г.). Как правило, под NAT понимают так называемый Source NAT (SNAT), при котором маршрутизатор, реализующий NAT, пропуская идущий из локальной сети пакет, заменяет адрес отправителя своим и меняет номер порта, чтобы различать ответные пакеты, адресованные разным локальным компьютерам. Комбинацию, нужную для обратной подстановки, роутер сохраняет у себя во временной NAT-таблице. Когда маршрутизатор получает ответ от сервера, он по таблице открытых соединений восстанавливает адресата и ретранслирует ему ответ.
Через NAT внутрисетевой компьютер может налаживать связь с любым сервером Интернета по любому прикладному протоколу. Но у NAT есть и недостатки. С машиной с частным IP-адресом связаться можно только изнутри локальной сети. С одной стороны, это делает локальную сеть недоступной для многих атак извне. С другой стороны, в некоторых службах Интернета (одноранговых сетях, сетевых играх, передаче файлов в мессенджерах) это создаёт проблемы: если у одного из компьютеров IP-адрес частный, а у другого внешний, инициатором соединения будет клиент с частным IP; если частные у обоих — прямой обмен между ними затруднён. Для решения этой задачи используется так называемый Destination NAT (DNAT).

Концепции DNAT

1. *Статический NAT.* Отображение незарегистрированного IP-адреса на зарегистрированный IP-адрес на основании один к одному.
2. *Динамический NAT.* Отображает незарегистрированный IP-адрес на зарегистрированный адрес из группы зарегистрированных IP-адресов. Динамический NAT также устанавливает непосредственное отображение между незарегистрированным и зарегистрированным адресом, но отображение может меняться в зависимости от зарегистрированного адреса, доступного в пуле адресов, во время коммуникации.
3. *Перегруженный NAT* (NAPT, NAT Overload, PAT, маскарадинг). Форма динамического NAT, который отображает несколько незарегистрированных адресов в единственный зарегистрированный IP-адрес, используя различные порты. Известен также как PAT (Port Address Translation). При перегрузке каждый компьютер в частной сети транслируется в тот же самый адрес, но с различным номером порта.

Типы NAT и технология NAT loopback

Типы NAT

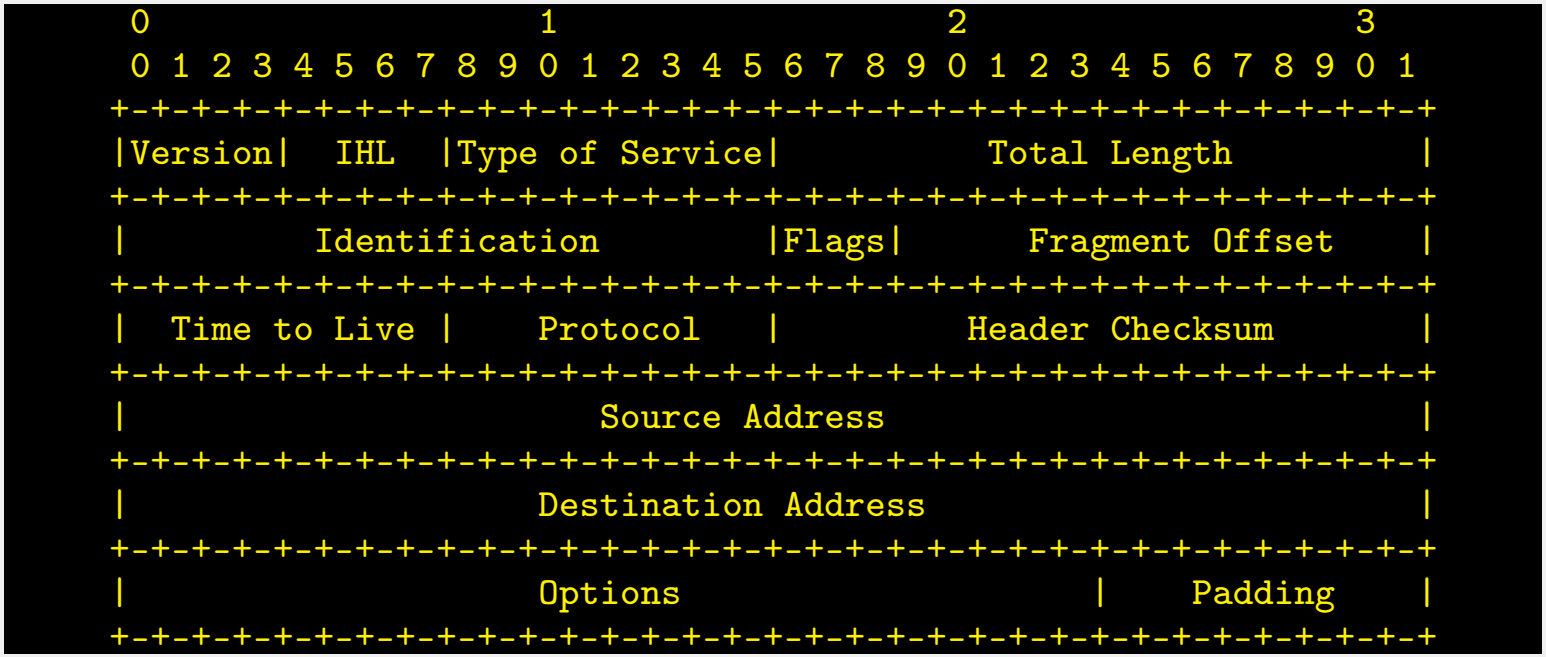
- Cone NAT, Full Cone NAT — Однозначная (взаимная) трансляция между парами «внутренний адрес: внутренний порт» и «публичный адрес: публичный порт». Любой внешний хост может инициировать соединение с внутренним хостом (если это разрешено в правилах межсетевого экрана).
- Address-Restricted cone NAT, Restricted cone NAT — Постоянная трансляция между парой «внутренний адрес: внутренний порт» и «публичный адрес: публичный порт». Любое соединение, инициированное с внутреннего адреса, позволяет в дальнейшем получать ему пакеты с любого порта того публичного хоста, к которому он отправлял пакеты ранее.
- Port-Restricted cone NAT — Трансляция между парой «внутренний адрес: внутренний порт» и «публичный адрес: публичный порт», при которой входящие пакеты проходят на внутренний хост только с одного порта публичного хоста — того, на который внутренний хост уже посылал пакет.
- Симметричный NAT (Symmetric NAT) — Трансляция, при которой каждое соединение, инициируемое парой «внутренний адрес: внутренний порт» преобразуется в свободную уникальную случайно выбранную пару «публичный адрес: публичный порт». При этом инициация соединения из публичной сети невозможна.

NAT loopback

Технология NAT loopback (NAT hairpinning) заключается в том, что пакет, приходящий из внутренней сети на внешний IP-адрес маршрутизатора, считается пришедшим извне — а значит, работают правила брандмауэра, относящиеся ко внешним соединениям. Если пакет успешно пройдёт сквозь брандмауэр, сработает NAT, взяв на себя посредничество между двумя внутрисетевыми машинами. Эта технология позволяет прямо изнутри локальной сети проверить, как настроены сетевые службы, и обеспечивает доступ к серверу, находящемуся в локальной сети, по доменному имени. Недостатком NAT loopback можно считать повышенную нагрузку на хаб и маршрутизатор (по сравнению с прямым доступом к серверу).

Формат пакета IPv4

Формат заголовка пакета IPv4



Заголовок пакета IPv4 содержит 14 полей, из которых 13 являются обязательными. Четырнадцатое поле предназначено для необязательных опций. Поля используют порядок байтов от старшего к младшему, старшие биты идут первыми. Первый бит имеет номер 0. Таким образом, например, поле с версией находится в четырёх старших битах первого байта.

Формат заголовка пакета IPv4. Поля заголовка

Версия протокола (Version)

Имеет размер в четыре бита. Для IPv4 значение этого поля равно 4.

Размер заголовка (Internet Header Length)

Имеет размер в четыре бита. Содержит размер заголовка пакета в 32-битных словах. Поскольку число опций не постоянно, указание размера важно для отделения заголовка от данных. Минимальное значение равно 5 ($5 \cdot 32 = 160$ бит, 20 байт), максимальное — 15 (60 байт).

Тип обслуживания» (Type of Service) или Differentiated Services Code Point (DSCP)

Изначально называлось «тип обслуживания» (Type of Service, ToS), в настоящее время определяется RFC 2474 как «Differentiated Services». Используется для разделения трафика на классы обслуживания, например для установки чувствительному к задержкам трафику, такому как VoIP, большего приоритета.

Размер пакета (Total Length)

16-битный полный размер пакета в байтах, включая заголовок и данные. Минимальный размер равен 20 байтам (заголовок без данных), максимальный — 65535 байт. Хосты должны поддерживать передачу пакетов размером до 576 байт, но современные реализации обычно поддерживают гораздо больший размер. Пакеты большего размера, чем поддерживает канал связи, фрагментируются.

Идентификатор (Identification)

Преимущественно используется для идентификации фрагментов пакета, если он был фрагментирован. Существуют эксперименты по его использованию для других целей, таких как добавление информации о трассировке пакета для упрощения отслеживания пути пакета с подделанным адресом источника.

Формат заголовка пакета IPv4. Поля заголовка

Флаги (Flags)

Поле размером три бита содержащее флаги контроля над фрагментацией. Биты, от старшего к младшему, означают:

0: Зарезервирован, должен быть равен 0.

1: Не фрагментировать

2: У пакета ещё есть фрагменты

Если установлен флаг «не фрагментировать», то в случае необходимости фрагментации такой пакет будет уничтожен. Может использоваться для передачи данных хостам, не имеющим достаточных ресурсов для обработки фрагментированных пакетов.
Флаг «есть фрагменты» должен быть установлен в 1 у всех фрагментов пакета, кроме последнего. У нефрагментированных устанавливается в 0 — такой пакет считается собственным последним фрагментом.

Смещение фрагмента (Fragment Offset)

Поле размером в 13 бит, указывает смещение текущего фрагмента от начала передачи фрагментированного пакета в блоках по 8 байт. Позволяет $(213 - 1) \cdot 8 = 65528$ байт смещения, что превышает максимальный размер пакета. Первый фрагмент в последовательности имеет нулевое смещение.

Время жизни (Time to Live, TTL) пакета

Позволяет предотвратить закольцовывание пакетов в сети путем уничтожения пакетов, превысивших время жизни. Указывается в секундах, интервалы менее секунды округляются до одной секунды. На практике каждый маршрутизатор уменьшает время жизни пакетов на единицу. Пакеты, время жизни которых стало равно нулю уничтожаются, а отправившему посылается сообщение ICMP Time Exceeded. На отправке пакетов с разным временем жизни основана трассировка их пути прохождения (traceroute).

Формат заголовка пакета IPv4. Поля заголовка

Протокол (Protocol)

Указывает, данные какого протокола содержит пакет (например, TCP или ICMP). Присвоенные номера протоколов можно найти на сайте IANA.

Контрольная сумма заголовка (Header Checksum)

16-битная контрольная сумма, используемая для проверки целостности заголовка. Каждый хост или маршрутизатор сравнивает контрольную сумму заголовка со значением этого поля и отбрасывает пакет, если они не совпадают. Целостность данных IP не проверяет — она проверяется протоколами более высоких уровней, которые тоже используют контрольные суммы. Поскольку TTL уменьшается на каждом шаге прохождения пакета, сумма тоже должна вычисляться на каждом шаге. Метод пересчета определён в RFC 1071.

Адрес источника (Source Address)

32-битный адрес отправителя пакета. Может не совпадать с настоящим адресом отправителя из-за трансляции адресов.

Адрес назначения (Destination Address)

32-битный адрес получателя пакета.

Опции (Options)

За адресом назначения может следовать поле дополнительных опций, но оно используется редко. Размер заголовка в этом случае должен быть достаточным чтобы вместить все опции (с учетом дополнения до целого числа 32-битных слов).

Контрольная сумма заголовка IPv4 (Header Checksum)

Контрольная сумма (КС, CS) заголовка IPv4 представляет собой 16-битовое поразрядное дополнение суммы всех 16-битовых слов заголовка. При вычислении контрольной суммы значение самого поля принимается нулевым.

Пример пакета IPv4. Заголовок IPv4 выделен зеленым. Поле КС — синим

```
0000: 00 50 FC 1E BF 8D 00 30 4F 0E 89 65 08 00 45 00
0010: 00 38 89 28 40 00 80 06 11 21 C0 A8 01 32 C3 13
0020: DB 88 04 50 00 15 00 4C 69 E7 3C 00 27 92 50 18
0030: 22 05 D3 39 00 00 55 53 45 52 20 61 6E 6F 6E 79
0040: 6D 6F 75 73 0D 0A
```

Вычисление КС заголовка IPv4

1. Заголовок разбивается на слова размером 16 бит (2 байта) каждое. Поле КС принимается равным нулю (не участвует в вычислении). Все полученные слова суммируются.

$$4500 + 0038 + 8928 + 4000 + 8006 + 0000 + C0A8 + 0132 + C313 + DB88 = 3EEDB$$

2. Если длина результата суммирования превышает 2 байта (4 шестнадцатеричные цифры), то результат делится на две части (правые 4 цифры и остаток), которые суммируются между собой.

$$0003 + EEDB = EEDE$$

3. Находится поразрядное дополнение от итоговой суммы. Результат и будет контрольной суммой заголовка пакета IPv4.

$$FFFF - EEDE = 1121 = CS_{IPv4}$$

Источники

- Материалы с сайта `https://wikipedia.org/`
- Материалы с сайта `https://www.rfc-editor.org/`
- Материалы с сайта `http://www.ibm.com/`
- Компьютерные сети. Принципы, технологии, протоколы: Учебник для вузов / В. Г. Олифер, Н. А. Олифер. — СПб. : Питер, 2010.
- Основы построения INTERNET : Электронный курс / Е. М. Доронин. URL: `http://opds.sut.ru/`
- RFC-791. Internet Protocol.
- RFC-1918. Address Allocation for Private Internets.
- RFC-3330. Special-Use IPv4 Addresses.

Сети связи
Протоколы, сервисы и услуги в Интернет и IP-сетях

Тема № 5
Протоколы ARP, InARP, RARP

доц. каф. СС и ПД, к.т.н. С. С. Владимиров

2020 г.

Протоколы ARP и Inverse ARP

ARP (Address Resolution Protocol) — Протокол определения (разрешения) адреса

Протокол в компьютерных сетях, предназначенный для определения канального адреса (MAC) по известному сетевому адресу (IPv4). Как правило используется совместно с протоколом IPv4 в сетях, на основе технологии Ethernet. Описание протокола было опубликовано в ноябре 1982 года в RFC 826 (с обновлениями в RFC 5227/2008 и RFC 5494/2009). ARP был спроектирован для случая передачи IP-пакетов через сегмент Ethernet. При этом общий принцип, предложенный для ARP, может, и был использован и для сетей других типов.

По модели OSI протокол ARP относят или к канальному, или к сетевому уровням. Также часто указывают, что он находится между этими уровнями модели OSI. С точки зрения взаимодействия с другими протоколами пакет ARP инкапсулируется напрямую в кадр Ethernet, т. е. протокол ARP работает поверх Ethernet. В поле «Ethertype» заголовка кадра Ethernet протоколу ARP соответствует $ID = \texttt{0x0806}$.

Типы ARP-пакетов

1. ARP запрос (ARP request)
2. ARP ответ (ARP reply)

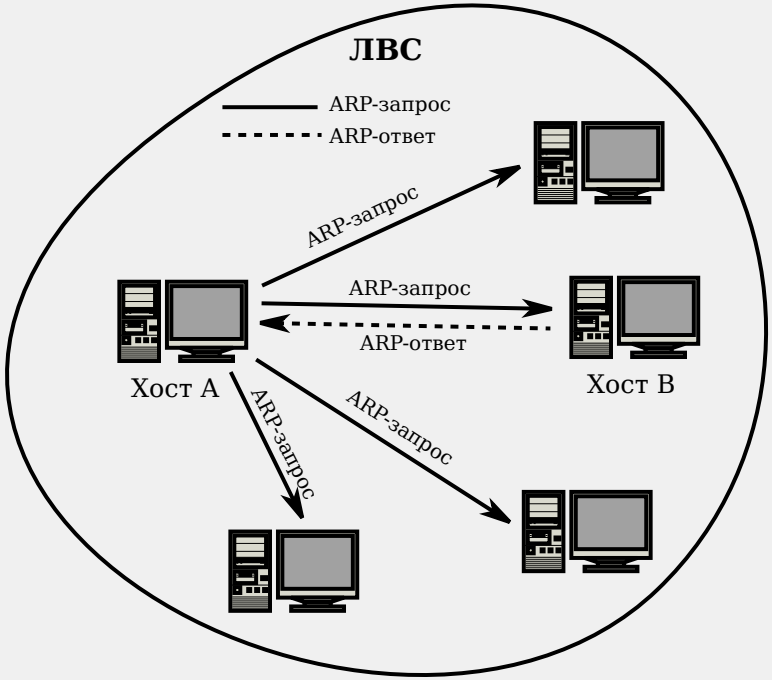
Inverse ARP (Inverse Address Resolution Protocol, InARP)

Протокол для получения адресов сетевого уровня (например IP адресов) других рабочих станций по их адресам канального уровня (например, DLCI в Frame Relay сетях). В основном используется во Frame Relay и ATM сетях. Описание протокола опубликовано в RFC 2390.
InARP реализовано как расширение ARP. Форматы пакетов этих протоколов одни и те же, различаются лишь коды операций и заполняемые поля.

Принцип работы ARP

Порядок разрешения адреса

1. В начале передачи данных от узла *A* на узел *B*, узел *A* должен определить его MAC-адрес, т. е. он должен выполнить процедуру отображения IP-адреса на локальный MAC-адрес. Для этого узел *A* формирует ARP запрос, указывая в нем известный IP-адрес *B*, и рассылает запрос широковещательно по протоколу канального уровня.
2. Все узлы локальной сети получают ARP запрос и сравнивают указанный в нем IP-адрес с собственным.
3. Определив, что IP-адрес в запросе совпадает с его IP-адресом, узел *B* формирует ARP-ответ, в котором указывает свой IP-адрес и свой локальный MAC-адрес, и отправляет его уже направленно, так как в ARP-запросе отправитель *A* указывает свой MAC-адрес.



4. Получив ARP-ответ, передающий узел *A* записывает соответствие MAC-адреса *B* его IP-адресу в специальную ARP-таблицу. Теперь, пока запись хранится в ARP-таблице (т. н. время жизни записи), данные могут передаваться от *A* к *B*. Необходимо отметить, что для передачи данных от *B* к *A* требуется провести обратную процедуру.
5. В случае когда данные от *A* к *B* всё ещё передаются, но время жизни записи в ARP-таблице истекает, узел *A* должен обновить запись. Для этого он посылает *повторный ARP-запрос*, который отправляется на известный MAC-адрес узла *B*. В том случае, если ответ на повторный запрос не приходит, процедура разрешения адреса повторяется с самого начала (посредством широковещательного ARP-запроса).

ARP-таблица. Самопроизвольный ARP

Пример ARP-таблицы

```
user@pc:/$ sudo arp -a
HostPC1.lan (192.168.1.111) at 24:08:34:2a:ef:34 [ether] on eth0
Server2.lan (192.168.1.3) at 1c:2b:d4:f9:78:35 [ether] on eth0
OpenWrt.lan (192.168.1.203) at e8:14:f6:43:3b:12 [ether] on eth0
Gateway.lan (192.168.1.1) at e8:d3:27:2e:0f:75 [ether] on eth0
```

Самопроизвольный ARP (gratuitous ARP)

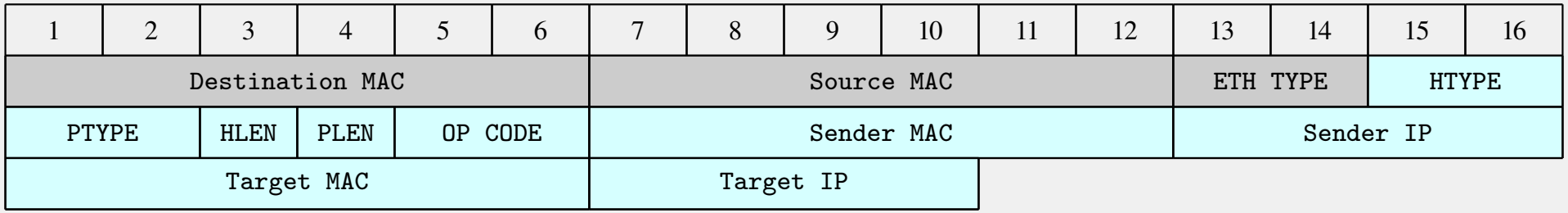
Это такое поведение ARP, когда ARP-ответ присылается, когда в этом (с точки зрения получателя) нет особой необходимости. Самопроизвольный ARP-ответ это пакет-ответ ARP, присланный без запроса. Он применяется для определения конфликтов IP-адресов в сети: как только станция получает адрес по DHCP или адрес присваивается вручную, рассылается самопроизвольный ARP-ответ.
Самопроизвольный ARP может быть полезен в следующих случаях:

- обновление ARP-таблиц, в частности, в кластерных системах;
- информирование коммутаторов;
- извещение о включении сетевого интерфейса.

Несмотря на эффективность самопроизвольного ARP, он является особенно небезопасным, поскольку с его помощью можно уверить удалённый узел в том, что MAC-адрес какой-либо системы, находящейся с ней в одной сети, изменился и указать, какой адрес используется теперь.

ARP-пакет

Структура ARP-пакета



Hardware type (HTYPE)

Номер протокола передачи канального уровня (`0x0001` для протокола Ethernet).

Protocol type (PTYPE)

Код протокола сетевого уровня (`0x0800` для протокола IPv4).

Hardware length (HLEN)

Длина физического адреса в байтах. Адреса Ethernet имеют длину 6 байт.

Protocol length (PLEN)

Длина логического адреса в байтах. IPv4 адреса имеют длину 4 байта.

ARP-пакет

Operation code (OP CODE)

Код операции: `0x01` в случае ARP-запроса и `0x02` в случае ARP-ответа

Sender MAC. Sender hardware address (SHA)

Физический адрес отправителя.

Sender IP. Sender protocol address (SPA)

Сетевой адрес отправителя.

Target MAC. Target hardware address (THA)

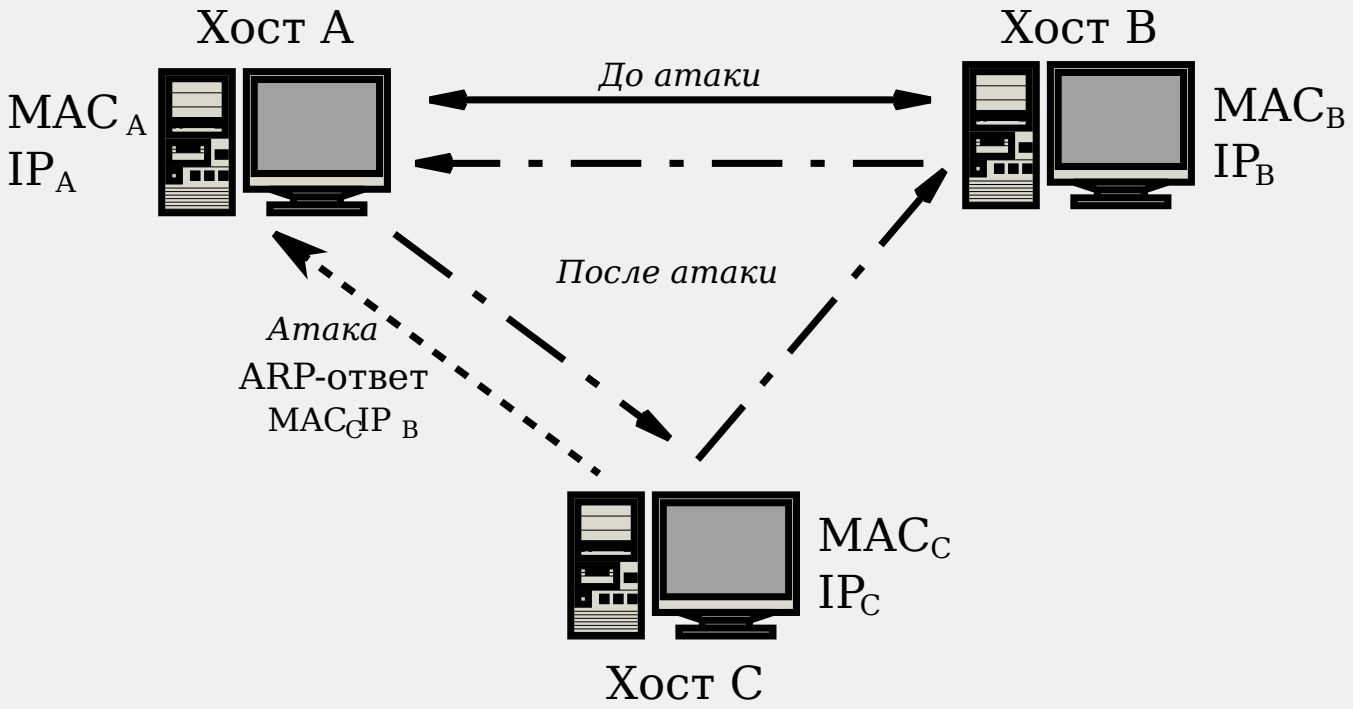
Физический адрес получателя. При запросе поле заполняется нулями.

Target IP. Target protocol address (TPA)

Сетевой адрес получателя.

ARP-spoofing (ARP-poisoning)

Схема сетевой атаки ARP-спуфинг

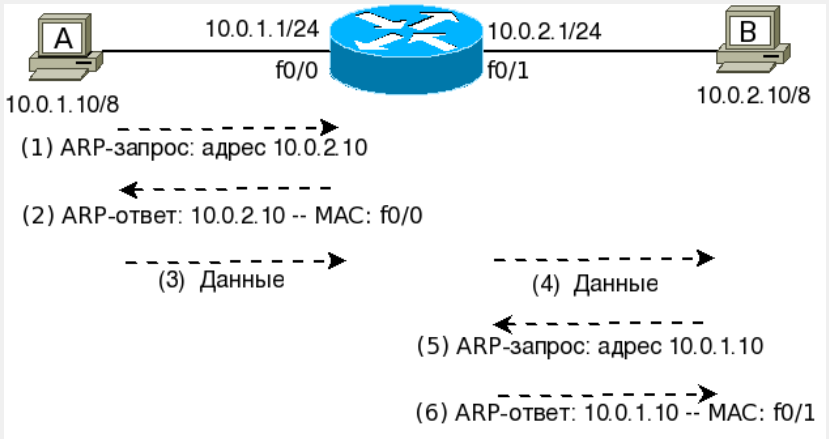


Сетевая атака ARP-спуфинг (ARP-spoofing) основана на использовании самопроизвольного ARP.
Чтобы перехватить сетевые пакеты, которые атакуемый хост (*A*) отправляет на хост *B*, атакующий хост (*C*) формирует ARP-ответ, в котором ставит в соответствие IP-адресу хоста *B* свой MAC-адрес. Далее этот пакет отправляется на хост *A*. В том случае, если хост *A* поддерживает самопроизвольный ARP, он модифицирует собственную ARP-таблицу и помещает туда запись, где вместо настоящего MAC-адреса хоста *B* стоит MAC-адрес атакующего хоста *C*.
Теперь пакеты, отправляемые хостом *A* на хост *B*, будут передаваться хосту *C*.

Proxy ARP

Техника использования ARP-протокола, позволяющая объединить две не связанные на канальном уровне сети в одну. Хосты, находящиеся в этих сетях, могут использовать адреса из одной IP-подсети и обмениваться трафиком между собой без использования маршрутизатора (как им кажется).

На рисунке изображены два хоста *A* и *B*, которые находятся на канальном уровне в разных сегментах. На хостах не настроен шлюз по умолчанию. Маски подсетей на маршрутизаторе и на хостах отличаются.



1. Хост *A* хочет отправить какие-то данные хосту *B*. Так как, на хосте *A* IP-адрес `10.0.1.10` с маской `/8`, то он считает, что хост *B* с IP-адресом `10.0.2.10/8`, также находится с ним в одной сети (хосты считают, что они в сети `10.0.0.0/8`). Хосту *A* необходимо узнать MAC-адрес хоста *B*. Он отправляет ARP-запрос в сеть.
2. Маршрутизатор получает ARP-запрос, но не перенаправляет его, так как получатель в другой сети. Если на маршрутизаторе включен Proxy ARP, то маршрутизатор отправляет хосту *A* ARP-ответ, в котором подставляет свой MAC-адрес. То есть, для хоста *A*, создается соответствие `10.0.2.10` — MAC `f0/0`.
3. Теперь хост *A* может отправить данные.
4. Маршрутизатор получается пакет, смотрит на IP-адрес получателя и перенаправляет пакет на него (при условии, что в ARP кеше маршрутизатора уже есть запись для хоста *B*).
5. Хост *B* аналогичным образом считает, что хост *A* с ним в одной сети. Хосту *B* необходимо узнать MAC-адрес хоста *A*. Он отправляет ARP-запрос в сеть.
6. Маршрутизатор получает ARP-запрос, но не перенаправляет его, так как получатель в другой сети. Если на маршрутизаторе включен Proxy ARP, то маршрутизатор отправляет хосту *B* ARP-ответ, в котором подставляет свой MAC-адрес. То есть, для хоста *B*, создается соответствие `10.0.1.10` — MAC `f0/1`.

Протокол RARP

RARP (Reverse Address Resolution Protocol — Обратный протокол преобразования адресов)

Протокол сетевого уровня модели OSI, выполняет обратное отображение адресов, то есть преобразует физический адрес в IP-адрес. Используется для систем, не имеющих диска, таких как X терминалы или бездисковые рабочие станции для определения собственного IP адреса. RARP является дополнением к ARP, и описан в RFC 903. Формат пакета RARP практически идентичен пакету ARP. С точки зрения выполняемых функций, RARP является скорее аналогом DHCP/BOOTP.

Протокол применяется во время загрузки узла (например компьютера), когда он посылает групповое сообщение-запрос со своим физическим адресом. Сервер принимает это сообщение и просматривает свои таблицы (либо перенаправляет запрос куда-либо ещё) в поисках IP-адреса, соответствующего физическому. После обнаружения найденный адрес отсылается обратно на запросивший его узел. Другие станции также могут «слышать» этот диалог и локально сохранить эту информацию в своих ARP-таблицах.
RARP позволяет разделять IP-адреса между не часто используемыми хост-узлами. После использования каким-либо узлом IP-адреса он может быть освобождён и выдан другому узлу.
При работе с бездисковыми PC RARP служит также для передачи ссылки на образ системы в сети.

Несколько RARP серверов в сети

Оособенность протокола заключается в том, что RARP запросы посылаются в виде широковещательных запросов аппаратного уровня. Это означает, что они не перенаправляются маршрутизаторами. Чтобы позволить бездисковым системам загружаться, даже если RARP сервер выключен, в сети обычно существуют несколько RARP серверов.
По мере того как количество серверов растет (чтобы повысить надежность), увеличивается сетевой траффик, так как каждый сервер посылает RARP отклик на каждый RARP запрос. Бездисковые системы, которые посылают RARP запросы, обычно используют первый полученный ими RARP отклик. Более того, существует вероятность, что несколько RARP серверов отправят отклики одновременно, увеличивая тем самым количество коллизий в Ethernet.

Источники

- Материалы с сайта `https://wikipedia.org/`
- Материалы с сайта `https://www.rfc-editor.org/`
- Telecommunication technologies — телекоммуникационные технологии / Ю. А. Семенов. URL: `http://book.itep.ru/`
- Proxy ARP. URL: `http://xgu.ru/wiki/Proxy_ARP`

Сети связи
Протоколы, сервисы и услуги в Интернет и IP-сетях

Тема № 6
Протокол ICMP

доц. каф. СС и ПД, к.т.н. С. С. Владимиров

2020 г.

Протокол ICMP

ICMP (Internet Control Message Protocol — протокол межсетевых управляющих сообщений)

Сетевой протокол, входящий в стек протоколов TCP/IP. В основном ICMP используется для передачи сообщений об ошибках и других исключительных ситуациях, возникших при передаче данных, например, запрашиваемая услуга недоступна, или хост, или маршрутизатор не отвечают. Также на ICMP возлагаются некоторые сервисные функции.

Протокол ICMP описан в RFC 792 (с дополнениями в RFC 950, RFC 4884, RFC 6633, RFC 6918) и является стандартом Интернета (STD 5). Протокол ICMP является неотъемлемой частью IP и обязателен при реализации стека TCP/IP. ICMP-пакеты для передачи инкапсулируются в IP-пакеты. Код протокола `0x01`. Текущая версия ICMP для IPv4 называется ICMPv4. В IPv6 существует аналогичный протокол ICMPv6.

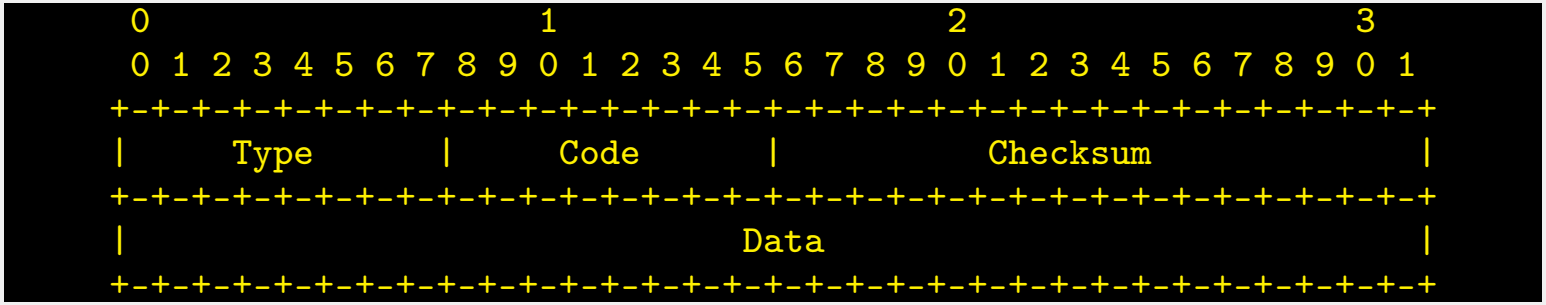
Каждое ICMP-сообщение инкапсулируется непосредственно в пределах одного IP-пакета, и, таким образом ICMP является т. н. «ненадежным» (не контролирующим доставку и её правильность). В отличие от UDP, где реализация надёжности возложена на ПО прикладного уровня, ICMP (в силу специфики применения) обычно не нуждается в реализации надёжной доставки. Тот же Ping, например, служит как раз для проверки потерь IP-пакетов на маршруте.

Правила генерации ICMP-пакетов

- При потере ICMP-пакета никогда не генерируется новый.
- ICMP-пакеты никогда не генерируются в ответ на IP-пакеты с широковещательным или групповым адресом, чтобы не вызывать перегрузку в сети (т. н. «широковещательный шторм»).
- При повреждении фрагментированного IP-пакета ICMP-сообщение отправляется только после получения первого повреждённого фрагмента, поскольку отправитель всё равно повторит передачу всего IP-пакета целиком.

Формат ICMP-пакета

Формат ICMP-пакета



Type (тип сообщения) и Code (код сообщения)

Эти поля определяют назначение ICMP-пакета (сообщения) и формат поля данных (Data). Поле Type определяет общее назначение сообщения, а поле Code конкретизирует его. Некоторые ICMP-сообщения, предложенные изначально в RFC 792 и некоторых других, впоследствии были признаны устаревшими (не используемыми) и отменены в RFC 6918.

Checksum (контрольная сумма)

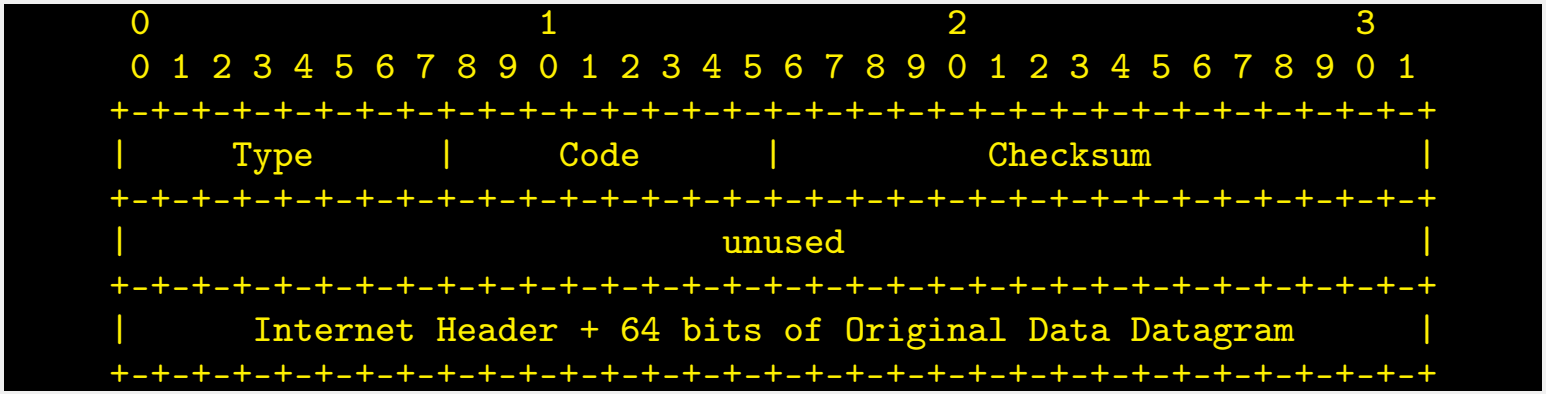
16-битная контрольная сумма, используемая для проверки целостности ICMP-сообщения. Представляет собой 16-битовое поразрядное дополнение суммы всех 16-битовых слов ICMP-сообщения, начиная с поля Type и включая все данные. При вычислении контрольной суммы значение самого поля принимается нулевым.

Data (данные)

Значение этого поля зависит от типа сообщения. Например, для сообщений об ошибках это поле содержит заголовок IP-пакета, вызвавшего ошибку, и первые 64 бита данных.

ICMP-сообщения «Адресат недоступен»

Формат сообщения



Значения полей

`Type` 3

`Code` 0 — сеть недоступна

1 — хост недоступен

2 — неверный протокол

3 — порт закрыт

4 — требуется фрагментация

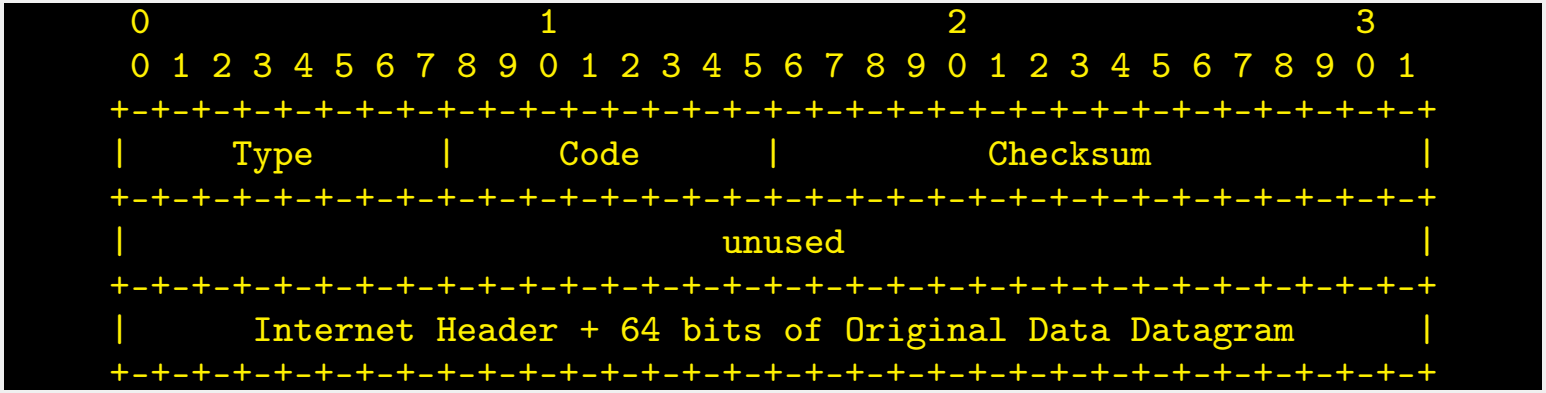
5 — необходима фрагментация, но установлен флаг её запрета (DF)

...

`Data` Содержит 4 неиспользуемых (нулевых) байта, заголовок IP-пакета, вызвавшего ошибку, и первые 64 бита данных.

ICMP-сообщения «Время истекло»

Формат сообщения



Значения полей

`Type` 11

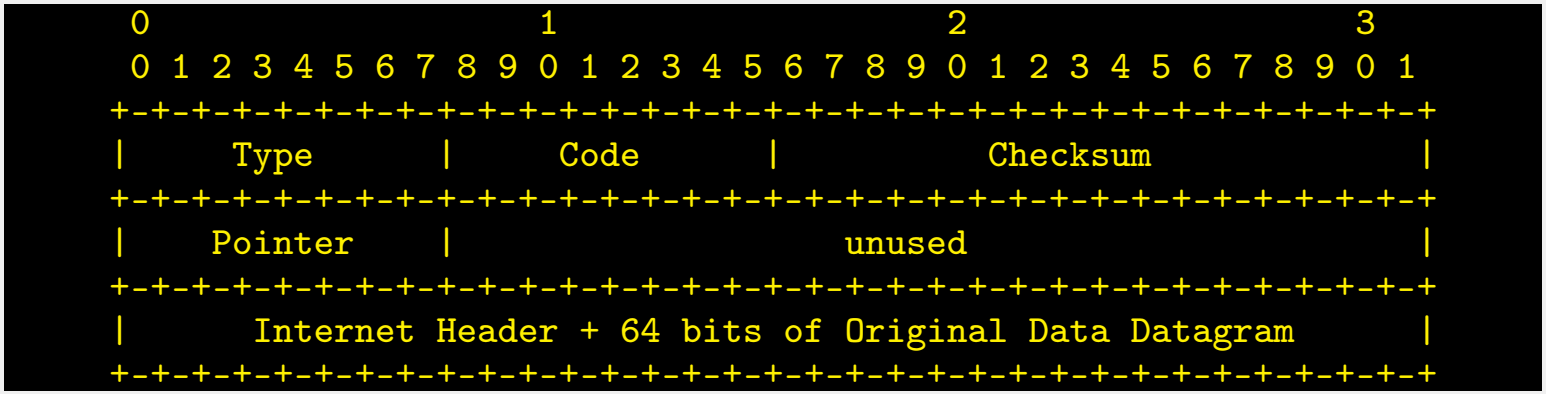
`Code` 0 — время жизни (TTL) истекло

1 — истекло время восстановления пакета при фрагментации

`Data` Содержит 4 неиспользуемых (нулевых) байта, заголовок IP-пакета, вызвавшего ошибку, и первые 64 бита данных.

ICMP-сообщение «Неверный параметр»: ошибка в IP-заголовке

Формат сообщения



Значения полей

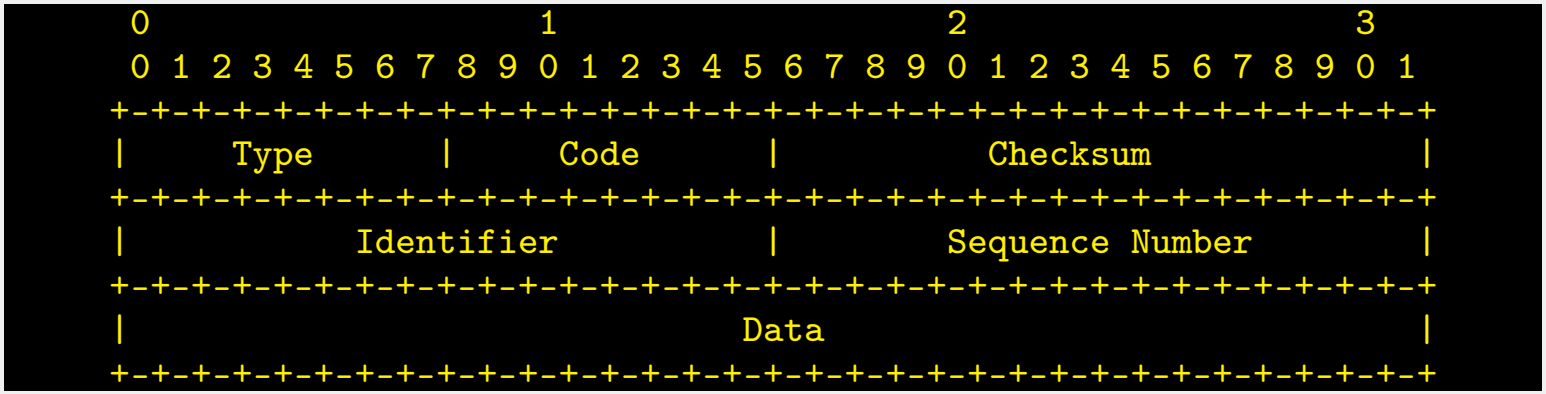
`Type` 12

`Code` 0 — место ошибки указано в поле `Pointer`

`Data` Если `Code=0`, то в поле `Pointer` указывается октет, в котором отмечена ошибка. Далее следуют 3 неиспользуемых (нулевых) байта, заголовок IP-пакета, вызвавшего ошибку, и первые 64 бита данных.

ICMP-сообщения «Эхо-запрос» и «Эхо-ответ»

Формат сообщения



Значения полей

`Type` 12

`Code` 0 — место ошибки указано в поле `Pointer`

`Identifier` Идентификатор последовательности эхо-запросов/эхо-ответов.

`Sequence Number` Номер пакета в последовательности эхо-запросов/эхо-ответов.

`Data` Набор случайных данных для увеличения размера пакета. Зависит от используемого ПО.

ICMP туннелирование

ICMP туннель

Скрытый канал для передачи данных, организованный между двумя узлами, использующий IP-пакеты с типом протокола ICMP (обычно echo request, echo reply). Используется для обхода запретов на передачу информации на межсетевых экранах.

Принцип работы

Узлы обмениваются сообщениями echo request/echo reply, напоминающими работу утилиты ping, однако содержимое сообщений является информацией, передаваемой внутри канала. В случае, если оба узла имеют возможность принимать/отправлять запросы, передача может осуществляться любым узлом, в случае, если один из узлов находится за NAT, он может только отправлять запросы (и получать ответы).

Утилита PingTunnel

Утилита PingTunnel, разработанная Дэниелом Стодлом, позволяет организовать TCP-соединение поверх протокола ICMP или 53 UDP порта.
Для работы PingTunnel необходим запуск прокси-процесса на удаленной машине, имеющей выход в сеть. Пример команд показан ниже.

Прокси: `ptunnel -x пароль`

Клиент: `ptunnel -p хост_прокси -lp лок_порт_туннеля -da адрес_назн -dp порт_назн -x пароль`

Для доступа к адресу и порту назначения необходимо обратиться к заданному порту локального компьютера на loopback интерфейс.

Источники

- Материалы с сайта `https://wikipedia.org/`
- RFC 792. Internet Control Message Protocol
- RFC 6918. Formally Deprecating Some ICMPv4 Message Types
- Проброс TCP соединения через ICMP туннель. URL: `https://www.opennet.ru/tips/1896_proxy_icmp_tunnel.shtml`
- Ping Tunnel. URL: `http://www.cs.uit.no/~daniels/PingTunnel/`

Сети связи
Протоколы, сервисы и услуги в Интернет и IP-сетях

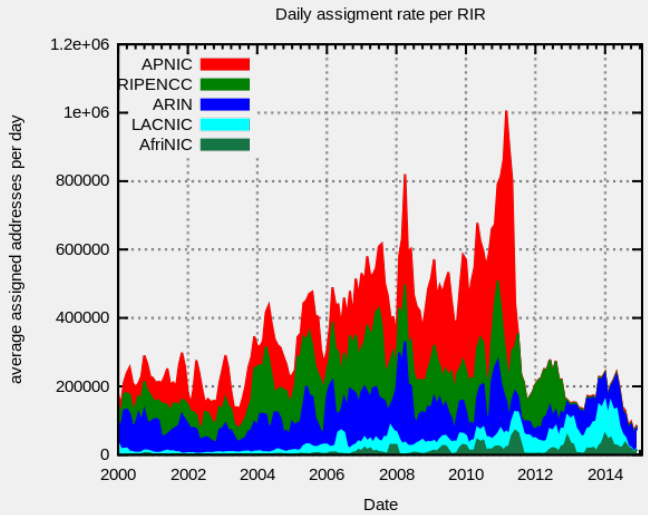
Тема № 7
Протокол IPv6

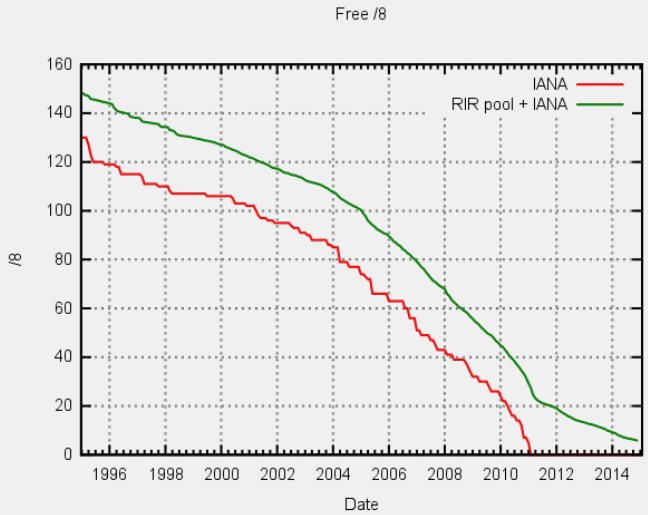
доц. каф. СС и ПД, к.т.н. С. С. Владимиров

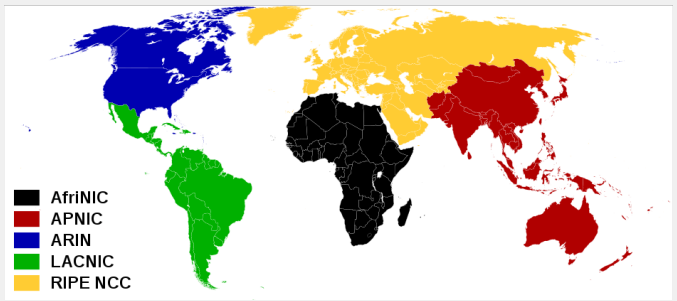
2020 г.

IPv6 (Internet Protocol version 6)

Новая версия протокола IP, использующая длину адреса 128 бит вместо 32. Основной причиной появления IPv6 стало понимание того, что адресное пространство IPv4 рано или поздно будет исчерпано. Первые варианты новых сетевых протоколов появились в 1992. Тогда же IETF объявила конкурс для рабочих групп на создание интернет-протокола следующего поколения (IP Next Generation — IPng). 25 июля 1994 года IETF утвердила модель IPng, с образованием нескольких рабочих групп IPng. Первым документом, определяющим спецификации IPv6, стал RFC 1883, опубликованный в декабре 1995 года. IETF назначила новому протоколу версию 6, так как версия 5 была ранее назначена экспериментальному протоколу, предназначенному для передачи видео и аудио. Позднее, в 1998, RFC 1883 был заменен на RFC 2460 и в дальнейшем дополнялся другими RFC.



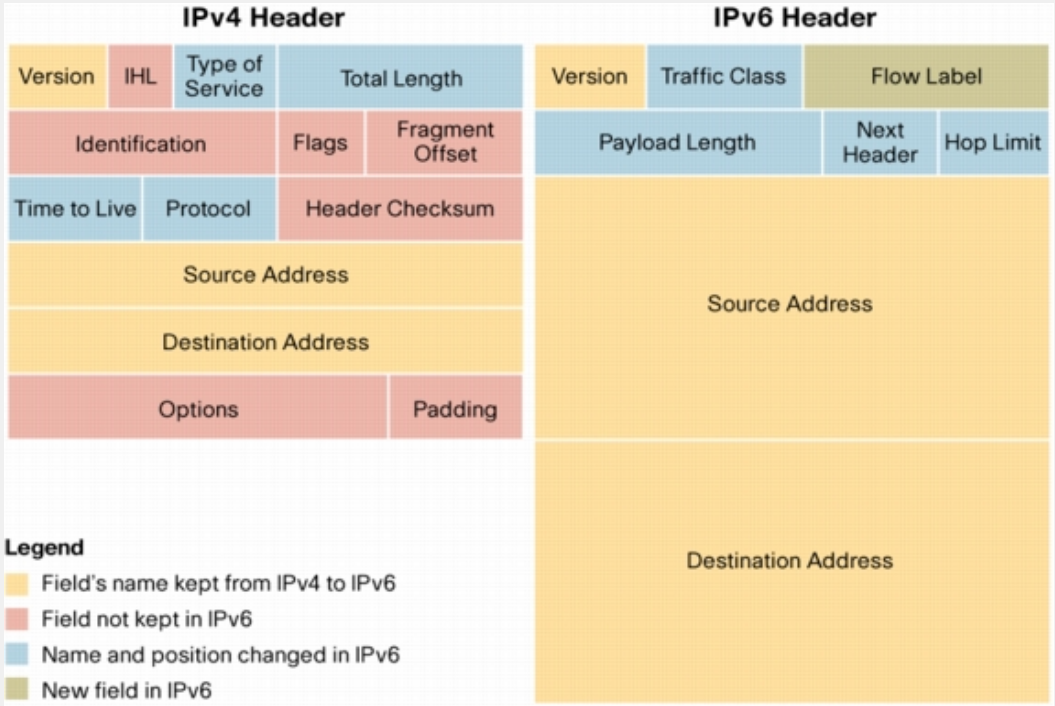




На рисунках показано распределение адресов IPv4 региональными интернет-регистраторами (RIR), а также процесс исчерпания пула IPv4 сетей с маской `/8`.

Формат пакета IPv6

Заголовок IPv6 в сравнении с IPv4



Одним из основных конструктивных улучшений протокола IPv6 по сравнению с IPv4 является упрощённый заголовок IPv6. Заголовок IPv4 состоит из 20 октетов (до 60 байт, если используется поле «Параметры») и 12 основных полей заголовка, не учитывая поля «Параметры» и «Заполнитель». Заголовок IPv6 состоит из 40 октетов (главным образом из-за длины адресов IPv6 источника и назначения) и 8 полей заголовков (3 основных поля заголовков IPv4 и 5 дополнительных полей). Кроме того, в IPv6 добавлено новое поле, которое не используется в протоколе IPv4.

Упрощённый заголовок IPv6 предлагает ряд преимуществ по сравнению с IPv4: повышенная эффективность маршрутизации для масштабируемости производительности и скорости пересылки; не требуется обработка контрольных сумм (выигрыш в том, что не требуется пересчитывать CS при уменьшении TTL); упрощённые и более эффективные механизмы заголовков расширений (в отличие от поля «Параметры» в IPv4); поле «Метка потока» предназначена для обработки по потокам без необходимости открывать транспортный внутренний пакет для определения различных потоков трафика.

Поля заголовка IPv6

Версия (Version)

Поле, содержащее 4-битное двоичное значение, которое определяет версию IP-пакета. Для пакетов IPv6 в этом поле всегда указано значение `0b0110`, т. е. 6.

Класс трафика (Traffic Class)

8-битное поле, соответствующее полю «Type of service» в заголовке IPv4. Оно содержит 6-битное значение точки кода дифференцированных сервисов (DSCP), которое используется для классификации пакетов, а также 2-битное значение явного уведомления о перегрузке (ECN), используемое для управления перегрузками трафика.

Метка потока (Flow Label)

20-битное поле, предоставляющее специальную службу для приложений реального времени. Используя это поле, маршрутизаторам и коммутаторам передается информация о необходимости поддерживать один и тот же путь для потока пакетов, что поможет избежать их переупорядочивания.
Метка потока присваивается узлом-отправителем путём генерации псевдослучайного 20-битного числа. Все пакеты одного потока должны иметь одинаковые заголовки, обрабатываемые маршрутизатором.
При получении первого пакета с меткой потока маршрутизатор анализирует дополнительные заголовки, выполняет предписанные этими заголовками функции и запоминает результаты обработки (адрес следующего узла, опции заголовка переходов, перемещение адресов в заголовке маршрутизации и т. д.) в локальном кэше. Ключом для такой записи является комбинация адреса источника и метки потока. Последующие пакеты с той же комбинацией адреса источника и метки потока обрабатываются с учётом информации кэша без детального анализа всех полей заголовка. Время жизни записи в кэше составляет не более 6 секунд, даже если пакеты этого потока продолжают поступать. При обнулении записи в кэше и получении следующего пакета потока пакет обрабатывается в обычном режиме, и для него происходит новое формирование записи в кэше. Указанное время жизни потока может быть явно определено узлом отправителем с помощью протокола управления или опций заголовка переходов и может превышать 6 секунд.

Поля заголовка IPv6

Длина полезной нагрузки (Payload Length)

16-битное поле, соответствующее полю «Общая длина» в заголовке IPv4. Оно определяет размер всего пакета (фрагмента) после заголовка, включая дополнительные расширения.

Следующий заголовок (Next Header)

8-битное поле, соответствующее полю «Протокол» в заголовке IPv4. Оно указывает тип полезной нагрузки данных, которые переносит пакет, что позволяет сетевому уровню пересылать данные на соответствующий протокол более высокого уровня. Это поле также используется в тех случаях, когда в пакет IPv6 добавляются дополнительные заголовки расширений.

Предел перехода (Hop Limit)

8-битное поле, заменяющее поле «Время жизни» (TTL) в IPv4. Это значение уменьшается на единицу каждым маршрутизатором, пересылающим пакет. Когда счетчик достигает 0, пакет отбрасывается, и на отправляющий узел пересылается сообщение ICMPv6, которое означает, что пакет не достиг своего назначения.

Адрес источника (Source Address)

128-битовое поле, определяющее IPv6-адрес принимающего узла.

Адрес назначения (Destination Address)

128-битное поле, определяющее IPv6-адрес принимающего узла.

Расширенные заголовки (Extension headers)

Расширенные заголовки (или *заголовки расширений*) содержат дополнительную информацию и размещены между фиксированным заголовком и заголовком протокола более высокого уровня. Тип первого расширенного заголовка указывается в поле Next Header фиксированного заголовка, а каждый расширенный заголовок имеет аналогичное поле в котором хранится тип следующего расширенного заголовка. В поле Next Header последнего заголовка находится тип протокола более высокого уровня, находящегося в качестве полезных данных.
Каждый расширенный заголовок должен иметь размер в октетах, кратный 8. Некоторые заголовки необходимо расширить до нужного размера.
Расширенные заголовки должны быть обработаны только конечным узлом, за исключением заголовка Hop-By-Hop Options, который должен быть обработан каждым промежуточным узлом на пути пакета, включая отправителя и получателя. Если расширенных заголовков в пакете несколько, то рекомендуется отсортировать их как указано в таблице ниже. Отметим, что все расширенные заголовки являются необязательными и не должны появиться в пакете более одного раза, за исключением заголовка Destination Options, который может появиться дважды.
Если узел не может обработать какой-то расширенный заголовок, то он должен отбросить пакет и отправить сообщение Parameter Problem (ICMPv6 тип 4, код 1). Если в поле Next Header расширенного заголовка будет 0, то узел должен сделать то же самое.

Некоторые расширенные заголовки

Расширенный заголовок	**Тип**	**Описание**
Hop-by-Hop Options	0	Параметры, которые должны быть обработаны каждым транзитным узлом.
Destination Options	60	Параметры которые должны быть обработаны только получателем.
Routing	43	Позволяет отправителю определять список узлов, которые пакет должен пройти.
Fragment	44	Заголовок содержит информацию по фрагментации пакета.
Authentication Header (AH)	51	Содержит информацию, для аутентификацию большей части пакета.
Encapsulating Security Payload (ESP)	50	Осуществляет шифрование данных для безопасных подключений.

Определение размера пакета в IPv6

Маршрутизаторы IPv6 больше не должны фрагментировать пакет, вместо этого пакет отбрасывается с ICMP-уведомлением о превышении MTU. Передающая сторона в IPv6, таким образом, должна или использовать технологию Path MTU discovery (RFC 1191), или использовать минимальный MTU. Для лучшей работы протоколов, требовательных к потерям, минимальный MTU поднят до 1280 байт. Фрагментация поддерживается как опция (информация о фрагментации пакетов вынесена из основного заголовка в расширенные) и возможна только по инициативе передающей стороны.

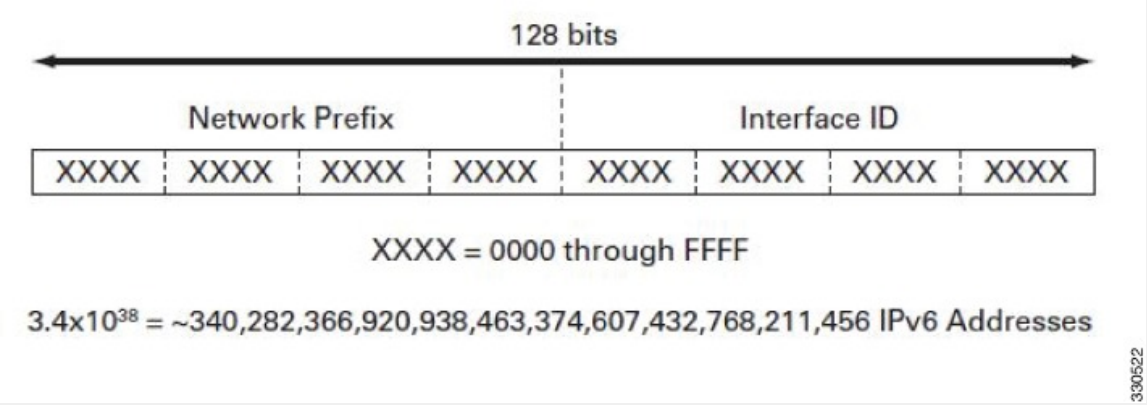
Термин Path MTU означает наименьший MTU на пути следования пакета в сети.

Узел, значение MTU которого меньше размера пакета, отклоняет передачу пакета и отправляет сообщение ICMP «необходима фрагментация, но установлен флаг её запрета (Don't Fragment)». Хост-отправитель уменьшает размер пакета и отсылает его заново. Такая операция происходит до тех пор, пока пакет не будет достаточно мал, чтобы дойти до хоста-получателя без фрагментации.

Существуют потенциальные проблемы. Некоторые маршрутизаторы настраиваются администраторами на полное блокирование ICMP пакетов. В результате, если размер пакета не соответствует значению MTU на определённом участке, пакет отбрасывается, а хост-отправитель не может получить информацию о значении MTU и не отправляет пакет заново. Поэтому соединение между хостами не устанавливается. Проблема получила название MTU Discovery Black Hole (RFC 2923), и протокол был модифицирован для детектирования таких маршрутизаторов.

Существует несколько вариантов решения этой проблемы. Самым простым является отмена фильтрации пакетов ICMP. Поскольку зачастую подобная операция находится вне компетенции пользователя, проблему решают путем ручной настройки размера передаваемого пакета на шлюзе пользователя. Для этого меняют значение MSS (максимальный размер сегмента, то есть величина, меньшая MTU на 40 байт в случае протокола IPv4). При установке соединения хосты обмениваются информацией о максимальном размере сегмента, который каждый из них сможет принять. Поэтому, меняя значение MSS, заставляют оба хоста обмениваться пакетами, которые шлюз пользователя сможет заведомо принять без фрагментации.

Адрес IPv6



Длина IPv6-адресов составляет 128 бит, написанных в виде строки шестнадцатеричных значений. 4 бита представлены одной шестнадцатеричной цифрой. Всего 32 цифры. Адресация IPv6 определена в RFC 4291.

Правила сокращенного представления IPv6 адресов

```
2001:0db8:85a3:0000:0000:8a2e:0370:7334
```

1. Пропуск всех ведущих 0 в шестнадцатеричной записи

```
01AB -> 1AB     09F0 -> 9F0     0A00 -> A00     00AB -> AB
```

```
2001:db8:85a3:0:0:8a2e:370:7334
```

Это правило применяется только к ведущим нулям, а НЕ к последующим, иначе адрес будет записан неясно.

2. Двойное двоеточие (::) может заменить любую единую, смежную строку одного или нескольких 16-битных сегментов (хекстетов), состоящих из нулей. Двойное двоеточие (::) может использоваться в адресе только один раз.

```
2001:db8:85a3::8a2e:370:7334
```

Неверный адрес:

```
2001:0DB8::ABCD::1234
```

Типы IPv6-адресов. Unicast адреса

Классификация по способу адресации

- Unicast адреса идентифицируют только один сетевой интерфейс. Протокол IPv6 доставляет пакеты, отправленные на такой адрес, на конкретный интерфейс. Существует шесть типов Unicast адресов
- Anycast адреса назначаются группе интерфейсов, обычно принадлежащих различным узлам. Пакет, отправленный на такой адрес, доставляется на один из интерфейсов данной группы, как правило наиболее близкий к отправителю с точки зрения протокола маршрутизации.
- Multicast адрес также используется группой узлов, но пакет, отправленный на такой адрес, будет доставлен каждому узлу в группе. Различают два типа Multicast адресов.

В IPv6 не реализованы широковещательные адреса. Традиционная роль широковещательной рассылки реализована с помощью групповой рассылки на адрес `ff02::1`, однако использование не рекомендуется.

Global unicast

Global unicast адрес мало чем отличается от публичного IPv4-адреса. Эти адреса, к которым можно проложить маршрут по Интернету, являются уникальными по всему миру. Глобальные индивидуальные адреса могут быть настроены статически или присвоены динамически.

Loopback

Loopback-адрес используется узлом для отправки пакета самому себе и не может быть назначен физическому интерфейсу. Как и на loopback-адрес IPv4, для проверки настроек TCP/IP на локальном узле можно послать эхо-запрос на loopback-адрес IPv6. Loopback-адрес IPv6 состоит из нулей, за исключением последнего бита, который выглядит как `::1/128` или просто `::1` в сжатом формате.

Unicast адреса

Link-local

Local IPv6-адрес канала позволяет устройству обмениваться данными с другими устройствами под управлением IPv6 по одному и тому же каналу и только по данному каналу (подсети). Пакеты с локальным адресом канала источника или назначения не могут быть направлены за пределы того канала, в котором пакет создаётся. В отличие от локальных IPv4-адресов канала, локальные адреса канала IPv6 играют важную роль в различных аспектах сети. Глобальный индивидуальный адрес не обязателен. Однако для содержания локального адреса канала необходим сетевой интерфейс под управлением протокола IPv6. Если локальный адрес канала не настроен вручную на интерфейсе, устройство автоматически создаёт собственный адрес, не обращаясь к DHCP-серверу. Узлы под управлением IPv6 создают локальный IPv6-адрес канала даже в том случае, если устройству не был назначен глобальный IPv6-адрес. Это позволяет устройствам под управлением IPv6 обмениваться данными с другими устройствами под управлением IPv6 в одной подсети, в том числе со шлюзом по умолчанию (маршрутизатором). Локальные IPv6-адреса канала находятся в диапазоне `FE80::/10`.

Unspecified address

Неопределённый адрес состоит из нулей и в сжатом формате представлен как `::/128` или просто `::` Он не может быть назначен интерфейсу и используется только в качестве адреса источника в IPv6-пакете. Неопределённый адрес используется в качестве адреса источника, когда устройству еще не назначен постоянный IPv6-адрес или когда источник пакета не относится к месту назначения.

Unicast адреса

Unique local

Unique local IPv6-адреса (RFC 4193) имеют некоторые общие особенности с частными («серыми») адресами для IPv4, но при этом между ними имеются и значительные различия. Уникальные локальные адреса используются для локальной адресации в пределах узла или между ограниченным количеством узлов. Эти адреса не следует маршрутизировать в глобальном протоколе IPv6. Уникальные локальные адреса находятся в диапазоне от `FC00::/7` до `FDFF::/7`. В случае с IPv4 частные адреса объединены с преобразованием сетевых портов и адресов (NAT/PAT) для обеспечения преобразования адресов из частных в публичные. Это делается из-за недостатка адресного пространства IPv4. Во многих сетях также используют частный характер адресов RFC 1918, чтобы обеспечить безопасность или защитить сеть от потенциальных угроз. Однако такая мера никогда не была целью использования данных технологий, и организация IETF всегда рекомендовала предпринимать правильные меры предосторожности при работе маршрутизатора в Интернете. Хотя протокол IPv6 обеспечивает особую адресацию для сайтов, он не предназначен для того, чтобы скрывать внутренние устройства под управлением IPv6 от Интернета IPv6. IETF рекомендует ограничивать доступ к устройствам с помощью наилучших мер безопасности.

IPv4 embedded

Встроенные IPv4-адреса. Использование этих адресов способствует переходу с протокола IPv4 на IPv6. Эти адреса определены в RFC 6052. Выделяют так называемый *IPv4 совместимый IPv6 адрес* вида `::FFFF:xx.xx.xx.xx/96`, в котором нижние 32 бита это адрес IPv4. Устарел и больше не используется. Также выделяют *адрес IPv4, отображённый на IPv6* вида `::xx.xx.xx.xx/96`.

Global unicast IPv6

Global unicast IPv6-адреса уникальны по всему миру и доступны для маршрутизации через Интернет IPv6. Эти адреса эквивалентны публичным IPv4-адресам. В настоящее время назначаются только глобальные индивидуальные адреса с первыми тремя битами `0b001` или `2000::/3`. Это лишь 1/8 от всего доступного адресного пространства IPv6. Адрес `2001:0DB8::/32` был зарезервирован для документации, в том числе для использования в примерах.

Структура Global unicast адреса

- Префикс глобальной маршрутизации — Префикс глобальной маршрутизации — это префиксальная или сетевая часть адреса, назначаемая интернет-провайдером заказчику или узлу. В настоящее время `/48` является префиксом глобальной маршрутизации, который в настоящее время интернет-регистраторы назначают своим заказчикам — корпоративным сетям и индивидуальным пользователям. Этого адресного пространства более чем достаточно для большинства заказчиков.
- Идентификатор подсети — Идентификатор подсети используется организациями для обозначения подсетей в каждом узле.
- Идентификатор интерфейса — Идентификатор IPv6-интерфейса эквивалентен узловой части адреса IPv4-адреса. Термин «идентификатор интерфейса» используется в том случае, когда один узел может иметь несколько интерфейсов, каждый из которых обладает одним или более IPv6-адресами.

Процесс EUI-64

Организация IEEE разработала расширенный уникальный идентификатор (EUI) или изменённый процесс EUI-64. Этот процесс использует 48-битный MAC-адрес Ethernet клиента и в середину этого адреса вставляет ещё 16 бит для создания 64-битного идентификатора интерфейса. Преимущество EUI-64 MAC-адреса Ethernet заключается в том, что его можно использовать для определения идентификатора интерфейса. Кроме того, сетевые администраторы могут легко отслеживать IPv6-адрес до конечных устройств с помощью уникального MAC-адреса.
Однако, именно возможность отследить как пакеты устройства, так и перемещение самого устройства между сетями привела к тому, что были высказаны опасения о нарушении приватности пользователей, а также ою уменьшении уровня безопасности сети. Соответственно, современные ОС на конечных устройствах генерируют идентификатор интерфейса случайным образом.

1 шаг: EUI-48 $\longrightarrow$ EUI-64

```
02:0C:29:0C:47:D5 ==> 02:0C:29:FF:FE:0C:47:D5
```

2 шаг: инверсия бита Unique/Local

```
02:0C:29:FF:FE:0C:47:D5 ==> 00:0C:29:FF:FE:0C:47:D5
```

Пример получения адреса IPv6 из локального MAC-адреса

```
02:00:00:00:00:01 ==> FE80::FF:FE00:1
```

Multicast IPv6. Assigned multicast

Multicast IPv6-адреса мало чем отличаются от multicast IPv4-адресов. Multicast адрес используется для отправки одного пакета по одному или нескольким назначениям (группе мультивещания). Multicast IPv6-адреса имеют префикс `FF00::/8`. Multicast адреса могут быть только адресами назначения, а не адресами источника. Существует два типа:

1. Назначенные (присвоенные) (Assigned multicast)
2. Запрошенные (Solicited multicast)

Присвоенные групповые адреса (Assigned multicast)

Присвоенные групповые адреса зарезервированы для заданных групп устройств. Присвоенный групповой адрес — это один адрес, используемый для осуществления связи с группой устройств, работающих на одном протоколе или сервисе. Присвоенные групповые адреса используются вместе с конкретными протоколами, например с протоколом DHCPv6. Есть две распространённые группы присвоенных групповых IPv6-адресов.

1. Группа мультивещания для всех IPv6-узлов `FF02::1`, к которой подключены все устройства под управлением протокола IPv6. Пакет, отправленный этой группе, получается и обрабатывается всеми IPv6-интерфейсами в канале или сети. Эта группа адресов работает так же, как широковещательный адрес в протоколе IPv4.
2. Группа мультивещания для всех маршрутизаторов `FF02::2`, к которой подключены все IPv6-маршрутизаторы. Пакет для этой группы получается и обрабатывается всеми IPv6-маршрутизаторами в канале или сети.

Групповой IPv6-адрес запрашиваемого узла (Solicited multicast)

Групповой IPv6-адрес запрашиваемого узла создаётся автоматически при назначении глобального индивидуального адреса или локального адреса канала. Групповой IPv6-адрес запрашиваемого узла создаётся посредством объединения специального префикса `FF02:0:0:0:0:1:FF00::/104` с крайними правыми 24 битами его индивидуального адреса.
Очень редко в идентификаторах интерфейса устройств встречаются одинаковые крайние правые 24 бита. Это не влечёт за собой никаких проблем, поскольку устройство по-прежнему будет обрабатывать инкапсулированное сообщение, в котором содержится полный IPv6-адрес запрашиваемого устройства.

Разбиение на подсети

Разбиение IPv6-сети на подсети подразумевает использование другого подхода, чем разбиение на подсети IPv4-сети. Пространство IPv6-адресов разбивается не с целью экономии адресов, а для обеспечения иерархической логической структуры сети. Разбиение на подсети в IPv6 возможно провести двумя вариантами.

Разбиение на подсети с использованием идентификатора подсети

Блок IPv6-адресов с префиксом `/48` содержит 16 бит идентификатора подсети. Разбиение на подсети с использованием 16 бит идентификатора подсети даёт 65536 возможных подсетей `/64`. Поэтому нет необходимости заимствовать биты из идентификатора интерфейса или узловой части адреса. Каждая IPv6-подсеть `/64` содержит примерно $18 \cdot 10^{18}$ адресов, что гораздо больше, чем когда-либо понадобится в одном сегменте IP-сети. Подсети, созданные из идентификатора подсети, легко представить, поскольку не нужно выполнять преобразование в двоичный формат. Чтобы определить следующую доступную подсеть, достаточно рассчитать следующее шестнадцатеричное число. Необходимо применить расчёт части идентификатора подсети в шестнадцатеричной системе счисления. Префикс глобальной маршрутизации является одинаковым для всех подсетей. Для каждой подсети увеличивается только четырёхразрядный идентификатор подсети.

Разбиение на подсети

Разбиение на подсети с использованием идентификатора интерфейса

В IPv6-сетях по аналогии с заимствованием бит из узловой части IPv4-адреса можно позаимствовать биты из идентификатора интерфейса для создания дополнительных IPv6-подсетей. Как правило, это делается по соображениям безопасности, чтобы уменьшить число узлов в подсети и создавать дополнительные подсети. При расширении идентификатора подсети путём заимствования бит из идентификатора интерфейса рекомендуется создавать подсеть на границе полубайта (4 бита или одна шестнадцатеричная цифра). Префикс подсети `/64` расширяется на четыре бита или один полубайт до подсети `/68`. Это позволяет уменьшить размер идентификатора на 4 бита (с 64 до 60). Разбиение на подсети по границе полубайта имеет значение только для масок подсетей, выровненных по полубайту. Начиная с `/64`, масками подсети, выровненными по полубайту, будут являться маски `/68`, `/72`, `/76`, `/80` и т. д. Разбиение на подсети по границе полубайта позволяет создать подсети с использованием дополнительного шестнадцатеричного значения. Можно создать подсеть в пределах полубайта, используя шестнадцатеричную цифру, однако это не рекомендуется и, кроме того, в этом нет необходимости. Разбиение на подсети в пределах полубайта сводит на нет преимущества быстрого определения префикса из идентификатора интерфейса. Например, если используется длина префикса `/66`, первые два бита были бы частью идентификатора подсети, а вторые два бита — частью идентификатора интерфейса.

Источники

- Материалы с сайта `https://wikipedia.org/`
- Материалы с сайта `https://www.rfc-editor.org/`
- RFC 4291. IP Version 6 Addressing Architecture.
- IPV6 — это весело. Часть 1. URL: `https://habrahabr.ru/post/253803/`

Сети связи
Протоколы, сервисы и услуги в Интернет и IP-сетях

Тема № 8
Протокол ICMPv6

доц. каф. СС и ПД, к.т.н. С. С. Владимиров

2020 г.

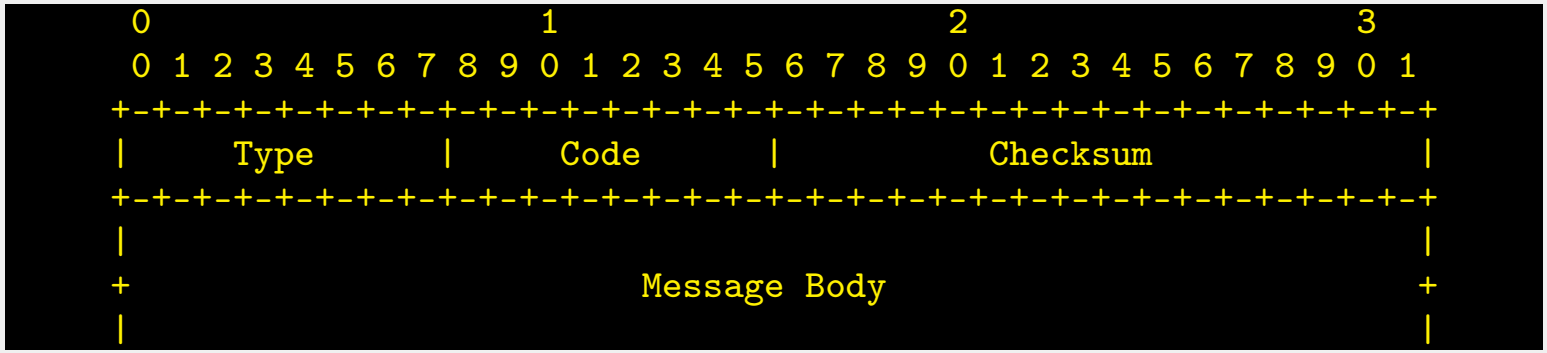
Протокол ICMPv6

Реализация ICMP для IPv6. Неотъемлемая часть IPv6, отвечающая за сообщения об ошибках, диагностические функции (например, ping), поиск соседей, определение MTU и основа для расширения и реализации будущих аспектов управления межсетевым протоколом. Протокол был определен в RFC 1885 (декабрь 1995), который в 1998 был заменен на RFC 2463, а в 2006 на RFC 4443, который и используется по настоящее время.
ICMPv6 сообщения могут быть разделены на две категории:

- Сообщения об ошибках. У них старший бит всегда установлен в «0», то есть тип может принимать значения от 0 до 127.
- Информационные сообщения. Старший бит всегда установлен в «1», то есть тип может принимать значения от 128 до 255.

ICMPv6 сообщения инкапсулированы в пакеты IPv6, с полем Next Header установленным в 58.

Формат пакета ICMPv6



Некоторые сообщения об ошибках

Type:Code

1 Destination Unreachable (RFC 4443)

- 1:0 No route to destination — В таблице маршрутизации нет нужного маршрута
- 1:1 Administratively Prohibited — Пакет попал под запрещающее правило в фильтре-файерволе
- 1:2 Beyond the scope of source address — Пакет попал не на тот интерфейс
- 1:3 Address unreacheable — Не определяется link-layer адрес получателя
- 1:4 Port unreacheable — Получатель не «слушает» на этом порту
- 1:5 Source address failed ingress/egress policy — С таким адресом источника сюда не пускают
- 1:6 Reject route to destination — Пакет попал под действие запрещающего фильтра на роутере

2:0 Packet Too Big. Используется в Path MTU Discovery (RFC 4443)

3 Time Exceeded (RFC 4443)

- 3:0 Time exceeded:Hop limit reached — После обработки пакета поле Hop Limit стало равным нулю.
- 3:1 Fragmentation reassembly timeout — За 60 секунд не удалось собрать пакет из фрагментов

4 Parameter Problem (RFC 4443)

- 4:0 Error in header — Ошибка в заголовке IPv6-пакета или ошибка в расширении заголовка
- 4:1 Unknown next header — В поле Next Header обнаружена ошибка
- 4:2 Unknown option — В заголовке просто что-то непонятное

127 Reserved for expansion of ICMPv6 error messages

Некоторые информационные сообщения

Type:Code

128:0 Echo Request (RFC 4443)

129:0 Echo Reply (RFC 4443)

130:0 Multicast Listener Query (RFC 2710 и RFC 3810)

131:0 Version 1 Multicast Listener Report (RFC 2710)

132:0 Multicast Listener Done (RFC 2710)

133:0 Router Solicitation (RFC 4861)

134:0 Router Advertisement (RFC 4861)

135:0 Neighbor Solicitation (RFC 4861)

136:0 Neighbor Advertisement (RFC 4861)

137:0 Redirect (RFC 4861)

141:0 Inverse Neighbor Discovery Solicitation Message (RFC 3122)

142:0 Inverse Neighbor Discovery Advertisement Message (RFC 3122)

151:0 Multicast Router Advertisement (RFC 4286)

152:0 Multicast Router Solicitation (RFC 4286)

153:0 Multicast Router Termination (RFC 4286)

255:0 Reserved for expansion of ICMPv6 informational messages

Neighbor Discovery Protocol (NDP)

Протокол NDP

Протокол из набора Internet Protocol Suite, используемый совместно с IPv6. Отвечает за автонастройку адреса конечных точек сети, обнаружение других узлов на линии, обнаружение адреса других узлов на уровне канала связи, обнаружение конфликта адресов, поиск доступных путей и DNS-серверов, обнаружение подсетей и поддержку доступности информации о путях к другим активным соседним узлам. Определен в RFC 4861 (2007). Ранее был определен в RFC 1970 (1996) и RFC 2461 (1998). Устанавливает пять различных типов пакета ICMPv6 для выполнения функций IPv6, сходных с ARP, ICMP, Router Discovery и Router Redirect протоколов для IPv4.
NDP используется в процессах:

- определение link-layer адресов (включая проверку на занятость адреса)
- обнаружение роутера
- определение недостижимости хостов
- перенаправление трафика (определение оптимального next-hop)

Сообщения NDP

133:0 Router Solicitation — Запрос на доступность маршрутизаторов
134:0 Router Advertisement — Ответ маршрутизатора
135:0 Neighbor Solicitation — Запрос доступных соседей
136:0 Neighbor Advertisement — Ответ соседа
137:0 Redirect — Перенаправление

Поскольку эти сообщения не должны покидать local-link, у них устанавливается Hop-limit равным 255. Если обнаруживается NDP-пакет с другим значением этого параметра, то он должен быть уничтожен. Роутеры не пересылают такие пакеты.

Протокол NDP

Таблицы протокола NDP, хранящиеся на хосте

1. Neighbor cache — кэш с информацией о соседях в пределах общего линка: IP-адреса, соответствующие им link-layer адреса, индикатор доступности
2. Destination cache — кэш с информацией о адресах, служащих шлюзом для достижения получателей. Содержит информацию: адрес получателя, соотвествующий ему next-hop адрес, значение MTU для данного получателя
3. Prefix list — список префиксов — список префиксов для данного линка. Составляется из RA.
4. Default router list IP addresses — список адресов, которые могут служить шлюзом по умолчанию. Составлен из RA.

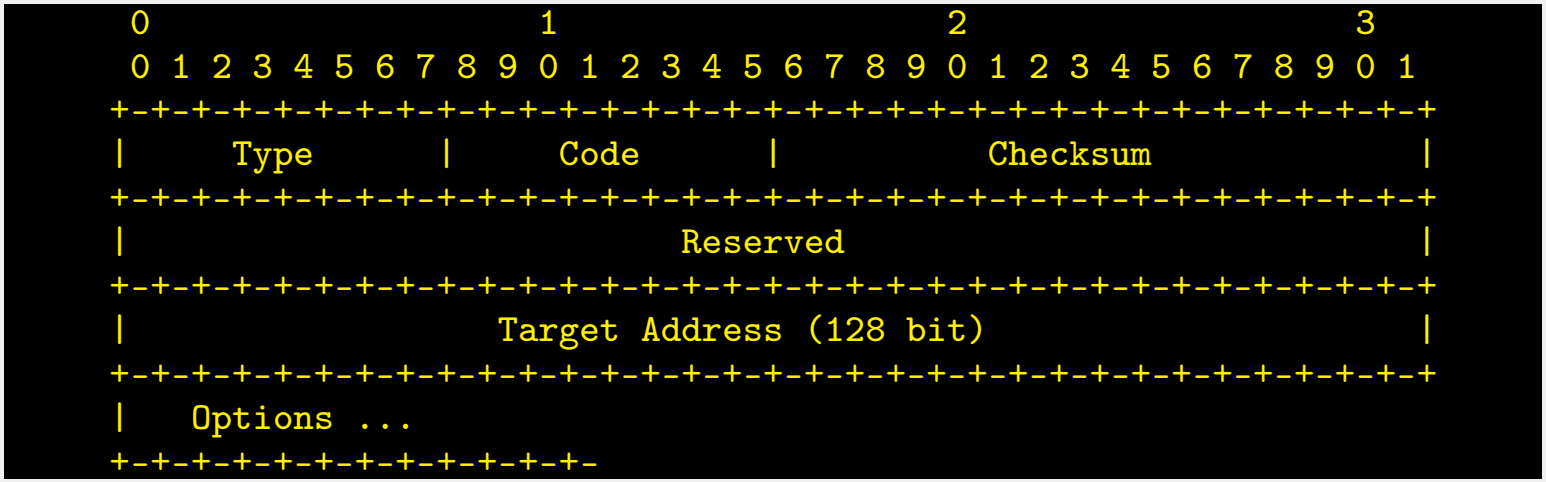
Процедура разрешения адресов

«Разрешение» адресов, то есть получение канального (MAC, link-layer) адреса по сетевому IPv6-адресу производится путем обмена сообщениями Neighbor Solicitation и Neighbor Advertisement (NS-NA). Neighbor Solicitation отправляется на multicast адрес, получатель обновляет свою таблицу соседей, и отвечает unicast пакетом Neighbor Advertisement. Отправитель добавляет данные в свою таблицу соседей (Neighbor cache).

Протокол NDP. Neighbor Solicitation

Neighbor Solicitation

Запрос link-layer address целевого узла и сообщение ему своего link-layer address. Отправляется либо как multicast (при получении адреса), либо как unicast (при верификации).



IPv6 заголовок

Source Address — Адрес интерфейса, с которого отправлено сообщение, или неопределенный адрес (`::/128`)

Destination Address — Solicited-node multicast адрес, соответствующий целевому узлу, или unicast адрес целевого узла

Hop Limit — 255

Протокол NDP. Neighbor Solicitation

ICMP заголовок

Type	— 135
Code	— 0
Reserved	— Неиспользуемое поле. Оно ***должно*** быть проинициализировано нулями отправителем и ***должно*** игнорироваться получателем
Target Address	— IPv6-адрес целевого узла. ***Не должен*** быть multicast-адресом.

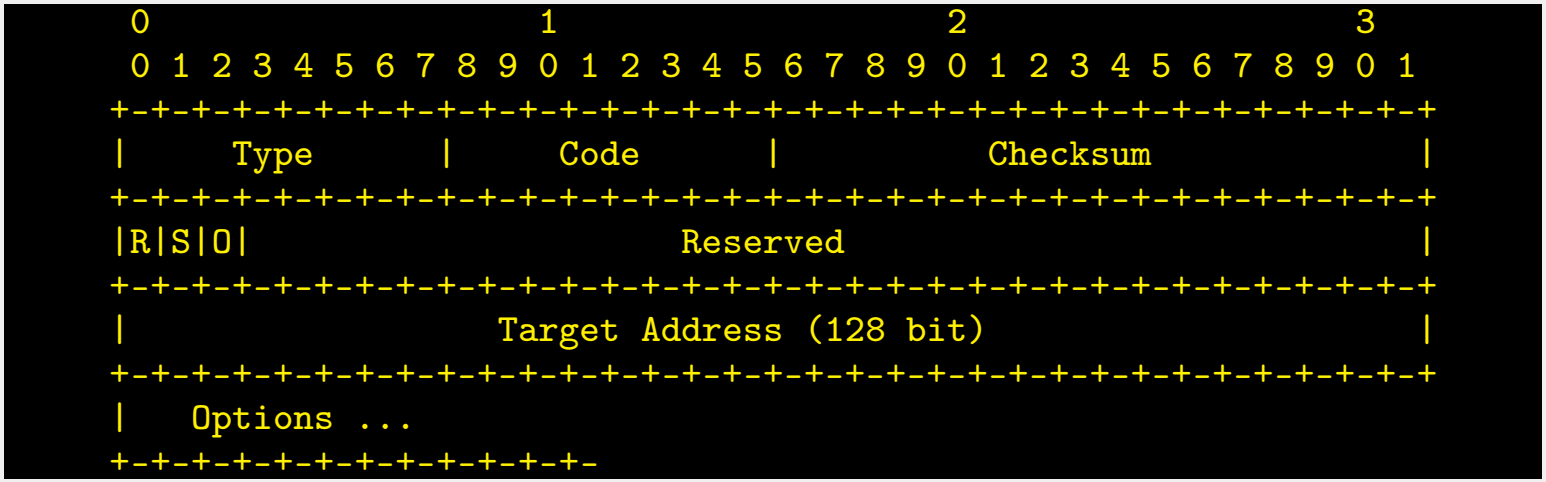
Опции

Source link-layer address	— Канальный (link-layer) адрес отправителя. Опция ***не должна*** использоваться, если IPv6-адрес источника является неопределенным. Если же адрес есть, то опция ***должна*** использоваться при multicast-рассылке запроса и ***рекомендуется*** к использованию при unicast-отправке запроса.

Протокол NDP. Neighbor Advertisement

Neighbor Advertisement

Посылается в ответ на NS или для быстрого оповещения о смене адреса (unsolicited NA).



IPv6 заголовок

Source Address — Адрес интерфейса, с которого отправлен ответ

Destination Address — Для solicited NA указывается значение из поля Source Address соответствующего запроса, либо, если в запросе был задан неопределенный адрес, указывается All-nodes multicast. Для unsolicited NA указывается All-nodes multicast.

Hop Limit — 255

Протокол NDP. Neighbor Advertisement

ICMP заголовок

Type	— 136
Code	— 0
Router flag	— Установлен в 1, если отправителем является роутер. У хоста установлен в 0.
Solicited flag	— Установлен в 1, если это ответ на запрос NA. Используется как индикатор достижимости.
Override flag	— Если установлен в 1, то следует переписать link-layer адрес на тот, что в пакете.
Target Address	— Для solicited NA указывается то же значение, что и в поле Target Address NS-запроса. Для unsolicited NA указывается адрес узла, чей канальный адрес меняется. ***Не должен*** быть multicast-адресом.

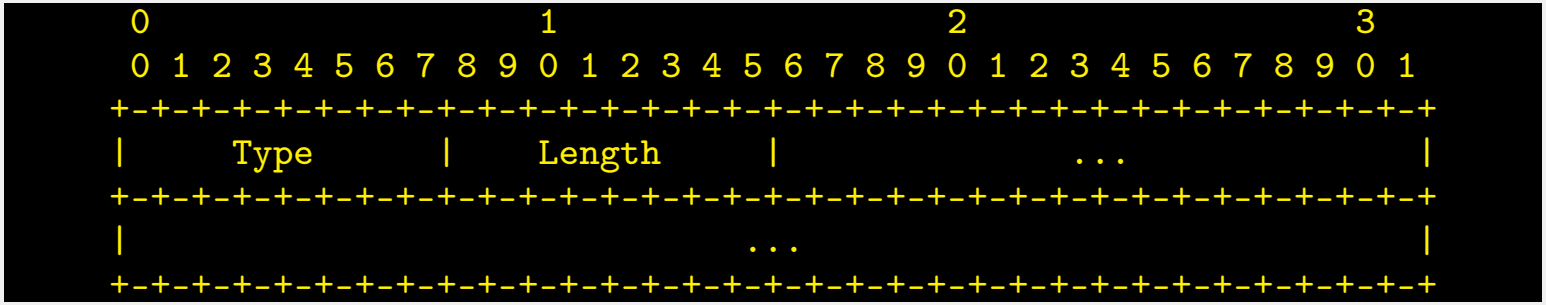
Опции

Target link-layer address	— Канальный (link-layer) адрес цели, т. е. отправителя NA. Опция ***должна*** использоваться в ответе на multicast-запрос. Если NA отправляется в ответ на unicast-запрос, то опцию ***рекомендуется*** использовать.

Протокол NDP. Формат опций

Формат опций

Размер блока опций должен быть кратен 64 бит (8 октетов).



Type

8-битный идентификатор типа опции.

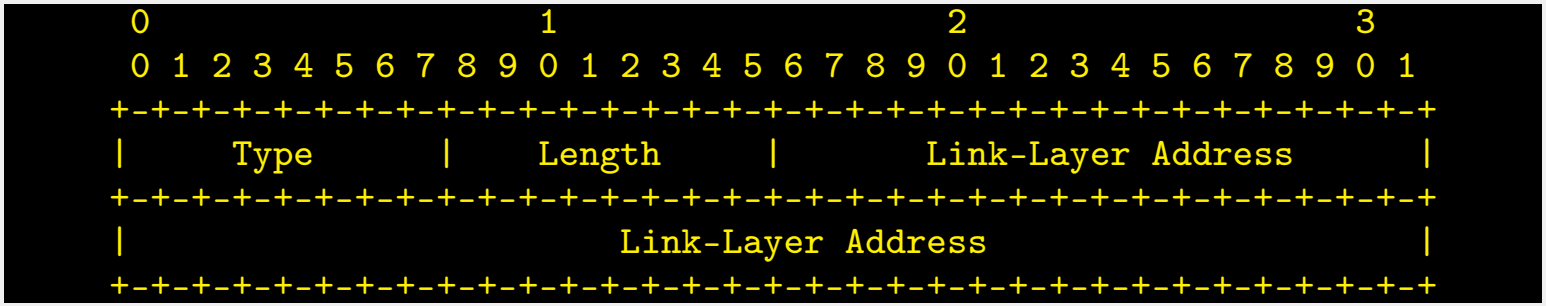
- Source Link-Layer Address — 1
- Target Link-Layer Address — 2
- Prefix Information — 3
- Redirected Header — 4
- MTU — 5

Length

Длина опции, включая поля Type и Length в словах по 8 октетов. ***Не должна*** равняться 0. Узлы ***должны*** молча отбрасывать NDP-пакеты, содержащие опцию с нулевым значением в поле Length.

Протокол NDP. Опция Source/Target Link-Layer Address

Формат опции



Type

- Source Link-Layer Address — 1
- Target Link-Layer Address — 2

Length

Длина опции, включая поля Type и Length в словах по 8 октетов. В случае использования адресов EUI-48 (MAC-адресов) значение поля равняется 1.

Link-Layer Address

Поле переменной длины, в которое записывается адрес канального уровня. Для адреса EUI-48 (MAC-адреса) имеет длину 6 байт.

Протокол NDP. Проверка достижимости узла

Neighbor Unreachability Detection. Проверка достижимости узла

Узел считается достижимым, если есть подтверждение того, что пакет отправленный к нему, был им получен. Поскольку между узлами могут быть другие (транзитные) узлы, недостижимость может быть связана с проблемами на транзитном оборудовании. По этой причине под достижимостью понимается только достижимость на первом хопе. Один из путей проверки достижимости — обмен NS-NA. Полученные unsolicitated NA и RA не считаются индикаторами достижимости.
Если A отправил NS к B, а тот ответил NA к A, то A считает B достижимым, но не наоборот. Для того, чтобы B считал A достижимым, он должен отправить NS к A и получить от A NA в ответ. Другой метод получения информации о достижимости — это информация от протоколов верхнего уровня, например от TCP.

Состояния Neighbor cache

Incomplete	— Запрос отправлен, но link-layer адрес еще не получен.
Reachable	— Получено подтверждение от соседа.
Stale	— Истекло время, в течение которого сосед считался достижимым, или получено unsolicited NA.
Delay	— Ожидается подтверждение от протокола верхнего уровня, как правило в течение 5 сек. При неполучении — переход в состояние Probe.
Probe	— Отправлен NS для соседей в состоянии Delay или Stale.
No entry exists	— Нет такой записи.

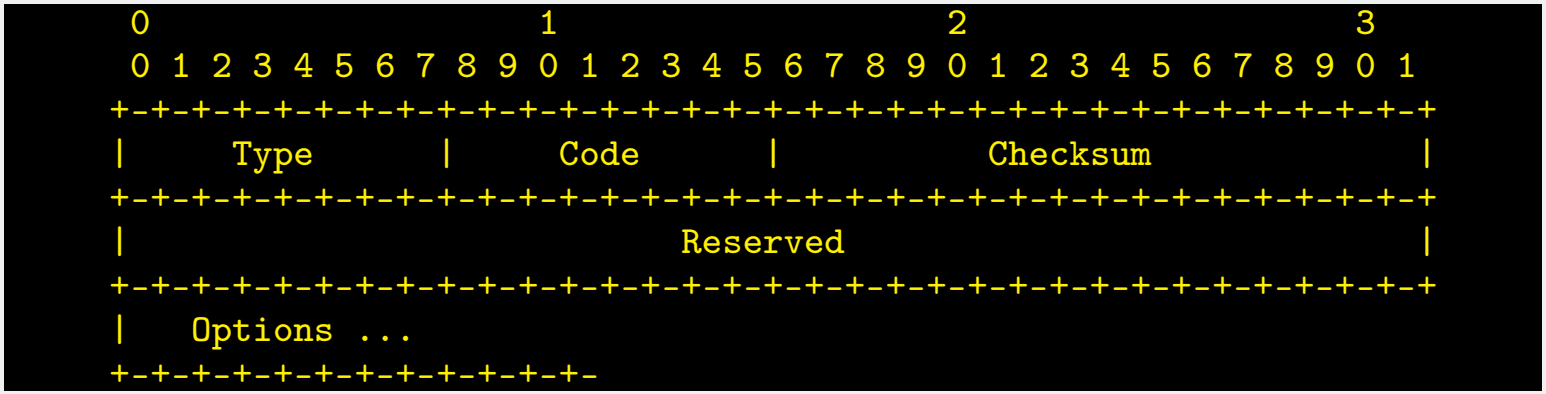
Duplicate address detection. Проверка дублирования адресов

Хост отправляет NS со своим предполагаемым адресом в качестве адреса получателя и слушает ответ. Если услышит NA — то адрес занят. После такого события требуется конфигурирование адреса вручную.

Протокол NDP. Router Solicitation

Router Solicitation

Сообщение отправляется хостом для нахождения роутера. Отправка такого сообщения позволяет получить информацию о роутерах, не дожидаясь прихода RA. В IPv6-заголовке такого сообщения в качестве источника указывается неопределенный адрес (`::/128`), или link-local адрес хоста. Адрес получателя устанавливается `FF02::2` (All-routers multicast), hop-limit задается равным 255.



ICMP заголовок

Type — 133

Code — 0

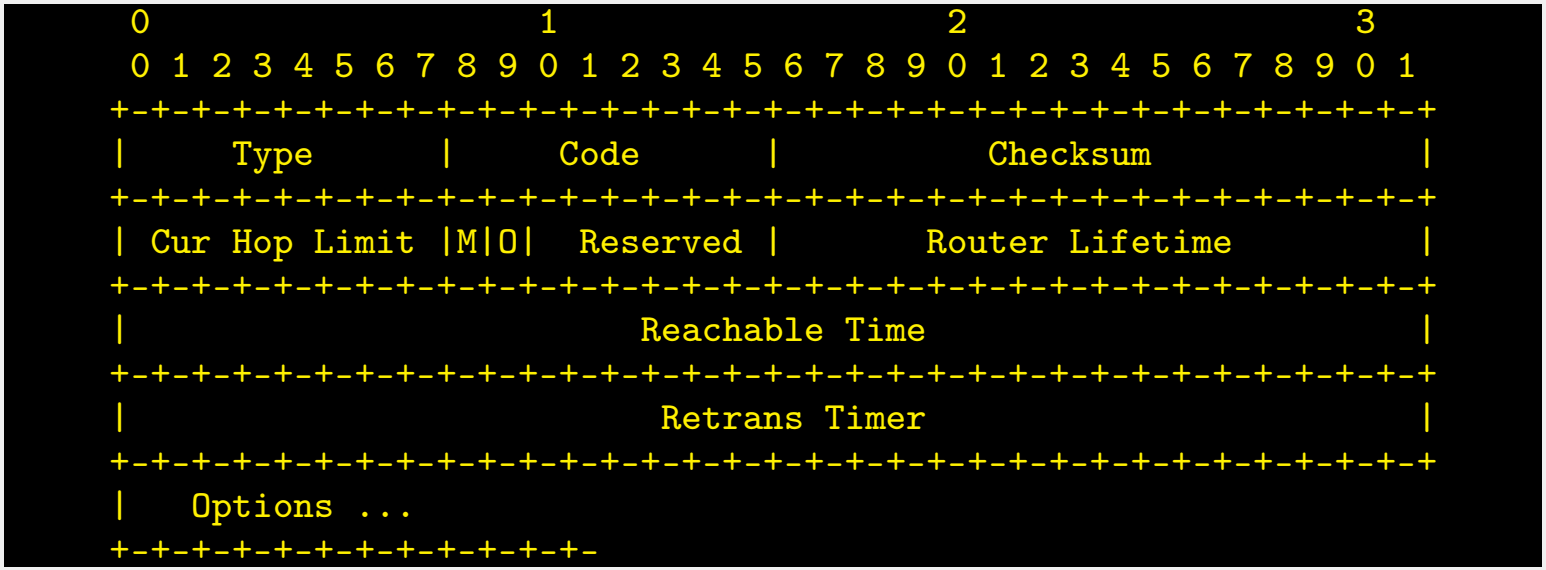
Опции

Source link-layer address — Канальный (link-layer) адрес отправителя. ***Не должен*** использоваться, если IPv6-адрес источника неопределен. В ином случае, для технологий, имеющих адресацию канального уровня, опцию ***рекомендуется*** использовать.

Протокол NDP. Router Advertisement

Router Advertisement

Роутер отправляет такое сообщение по своей инициативе через заданный интервал плюс-минус случайная величина, или в ответ на RS, при этом ответ следует с произвольной задержкой, чтобы несколько роутеров не ответили одновременно.



IPv6 заголовок

Source Address — Адрес интерфейса, с которого отправлено сообщение.

Destination Address — Адрес источника из соответствующего сообщения Router Solicitation или All-nodes multicast (`FF02::1`).

Hop Limit — 255

Протокол NDP. Router Advertisement

ICMP заголовок

Type — 134

Code — 0

M — «Managed address configuration» flag. Если установлен в 1, то хост должен использовать DHCP-сервер для получения адреса. При установленном флаге M следующий флаг O не обязателен и может игнорироваться, поскольку DHCP-сервер и так вернет всю необходимую информацию.

O — «Other configuration» flag. Если установлен в 1, то хост должен использовать DHCP-сервер для получения дополнительных параметров, например адреса DNS-сервера. Если же не установлены ни флаг M, ни флаг O, то это означает, что DHCP-сервер в данной сети не используется.

Router Lifetime — Время, в течение которого данный роутер может быть маршрутизатором по умолчанию. От нуля до 65535 секунд. Значение 0 запрещает использование роутера в качестве маршрутизатора по умолчанию.

Reachable time — Время в милисекундах, в течение которого хост считает этот роутер достижимым (32 бита). 0 означает, что роутер не устанавливает этот параметр.

Retransmission timer — Интервал в милисекундах между NS-сообщениями (32 бита). Используется для проверки достижимости хостов.

Опции

Source link-layer address — Канальный (link-layer) адрес отправителя RA.

MTU — Maximum transmission unit. ***Рекомендуется*** к использованию в сетях, где возможны различные MTU, т. е. передача IPv6 поверх различных технологий канального уровня. В иных случаях ***может*** использоваться.

Prefix Information — Содержит префиксы, которые можно использовать для автоконфигурации. ***Рекомендуется*** передавать все префиксы, прописанные на маршрутизаторе.

Источники

- Материалы с сайта `https://wikipedia.org/`
- RFC 4443. Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification
- RFC 4861. Neighbor Discovery for IP version 6 (IPv6)

Сети связи
Протоколы, сервисы и услуги в Интернет и IP-сетях

Тема № 9
Протоколы транспортного уровня TCP и UDP

доц. каф. СС и ПД, к.т.н. С. С. Владимиров

2020 г.

Протокол TCP

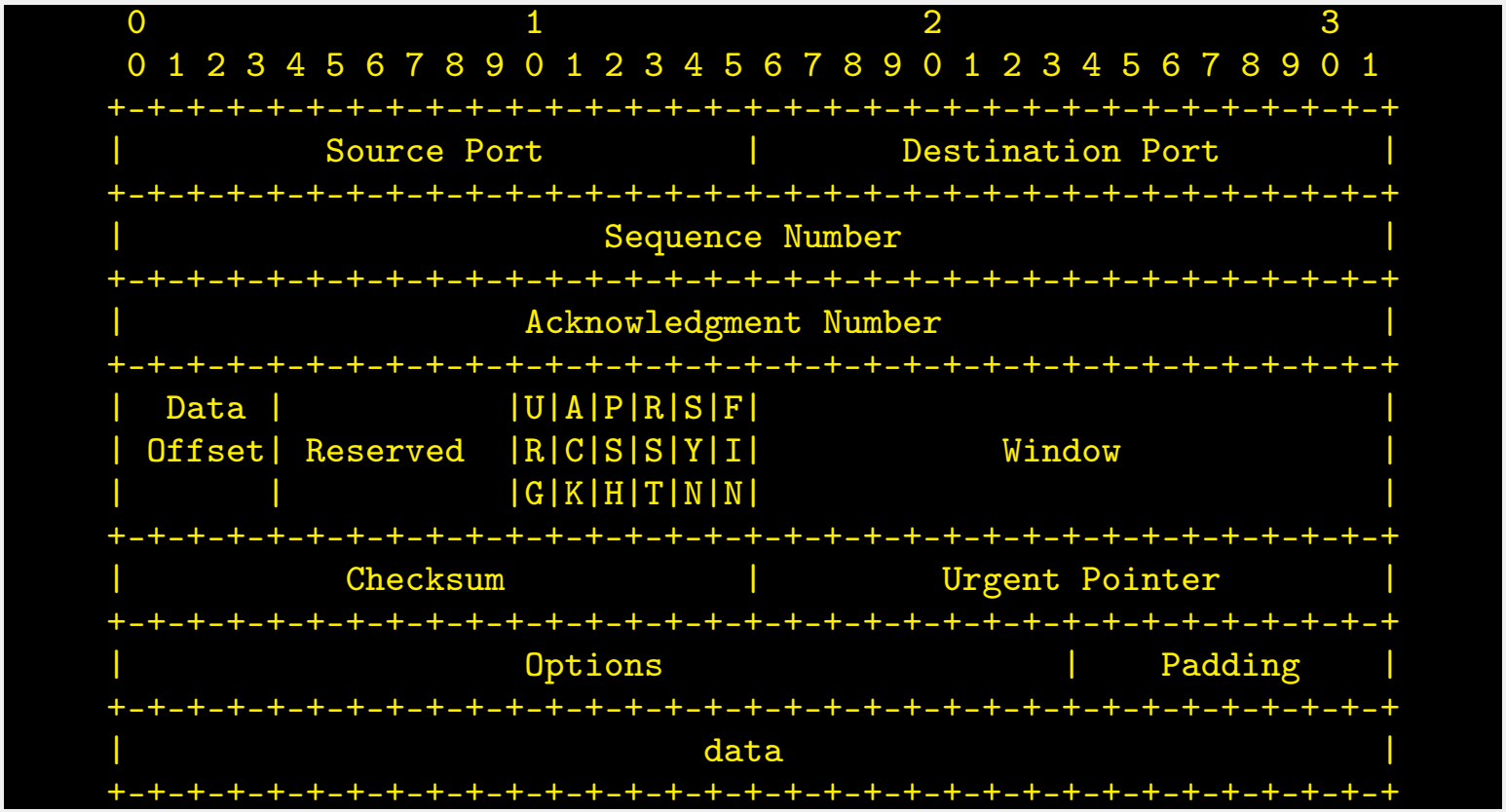
TCP (Transmission Control Protocol) — Протокол управления передачей

Один из основных протоколов транспортного уровня в стеке протоколов TCP/IP. Создан в 1981 г. Основной стандарт для протокола TCP — RFC 793 (STD 7) с обновлениями в RFC 1122, 3168, 6093, 6528.

TCP обеспечивает транспортный механизм, предоставляющий поток данных с предварительной установкой соединения, за счёт этого дающий уверенность в достоверности получаемых данных, осуществляет повторный запрос данных в случае потери данных и устраняет дублирование при получении двух копий одного пакета. Гарантирует целостность передаваемых данных и уведомление отправителя о результатах передачи.
Реализация TCP, как правило, встроена в ядро ОС, хотя есть и реализации TCP в контексте приложения.

Когда осуществляется передача от компьютера к компьютеру через Интернет, TCP работает на верхнем уровне между двумя конечными системами, например, браузером и веб-сервером. Также TCP осуществляет надежную передачу потока байтов от одной программы на некотором компьютере к другой программе на другом компьютере. Программы для электронной почты и обмена файлами используют TCP. TCP контролирует длину сообщения, скорость обмена сообщениями, сетевой трафик.

Формат заголовка TCP



Поля заголовка TCP

Source Port. Порт источника

16-битное поле, идентифицирующее приложение клиента, с которого отправлены пакеты. Ответные данные передаются клиенту на основании этого номера.

Destination Port. Порт назначения

Идентифицирует порт (приложение/протокол), на который отправлен пакет.

Sequence Number. Номер последовательности

Номер последовательности выполняет две задачи:

1. Если установлен флаг SYN, то это начальное значение номера последовательности — ISN (Initial Sequence Number), и первый байт данных, которые будут переданы в следующем пакете, будет иметь номер последовательности, равный ISN + 1.
2. В противном случае, если SYN не установлен, первый байт данных, передаваемый в данном пакете, имеет этот номер последовательности.

Поскольку поток TCP в общем случае может быть длиннее, чем число различных состояний этого поля, то все операции с номером последовательности должны выполняться по модулю 2^{32}. Это накладывает практическое ограничение на использование TCP. Если скорость передачи коммуникационной системы такова, чтобы в течение MSL (максимального времени жизни сегмента) произошло переполнение номера последовательности, то в сети может появиться два сегмента с одинаковым номером, относящихся к разным частям потока, и приёмник получит некорректные данные.

Поля заголовка TCP

Acknowledgment Number. Номер подтверждения

Если установлен флаг ACK, то это поле содержит номер последовательности, ожидаемый получателем в следующий раз. Помечает этот сегмент как подтверждение получения.

Data Offset. Длина заголовка (смещение данных)

Это поле определяет размер заголовка пакета TCP в 4-байтных (4-октетных) словах. Минимальный размер составляет 5 слов, а максимальный — 15, что составляет 20 и 60 байт соответственно. Смещение считается от начала заголовка TCP.

Reserved. Зарезервировано

Зарезервировано (6 бит) для будущего использования и должно устанавливаться в ноль. На данный момент в RFC 3168 определены биты 5 и 6:

- CWR (Congestion Window Reduced) — «Окно перегрузки уменьшено» — флаг установлен отправителем, чтобы указать, что получен пакет с установленным флагом ECE.
- ECE (ECN-Echo) — «Эхо ECN» — указывает, что данный узел способен на ECN (явное уведомление перегрузки) и для указания отправителю о перегрузках в сети.

Поля заголовка TCP

Flags. Флаги (управляющие биты)

- URG — Поле «Указатель важности» задействовано (Urgent pointer field is significant).
- ACK — Поле «Номер подтверждения» задействовано (Acknowledgement field is significant).
- PSH — Инструктирует получателя протолкнуть данные, накопившиеся в приемном буфере, в приложение пользователя (Push function).
- RST — Оборвать соединения, сбросить буфер (очистка буфера) (Reset the connection).
- SYN — Синхронизация номеров последовательности (Synchronize sequence numbers).
- FIN — Указывает на завершение соединения (FIN bit used for connection termination).

Window. Размер окна

В этом поле содержится число, определяющее в байтах размер данных, которые отправитель готов принять.

Checksum. Контрольная сумма

Поле контрольной суммы — это 16-битное дополнение к сумме всех 16-битных слов заголовка (включая псевдозаголовок) и данных. Если сегмент, по которому вычисляется контрольная сумма, имеет длину не кратную 16-ти битам, то длина сегмента увеличивается до кратной 16-ти, за счет дополнения к нему справа нулевых бит заполнения. Биты заполнения (0) не передаются в сообщении и служат только для расчёта контрольной суммы. При расчёте контрольной суммы значение самого поля контрольной суммы принимается равным 0.

Поля заголовка TCP

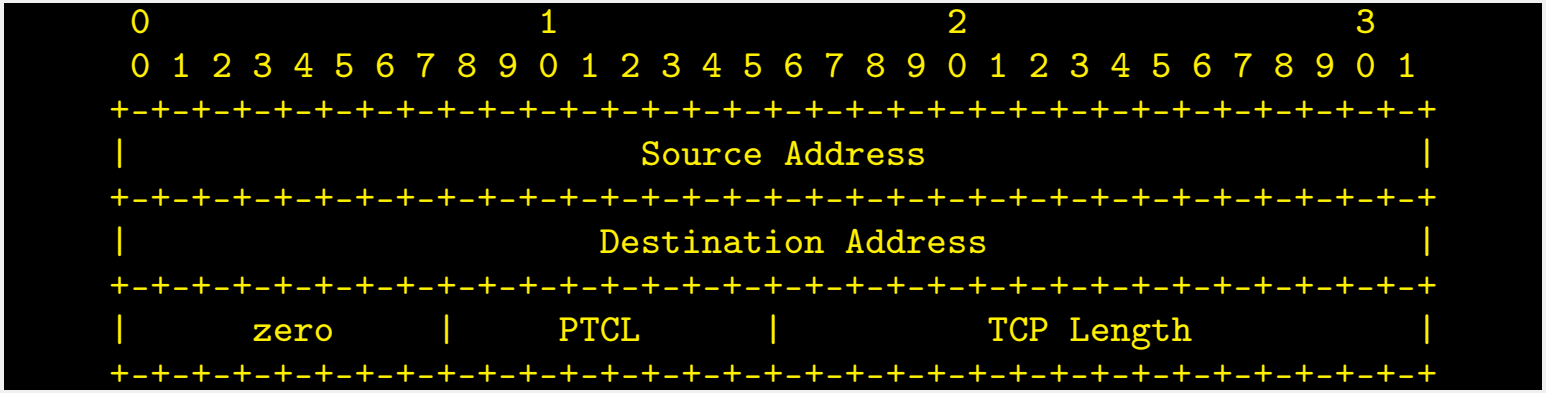
Urgent Pointer. Указатель важности

16-битовое значение положительного смещения от порядкового номера в данном сегменте. Это поле указывает порядковый номер октета, которым заканчиваются важные (urgent) данные. Поле принимается во внимание только для пакетов с установленным флагом URG.

Options. Опции

Могут применяться в некоторых случаях для расширения протокола. Иногда используются для тестирования. На данный момент в опции практически всегда включают 2 байта NOP (в данном случае `0x01`) и 10 байт, задающих временные метки (timestamps). Вычислить длину поля опции можно через значение поля смещения.

Псевдозаголовок TCP



TCP-заголовок не содержит информации об адресе отправителя и получателя, поэтому даже при совпадении порта получателя нельзя с точностью сказать, что сообщение пришло в нужное место. Поскольку назначением протокола TCP является надёжная доставка сообщений, то этот момент имеет принципиальное значение. Для того, чтобы не дублировать адресную информацию в заголовке TCP, был использован дополнительный псевдозаголовок.
Псевдозаголовок не включается в TCP-сегмент. Он используется для расчета контрольной суммы перед отправлением сообщения и при его получении (получатель составляет свой псевдозаголовок, используя адрес хоста, с которого пришло сообщение, и собственный адрес, а затем считает контрольную сумму).

Поля псевдозаголовка

Source Address — IP-адрес источника.

Dest. Address — IP-адрес получателя.

zero — Четыре нулевых бита.

PTCL — Тип протокола верхнего относительно IP уровня. В случае TCP Это поле равно `0x06`.

TCP Length — Содержит в себе длину TCP-сегмента в байтах (TCP-заголовок + данные; длина псевдозаголовка не учитывается).

Пример расчета контрольной суммы заголовка TCP

Контрольная сумма (КС, CS) заголовка TCP представляет собой 16-битовое поразрядное дополнение суммы всех 16-битовых слов заголовка, данных и псевдозаголовка. При вычислении контрольной суммы значение самого поля принимается нулевым.

Пример пакета TCP. Заголовок — зелёный. КС — синяя

```
0000: 00 50 FC 1E BF 8D 00 30 4F 0E 89 65 08 00 45 00
0010: 00 38 89 28 40 00 80 06 11 21 C0 A8 01 32 C3 13
0020: DB 88 04 50 00 15 00 4C 69 E7 3C 00 27 92 50 18
0030: 22 05 D3 39 00 00 55 53 45 52 20 61 6E 6F 6E 79
0040: 6D 6F 75 73 0D 0A
```

Псевдозаголовок

```
C0 A8 01 32
C3 13 DB 88
00 06 00 24
```

Вычисление КС заголовка TCP

1. Заголовок TCP разбивается на слова размером 16 бит (2 байта) каждое. Поле КС принимается равным нулю (не участвует в вычислении). Все полученные слова суммируются.

$$0450 + 0015 + 004C + 69E7 + 3C00 + 2792 + 5018 + 2205 + 0000 + 0000 = 14447$$

2. Аналогично считаем данные и псевдозаголовок.

$$5553 + 4552 + 2061 + 6E6F + 6E79 + 6D6F + 7573 + 0D0A = 287DA$$

$$C0A8 + 0132 + C313 + DB88 + 0006 + 0024 = 2609F$$

3. Итого: $14447 + 287da + 2609F = 62CC0$
4. Длина результата суммирования превышает 2 байта (4 шестнадцатеричные цифры), следовательно, производится перенос разряда: $0006 + 2CC0 = 2CC6$.
5. Находится поразрядное дополнение от итоговой суммы. Результат и будет контрольной суммой заголовка пакета TCP.

$$FFFF - 2CC6 = D339 = CS_{TCP}$$

Сеанс протокола TCP

Стадии TCP-соединения

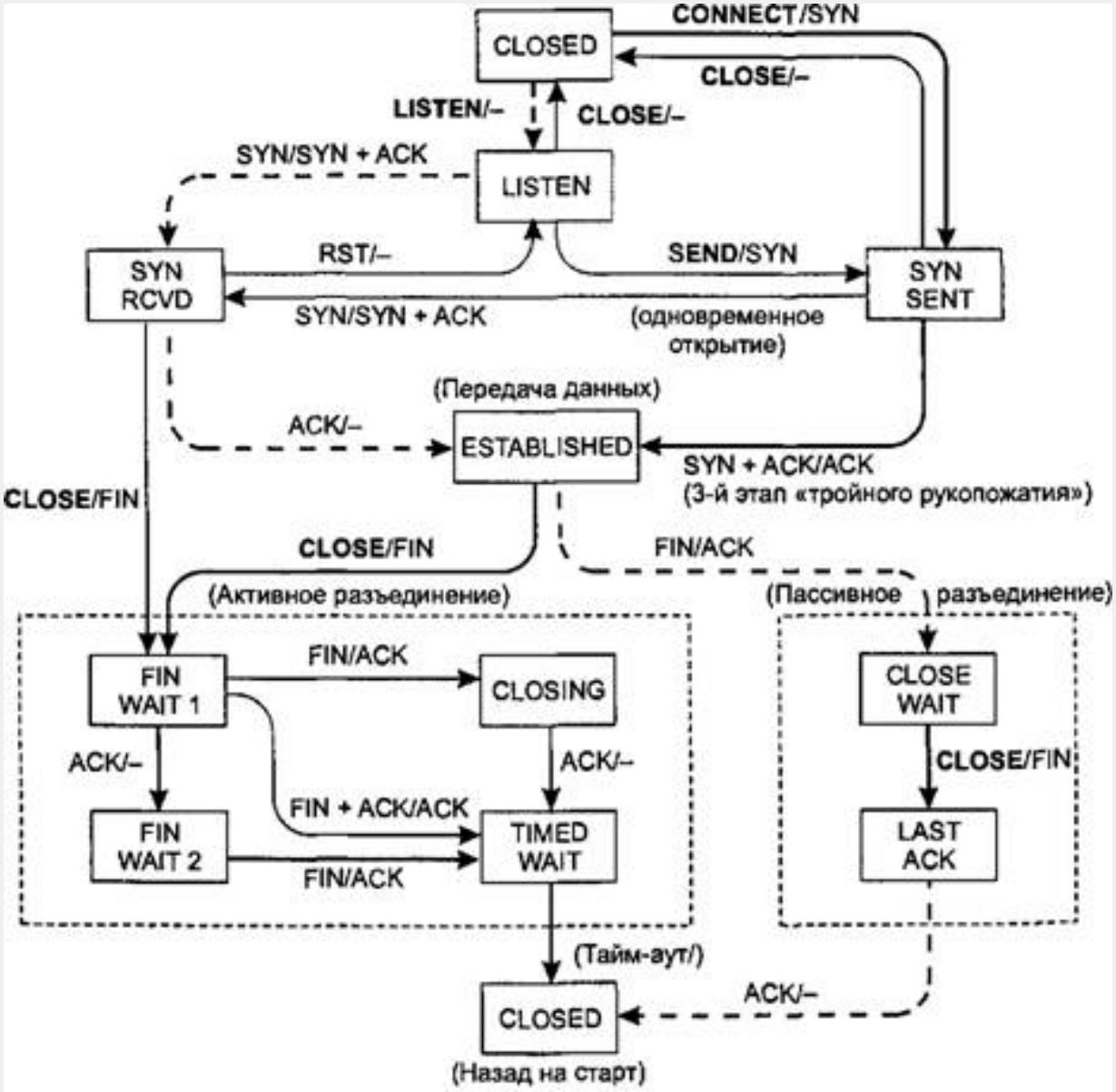
1. Установка соединения
2. Передача данных
3. Завершение соединения

Состояния сеанса TCP

CLOSED	Начальное состояние узла. Фактически фиктивное
LISTEN	Сервер ожидает запросов установления соединения от клиента
SYN-SENT	Клиент отправил запрос серверу на установление соединения и ожидает ответа
SYN-RECEIVED	Сервер получил запрос на соединение, отправил ответный запрос и ожидает подтверждения
ESTABLISHED	Соединение установлено, идёт передача данных
FIN-WAIT-1	Одна из сторон (назовём её узел-1) завершает соединение, отправив сегмент с флагом FIN
CLOSE-WAIT	Другая сторона (узел-2) переходит в это состояние, отправив, в свою очередь сегмент ACK и продолжает одностороннюю передачу
FIN-WAIT-2	Узел-1 получает ACK, продолжает чтение и ждёт получения сегмента с флагом FIN
LAST-ACK	Узел-2 заканчивает передачу и отправляет сегмент с флагом FIN
TIME-WAIT	Узел-1 получил сегмент с флагом FIN, отправил сегмент с флагом ACK и ждёт 2*MSL секунд, перед окончательным закрытием соединения
CLOSING	Обе стороны инициировали закрытие соединения одновременно: после отправки сегмента с флагом FIN узел-1 также получает сегмент FIN, отправляет ACK и находится в ожидании сегмента ACK (подтверждения на свой запрос о разъединении)

Сеанс протокола TCP

Конечный автомат TCP



Установление TCP-соединения

Процедура установления TCP-соединения называется «трехэтапным согласованием» («three way handshake»).

1. Клиент, который намеревается установить соединение, посылает серверу сегмент с номером последовательности и флагом SYN.
   - Сервер получает сегмент, запоминает номер последовательности и пытается создать сокет (буферы и управляющие структуры памяти) для обслуживания нового клиента.
   - В случае успеха сервер посылает клиенту сегмент с номером последовательности и флагами SYN и ACK, и переходит в состояние SYN-RECEIVED.
   - В случае неудачи сервер посылает клиенту сегмент с флагом RST.
2. Если клиент получает сегмент с флагом SYN, то он запоминает номер последовательности и посылает сегмент с флагом ACK.
   - Если он одновременно получает и флаг ACK (что обычно и происходит), то он переходит в состояние ESTABLISHED.
   - Если клиент получает сегмент с флагом RST, то он прекращает попытки соединиться.
   - Если клиент не получает ответа в течение 10 секунд, то он повторяет процесс соединения заново.
3. Если сервер в состоянии SYN-RECEIVED получает сегмент с флагом ACK, то он переходит в состояние ESTABLISHED. В противном случае после тайм-аута он закрывает сокет и переходит в состояние CLOSED.

Существуют экспериментальные расширения протокола TCP, сокращающие количество пакетов при установлении соединения, например TCP Fast Open. Ранее также существовало расширение T/TCP.

Пример установления TCP-соединения

```
      TCP A                                                TCP B
1.  CLOSED                                               LISTEN
2.  SYN-SENT    --> <SEQ=100><CTL=SYN>               --> SYN-RECEIVED
3.  ESTABLISHED <-- <SEQ=300><ACK=101><CTL=SYN,ACK>  <-- SYN-RECEIVED
4.  ESTABLISHED --> <SEQ=101><ACK=301><CTL=ACK>      --> ESTABLISHED
5.  ESTABLISHED <-- <SEQ=301><ACK=101><CTL=ACK>      <-- ESTABLISHED
```

В строке 2 TCP A начинает передачу сегмента SYN, говорящего об использовании номеров последовательности, начиная со 100. В строке 3 TCP B передает SYN и подтверждение для принятого SYN в адрес TCP A. Надо отметить, что поле подтверждения показывает ожидание TCP B приема номера последовательности 101, подтверждающего SYN с номером 100.
В строке 4 TCP A отвечает пустым сегментом с подтверждением ACK для сегмента SYN от TCP B; в строке 5 TCP B передает некоторые данные. Отметим, что номер последовательности сегмента в строке 5 совпадает с номером в строке 4, поскольку ACK не занимает пространства номеров последовательности (если это сделать, придется подтверждать подтверждения — ACK для ACK).

Передача данных в TCP

При обмене данными приемник использует номер последовательности, содержащийся в получаемых сегментах, для восстановления их исходного порядка. Приемник уведомляет передающую сторону о номере последовательности байт, до которой он успешно получил данные, включая его в поле «номер подтверждения». Все получаемые данные, относящиеся к промежутку подтвержденных последовательностей, игнорируются. Если полученный сегмент содержит номер последовательности больший, чем ожидаемый, то данные из сегмента буферизируются, но номер подтвержденной последовательности не изменяется. Если впоследствии будет принят сегмент, относящийся к ожидаемому номеру последовательности, то порядок данных будет автоматически восстановлен исходя из номеров последовательностей в сегментах.

Для того, чтобы передающая сторона не отправляла данные интенсивнее, чем их может обработать приемник, TCP содержит средства управления потоком. Для этого используется поле «окно». В сегментах, направляемых от приемника передающей стороне в поле «окно» указывается текущий размер приемного буфера. Передающая сторона сохраняет размер окна и отправляет данных не более, чем указал приемник. Если приемник указал нулевой размер окна, то передача данных в направлении этого узла не происходит, до тех пор пока приемник не сообщит о большем размере окна.

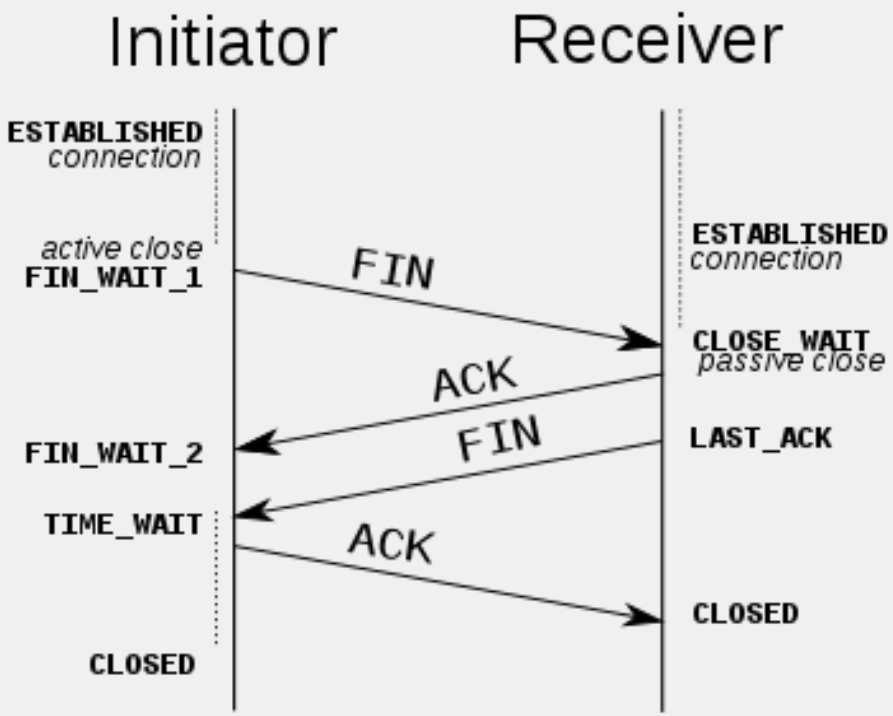
В некоторых случаях передающее приложение может явно затребовать протолкнуть данные до некоторой последовательности принимающему приложению, не буферизируя их. Для этого используется флаг PSH. Если в полученном сегменте обнаруживается флаг PSH, то реализация TCP отдает все буферизированные на текущий момент данные принимающему приложению. «Проталкивание» используется, например, в интерактивных приложениях. В сетевых терминалах нет смысла ожидать ввода пользователя после того, как он закончил набирать команду. Поэтому последний сегмент, содержащий команду, обязан содержать флаг PSH, чтобы приложение на принимающей стороне смогло начать её выполнение.

Завершение TCP-соединения

Этапы завершения соединения

1. Посылка серверу от клиента флагов FIN и ACK на завершение соединения.
2. Сервер посылает клиенту флаги ответа ACK, FIN, что соединение закрыто.
3. После получения этих флагов клиент закрывает соединение и в подтверждение отправляет серверу ACK , что соединение закрыто.

Диаграмма завершения соединения



Особенности протокола TCP

Максимальный размер сегмента

TCP требует явного указания максимального размера сегмента (MSS) в случае, если виртуальное соединение осуществляется через сегмент сети, где максимальный размер блока (MTU) менее, чем стандартный MTU Ethernet (1500 байт).
В протоколах туннелирования, таких как GRE, IPIP, а также PPPoE MTU туннель меньше чем стандартный, поэтому сегмент TCP максимального размера имеет длину пакета больше, чем MTU. Это приводит к фрагментации и уменьшению скорости передачи полезных данных. Если на каком-либо узле фрагментация запрещена, то со стороны пользователя это выглядит как «зависание» соединений. При этом «зависание» может происходить в произвольные моменты времени, а именно тогда, когда отправитель использовал сегменты длиннее допустимого размера. Для решения этой проблемы на маршрутизаторах применяются правила Firewall, добавляющие параметр MSS во все пакеты, инициирующие соединения, чтобы отправитель использовал сегменты допустимого размера.
MSS может также управляться параметрами операционной системы.

Обнаружение ошибок при передаче данных

Хотя протокол осуществляет проверку контрольной суммы по каждому сегменту, используемый алгоритм считается слабым. Так, в 2008 году ошибка в передаче одного бита, не обнаруженная сетевыми средствами, привела к остановке серверов системы Amazon Web Services.

Освобождение от расчёта контрольной суммы

Многие реализации стека TCP/IP предоставляют возможности использования аппаратной поддержки для автоматического расчёта контрольной суммы в сетевом адаптере до передачи в сеть или после приёма из сети для верификации. Это может освобождать операционную систему от использования ценных тактов процессора при вычислении контрольной суммы. Эта функция может приводить к тому, что анализаторы трафика, перехватывающие исходящие пакеты до их передачи в сетевой адаптер и не знающие о делегировании расчёта контрольной суммы сетевому адаптеру, могут сообщать об ошибке контрольной суммы в исходящих пакетах.

Протокол UDP

UDP (User Datagram Protocol)

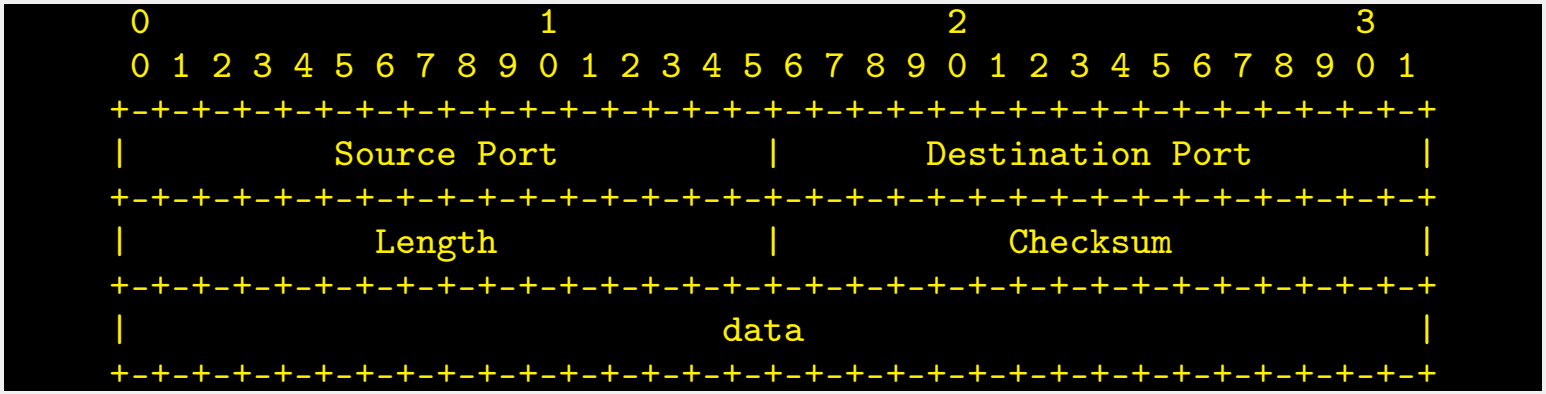
UDP позволяет посылать сообщения (датаграммы) другим хостам по IP-сети без необходимости предварительного установления соединений. Был разработан Дэвидом П. Ридом в 1980 году и официально определён в RFC 768 (STD 6).

UDP использует простую модель передачи, без неявных «рукопожатий» для обеспечения надёжности, упорядочивания или целостности данных. Таким образом, UDP предоставляет ненадёжный сервис, и датаграммы могут прийти не по порядку, дублироваться или вовсе исчезнуть без следа. По этой причине UDP иногда называют Unreliable Datagram Protocol (Ненадёжный протокол датаграмм). UDP подразумевает, что проверка ошибок и исправление либо не нужны, либо должны исполняться в приложении. Чувствительные ко времени приложения часто используют UDP, так как предпочтительнее сбросить пакеты, чем ждать задержавшиеся пакеты, что может оказаться невозможным в системах реального времени.

Природа UDP как протокола без сохранения состояния также полезна для серверов, отвечающих на небольшие запросы от огромного числа клиентов, например DNS и потоковые мультимедийные приложения вроде IPTV.

Структура пакета UDP. Поля заголовка UDP

Заголовок UDP



Source Port. Порт отправителя

В этом поле указывается номер порта отправителя. Предполагается, что это значение задаёт порт, на который при необходимости будет посылаться ответ. В противном же случае, значение должно быть равным 0. Если хостом-источником является клиент, то номер порта будет, скорее всего, динамическим. Если источником является сервер, то его порт будет одним из зарегистрированных в IANA.

Destination Port. Порт получателя

Это поле обязательно и содержит порт получателя. Аналогично порту отправителя, если хостом-получателем является клиент, то номер порта динамический, если получатель — сервер, то это будет порт из списка IANA.

Поля заголовка UDP

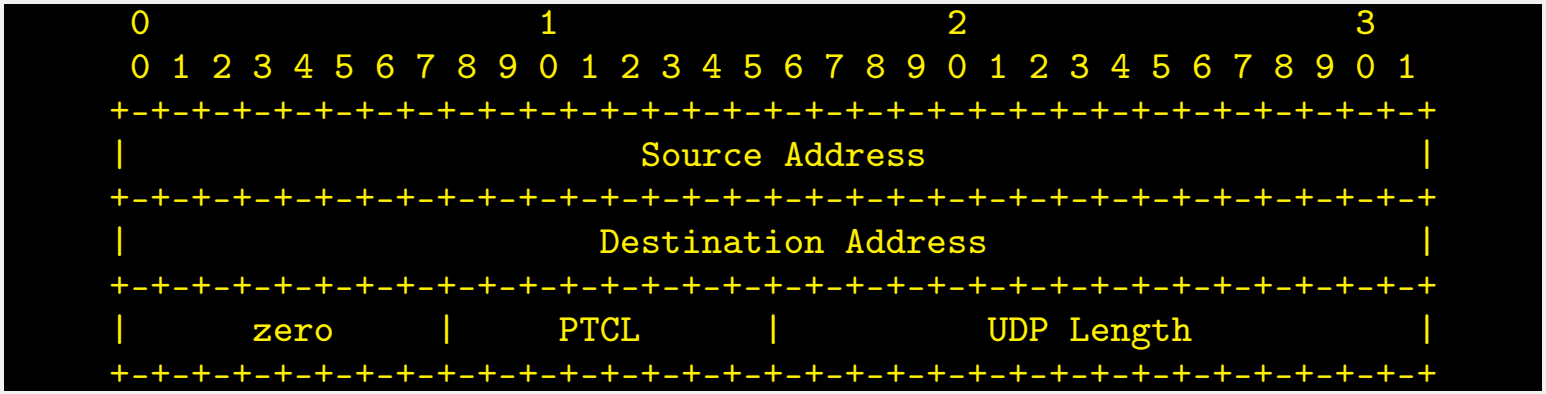
Length. Длина датаграммы

Поле, задающее длину всей датаграммы (заголовка и данных) в байтах. Минимальная длина равна длине заголовка — 8 байт. Теоретически, максимальный размер поля — 65535 байт для UDP-датаграммы (8 байт на заголовок и 65527 на данные). Фактический предел для длины данных при использовании IPv4 — 65507 (помимо 8 байт на UDP-заголовок требуется ещё 20 на IP-заголовок).
На практике также следует учитывать, что если длинна IPv4 пакета с UDP будет превышать MTU (для Ethernet по умолчанию 1500 байт), то отправка такого пакета вызовет его фрагментацию, что может привести к тому, что он вообще не сможет быть доставлен, если промежуточные маршрутизаторы или конечный хост не будут поддерживать фрагментированные IP пакеты. Также в RFC 791 указывается минимальная длина IP пакета не менее 576 байт и рекомендуется отправлять IP пакеты большего размера только в том случае если вы уверены, что принимающая сторона может принять пакеты такого размера. Следовательно, чтобы избежать фрагментации UDP пакетов (и возможной их потери), размер данных в UDP не должен превышать: $MTU - (MaxIPHeaderSize) - (UDPHedaerSize) = 1500 - 60 - 8 = 1432$ байт. Для того чтобы быть уверенным, что пакет будет принят любым хостом, размер данных в UDP не должен превышать: $(MinIP - packetLength) - (MaxIPHeaderSize) - (UDPHeaderSize) = 576 - 60 - 8 = 508$ байт.
В Jumbogram IPv6 пакеты UDP могут иметь больший размер. Максимальное значение составляет 4 294 967 295 байт ($2^{32} - 1$), из которых 8 байт соответствуют заголовку, а остальные 4 294 967 287 байт — данным.

Checksum. Контрольная сумма

Поле контрольной суммы используется для проверки заголовка и данных на ошибки. Если сумма не сгенерирована передатчиком, то поле заполняется нулями. Поле не является обязательным для IPv4. Метод для вычисления контрольной суммы определён в RFC 768. Методика расчета контрольной суммы заголовка UDP полностью аналогична методике расчета контрольной суммы заголовка TCP.

Псевдозаголовок UDP



Поля псевдозаголовка

Source Address — IP-адрес источника.

Dest. Address — IP-адрес получателя.

zero — Четыре нулевых бита.

PTCL — Тип протокола верхнего относительно IP уровня. В случае UDP Это поле равно `0x11`.

UDP Length — Содержит в себе длину UDP-датаграммы в байтах (UDP-заголовок + данные; длина псевдозаголовка не учитывается). Соответствует полю Length заголовка UDP.

Надёжность и решения проблемы перегрузок

Из-за недостатка надёжности приложения UDP должны быть готовы к некоторым потерям, ошибкам и дублированиям. Некоторые из них (например, TFTP) могут при необходимости добавить элементарные механизмы обеспечения надёжности на прикладном уровне.
Но чаще такие механизмы не используются UDP-приложениями и даже мешают им. Потоковые медиа, многопользовательские игры в реальном времени и VoIP — примеры приложений, часто использующих протокол UDP. В этих конкретных приложениях потеря пакетов обычно не является большой проблемой. Если приложению необходим высокий уровень надёжности, то можно использовать другой протокол (TCP) или воспользоваться методами помехоустойчивого кодирования.

Более серьёзной потенциальной проблемой является то, что в отличие от TCP, основанные на UDP приложения не обязательно имеют хорошие механизмы контроля и избегания перегрузок. Чувствительные к перегрузкам UDP-приложения, которые потребляют значительную часть доступной пропускной способности, могут поставить под угрозу стабильность в Интернете.
Сетевые механизмы были предназначены для того, чтобы свести к минимуму возможные эффекты от перегрузок при неконтролируемых, высокоскоростных нагрузках. Такие сетевые элементы, как маршрутизаторы, использующие пакетные очереди и техники сброса, часто являются единственным доступным инструментом для замедления избыточного UDP-трафика. DCCP (Datagram Congestion Control Protocol — протокол контроля за перегрузками датаграмм) разработан как частичное решение этой потенциальной проблемы с помощью добавления конечному хосту механизмов для отслеживания перегрузок для высокоскоростных UDP-потоков вроде потоковых медиа.

Приложения и протоколы, использующие UDP

Многочисленные ключевые Интернет-протоколы используют UDP, в их числе — DNS (где запросы должны быть быстрыми и состоять только из одного запроса, за которым следует один пакет ответа), Простой Протокол Управления Сетями (SNMP), Протокол Маршрутной Информации (RIP), Протокол Динамической Конфигурации Узла (DHCP).

Голосовой и видеотрафик обычно передается с помощью UDP. Протоколы потокового видео в реальном времени и аудио разработаны для обработки случайных потерь пакетов так, что качество лишь незначительно уменьшается вместо больших задержек при повторной передаче потерянных пакетов. Поскольку и TCP, и UDP работают с одной и той же сетью, многие компании замечают, что недавнее увеличение UDP-трафика из-за этих приложений реального времени мешает производительности TCP-приложений вроде систем баз данных или бухгалтерского учета. Так как и бизнес-приложения, и приложения в реальном времени важны для компаний, развитие качества решений проблемы некоторыми рассматривается в качестве важнейшего приоритета.

Сравнение UDP и TCP

TCP

Ориентированный на соединение протокол, что означает необходимость «рукопожатия» для установки соединения между двумя хостами. Как только соединение установлено, пользователи могут отправлять данные в обоих направлениях.

- Надёжность — управляет подтверждением, повторной передачей и тайм-аутом сообщений. Производятся многочисленные попытки доставить сообщение. Если оно потеряется на пути, сервер вновь запросит потерянную часть. Нет ни пропавших данных, ни (в случае многочисленных тайм-аутов) разорванных соединений.
- Упорядоченность — если два сообщения последовательно отправлены, первое сообщение достигнет приложения-получателя первым. Если участки данных прибывают в неверном порядке, TCP отправляет неупорядоченные данные в буфер до тех пор, пока все данные не могут быть упорядочены и переданы приложению.
- Тяжеловесность — необходимо три пакета для установки соединения перед отправкой данных. Следит за надёжностью и перегрузками.
- Потоковость — данные читаются как поток байтов без особых обозначений для границ сообщения или сегментов.

UDP

Простой, основанный на сообщениях протокол без установления соединения. Связь достигается путем передачи информации в одном направлении от источника к получателю без проверки готовности или состояния получателя.

- Ненадёжный — неизвестно, достигнет ли посланное своего назначения — оно может потеряться по пути. Нет таких понятий, как подтверждение, повторная передача, тайм-аут.
- Неупорядоченность — если два сообщения отправлены одному получателю, то порядок их достижения цели не может быть предугадан.
- Легковесность — нет упорядочивания сообщений, отслеживания соединений и т. д.
- Датаграммы — пакеты посылаются по отдельности и проверяются на целостность только если они прибыли. Пакеты имеют определенные границы, которые соблюдаются после получения, то есть операция чтения на сокете-получателе выдаст сообщение таким, каким оно было изначально послано.
- Нет контроля перегрузок — UDP сам по себе не избегает перегрузок. Для приложений с большой пропускной способностью возможно вызвать коллапс перегрузок, если они не реализуют меры контроля на прикладном уровне.

Источники

- Материалы с сайта `https://wikipedia.org/`
- Telecommunication technologies — телекоммуникационные технологии / Ю. А. Семенов.
  URL: `http://book.itep.ru/`
- RFC 793. Transmission Control Protocol.
- RFC 768. User Datagram Protocol.

Сети связи
Протоколы, сервисы и услуги в Интернет и IP-сетях

Тема № 10
Протоколы удаленного управления Telnet и SSH

доц. каф. СС и ПД, к.т.н. С. С. Владимиров

2020 г.

Протокол/служба Telnet

TELNET (TErminaL NETwork)

Служба прикладного уровня модели OSI для реализации текстового интерфейса по сети (в современной форме — при помощи транспорта TCP). Название «telnet» имеют также некоторые утилиты, реализующие клиентскую часть протокола. Современный стандарт протокола описан в RFC 854.

Назначение протокола TELNET в предоставлении достаточно общего, двунаправленного, восьмибитного байт-ориентированного средства связи. Его основная задача заключается в том, чтобы позволить терминальным устройствам и терминальным процессам взаимодействовать друг с другом. Протокол может быть использован для связи вида терминал-терминал («связывание») или для связи процесс-процесс («распределенные вычисления»).

В сессии Telnet выделяют клиентскую и серверную стороны, однако сам протокол на самом деле полностью симметричен. После установления транспортного соединения оба его конца играют роль «сетевых виртуальных терминалов» (Network Virtual Terminal, NVT), обменивающихся двумя типами данных:

- прикладные данные;
- команды протокола Telnet.

Хотя Telnet-сессии, выполняющейся по TCP, свойственен полный дуплекс, NVT должен рассматриваться как полудуплексное устройство, работающее по умолчанию в буферизированном строковом режиме.

Прикладные данные проходят через протокол без изменений, то есть на выходе второго виртуального терминала мы видим именно то, что было введено на вход первого. С точки зрения протокола данные представляют просто последовательность байтов (октетов), по умолчанию принадлежащих набору ASCII, но при включенной опции Binary — любых. Были предложены расширения для идентификации набора символов, но на практике ими не пользуются.

Протокол/служба Telnet

Все значения октетов прикладных данных кроме \377 (десятичное: 255) передаются по транспорту как есть. Октет \377 передаётся последовательностью \377\377 из двух октетов. Это связано с тем, что октет \377 используется на транспортном уровне для кодирования опций.
Протокол предоставляет по умолчанию минимальную функциональность и набор расширяющих её опций. Принцип оговоренных опций требует проводить переговоры при включении каждой из опций. Одна сторона инициирует запрос, а другая сторона может либо принять, либо отвергнуть предложение. Если запрос принимается, то опция немедленно вступает в силу. Опции описаны отдельно от протокола как такового, и их поддержка программным обеспечением произвольна. Клиенту протокола (сетевому терминалу) предписывается отвергать запросы на включение неподдерживаемых и неизвестных опций.
Терминал NVT имеет неопределённую ширину каретки и длину страницы и должен иметь представление всех 95 печатных символов US-ASCII (коды с 32 по 126).
Каждая команда TELNET является многобайтовой последовательностью, начинающейся с кода \377 «Interpret as Command» (IAC) и кода команды. Команды, отвечающие за договоренности по опции, являются трехбайтовыми последовательностями, где третий байт является кодом опции.

Исторически Telnet служил для удалённого доступа к интерфейсу командной строки операционных систем. Впоследствии его стали использовать для прочих текстовых интерфейсов, вплоть до игр MUD и анимированного ASCII-art. Теоретически, даже обе стороны протокола могут являться не только людьми, но и программами.
Иногда клиенты telnet используются для доступа к другим протоколам на основе транспорта TCP. Например, telnet используется в управляющем соединении FTP.
В протоколе не предусмотрено использование ни шифрования, ни проверки подлинности данных. Поэтому он уязвим для любого вида атак, к которым уязвим его транспорт, то есть протокол TCP. Сессия Telnet беззащитна, если только не осуществляется в полностью контролируемой сети или с применением защиты на сетевом уровне (различные реализации виртуальных частных сетей). По причине ненадёжности от Telnet как средства управления операционными системами отказались.

Протокол SSH

SSH (Secure Shell)

Сетевой протокол прикладного уровня, позволяющий производить удалённое управление операционной системой и туннелирование TCP-соединений (например, для передачи файлов). Схож по функциональности с протоколами Telnet и rlogin, но, в отличие от них, шифрует весь трафик, включая и передаваемые пароли. SSH допускает выбор различных алгоритмов шифрования. SSH-клиенты и SSH-серверы доступны для большинства сетевых операционных систем.

SSH позволяет безопасно передавать в незащищённой среде практически любой другой сетевой протокол. Таким образом, можно не только удалённо работать на компьютере через командную оболочку, но и передавать по шифрованному каналу звуковой поток или видео. Также SSH может использовать сжатие передаваемых данных для последующего их шифрования, что удобно, например, для удалённого запуска клиентов X Window System.

Первая версия протокола, SSH-1, была разработана в 1995 году исследователем Тату Улёненом из Технологического университета Хельсинки (Финляндия). SSH-1 был написан для обеспечения большей конфиденциальности, чем протоколы rlogin, telnet и rsh. В 1996 году была разработана более безопасная версия протокола, SSH-2, несовместимая с SSH-1. Протокол приобрел ещё большую популярность, и к 2000 году у него было около двух миллионов пользователей. В настоящее время под термином «SSH» обычно подразумевается именно SSH-2, т. к. первая версия протокола ввиду существенных недостатков сейчас практически не применяется.

В 2006 году протокол был утвержден рабочей группой IETF в качестве Интернет-стандарта. SSH-2 определен в RFC 4250–4256.

В некоторых странах (Франция, Россия, Ирак и Пакистан) требуется специальное разрешение в соответствующих структурах для использования определённых методов шифрования, включая SSH.

Распространены две реализации SSH: частная коммерческая и бесплатная свободная. Свободная реализация называется OpenSSH.

Протокол SSH

Протокол SSH-1, в отличие от протокола telnet, устойчив к атакам прослушивания трафика («снифинг»), но неустойчив к атакам «человек посередине». Протокол SSH-2 также устойчив к атакам путем присоединения посредине (session hijacking), так как невозможно включиться в уже установленную сессию или перехватить её.
Для предотвращения атак «человек посередине» при подключении к хосту, ключ которого ещё не известен клиенту, клиентское ПО показывает пользователю «слепок ключа» (key fingerprint). Рекомендуется тщательно сверять показываемый клиентским ПО «слепок ключа» со слепком ключа сервера, желательно полученным по надёжным каналам связи или лично.
Поддержка SSH реализована во всех UNIX-подобных системах, и на большинстве из них в числе стандартных утилит присутствуют клиент и сервер ssh. Существует множество реализаций SSH-клиентов и для не-UNIX ОС. Большую популярность протокол получил после широкого развития анализаторов трафика и способов нарушения работы локальных сетей, как альтернативное небезопасному протоколу Telnet решение для управления важными узлами.

Для работы по SSH нужен SSH-сервер и SSH-клиент. Сервер прослушивает соединения от клиентских машин и при установлении связи производит аутентификацию, после чего начинает обслуживание клиента. Клиент используется для входа на удалённую машину и выполнения команд.
Для соединения сервер и клиент должны создать пары ключей — открытых и закрытых — и обменяться открытыми ключами. Обычно используется также и пароль.

Протокол SSH

Техническая информация о протоколе

SSH-сервер для работы использует TCP-порт 22. Для аутентификации сервера в SSH используется протокол аутентификации сторон на основе алгоритмов электронно-цифровой подписи RSA или DSA. Для аутентификации клиента также может использоваться ЭЦП RSA или DSA, но допускается также аутентификация при помощи пароля (режим обратной совместимости с Telnet) и IP-адреса хоста (режим обратной совместимости с rlogin). Аутентификация по паролю наиболее распространена; она безопасна, так как пароль передаётся по зашифрованному виртуальному каналу. Аутентификация по ip-адресу небезопасна, эту возможность чаще всего отключают. Для создания общего секрета (сеансового ключа) используется алгоритм Диффи–Хеллмана (DH). Для шифрования передаваемых данных используется симметричное шифрование, алгоритмы AES, Blowfish или 3DES. Целостность передачи данных проверяется с помощью CRC32 в SSH1 или HMAC-SHA1/HMAC-MD5 в SSH2.
Для сжатия шифруемых данных может использоваться алгоритм LempelZiv (LZ77).

Советы по безопасности использования SSH

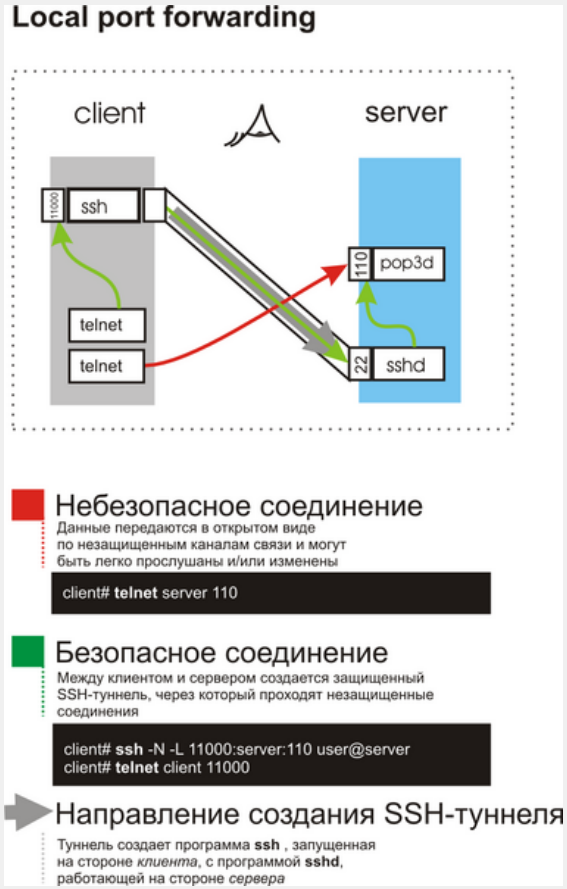
- Запрещение удалённого root-доступа.
- Запрещение подключения с пустым паролем или отключение входа по паролю.
- Выбор нестандартного порта для SSH-сервера.
- Использование длинных SSH2 RSA-ключей (2048 бит и более). Системы шифрования на основе RSA считаются надёжными, если длина ключа не менее 1024 бит.
- Ограничение списка IP-адресов, с которых разрешён доступ.
- Запрещение доступа с некоторых потенциально опасных адресов.
- Регулярный просмотр сообщений об ошибках аутентификации.
- Установка систем обнаружения вторжений (IDS).

SSH-туннелирование

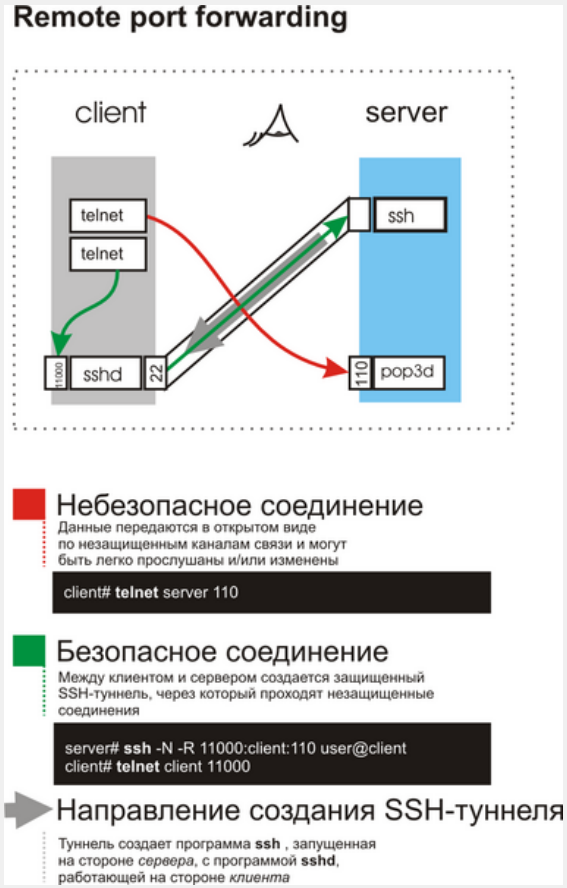
SSH-туннелирование

SSH-туннель — это туннель, создаваемый посредством SSH-соединения и используемый для шифрования туннелированных данных. Используется для того, чтобы обезопасить передачу данных в Интернете (аналогичное назначение имеет IPsec). При пересылке через SSH-туннель незашифрованный трафик любого протокола шифруется на одном конце SSH-соединения и расшифровывается на другом.

Local Port Forwarding



Remote Port Forwarding



Источники

- Материалы с сайта `https://wikipedia.org/`
- Telecommunication technologies — телекоммуникационные технологии / Ю. А. Семенов. URL: `http://book.itep.ru/`

Сети связи
Протоколы, сервисы и услуги в Интернет и IP-сетях

Тема № 11
Протоколы передачи файлов. FTP, TFTP. SFTP

доц. каф. СС и ПД, к.т.н. С. С. Владимиров

2020 г.

Протокол FTP

FTP (File Transfer Protocol)

Стандартный протокол, предназначенный для передачи файлов по TCP-сетям.
Протокол построен на архитектуре «клиент–сервер» и использует разные сетевые соединения для передачи команд и данных между клиентом и сервером. Пользователи FTP могут пройти аутентификацию, передавая логин и пароль открытым текстом, или же, если это разрешено на сервере, они могут подключиться анонимно. Можно использовать протокол SSH для безопасной передачи, скрывающей (шифрующей) логин и пароль, а также шифрующей содержимое.
FTP является одним из старейших прикладных протоколов, появившимся задолго до HTTP, и даже до TCP/IP, в 1971 году. В первое время он работал поверх протокола NCP (Network Control Protocol — сетевой протокол, который был первым стандартом сетевого протокола в ARPAnet).

Особенность протокола FTP в том, что он использует множественное (как минимум двойное) подключение. При этом один канал является управляющим, через который поступают команды серверу и возвращаются его ответы (обычно через TCP-порт 21), а через остальные происходит собственно передача данных, по одному каналу на каждую передачу. Поэтому в рамках одной сессии по протоколу FTP можно передавать одновременно несколько файлов, причём в обоих направлениях. Для каждого канала данных открывается свой TCP порт, номер которого выбирается либо сервером, либо клиентом, в зависимости от режима передачи.
Протокол FTP имеет двоичный режим передачи, что сокращает накладные расходы трафика и уменьшает время обмена данными при передаче больших файлов. Протокол же HTTP обязательно требует кодирования двоичной информации в текстовую форму, например при помощи алгоритма Base64.

Протокол определен в RFC 959. Сервер отвечает по потоку управления трехзначными ASCII-кодами состояния с необязательным текстовым сообщением. Например, «200» (или «200 OK») означает, что последняя команда была успешно выполнена. Цифры представляют код ответа, а текст — разъяснение или запрос. Текущая передача по потоку данных может быть прервана с помощью прерывающего сообщения, посылаемого по потоку управления.

Режимы работы FTP

Режимы работы FTP

FTP может работать в активном или пассивном режиме, от выбора которого зависит способ установки соединения. В активном режиме клиент создаёт управляющее TCP-соединение с сервером и отправляет серверу свой IP-адрес и произвольный номер клиентского порта, после чего ждёт, пока сервер не запустит TCP-соединение с этим адресом и номером порта. В случае, если клиент находится за брандмауэром и не может принять входящее TCP-соединение, может быть использован пассивный режим. В этом режиме клиент использует поток управления, чтобы послать серверу команду PASV, и затем получает от сервера его IP-адрес и номер порта, которые затем используются клиентом для открытия потока данных с произвольного клиентского порта к полученному адресу и порту.

Режимы передачи данных в FTP

1. Поточный режим — данные посылаются в виде непрерывного потока, освобождая FTP от выполнения какой бы то ни было обработки. Вместо этого, вся обработка выполняется TCP. Индикатор конца файла не нужен, за исключением разделения данных на записи.
2. Блочный режим — FTP разбивает данные на несколько блоков (блок заголовка, количество байт, поле данных) и затем передаёт их TCP.
3. Режим сжатия — данные сжимаются единым алгоритмом (обыкновенно, кодированием длин серий).

Представление данных и аутентификация в FTP

Форматы представления данных в FTP

1. ASCII — используется для текста. Данные, если необходимо, до передачи конвертируются из символьного представления на хосте-отправителе в "восьмибитный ASCII и (опять же, если необходимо) в символьное представление принимающего хоста. Как следствие, этот режим не подходит для файлов, содержащих не только обычный текст.
2. Режим изображения (бинарный режим) — устройство-отправитель посылает каждый файл байт за байтом, а получатель сохраняет поток байтов при получении. Поддержка данного режима была рекомендована для всех реализаций FTP.
3. EBCDIC — используется для передачи обычного текста между хостами в кодировке EBCDIC. В остальном, этот режим аналогичен ASCII-режиму.
4. Локальный режим — позволяет двум компьютерам с идентичными установками посылать данные в собственном формате без конвертации в ASCII.

Аутентификация

FTP-аутентификация использует обычную схему имя пользователя/пароль для предоставления доступа. Имя пользователя посылается серверу командой USER, а пароль - командой PASS. Если предоставленная клиентом информация принята сервером, то сервер отправит клиенту приглашение и начинается сессия. Пользователи могут, если сервер поддерживает эту особенность, войти в систему без предоставления учётных данных, но сервер может предоставить только ограниченный доступ для таких сессий.

Анонимный FTP

FTP-сервер, может предоставить анонимный доступ к FTP. Традиционно для этого используются логины «anonymous» и «ftp». Хотя обычно пользователей просят прислать адрес их электронной почты вместо пароля, никакой проверки фактически не производится. Многие FTP-хосты, предоставляющие обновления программного обеспечения, поддерживают анонимный доступ.

Безопасность в FTP

FTP не разрабатывался как защищённый (особенно по нынешним меркам) протокол и имеет многочисленные уязвимости в защите.
FTP не может зашифровать свой трафик, все передачи — открытый текст, поэтому имена пользователей, пароли, команды и данные могут быть прочитаны кем угодно, способным перехватить пакет по сети. Эта проблема характерна для многих спецификаций Интернет-протокола (в их числе SMTP, Telnet, POP, IMAP), разработанных до создания таких механизмов шифрования, как TLS и SSL. Обычное решение этой проблемы — использовать "безопасные TLS-защищенные версии уязвимых протоколов (FTPS для FTP, TelnetS для Telnet и т. д.) или же другой, более защищённый протокол, вроде SFTP/SCP, предоставляемого с большинством реализаций протокола Secure Shell.

FTPS

Явный FTPS — расширение стандарта FTP, позволяющее клиентам требовать того, чтобы FTP-сессия была зашифрована. Это реализуется отправкой команды "AUTH TLS". Сервер обладает возможностью позволить или отклонить соединения, которые не запрашивают TLS. Это расширение протокола определено в спецификации RFC 4217. Неявный FTPS — устаревший стандарт для FTP, требующий использования SSL- или TLS-соединения. Этот стандарт должен был использовать отличные от обычного FTP порты.

FTP через SSH

Относится к практике туннелирования обычной FTP-сессии через SSH-соединение. Поскольку FTP использует несколько TCP-соединений, туннелирование через SSH особенно затруднительно. Когда много SSH-клиентов пытаются установить туннель для канала управления (изначальное "клиент-сервер"соединение по порту 21), защищён будет только этот канал; при передаче данных программное обеспечение FTP на любом конце установит новые TCP-соединения (каналы данных), которые обойдут SSH-соединение и, таким образом, лишатся целостной защиты.

Коды ответов FTP

Код ответа — трёхзначное число. Первая цифра отвечает за один из трёх исходов: успех, отказ или указание на ошибку либо неполный ответ.

2xx — Успешный ответ

4xx/5xx — Команда не может быть выполнена

1xx/3xx — Ошибка или неполный ответ

Вторая цифра определяет тип ошибки:

x0z — Синтаксическая.

x1z — Информация. Соответствует информационному сообщению.

x2z — Соединения. Сообщение относится к управляющему соединению либо к соединению данных.

x3z — Соответствует сообщениям об аутентификации пользователя и его правах.

x4z — Не определено.

x5z — Файловая система. Соответствует сообщению о состоянии файловой системы.

Третья цифра окончательно специфицирует ошибку.

Протокол FXP

FXP (File eXchange Protocol — протокол обмена файлами)

Способ передачи файлов между двумя FTP-серверами напрямую, не закачивая их на свой компьютер. При FXP-сессии клиент открывает два FTP-соединения к двум разным серверам, запрашивая файл на первом сервере, указывая в команде PORT IP-адрес второго сервера.

Несомненным преимуществом поддержки стандарта FXP является то, что на конечных пользователей, желающих скопировать файлы с одного FTP-сервера на другой, уже не действует ограничение пропускной способности их собственного интернет-соединения. Нет необходимости скачивать себе файл, чтобы потом загрузить его на другой FTP-сервер. Таким образом, время передачи файлов будет зависеть только от скорости соединения между двумя удаленными FTP-серверами, которая в большинстве случаев заведомо больше «пользовательской».

FXP стал использоваться злоумышленниками для атак на другие серверы: в команде PORT указывается IP-адрес и порт атакуемого сервиса на компьютере жертвы, и командами RETR/STOR производится обращение на этот порт от лица FTP-сервера, а не атакующей машины, что позволяло устраивать масштабные DDoS-атаки с использованием сразу многих FTP-серверов, либо обходить систему безопасности компьютера жертвы, если он полагается только на проверку IP клиента и используемый для атаки FTP-сервер находится в доверенной сети или на шлюзе. В результате сейчас практически все серверы проверяют соответствие IP-адреса, указанного в команде PORT, IP-адресу FTP-клиента и по умолчанию запрещают использование там IP-адресов третьих сторон. Таким образом, использование FXP невозможно при работе с публичными FTP-серверами.

Протокол TFTP

TFTP (Trivial File Transfer Protocol — простой протокол передачи файлов)

Используется главным образом для первоначальной загрузки бездисковых рабочих станций. В отличие от FTP, не содержит возможностей аутентификации (хотя возможна фильтрация по IP-адресу) и основан на транспортном протоколе UDP.
TFTP был разработан в 1980 году и определен в RFC 1350 (STD 33). За ним закреплен 69 порт UDP.
Основное назначение TFTP — обеспечение простоты реализации клиента. В связи с этим он используется для загрузки бездисковых рабочих станций, загрузки обновлений и конфигураций в «умные» сетевые устройства, записи статистики с мини-АТС и аппаратных маршрутизаторов/файрволов.

Безопасность в TFTP

Поскольку протокол не поддерживает аутентификации, единственный метод идентификации клиента — это его сетевой адрес. Обычно в Unix-системах серверу `tftpd` доступен только каталог `/tftpboot`. Однако в старых TFTP-серверах было возможным получить файл паролей командой `RRQ ../etc/passwd`.
Демон `tftpd` (одна из реализаций tftp-сервера) отказывается обрабатывать файлы, содержащие в своём имени комбинацию «`/../`» или начинающуюся с «`../`». Запись разрешается только в файлы, которые уже существуют (любого размера, например нулевого), и доступны для публичной записи (права доступа: `-rw-rw-rw-`).
Дополнительная защита от доступа к произвольным файлам осуществляется с помощью смены корневого каталога на каталог `tftpd` (обычно `/usr/TFTPRoot`).

Протокол SCP

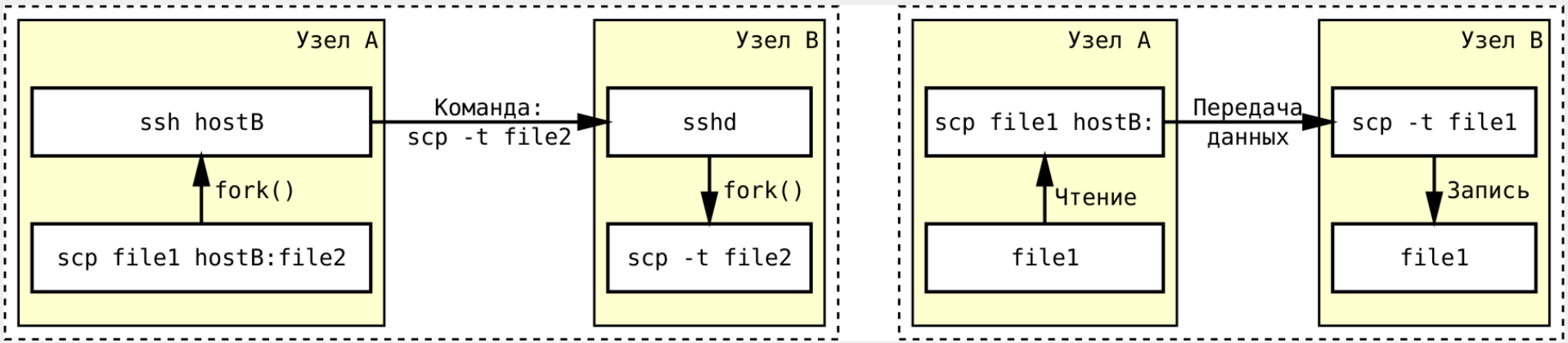
SCP (Secure Copy Protocol)

Сетевой протокол прикладного уровня (и одноименная утилита), предназначенные для передачи данных между двумя сетевыми узлами (локальным и удаленным или парой удаленных). Работает поверх протокола SSH, используя его механизмы аутентификации и защиты данных при передаче. Соответственно, так же использует 22 порт TCP.
SCP основан на протоколе/утилите RCP (Secure Copy), являвшейся частью пакета программ Berkeley r-commands (1982), который долгое время использовался в ОС Unix и стал стандартом де-факто для удаленного управления в Unix. R-commands не обеспечивали необходимый уровень безопасности (в частности, данные не шифровались при передаче) и были заменены на SSH вскоре после его появления.

Порядок работы SCP

SCP при передаче использует клиент-серверный принцип. Вначале на удаленной стороне запускается SCP в режиме записи или чтения файла (используются скрытые опции утилиты: `-t` и `-f`, соответственно), а затем производится подключение со стороны локальной машины. При работе с директорией целиком используется дополнительная скрытая опция `-d`.

Принцип передачи файла на удаленный хост



Протокол SCP. Режимы работы и команды

Режимы работы SCP на удаленном узле

1. Режим чтения (source mode). Соответствует флагу `-f`.
2. Режим записи (sink mode). Соответствует флагу `-t`.

Команды протокола SCP

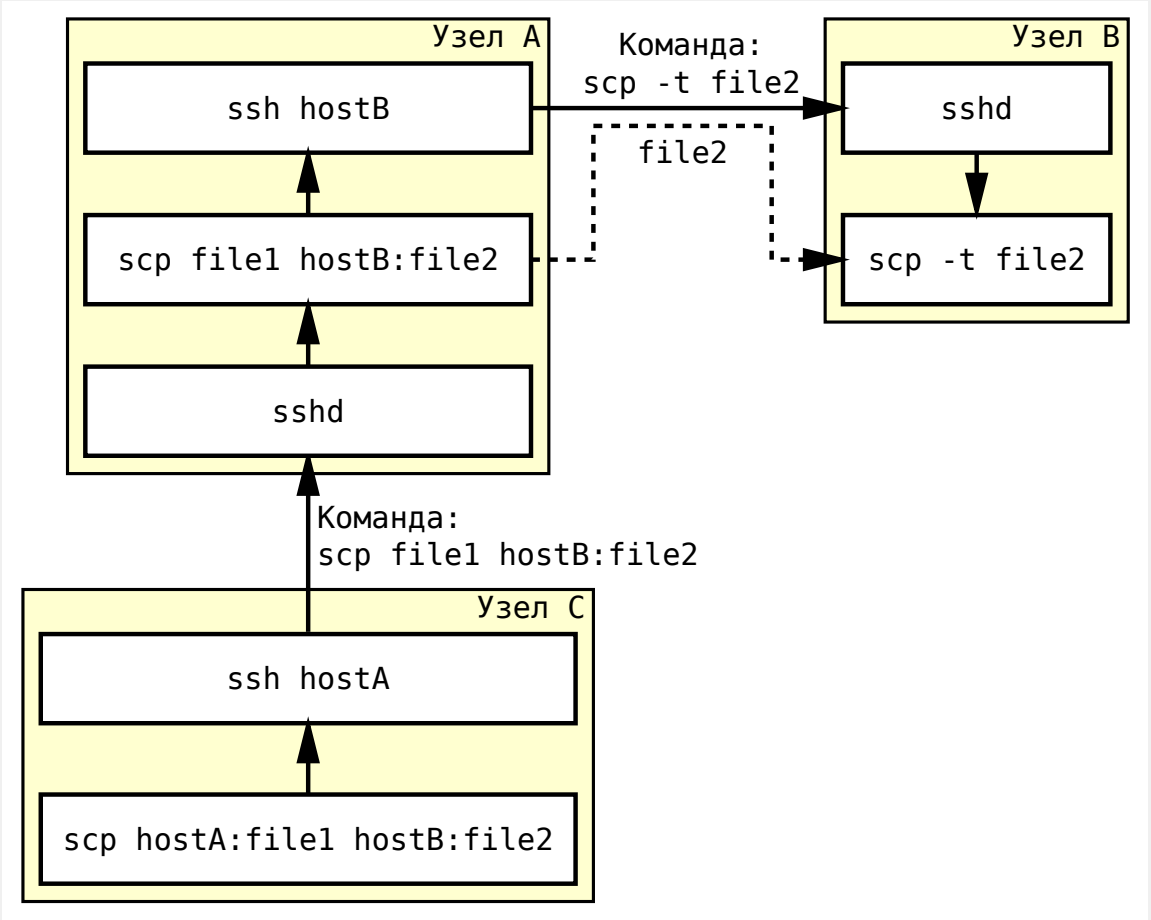
- `Cmmmm <length> <filename>` — Копирование одиночного файла: `mmmm` — права на файл; `<length>` — размер файла в байтах. Пример: `C0644 299 file1`. После этой команды должны следовать данные. Принимающая сторона считывает ровно столько данных, сколько указано в параметре `<length>`.
- `Dmmmm <length> <dirname>` — Рекурсивное копирование каталога. Параметр <length> игнорируется, но должен быть указан (обычно 0). Пример: `D0755 0 docs`. После этой команды должны следовать или команды передачи файла C, или команда E. На стороне «сервера» в этом случае необходимо использовать флаг «-r».
- `E` — Окончание рекурсивного копирования каталога.
- `T<mtime> 0 <atime> 0` — время доступа и изменения файла/директории в формате UNIX (в секундах от 00:00:00 UTC, 1 января 1970). Нули были введены на тот случай, если понадобится точность до микросекунд. Пример: `T1183828267 0 1183828267 0`. Передается до команд передачи файлов.

Размер передаваемых файлов

Поскольку в реализации протокола OpenSSH для параметра `<length>` используется переменная типа `long long int`, максимально можно передать 2^{63} байт (чуть больше $9 \cdot 10^{18}$ байт). Учитывая, что даже 2^{40} байт — это примерно 1 ТБ, можно считать, что ограничений на размер передаваемого файла нет.
При передаче большого количества небольших файлов (или передаче директории) оптимально использовать предварительную архивацию. Это значительно увеличивает скорость передачи по сети (даже без сжатия архива) за счет уменьшения накладных расходов на обработку каждого файла в отдельности.

Протокол SCP. Передача между удаленными узлами

Передача между удаленными узлами



При таком способе передачи не используется парольная аутентификация между узлами A и B. Допустимо использование host-based аутентификации или аутентификации по публичному ключу (незапароленному).

Протокол SFTP

SFTP (SSH File Transfer Protocol)

Протокол прикладного уровня, предназначенный для копирования и выполнения других операций с файлами поверх надёжного и безопасного соединения. Протокол разработан группой IETF как расширение к SSH-2, однако SFTP допускает реализацию и с использованием иных протоколов сеансового уровня.
Протокол предполагает, что он работает поверх установленного безопасного канала, что сервер уже аутентифицировал клиента и что идентификатор клиента доступен протоколу. Сервер SFTP обычно использует TCP порт 22.
SSH File Transfer Protocol не является протоколом FTP работающим поверх SSH — это другой, новый протокол. Также SFTP иногда путают с Simple File Transfer Protocol из-за совпадающего сокращения «SFTP».

В сравнении с другим протоколом, тоже предназначенным для копирования файлов поверх SSH — протоколом SCP, который позволяет только копировать файлы, SFTP даёт возможность выполнять намного больше операций с ними: например, докачивать файл после разрыва соединения или удалять файл на сервере и многие другие операции. По этой причине существуют графические и псевдографические клиенты для SFTP, но нет таких, кто использовал бы только SCP в чистом виде.

Сам по себе протокол SFTP не обеспечивает безопасность работы; это делает нижележащий протокол. Как правило, SFTP используется в сочетании с протоколом SSH2. Можно использовать SFTP и с другими протоколами, например SSH1, но это сопряжено с дополнительными трудностями.
Разработкой протокола занималась одна из групп IETF под названием Secsh — группа, ранее подготовившая стандарт SSH-2. Рабочая документация к новому протоколу SFTP не стала официальным стандартом, однако начала активно применяться для разработки приложений. В ходе развития протокола было выпущено шесть версий протокола. Постепенное наращивание функциональности протокола привело к тому, что 14 августа 2006 года было принято решение о прекращении работы над развитием протокола в связи с выполнением основной задачи проекта (разработка SSH) и отсутствием достаточного экспертного уровня для перехода к разработке полноценного протокола удалённой файловой системы.
Последней разработанной версией протокола является Draft 13 от 10 июля 2006 года.

Источники

- Материалы с сайта `https://wikipedia.org/`
- Telecommunication technologies — телекоммуникационные технологии / Ю. А. Семенов.
  URL: `http://book.itep.ru/`
- SFTP // Xgu.ru. URL: `http://xgu.ru/wiki/Sftp`
- How the SCP protocol works / Jan Pechanec.
  URL: `https://web.archive.org/web/20170215184048/https://blogs.oracle.com/janp/entry/how_the_scp_protocol_works`

Сети связи
Протоколы, сервисы и услуги в Интернет и IP-сетях

Тема № 12
Система доменных имён. Протокол DNS

доц. каф. СС и ПД, к.т.н. С. С. Владимиров

2020 г.

Система доменных имён DNS

DNS (Domain Name System — система доменных имён)

Компьютерная распределённая система для получения информации о доменах. Чаще всего используется для получения IP-адреса по имени хоста (компьютера или устройства), получения информации о маршрутизации почты, обслуживающих узлах для протоколов в домене (SRV-запись). Для работы протокола испольузуются протоколы TCP и UDP. Порт 53.

DNS была разработана Джоном Постелом и Полом Мокапетрисом из Института информационных наук Университета Южной Каролины в 1983 году. Оригинальное описание механизмов работы содержится в RFC 882 и RFC 883 с обновлением в RFC 973 (1986 год). В 1987 были опубликованы RFC 1034 и RFC 1035 с изменениями спецификации DNS, которые отменили RFC 882, 883 и 973 как устаревшие. Эти RFC были подписаны Полом Мокапетрисом, поэтому именно его называют автором системы DNS. Позднее в 1999 году он создал компанию Nominum, которая участвовала в разработке DNS-сервера BIND 9 (Berkeley Internet Name Domain) и протокола DHCPv6.

Распределённая база данных DNS поддерживается с помощью иерархии DNS-серверов, взаимодействующих по определённому протоколу.
Основой DNS является представление об иерархической структуре доменного имени и зонах. Каждый сервер, отвечающий за имя, может делегировать ответственность за дальнейшую часть домена другому серверу (с административной точки зрения — другой организации или человеку), что позволяет возложить ответственность за актуальность информации на серверы различных организаций (людей), отвечающих только за «свою» часть доменного имени.
Начиная с 2010 года, в систему DNS внедряются средства проверки целостности передаваемых данных, называемые DNS Security Extensions (DNSSEC). Передаваемые данные не шифруются, но их достоверность проверяется криптографическими способами. Внедряемый стандарт DANE обеспечивает передачу средствами DNS достоверной криптографической информации (сертификатов), используемых для установления безопасных и защищённых соединений транспортного и прикладного уровней.

Система доменных имён DNS

DNS важна для работы Интернета, так как для соединения с узлом необходима информация о его IP-адресе, а для людей проще запоминать буквенные (обычно осмысленные) адреса, чем последовательность цифр IP-адреса. В некоторых случаях это позволяет использовать виртуальные серверы, например, HTTP-серверы, различая их по имени запроса. Первоначально преобразование между доменными и IP-адресами производилось с использованием специального текстового файла `hosts`. Этот файл был уникальным и формировался в единственном экземпляре на сервере, размещенном в Стэнфордском исследовательском институте (SRI International). Как своего рода рудимент того времени, в усеченном виде этот файл сохраняется в большинстве операционных систем, но сейчас он зачастую содержит всего одну запись `localhost` с указанием на адрес `127.0.0.1` и иногда записи для основных multicast-адресов IPv6. Тем не менее, файл `hosts` до настоящего времени имеет преимущество перед системой DNS — если в файле `hosts` есть запись для определенного доменного имени, то обращения к системе DNS для разрешения этого имени проводиться не будет. Эта особенность широко используется администраторами при тестировании сетей, а также вирусописателями, которые встраивают в файл `hosts` запись, отправляющую пользователя на сервер злоумышленника, что позволяет ему перехватить данные пользователя, например, пароли.

Пример файла `/etc/hosts`

```
127.0.0.1       localhost
127.0.1.1       pcname

# The following lines are desirable for IPv6 capable hosts
::1     localhost ip6-localhost ip6-loopback
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
```

Ключевые характеристики и понятия DNS

Ключевые характеристики DNS

- *Распределённость администрирования.* Ответственность за разные части иерархической структуры несут разные люди или организации.
- *Распределённость хранения информации.* Каждый узел сети в обязательном порядке должен хранить только те данные, которые входят в его зону ответственности, и (возможно) адреса корневых DNS-серверов.
- *Кеширование информации.* Узел может хранить некоторое количество данных не из своей зоны ответственности для уменьшения нагрузки на сеть.
- *Иерархическая структура*, в которой все узлы объединены в дерево, и каждый узел может или самостоятельно определять работу нижестоящих узлов, или делегировать (передавать) их другим узлам.
- *Резервирование.* За хранение и обслуживание своих узлов (зон) отвечают (обычно) несколько серверов, разделённые как физически, так и логически, что обеспечивает сохранность данных и продолжение работы даже в случае сбоя одного из узлов.

Домен (domain — область)

Узел в дереве имён, вместе со всеми подчинёнными ему узлами (если таковые имеются), то есть именованная ветвь или поддерево в дереве имен. Структура доменного имени отражает порядок следования узлов в иерархии; доменное имя читается слева направо от младших доменов к доменам высшего уровня (в порядке повышения значимости): вначале корневой домен (не имеющий идентификатора), ниже идут домены первого уровня (доменные зоны), затем — домены второго уровня, третьего и т. д.

Поддомен (subdomain)

Подчинённый домен. Теоретически деление может достигать глубины 127 уровней, а каждая метка может содержать до 63 символов, пока общая длина вместе с точками не достигнет 254 символов. На практике больше трёх уровней встречается крайне редко.

Ключевые понятия DNS

Ресурсная запись

Единица хранения и передачи информации в DNS. Каждая ресурсная запись имеет имя (то есть привязана к определенному Доменному имени, узлу в дереве имен), тип и поле данных, формат и содержание которого зависит от типа.

Зона

Часть дерева доменных имен (включая ресурсные записи), размещаемая как единое целое на некотором сервере доменных имен, а чаще — одновременно на нескольких серверах. Целью выделения части дерева в отдельную зону является передача ответственности за соответствующий домен другому лицу или организации (делегирование).

Делегирование

Операция передачи ответственности за часть дерева доменных имен другому лицу или организации. За счет делегирования в DNS обеспечивается распределенность администрирования и хранения. Технически делегирование выражается в выделении этой части дерева в отдельную зону, и размещении этой зоны на DNS-сервере, управляемом этим лицом или организацией.

Ключевые понятия DNS

DNS-сервер

Специализированное ПО для обслуживания DNS, а также компьютер, на котором это ПО выполняется.

DNS-клиент

Специализированная библиотека (или программа) для работы с DNS. В ряде случаев DNS-сервер выступает в роли DNS-клиента.

Авторитетность

Признак размещения зоны на DNS-сервере. Ответы DNS-сервера могут быть двух типов: авторитетные (когда сервер заявляет, что сам отвечает за зону) и неавторитетные (англ. Non-authoritative), когда сервер обрабатывает запрос, и возвращает ответ других серверов. В некоторых случаях вместо передачи запроса дальше DNS-сервер может вернуть уже известное ему (по запросам ранее) значение (режим кеширования).

DNS-запрос (DNS query)

Запрос от клиента (или сервера) серверу. Запрос может быть рекурсивным или нерекурсивным.

Система DNS содержит иерархию DNS-серверов, соответствующую иерархии зон. Каждая зона поддерживается как минимум одним авторитетным сервером DNS (authoritative — авторитетный), на котором расположена информация о домене.

Корневой домен

Корневой домен (нулевой домен, root domain)

Домен самого верхнего уровня в любой системе доменных имён. Обслуживается корневыми серверами системы доменных имен, которые располагаются в различных странах мира.

Обозначается пустым именем. При записи доменного имени, каждый домен отделяется точкой; в конце имени может присутствовать точка, которая отделяет пустое имя, соответствующее корневому домену. Если эта точка есть (например «`www.example.com.`»), то доменное имя считается полным (абсолютным). Если точки в конце имени нет («`www.example`» или «`www.example.com`»), то имя считается относительным.

Каждое интернет-приложение должно правильно обрабатывать завершающую точку, однако большинство приложений позволяют вводить доменное имя без точки в конце; обработка зависит от реализации. В простейшем случае к адресу добавляется завершающая точка, и он трактуется как абсолютный. В ряде случаев для получения полного доменного имени локальное программное обеспечение (либо приложение, либо операционная система) может присоединить к относительному имени некоторый домен по умолчанию, который определяется по доменному имени компьютера или может быть задан в настройках. Иногда в настройках может быть задано несколько таких доменов, которые перебираются по очереди, до тех пор, пока не будет найдено существующее в DNS имя. Такой подход может приводить к неоднозначности, которая может быть разрешена с помощью задания полного имени.

Корневые серверы DNS

Корневые серверы DNS

DNS-серверы, содержащие информацию о доменах верхнего уровня, указывающую на DNS-серверы, поддерживающие работу каждого из этих доменов. Основные корневые серверы DNS размещены в домене `root-servers.org` и обозначаются латинскими буквами от `A` до `M`. Они управляются различными организациями, действующими по согласованию с ICANN. Из-за существовавших в прошлом ограничений на размеры DNS-пакета (512 байт) в DNS-ответ могло быть помещено всего 13 серверов (от `A` до `M`), сейчас за этими 13 именами стоят более 200 серверов. В частности, российское зеркало сервера `F` расположено в РосНИИРОС в Москве, сервера `K` в Новосибирске, а сервера `L` в Ростове-на-Дону. Ближайший (к пользователю) адрес «зеркала» корневого сервера выбирается автоматически благодаря IP AnyCast. Так, при обращении к `K.root-servers.net`, пользователь из Новосибирска скорее всего обратится к новосибирскому серверу.

Альтернативные корневые серверы DNS

Различные организации управляют альтернативными корневыми DNS-серверами. Альтернативные системы доменных имён используют собственные DNS-серверы и управляют пространствами имён, состоящими из собственных доменов верхнего уровня. Совет по архитектуре Интернета высказался категорически против альтернативных корневых серверов в RFC 2826.

- Chaos Computer Club DNS
- OpenNIC
- Google Public DNS
- Open Root Server Confederation
- Open Root Server Network

Некоторые из альтернативных систем DNS предоставляют новые домены верхнего уровня.

Особенности доменных имен

Псевдоинтернациональные домены

Домены верхнего уровня, чьи аббревиатуры созвучны с теми или иными сокращениями. Так, домен Тувалу `.tv` широко используется как ненациональный домен верхнего уровня для телевидения; домен Федеративных Штатов Микронезии `.fm` — для FM-радиостанций; домен Острова Мэн `.im` — для интернет-мессенджеров; домен Туркменистана `.tm` — для товарных знаков; домен Молдавии `.md` — для медицинских структур; домен Западного Самоа `.ws` — для веб-сайтов; а домен Лаоса `.la` — для организаций, зарегистрированных в Лос-Анджелесе.

Зарезервированные доменные имена

Документ RFC 2606 (Reserved Top Level DNS Names — Зарезервированные имена доменов верхнего уровня) определяет названия доменов, которые следует использовать в качестве примеров (например, в документации), а также для тестирования. Кроме `example.com`, `example.org` и `example.net`, в эту группу также входят `test`, `invalid` и др.

Интернациональные доменные имена

Доменное имя может состоять только из ограниченного набора ASCII символов, позволяя набрать адрес домена независимо от языка пользователя. ICANN утвердил основанную на Punycode систему IDNA, преобразующую любую строку в кодировке Unicode в допустимый DNS набор символов.

Имя и IP-адрес не тождественны

Один IP-адрес может иметь множество имён, что позволяет поддерживать на одном компьютере множество веб-сайтов (виртуальный хостинг). Обратное тоже справедливо — одному имени может быть сопоставлено множество IP-адресов: это позволяет создавать балансировку нагрузки.

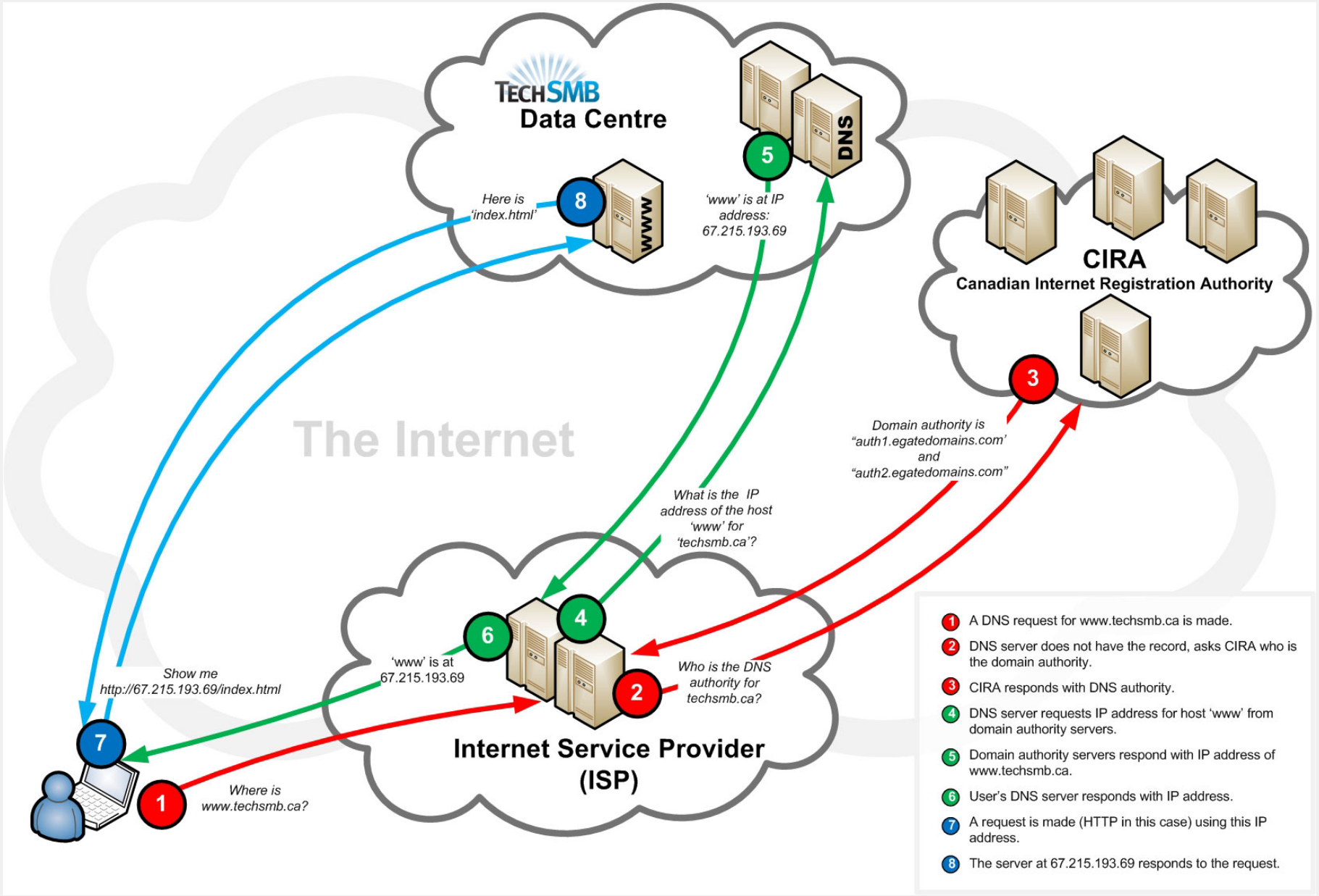
Рекурсия в DNS

Рекурсия

Алгоритм поведения DNS-сервера, при котором сервер выполняет от имени клиента полный поиск нужной информации во всей системе DNS, при необходимости обращаясь к другим DNS-серверам.

DNS-запрос может быть рекурсивным — требующим полного поиска, — и нерекурсивным (или итеративным) — не требующим полного поиска.
Аналогично, DNS-сервер может быть рекурсивным (умеющим выполнять полный поиск) и нерекурсивным (не умеющим выполнять полный поиск). Некоторые программы DNS-серверов, например, BIND, можно сконфигурировать так, чтобы запросы одних клиентов выполнялись рекурсивно, а запросы других — нерекурсивно.
При ответе на нерекурсивный запрос, а также при неумении или запрете выполнять рекурсивные запросы, DNS-сервер либо возвращает данные о зоне, за которую он ответствен, либо возвращает ошибку. Настройки нерекурсивного сервера, когда при ответе выдаются адреса серверов, которые обладают большим объёмом информации о запрошенной зоне, чем отвечающий сервер (чаще всего — адреса корневых серверов), являются некорректными и такой сервер может быть использован для организации DoS-атак.
В случае рекурсивного запроса DNS-сервер опрашивает серверы (в порядке убывания уровня зон в имени), пока не найдёт ответ или не обнаружит, что домен не существует. (На практике поиск начинается с наиболее близких к искомому DNS-серверов, если информация о них есть в кэше и не устарела, сервер может не запрашивать другие DNS-серверы.) Иногда допускается, чтобы запрошенный сервер передавал рекурсивный запрос «вышестоящему» DNS-серверу и дожидался готового ответа. При рекурсивной обработке запросов все ответы проходят через DNS-сервер, и он получает возможность кэшировать их. Повторный запрос на те же имена обычно не идет дальше кэша сервера, обращения к другим серверам не происходит вообще. Допустимое время хранения ответов в кэше приходит вместе с ответами (поле TTL ресурсной записи).
Рекурсивные запросы требуют больше ресурсов от сервера (и создают больше трафика), так что обычно принимаются от «известных» владельцу сервера узлов (например, провайдер предоставляет возможность делать рекурсивные запросы только своим клиентам, в корпоративной сети рекурсивные запросы принимаются только из локального сегмента). Нерекурсивные запросы обычно принимаются ото всех узлов сети, но содержательный ответ даётся только на запросы о зоне, размещенной на узле, а в ином случае возвращаются адреса других серверов.

Пример выполнения DNS-запроса



Записи DNS

Записи DNS, или Ресурсные записи (Resource Records, RR)

Единицы хранения и передачи информации в DNS. Каждая ресурсная запись состоит из следующих полей:

NAME	Доменное имя, к которому привязана или которому «принадлежит» данная ресурсная запись.
TTL	Time To Live — допустимое время хранения данной ресурсной записи в кэше неответственного DNS-сервера.
TYPE	Тип ресурсной записи — определяет формат и назначение данной ресурсной записи.
CLASS	Класс ресурсной записи. Теоретически считается, что DNS может использоваться не только с TCP/IP, но и с другими типами сетей, код в поле класс определяет тип сети.
RDLEN	Длина поля данных.
RDATA	Поле данных, формат и содержание которого зависит от типа записи.

Важные типы DNS-записей

A	Address record. Связывает имя хоста с адресом протокола IPv4. Например, запрос A-записи на имя `referrals.icann.org` вернёт его IPv4-адрес — `192.0.34.164`.
AAAA	IPv6 address record. Связывает имя хоста с адресом протокола IPv6. Например, запрос AAAA-записи на имя `K.root-servers.net` вернёт его IPv6-адрес — `2001:7fd::1`.
CNAME	Canonical name record. Каноническая запись имени (псевдоним). Используется для перенаправления на другое имя.
MX	Mail exchange. Указывает сервер(ы) обмена почтой для данного домена.
NS	Name server. Указывает на DNS-сервер для данного домена.
SRV	Server selection. Указывает на серверы для сервисов, используется, в частности, для Jabber и Active Directory.

Записи DNS

Важные типы DNS-записей

PTR — Pointer — запись указателя. Связывает IP-адрес хоста с его каноническим именем. Запрос в домене `in-addr.arpa` на IP-адрес хоста в обратной форме вернёт имя данного хоста. Например, для IP-адреса `192.0.34.164` запрос записи PTR `164.34.0.192.in-addr.arpa` вернёт его каноническое имя `referrals.icann.org`. В целях уменьшения объёма нежелательной корреспонденции (спама) многие серверы-получатели электронной почты могут проверять наличие PTR-записи для хоста, с которого происходит отправка. В этом случае PTR-запись для IP-адреса должна соответствовать имени отправляющего почтового сервера, которым он представляется в процессе SMTP-сессии.

SOA — Start of Authority — начальная запись зоны. Указывает, на каком сервере хранится эталонная информация о данном домене, содержит контактную информацию лица, ответственного за данную зону, тайминги (параметры времени) кеширования зонной информации и взаимодействия DNS-серверов.

Пример запроса DNS

```
user@pcname:~$ host -a sut.ru
Trying ''sut.ru''
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 14859
;; flags: qr rd ra; QUERY: 1, ANSWER: 7, AUTHORITY: 0, ADDITIONAL: 0
;; QUESTION SECTION:
   ;sut.ru.             IN      ANY
;; ANSWER SECTION:
sut.ru.      21599   IN      SOA     sut.ru. noc.sut.ru. 2022041125 10800 3600 3600000 86400
sut.ru.      21599   IN      A       91.238.228.4
sut.ru.      21599   IN      MX      10 mail.sut.ru.
sut.ru.      21599   IN      NS      ns.sut.ru.

Received 176 bytes from 8.8.8.8#53 in 12 ms
```

Обратный DNS-запрос

Обратный DNS-запрос

DNS используется в первую очередь для преобразования символьных имён в IP-адреса, но он также может выполнять обратный процесс. Для этого используются уже имеющиеся средства DNS. Дело в том, что с записью DNS могут быть сопоставлены различные данные, в том числе и какое-либо символьное имя. Существует специальный домен `in-addr.arpa`, записи в котором используются для преобразования IP-адресов в символьные имена. Например, для получения DNS-имени для адреса `11.22.33.44` можно запросить у DNS-сервера запись `44.33.22.11.in-addr.arpa`, и тот вернёт соответствующее символьное имя. Обратный порядок записи частей IP-адреса объясняется тем, что в IP-адресах старшие биты расположены в начале, а в символьных DNS-именах старшие (находящиеся ближе к корню) части расположены в конце.

Пример обратного запроса

```
user@pcname:~$ host -a 4.228.238.91.in-addr.arpa
Trying ”4.228.238.91.in-addr.arpa”
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 22978
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 0

;; QUESTION SECTION:
;4.228.238.91.in-addr.arpa.     IN      ANY

;; ANSWER SECTION:
4.228.238.91.in-addr.arpa. 3599 IN      PTR     mail.sut.ru.

Received 68 bytes from 8.8.8.8#53 in 13 ms
```

Динамический DNS

Динамический DNS

Технология, позволяющая информации на DNS-сервере обновляться в реальном времени, и в автоматическом режиме. Она применяется для назначения постоянного доменного имени устройству (компьютеру, сетевому накопителю) с динамическим IP-адресом. Это может быть IP-адрес, полученный по DHCP или по IPCP в PPP-соединениях (например, при удалённом доступе через модем). Другие машины в Интернете могут устанавливать соединение с этой машиной по доменному имени и даже не знать, что IP-адрес изменился.
Время устаревания (TTL) для динамической записи делается очень маленьким (не более двух-трёх минут), иначе другие DNS-серверы поместят её в свой кэш, а когда она изменится — их клиенты долго будут получать устаревшую информацию.
Динамическая DNS часто применяется в локальных сетях, где клиенты получают IP-адрес по DHCP, а потом регистрируют свои имена в локальном DNS-сервере.

Протокол для обновления DNS описан в RFC 2136 и реализован, например, утилитой `nsupdate`. Для безопасной аутентификации клиента можно использовать технологию TSIG (RFC 2845), в которой используется заранее известный ключ. Минус этой технологии в том, что ключ должен быть установлен на каждом клиенте и на сервере.

Услуги динамического DNS предоставляют такие компании, как No-IP (дочерняя фирма компании Vitalwerks Internet Solutions, LLC.) и Dyn (сервис DynDNS).
Хостинговые и прочие компании, хранящие у себя DNS-информацию клиентов и позволяющие клиентам эту информацию изменять, по сути тоже предоставляют динамическую DNS. Чаще всего клиент может изменить информацию, зайдя через веб-интерфейс.

Регистратор доменных имён

Регистратор доменных имён

Организация, имеющая полномочия создавать (регистрировать) новые доменные имена и продлевать срок действия уже существующих доменных имён в домене, для которого установлена обязательная регистрация. Таковыми доменами являются:

- домен нулевого уровня (корневой домен);
- все домены верхнего уровня (первого уровня);
- некоторые домены второго уровня (например, com.ru или co.uk).

Во всех прочих доменах для создания поддоменов специальных полномочий не требуется.

Роль регистратора для корневого домена выполняет организация ICANN. Для многих доменов регистратор не единственный. При наличии нескольких регистраторов все они должны использовать единую (централизованную или распределённую) базу данных для исключения конфликтов и обеспечения уникальности доменного имени. Для того, чтобы стать регистратором доменов в зонах `.com`, `.net`, `.org`, `.biz`, `.info`, `.name`, `.mobi`, `.asia`, `.aero`, `.tel`, `.travel`, `.jobs`, необходимо получить аккредитацию ICANN.

Во многих случаях регистратор доменных имён прямо или косвенно контролируется государством. Например, регистратором домена `.mil` является подразделение Министерства обороны США (Defense Information Systems Agency), а создание российского Координационного центра национального домена курировало Министерство связи РФ.
Регистрация и продление регистрации домена осуществляется в разных доменах на разных условиях — от бесплатной до весьма дорогой (до 10 тыс. долларов). Как правило, финансовые условия одинаковые для всех владельцев (администраторов) поддоменов в рамках одного домена.

Сетевая DNS-атака: Подмена DNS-ответа; внедрение ложного DNS-сервера

Цели и угрозы

Подмена доверенных объектов сети; перехват практически любого траффика жертвы; подмена сетевых запросов/ответов.

Механизм атаки

Атакующий ждёт DNS-запроса от жертвы на хосте 1 (атакующий находится либо на хосте нарушителя 1, либо на хосте нарушителя 2; но может быть и где-либо ещё, где есть доступ к трафику хоста 1 (жертвы)).
После передачи хостом 1 DNS-запроса, атакующий принимает запрос, в котором запоминает ID и порт. Далее, такакующий отправляет ложный DNS-ответ, в котором подменяет поле IP-адрес DNS-сервера на свой IP, делая свой компьютер для жертвы валидным DNS-сервером.
Хост 1 принимает ложный DNS-ответ, принимает IP-адрес хакера за подлинный DNS-сервер и отправляет все последующие запросы ему.
Атакующий после получения DNS-запросов пересылает их на настоящий DNS-сервер, получает правильный ответ и пересылает его назад – жертве. Существует лёгкая возможность подменить в DNS-ответе IP любого запрашиваемого DNS-имени.

Сетевая DNS-атака:
Подмена DNS-ответа; внедрение ложного DNS-сервера

Схема реализации атаки в случае, если доступа к трафику жертвы нет

Атакующий не дожидается DNS-запроса (он его и не получит, ибо траффик жертвы через него не проходит), а отправляет массированный поток ложных DNS-ответов, подбирая на ходу нужные порт и ID запроса (простой перебор — bruteforce). При этом в ложном ответе атакующий подменяет поле IP-адрес DNS-сервера на свой IP, делая свой компьютер для жертвы валидным DNS-сервером.

Хост 1, отправив запрос, принимает ложный DNS-ответ, в котором указан IP-адрес хакера, как IP подлинного DNS-сервера. В итоге, жертва отправляет все последующие DNS-запросы злоумышленнику.

Атакующий после получения DNS-запросов пересылает их на настоящий DNS-сервер, получает правильный ответ и пересылает его назад – жертве. Существует лёгкая возможность подменить в DNS-ответе IP любого запрашиваемого DNS-имени.

В случае, если атакующий находится за отдельным маршрутизатором и не имеет доступа к трафику клиента, но находится в том же сегменте сети, что и DNS-сервер жертвы (сама жертва), схема остаётся почти тай же. Фаза 1 заменяется на фазу массированной отправки DNS-ответов, не дожидаясь запроса жертвы. В этом случае, жертва после запроса моментально получит ответ, один из которых окажется правильным.

Сетевая DNS-атака:
Атака на кеш DNS-сервера
Подмена вышестоящего DNS-сервера (атака Каминского).

Цели и угрозы

Подмена доверенных объектов сети; перехват практически любого траффика жертвы.

Схема реализации атаки

Злоумышленник посылает на целевой DNS-сервер (жертвы) запрос, которого заведомо нет в его кеше (или поток потенциальных запросов, в один прекрасный момент случится ситуация, когда в кеше локального DNS-сервера ответа не будет).
Далее злоумышленник создает направленный шторм ложных DNS-ответов от имени одного из вышестоящих (можно взять и корневые, как на картинке) DNS-серверов. Ситуация с подменой DNS-ответа от имени корневого проще: если в предыдущей схеме нужно было подбирать порт и ID, то в текущей подбирать порт не требуется: он стандартизован и постоянен во всех соединениях между DNS-серверами: порт 53.
DNS-сервер передает DNS-запрос на вышестоящий (корневой) DNS-сервер и немедленно получает ложный DNS-ответ от атакующего.
Хост нарушителя изменяет кэш-таблицу DNS-сервера и обеспечивает прохождение трафика через подставной хост злоумышленника (адресу `top.secret.com` в кеше DNS будет соответствовать ложный IP) по тому же алгоритму, что и в описанных выше схемах.
Как видно, схема реализации атаки достаточно проста и даже не требует разворачивания собственного DNS-сервера, что делает её очень удобной и привлекательной для злоумышленников.
В результате атаки, подмена целевых DNS-ответов (IP-адресов DNS-имён) происходит не только у одного хоста-жертвы, а у всех пользователей данного DNS-сервера.

Сетевая DNS-атака:
Атака посредством отражённых DNS-запросов

Цели и угрозы

Выведение хоста жертвы (в т. ч. DNS-сервера) из строя; падение шлюза (канала доступа в Интернет)/межсетевого экрана.

Схема реализации атаки

Атака посредством отражённых DNS основывается на том, что DNS-ответ всегда в 3-4 раза длиннее запроса. В некоторых случаях (у некоторых DNS-имён) размер ответного пакета может в 10 и более раз превышать размер запроса.
Злоумышленник отправляет DNS-запросы на один или несколько сторонних DNS-серверов, которые не являются реальными объектами нападения, с подменённым IP адресом источника: IP источника равен IP хоста-жертвы.
DNS-сервера получив запрос, формируют и отправляют ответы (намного длиннее запросов) на подложный IP (т.е. жертве). Источник оказывается под мощным штормом DNS-ответов и не выдерживая нагрузки, падает.
Данный тип атаки может быть организован в условиях довольно ограниченных ресурсов, достигая 4–10-кратного эффекта усиления атаки. Если при этом злоумышленник ещё и создаст (или подберет) определенные домены, для отправки имен которых требуются DNS-пакеты огромных размеров, то отправляя запросы только на такие доменные имена, злоумышленник может достигать 100-кратного усиления эффекта атаки.
Фактически, данный тип атаки является эволюционной разновидностью обыкновенного DDoS.

Сетевая DNS-атака: Атаки типа DNS-флуд. Garbage-атаки Атаки с помощью рекурсивных DNS-запросов

Цели и угрозы

Выведение DNS-сервера — жертвы — из строя.

Основа данных видов атак — простой *DNS-флуд*: множество хостов злоумышленника посылает массированный поток запросов на целевой DNS-сервер с ложным Source IP. Стандартный компьютер способен генерировать 1000 запросов в секунду, стандартный DNS-сервер способен обрабатывать 10000 запросов в секунду. Таким образом 10 обычных домашних (или не совсем домашних) компьютеров вполне достаточно для выведения DNS-сервера из строя (условно). При том вычислить злоумышленника будет очень непросто.

Атака типа *Garbage-DNS* основывается на постоянно открытом 53 порту UDP. Схема атаки сводится к отправке злоумышленником (с множества хостов) больших (свыше 1500 байт) сетевых пакетов (не обязательно DNS). Таким образом, всё сводится к обычному DDoS, но на DNS-порт. Преимущество над обычным DDoS состоит в том, что 53 порт UDP всегда открыт, поскольку нужен для работоспособности DNS-системы.

Рекурсивная DNS-атака сводится к выявлению множества несуществующих в кеше DNS-сервера жертвы имён (возможно, фальшивых) и последующая отправка DNS-запросов с именами из этого множества. DNS-сервер в итоге вынужден пересылать подобные запросы на все соседние и вышестоящие DNS-сервера с целью получить IP заказанного хоста. В итоге, на каждый запрос сервер вынужден посылать ещё целое множество DNS-запросов другим серверам и принимать ответы от них, на что тратятся иногда в сотни раз большие ресурсы, чем отправка одного DNS-запроса. В игоге, как и в предыдущем типе атак, имея совсем не большие ресурсы, становится реальным осуществить достаточно мощную DDoS-атаку на DNS-сервер (в отличие от отражённых DNS-атак, целью которых может быть не только DNS-сервер).

Кибersквоттинг

Кибersквоттинг — захват доменов

Регистрация доменных имён, содержащих торговую марку, принадлежащую другому лицу, с целью их дальнейшей перепродажи или недобросовестного использования.

Виды кибersквоттинга

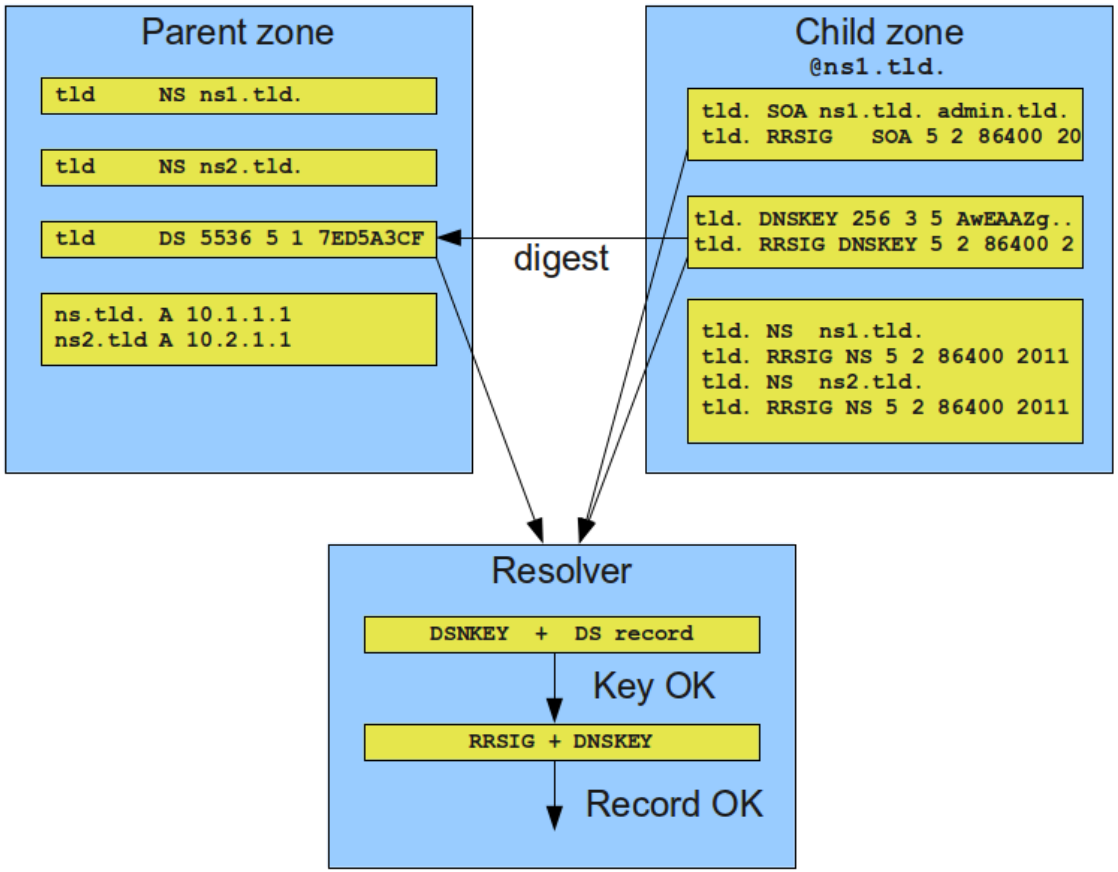
- Тайпсквоттинг — регистрация доменных имён, близких по написанию с адресами популярных сайтов в расчёте на ошибку части пользователей. Например, «`wwwsite.ru`» в расчёте на пользователя, который хотел попасть на «`www.site.ru`». При близости к очень популярным доменам тайпосквоттер может собрать на своём сайте некоторый процент «промахнувшихся» посетителей и за счёт показа рекламы заработать денег.
- Брендовый кибersквоттинг — регистрация доменных имён, содержащих товарные знаки, фирменные наименования, популярные имена собственные, то есть средства индивидуализации, охраняемые законом, а также регистрация «на перспективу», например, создатель фильма «ABC» регистрирует сайт «`ABC.com`», а кибersквоттер, с надеждой что выйдет продолжение фильма, постарается сразу зарегистрировать на себя «`ABC2.com`», «`ABC3.com`», «`ABC4.com`» и т. д. Хотя при этом у кибersквоттера есть риск лишиться домена и подвергнуться ответственности, но законные владельцы товарных знаков могут предпочесть не судиться, а выкупить захваченные домены, и цель кибersквотера будет достигнута — он заработает на этом деньги.
- Защитный кибersквоттинг — когда легальный владелец популярного сайта (товарного знака) регистрирует все доменные имена, близкие, созвучные, похожие, связанные по смыслу с его собственным доменным именем. Делается для того, чтобы не стать жертвой кибersквоттеров. Например, владелец популярного сайта «`www.firma.ru`» может захотеть также зарегистрировать домены «`firma-msk.ru`», «`firma-spb.ru`» и «`firma.org`», чтобы перенаправлять с них посетителей на свой основной сайт, а также «`anti-firma.ru`», чтобы недоброжелатели не смогли использовать его.

DNSSEC

DNSSEC — Domain Name System Security Extensions :: https://www.dnssec.net/

Набор расширений IETF протокола DNS, позволяющих минимизировать атаки, связанные с подменой DNS-адреса при разрешении доменных имён. Направлен на предоставление DNS-клиентам аутентичных ответов на DNS-запросы (или аутентичную информацию о факте отсутствия данных) и обеспечение их целостности. При этом используется криптография с открытым ключом. Не обеспечивается доступность данных и конфиденциальность запросов.

Принцип работы



Принцип работы DNSSEC основан на использовании цифровых подписей. У администратора имеются записи о соответствии имени домена и IP-адреса. DNSSEC ставит каждой из них в строгое соответствие специальную, строго определённую последовательность символов, которая представляет собой цифровую подпись. Главная особенность цифровой подписи в том, что есть открытые, публично доступные методы проверки достоверности подписи, а вот генерирование подписи для произвольных данных требует наличия в распоряжении подписывающего секретного ключа. Поэтому проверить подпись может каждый участник системы, но подписать новые или изменённые данные на практике может только тот, у кого есть секретный ключ.

DNSSEC. Типы ключей. Ресурсные записи

Типы ключей в DNSSEC

Особенность состоит в том, что DNSSEC использует два типа ключей:

1. Ключ подписывания зоны (ZSK, Zone Signing Key).
2. Ключ для подписи набора ключей (KSK, Key Signing Key).

Сделано это, исходя из следующего: зона может быть достаточно большой чтобы удалось подобрать закрытый ключ, поэтому его надо менять чаще, и сделать его можно покороче, чтобы зоны подписывались быстрее; KSK же используется для небольших объемов данных, поэтому его можно сделать длиннее и менять реже. Тем более, что хэш открытой части KSK должен размещаться в родительской зоне, и его частые обновления нежелательны.

DNSSEC вводит концепцию подписанных зон (signed zones). Подписанная зона имеет записи общедоступного ключа DNS (DNSKEY), сигнатуру ресурсной записи (RRSIG), Next Secure (NSEC), и (опционально) Delegation Signer (подписант делегирования) (DS).

Ресурсная запись DNSKEY

Хранит публичную часть ключа ZSK. Используется для проверки записей RRSIG.

Ресурсная запись RRSIG

Содержит подпись для ресурсной записи с определенным именем, классом и типом. RRSIG определяет интервал валидности для подписи и используемого алгоритма, имя подписанта и тэг ключа, для идентификации записи DNSKEY, содержащей публичный ключ, для верификации подписи.

```
ru.   345600  IN   DNSKEY 257 3 8 AwEAAaJ6PMvWiu64aN09Y2yfZ1Y4dK...
ru.   345600  IN   RRSIG DNSKEY 8 1 345600 20181130230000 20181110230000 15506 ru. UAEdycyucsV...
```

DNSSEC. Ресурсные записи

Ресурсная запись DS

Относится к записи DNSKEY и используется в процессе аутентификации DNS DNSKEY. Хранит тэг ключа, код алгоритма и дайджест (подпись) DNSKEY RR (ключом KSK). Ресурсная запись DS и соответствующая ей DNSKEY RR имеют идентичные имена владельца, но они записываются в разных местах. Ресурсная запись DS появляется только на верхней (родительской) стороне делегирования, и является аутентификационной информацией в родительской зоне. Например, DS RR для `example.com` записывается в родительской зоне `com`, а не в самой зоне `example.com` (дочерней зон). Соответствующая DNSKEY RR записывается в зоне `example.com` (дочерняя зона). Это упрощает управление DNS-зонами, но требует специальных требований к обработке откликов для DS RR (RFC4035).

```
ru.    86400    IN    DS 15506 8 2 331CBB1932E7CF201F81AB29...
ru.    86400    IN    RRSIG DS 8 1 86400 20181124170000 20181111160000 2134 . gD1eccLYsKT1Bez42jX...
```

Ресурсная запись NSEC — Next SECure

При подписи зоны доменные имена сортируются в алфавитном порядке, к каждому из них добавляется запись NSEC, в которой указывается какое следующее доменное имя защищено и какие записи для него присутствуют в зоне. Последняя NSEC запись указывает на SOA.

Ресурсная запись NSEC3

Аналог NSEC, но доменные имена хэшируются. NSEC3 для вычисления хэша может использовать соль, помимо соли можно задать количество итераций. Увеличение количества итераций приводит к увеличению нагрузки как на резолвер, так и на авторитетный сервер, причем на последний в большей степени. Это происходит из-за того, что для возвращения NXDOMAIN, авторитетный сервер должен вычислить хэш, и не один. Процесс описан в RFC 5155.

Механизм работы DNSSEC

Хотим узнать адрес `test.bar.example.com`

1. Рекурсивно запрашиваем доменное имя у DNS-сервера (резолвера). Резолвер выставляет бит DO и запрашивает `test.bar.example.com` у корневого сервера.
2. Резолвер знает, что корневая доменная зона подписана — у него есть ее ключ или хэш ключа (т. н. trust-anchor), поэтому он запрашивает у корневого сервера DNSKEY записи для корневой зоны и сверяет их с имеющимся.
3. Корневой сервер не знает такого доменного имени, но ему известно на каких серверах располагается зона `com`. Их он и сообщает резолверу вместе с подписанной DS записью для зоны `com`.
4. Резолвер проверяет DS запись полученным и проверенным ZSK ключом корневой зоны.
5. Теперь резолвер знает, что зона `com` подписана. Он спрашивает у ее DNS сервера DNSKEY и проверяет их, после чего запрашивает `bar.example.com`. Сервер зоны `com` знает, что зона `example.com` размещена на `ns.example.com` и `ns1.example.com`. Их он возвращает резолверу вместе с DS записью.
6. Таким образом резолвер выстраивает цепочку доверия до `example.com`, где он узнает серверы имен зоны `bar.example.com` и ее DS.
7. В конце резолвер итеративно узнает адреса DNS серверов, отвечающих за `bar.example.com`, получает с них итоговую запись, проверяет всю информацию и отдает адресную запись клиенту, выставив в ответе бит AD.

При невозможности что-то проверить резолвер вернет ответ `servfail`.

1. Исходящий трафик авторитетного DNS сервера увеличивается примерно в 4 раза. Размер файла зоны после подписи вырастает в 6-7 раз.
2. Увеличение длины ключа приводит к заметному снижению qps резолвера, а записи NSEC3 аналогично влияют на авторитетный сервер.
3. DNSSec приводит к значительному увеличению DNS ответа. Сетевое оборудование должно корректно работать с DNS пакетами более 512 байт.

DNS over TLS. DNS over HTTPS

DNS поверх HTTPS (DoH)

Экспериментальный протокол для выполнения разрешения DNS по протоколу HTTPS. Целью этого метода является повышение конфиденциальности и безопасности пользователей путём предотвращения перехвата и манипулирования данными DNS с помощью MitM-атак. Google и Mozilla Foundation тестируют версии DNS по протоколу HTTPS.
В публично реализованной версии протокола Google использует HTTP GET-запросы (через HTTPS) для доступа к информации DNS с использованием кодировки DNS-запроса и параметров результата, представленных в нотации JSON.
Другая аналогичная спецификация находится в статусе интернет-проекта под эгидой IETF (RFC 8484). В этой версии протокола используются протоколы HTTP/2 и HTTPS. Записи DNS в традиционном виде пакуются в полезную нагрузку HTTPS с MIME — application/dns-message.

DNS поверх TLS (DoT)

Предлагаемый стандартный протокол для выполнения разрешения удалённой системы DNS с использованием TLS (RFC 8310). Целью этого метода является повышение конфиденциальности и безопасности пользователей путём предотвращения перехвата и манипулирования данными DNS с помощью MitM-атак.
На порт TCP:853 выполняется TLS-подключение, при этом проверка сертификата резолвера происходит с использованием системных корневых сертификатов, точно так же, как HTTPS в браузере. Это избавляет от необходимости добавлять какие-либо ключи вручную. Внутри тоннеля выполняется обычный DNS-запрос. Это создает меньше накладных расходов по сравнению с DNS over HTTPS, который добавляет HTTP-заголовки к запросу и ответу.

Источники

- Материалы с сайта `https://wikipedia.org/`
- Telecommunication technologies — телекоммуникационные технологии / Ю. А. Семенов.
  URL: `http://book.itep.ru/`
- RFC 1034. Domain Names — Concepts And Facilities.
- RFC 1035. Domain Names — Implementation And Specification.
- Пол Мокапетрис — отец DNS / Л. Черняк // Открытые системы. СУБД. №4. 2013.
  URL: `https://www.osp.ru/os/2013/04/13035567/`
- How DNS Works. URL: `http://amar-linux.blogspot.ru/2012/05/how-dns-works.html`
- DNS-атаки: полный обзор по схемам атак / А. С Лысяк // Лаборатория информационной безопасности.
  URL: `http://inforsec.ru/technical-security/network-security/77-dns-attack`
- DNSSec: Что такое и зачем. // Хабр.
  URL: `https://habr.com/post/120620/`
- Внедрение DNSSEC в информационные системы.
  URL: `https://cctld.ru/ru/domains/dnssec/`
- QUIC, TLS 1.3, DNS-over-HTTPS, далее везде. // Хабр.
  URL: `https://habr.com/company/qrator/blog/416633/`
- A cartoon intro to DNS over HTTPS / Lin Clark // Mozilla Hacks.
  URL: `https://hacks.mozilla.org/2018/05/a-cartoon-intro-to-dns-over-https/`
- Google Public DNS тихо включили поддержку DNS over TLS. // Хабр.
  URL: `https://habr.com/post/427639/`

Сети связи
Протоколы, сервисы и услуги в Интернет и IP-сетях

Тема № 13
Электронная почта

доц. каф. СС и ПД, к.т.н. С. С. Владимиров

2020 г.

Электронная почта. Формат электронного адреса

Электронная почта (Electronic mail, e-mail)

Технология и предоставляемые ею услуги по пересылке и получению электронных сообщений по компьютерной сети.

Электронная почта по составу элементов и принципу работы практически повторяет систему обычной почты, заимствуя, в частности, термины (почта, письмо, конверт, вложение, ящик, доставка и другие).

Электронная почта в Internet использует *маршрутно-независимую адресацию*. Это значит, что адрес пользователя остается неизменным независимо от того, откуда посылается сообщение. Такая адресация очень удобна для пользователей, но усложняет процесс доставки сообщения, так как определение маршрута доставки полностью ложится на программное обеспечение электронной почты.
Допускается также указание маршрута сообщения в адресе получателя, но такие маршрутно-зависимые адреса используются редко, обычно, в отладочных целях. В общем случае их использование не имеет смысла.

Формат электронного адреса

Формат электронного адреса подробно описан в RFC 2822 . В общем виде он имеет следующий формат:

```
имя_пользователя@почтовый_домен
```

- `имя_пользователя` – идентификатор пользователя, уникальный в пределах одного почтового домена;
- `@` – символ-разделитель (коммерческое at);
- `почтовый_домен` – уникальный идентификатор почтовой системы.

Формат электронного адреса

Имя пользователя

Имя пользователя может состоять из цифр, латинских букв и символов:

```
! # $ % & ’’ * + - / = ? ^ _ ’ { | } ~
```

Оно может состоять из нескольких полей, разделенных точкой. Если имя пользователя содержит символы, отличные от перечисленных, его следует заключать в кавычки.
На практике имена пользователей обычно состоят только из цифр, латинских букв, символов «-», «_» и точки, которая интерпретируется не как разделитель полей, а как часть имени пользователя. Использование других символов может привести к неоднозначным интерпретациям, потому не желательно.

Почтовый домен

Имя почтового домена имеет тот же формат, какой используется в доменных именах Internet. Описан в RFC 1034.

Комментарии

Кроме значимой части, используемой при маршрутизации сообщения, адрес может содержать комментарии в виде произвольных текстовых строк до и после значимой части, отделенных от нее угловыми скобками.

```
комментарий < имя_пользователя@почтовый_домен > комментарий
```

Информация по обе стороны угловых скобок при доставке сообщения игнорируется.
Адрес электронной почты не обязательно указывает непосредственно на существующий почтовый ящик. Это также может быть почтовый псевдоним, указывающий на другой адрес, или адрес списка рассылки, указывающий на множество других адресов, или адрес, на который приходят сообщения, поступающие на обработку специальной программой, и т. д.
Некоторые системы допускают также использование одного лишь имени пользователя в качестве электронного адреса, если получатель зарегистрирован в том же почтовом домене, из которого посылается сообщение.

Формат сообщения электронной почты

Формат сообщения, передаваемого по электронной почте, описан в RFC 2822 . Оно состоит из трех частей:

- *Конверт* (envelope), содержащий адреса отправителя и получателей сообщения, эта информация используется только при пересылке сообщения по протоколу SMTP , получателю она недоступна.
- *Заголовок* (header), содержащий служебную информацию, формируемую программами, участвующими в передаче сообщения, такую как адреса отправителя и получателей, которые могут отличаться от используемых в конверте, тему сообщения, время отправки, сведения о пересылке и об используемых для создания сообщения программах и т. д. Заголовок завершается пустой строкой.
- *Тело* (body), содержащее само сообщение, созданное отправителем и подлежащее доставке получателю.

Таким образом, сообщение доставляется получателю в виде заголовка и отделенного от него пустой строкой тела.
Заголовок состоит из полей: текстовых строк, состоящих из имени поля: слова, заканчивающегося двоеточием, и содержимого поля.
В заголовке допускается использование только символов в кодировке US-ASCII. Другие символы должны быть закодированы таким образом, чтобы полученная кодовая последовательность содержала только символы кодировки US-ASCII. Это правило нередко нарушается, например, тема сообщения записывается в заголовке сообщения на русском языке без перекодирования. Этого следует избегать, так как на приемном конце не будет известна используемая кодировка русских букв, а значит, полученная последовательность может быть интерпретирована неправильно. Этого не произойдет, если текст будет закодирован в соответствии с RFC 2047 .

```
=? charset ? encoding ? encoded-text ?=
=?ISO-8859-1?Q?this=20is=20some=20text?=
=?UTF-8?B?2KfZhNiu2LfiNin2Kog2KfZhNiq2?=
```

Длинные поля заголовка разбиваются на несколько строк, при этом каждая строка, продолжающая предыдущую, начинается с пробельного символа. Заголовок обычно показывается не полностью. Получатель видит только некоторые поля: адреса отправителя и получателей, время отправки и тему сообщения.
Тело сообщения, если это не просто текст, записанный латинскими буквами, должно быть закодировано в соответствии со спецификацией MIME, как описано в RFC 2045. На приемной стороне оно при необходимости декодируется и преобразуется в понятный пользователю вид.

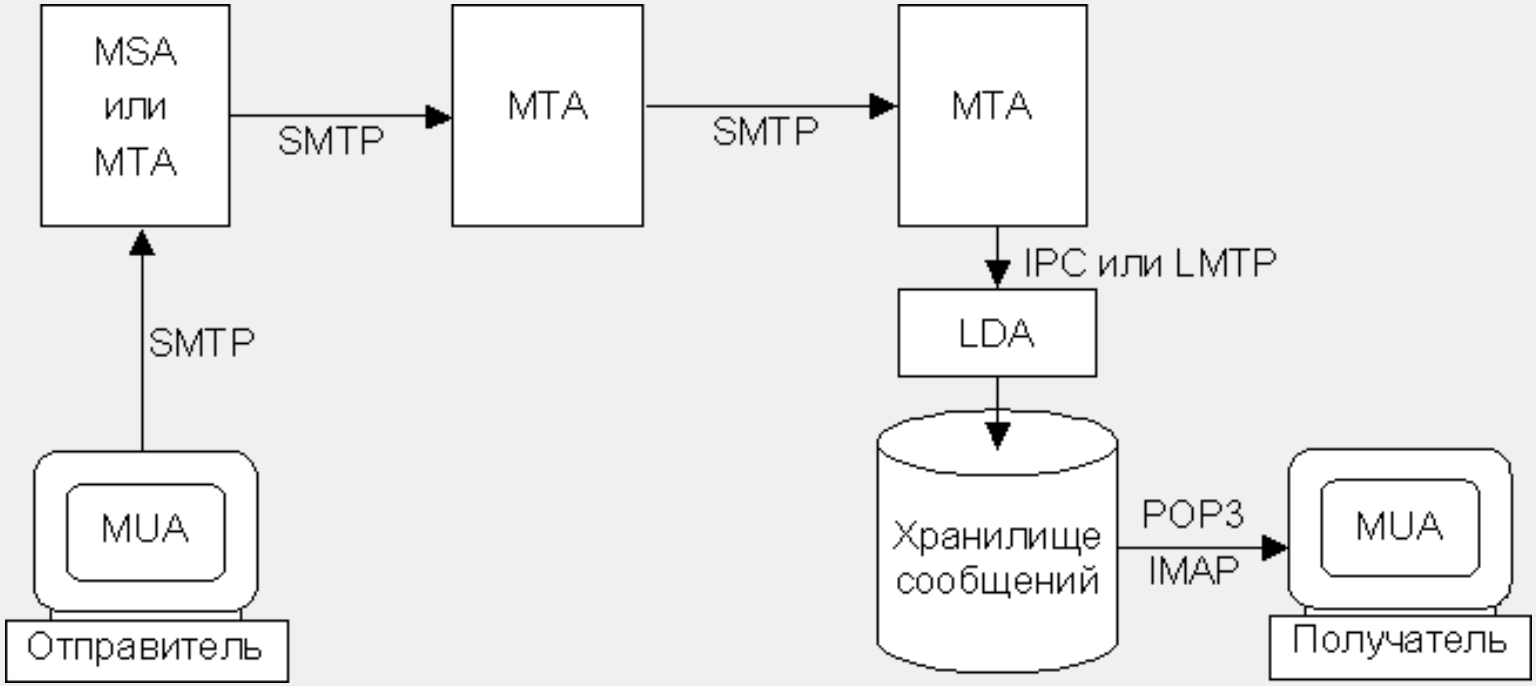
Пример сообщения электронной почты

Пример сообщения

```
Return-Path: <gmailaddr@gmail.com>
Received: from blacktower ([91.101.175.183])
        by mx.google.com with ESMTPSA id j2sm10233782lag.12.2014.03.23.03.44.05
        for <gmailaddr@gmail.com>
        (version=TLSv1 cipher=RC4-SHA bits=128/128);
        Sun, 23 Mar 2014 03:44:07 -0700 (PDT)
Message-ID: <532ebaf7.421a980a.4f0a.ffffc585@mx.google.com>
Received: by blacktower (sSMTP sendmail emulation); Sun, 23 Mar 2014 14:44:04 +0400
From: root <gmailaddr@gmail.com>
Date: Sun, 23 Mar 2014 14:44:04 +0400
To: root
Subject: Anacron job 'cron.daily' on blacktower
Content-Type: text/plain; charset=US-ASCII

/etc/cron.daily/logrotate:
ERROR Unable to contact server. Is it running?
error: error running non-shared postrotate script for /var/log/fail2ban.log of
'/var/log/fail2ban.log
'run-parts: /etc/cron.daily/logrotate exited with return code 1
```

Структура электронной почты в Internet



Компоненты электронной почты

MUA Mail User Agent – пользовательский агент, или клиентская почтовая программа.

MTA Mail Transfer Agent – транспортный агент, или почтовый сервер.

LDA Local Delivery Agent – агент локальной доставки.

MSA Message Submission Agent – агент подачи сообщения.

Mail User Agent (MUA)

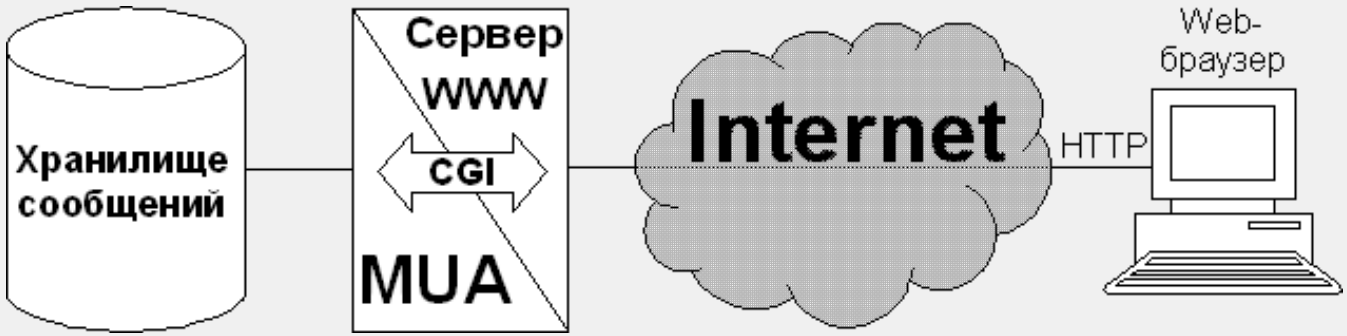
MUA предназначен для подготовки, отправки, получения и просмотра электронных писем. Это программа, установленная на компьютере пользователя. Задача электронной почты, по сути дела, сводится к тому, чтобы доставить сообщение от MUA отправителя на MUA получателя.
Подготовка к отправке заключается в приведении сообщения к принятому в Internet формату, описанному в RFC 2822.
MUA отправителя должен сформировать заголовок сообщения, а также закодировать и оформить его тело в соответствии со стандартом, чтобы MUA принимающей стороны смог правильно интерпретировать и представить как текст, так и вложения письма.
Так как MUA обычно устанавливается на машине пользователя, он, как правило, запускается только на время работы пользователя, а компьютер, на котором запущен MUA, может не иметь постоянного подключения к Internet. Поэтому MUA не может выступать в качестве сервера – он может быть только инициатором соединения, то есть клиентом.
MUA посылает сообщения по протоколу SMTP через MSA или MTA, используемый для отправки почты.
Входящие письма MUA забирает из хранилища сообщений по протоколу, предназначенному для получения почты. Как правило для этой цели используется один из двух протоколов:

1. Post Office Protocol Version 3 (POP3) – протокол почтового отделения, версия 3, описанный в RFC 1939, позволяет просматривать сообщения в почтовом ящике, забирать и удалять их.
2. Internet Message Access Protocol (IMAP) – протокол доступа к сообщениям, описанный в RFC 3501, обладает более широкими возможностями манипулирования почтовыми ящиками, чем POP3, в частности он позволяет работать с несколькими ящиками одновременно, не только считывать и удалять, но и создавать и исправлять сообщения.

Возможны и другие способы получения почты. Например, использование локальной доставки, если хранилище сообщений доступно MUA по локальной сети.

Mail User Agent (MUA). Web-mail

Довольно большое распространение получили агенты пользователя, использующие интерфейс CGI для доступа оконечного пользователя к его почтовому ящику по протоколу HTTP или более безопасному HTTPS при помощи web-браузера. Такую реализацию MUA часто называют web-mail.



Пользовательский интерфейс реализуется с помощью технологий WWW . Функции MUA выполняет приложение, взаимодействующее с web-сервером при помощи интерфейса CGI. MUA получает доступ к хранилищу сообщений по протоколам POP3 или IMAP или путем непосредственного обращения – MUA при такой реализации может быть включен в ту же локальную сеть, что и хранилище сообщений, они даже могут быть запущены на одной и той же машине.

Преимущество web-mail перед MUA, установленным на компьютере пользователя, это возможность работать со своей почтой с любого компьютера, подключенного к Internet, без предварительной настройки и без инсталляции программного обеспечения. Недостаток web-mail заключается в том, что пользователю для работы с почтой необходим постоянный доступ к Internet, так как каждый запрос выполняется не на пользовательской машине, а на web-сервере, и должен быть передан по сети.

Mail Transfer Agent (MTA)

MTA представляют собой узлы, через которые передаются электронные сообщения. Письмо, сформированное MUA, достигает хранилище сообщений, содержащее почтовый ящик получателя, проходя через один или несколько MTA, последний из которых передает письмо агенту локальной доставки (LDA).
Как правило, MTA должны быть доступны круглосуточно и постоянно ожидать подключения по протоколу SMTP. Иными словами, каждый MTA в Internet включает в себя сервер SMTP. Обмен данными между MTA происходит по этому протоколу. MTA, отправляющий почту, инициирует соединение и выступает в качестве клиента, MTA, принимающий почту, является сервером.
На MTA также возлагается разбор адресов получателей, раскрытие списков рассылки и почтовых псевдонимов и определение маршрута сообщения на основании анализа адресов получателей и записей MX, получаемых от сервера DNS.
MTA должен проверять соответствие действительности идентификационных данных получаемых им от встречного MTA. Следует проверять соответствие доменного имени, которое клиент сообщает в приветствии, его адресу IP. Также нужно удостовериться в существовании почтового домена, указанного в почтовом адресе отправителя. Если в доменной части адреса получателя указан почтовый домен, обслуживаемый данным MTA, то следует проверить, зарегистрирован ли в этом домене указанный адресат.
В целях предотвращения анонимных рассылок спама, RFC 2505 рекомендует принимать почту только при выполнении хотя бы одного из следующих условий:

- адрес IP клиента входит в список адресов клиентов, обслуживаемых данным MTA;
- получатель сообщения зарегистрирован в почтовом домене, обслуживаемом данным MTA;
- клиент прошел процедуру аутентификации.

Если не выполнено ни одно из названных условий, MTA должен отказать в приеме почты. MTA, принимающий почту, не отвечающую перечисленным требованиям, может быть внесен в списки серверов, не препятствующих распространению спама. В этом случае многие почтовые системы будут отказываться принимать от него почту.

Mail Transfer Agent (MTA)

MTA может производить обработку проходящих через него сообщений: проверку на наличие вирусов, фильтрацию спама и пр.
Каждый MTA, через который проходит почтовое сообщение, добавляет к его заголовку информацию о том, когда и откуда пришло это сообщение, а также результаты произведенных проверок.
В случае невозможности немедленной доставки сообщения, оно помещается в очередь. MTA регулярно предпринимает новые попытки отправить сообщения из очереди. Если это не удается за определенный срок, обычно за четыре часа, то отправителю посылается предупреждение о задержке доставки. Но сообщение остается в очереди, и попытки его отправить продолжаются. Если в течение длительного времени, обычно пяти дней, сообщение так и не удается доставить, оно удаляется из очереди, а отправителю посылается сообщение о невозможности доставки письма.
MTA могут также выполнять и другие функции, в зависимости от используемого программного обеспечения.
Основные требования к MTA и к MUA описаны в RFC 1123 и уточнены в RFC 2821 и в RFC 2822.
Существует множество разнообразных программных реализаций MTA. Старейшей из них и до сих пор одной из наиболее популярных является программа sendmail, разработанная в начале восьмидесятых годов Эриком Оллмэном, тогда еще студентом Калифорнийского университета в Беркли. Эта программа многократно дорабатывалась и стала фактически стандартом для этого типа программного обеспечения. Она продолжает совершенствоваться и по сей день. Доступна как свободно распространяемая версия для операционных систем, совместимых с UNIX, так и коммерческая версия.
Позже появились и другие программные продукты, реализующие функции MTA для различных операционных систем: Postfix, smail, qmail, exim, ZMailer и многие другие.

Message Submission Agent (MSA)

MTA, через которые проходит сообщение, добавляют некоторые строки в его заголовок. Однако информацию, уже содержащуюся в сообщении, MTA не изменяют, хотя необходимость в этом может возникнуть. Заголовок сообщения, полученного от MUA, может быть неправильно оформлен, например, там может быть не определено полностью имя домена, ошибочно указано время или дата. Может возникнуть необходимость в корректировке адреса отправителя, если в почтовой сети предприятия используется адресация, отличная от принятой в Internet. Например, если допускается использование адресов без указания почтового домена для пользователей, зарегистрированных в почтовой системе предприятия.

Функции корректировки заголовка сообщения можно возложить на MTA, принимающий почту от агентов пользователя, но, если поток почты велик, имеет смысл использовать для этого специальный процесс – MSA. Таким образом, можно сказать, что MSA это разновидность MTA, занимающаяся предварительной обработкой исходящей почты. Подробнее задачи и особенности реализации MSA описаны в RFC 2476.

Чтобы различать MTA и MSA, рекомендуется запускать MSA не на порту 25, предназначенном для MTA, а на другом порту TCP, либо использовать порт 25 на сервере, где не запущен MTA.

Local Delivery Agent (LDA)

Последний MTA на пути следования электронного почтового сообщения должен передать его агенту локальной доставки. Обычно LDA расположен на одной машине с MTA и представляет собой программу, которая вызывается агентом передачи сообщения при поступлении новых сообщений. В этом случае для взаимодействия между MTA и LDA используются механизмы межпроцессного взаимодействия (IPC). В некоторых случаях LDA также может быть реализован как сервер, принимающий от MTA почту по протоколу, аналогичному SMTP . Этот протокол описан в RFC 2033, он называется LMTP (Local Mail Transfer Protocol).

Агентом доставки называется программа, производящая обработку поступившей почты. В основном эта обработка заключается в помещении сообщений в почтовые ящики адресатов, то есть в добавлении сообщений к соответствующим файлам или в размещении их в специальных каталогах пользователей или в базах данных. Пользователь сможет получить сохраненные сообщения, соединившись с хранилищем сообщений по протоколу POP3 или IMAP.

Другой вид обработки сообщений – передача их каким-либо программам для дальнейшей обработки.

Для выполнения этих функций LDA должен при необходимости раскрывать почтовые псевдонимы и списки рассылки.

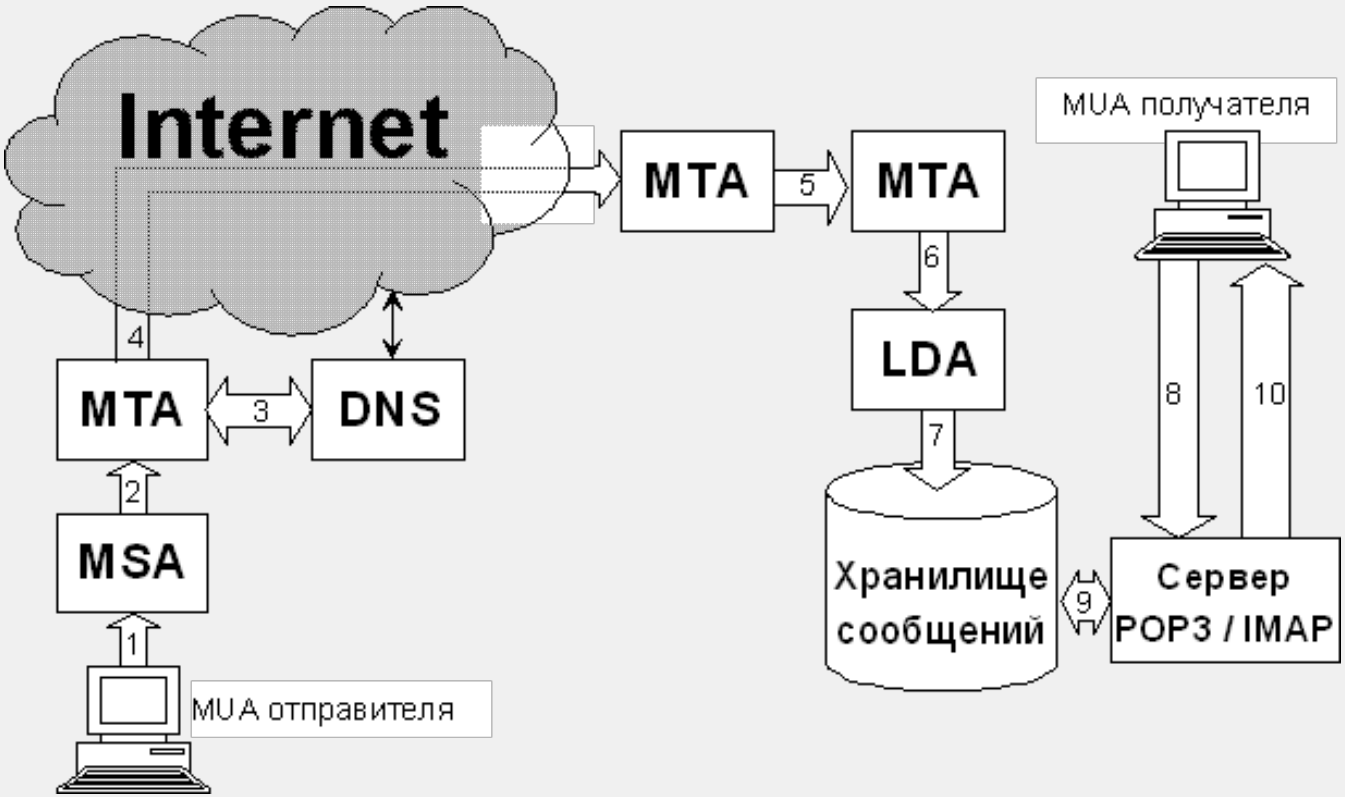
Хранилище сообщений

Электронные сообщения обычно не доставляются автоматически на машину пользователя, а помещаются в хранилище сообщений, откуда пользователь может их забрать в удобное для него время. Каждому пользователю выделяется ограниченный или неограниченный объем дискового пространства, физически реализованный в виде файла специального формата, каталога специальной структуры или набора записей в базе данных. Элемент хранилища сообщений, содержащий электронные сообщения, называется почтовым ящиком.

Доступ пользователей к сообщениям, находящимся в хранилище, обычно осуществляется по протоколам POP3 или IMAP. В качестве клиента выступает MUA пользователя, сервер имеет непосредственный доступ к хранилищу сообщений. Он ожидает подключений пользовательских агентов и, после обязательной аутентификации, определяет права доступа, установленные для данного пользователя. Пользователь должен иметь доступ не менее чем к одному почтовому ящику.

Какие именно манипуляции пользователь может проделывать со своими почтовыми ящиками и с содержащимися в них сообщениями, зависит от используемого программного обеспечения. При минимальной реализации пользователь получает доступ к одному почтовому ящику, сообщения в который помещаются LDA. Пользователь может получать и удалять отдельные сообщения. Такой вид доступа, в большом числе случаев достаточный, реализуется при использовании протокола POP3. Другой популярный протокол доступа к электронным почтовым ящикам – IMAP, предоставляет более широкие возможности.

Доставка почтового сообщения



Доставка почтового сообщения

1. Сообщение, сформированное MUA отправителя, по протоколу SMTP посылается MSA. MSA проверяет, имеет ли данный MUA или пользователь право посылать почту из этой почтовой системы. В случае положительного результата, сообщение принимается для дальнейшей доставки.
2. MSA проверяет заголовок сообщения и, при необходимости, исправляет его. Готовое к отправке сообщение по протоколу SMTP отправляется на MTA исходящей почты.
3. MTA исходящей почты анализирует адрес получателя. Если сообщение предназначено для получателя домена, обслуживаемого данной почтовой системой, то оно доставляется получателю (пункты 6–10), в противном случае MTA запрашивает информацию о почтовом домене, указанном в адресе получателя, сервер DNS. Получив запрашиваемые данные, сервер DNS сообщает MTA, какие узлы принимают почту для данного домена, их адреса IP и приоритеты.
4. MTA отправителя пытается установить соединение по протоколу с принимающими почту узлами в соответствии с приоритетами, указанными в записях MX, полученных от сервера DNS. Если соединение ни с одним узлом не удается установить, сообщение помещается в очередь, и через некоторое время попытки установить соединение повторяются. Если соединение установлено, то принимающий MTA, удостоверившись, что сообщение предназначено для пользователя его домена, и что почтовый ящик с указанным адресом действительно существует, принимает сообщение.
5. В принимающей почтовой системе сообщение может пройти через несколько промежуточных MTA, выполняющих различные виды обработки входящей почты: проверку на вирусы, фильтрацию спама, перенаправление к нужному хранилищу сообщений и пр. Внутри принимающей системы может использоваться как SMTP, так и LMTP.
6. Последний MTA, используя межпроцессное взаимодействие или протокол LMTP, передает сообщение LDA для локальной доставки.
7. LDA помещает сообщение в почтовый ящик адресата.
8. Получатель обращается к серверу POP3 или IMAP, чтобы проверить поступившую почту.
9. Сервер забирает сообщение из почтового ящика и посылает его пользовательскому агенту получателя.

Протокол SMTP

Simple Mail Transfer Protocol (SMTP)

Простой протокол передачи почты. Ообычно используется на участке от MUA отправителя до ближайшего к получателю MTA.
Протокол разрабатывался в начале восьмидесятых годов прошлого века. Окончательная версия была закреплена в RFC 821 1 августа 1982 года. Все годы, прошедшие с того времени, протокол SMTP оставался одним из наиболее часто используемых протоколов семейства TCP/IP. В 2001 был выпущена вторая версия описания протокола — RFC 2821, а в 2008 — третья версия (RFC 5321), которая и используется по настоящее время.

За это время принципиально изменились многие требования, касающиеся достоверности и защищенности передаваемых сообщений, значительно увеличились средний размер сообщений, и их количество, разнообразнее стала передаваемая информация: это уже не только текстовые сообщения на английском языке – сейчас электронные письма пишутся на многих языках и могут содержать вложения самых разных типов.
Однако протокол SMTP получил за время своего существования такое широкое распространение, что просто заменить его другим протоколом уже не представляется возможным. Вместо этого для него разрабатываются различные расширения (extensions), дополняющие возможности базового протокола. Дополненный расширениями протокол SMTP часто называют ESMTP (Extended SMTP).
Сам протокол изменился незначительно. На смену команде HELO , использовавшейся для начала диалога, пришла команда EHLO , позволяющая работать с расширениями ESMTP . Команды, применяемые для настройки почтовых систем и для получения справочной информации о пользователях, теперь используются значительно осторожнее, чем в восьмидесятые годы. Эти команды создают удобства не только для сетевых администраторов, но и для злоумышленников. Поэтому такие команды обычно используют только на этапе настройки почтовой системы. В работающей системе их, как правило, отключают.

Протокол SMTP. Команды SMTP

SMTP обычно использует TCP (25 порт). хотя может работать и с другими протоколами транспортного уровня.
Почта по протоколу SMTP посылается от клиента к серверу. Клиент запрашивает соединение с сервером. После успешного установления соединения сервер сообщает клиенту свое доменное имя. Он также может сообщить тип и версию установленного программного обеспечения. Однако, из соображений безопасности, чтобы не дать потенциальному взломщику воспользоваться известными ошибками данной версии сервера SMTP , передача этой информации часто блокируется системными администраторами.
Ответ сервера, свидетельствующий о готовности к приему команд клиента, служит сигналом к началу диалога, в котором клиент последовательно посылает серверу команды и ожидает ответы, либо подтверждающие исполнение команд, либо сообщающих о невозможности исполнения, либо содержащих информацию, запрошенную клиентом.

Команды SMTP

Каждая команда SMTP начинается с ключевого слова – названия команды. За ним могут следовать параметры, отделенные пробелом. Названия команд и, за редким исключением, параметры протокола, не зависят от регистра. В некоторых элементах расширений строчные и прописные символы могут различаться. Левая часть почтового адреса, до символа «@», может быть регистрозависимой. В командах допускается использование только семибитной кодировки ASCII — цифры, латинские буквы, и знаки препинания. Если информация передается восьмибитными блоками (октетами), старший бит должен быть равен нулю. Корректная интерпретация символов, старший, восьмой бит которых равен единице, например, русских букв, не гарантируется, использовать такие символы не следует.
Конец строк в протоколе SMTP обозначается последовательностью символов "возврат каретки"(`0x0D`) и "перевод строки"(`0x0A`). Эта последовательность обозначается CRLF. Сервер начинает выполнение команды только получив от клиента строку, завершающуюся последовательностью CRLF.
Сервера SMTP должны принимать командные строки длиной до 512 символов. Это значение может быть увеличено по желанию разработчиков. Для серверов, поддерживающих расширения ESMTP, требующие дополнительных параметров, максимально допустимая длина командной строки увеличивается. Соответствующие требования приведены в RFC, описывающих эти расширения. Если не используется расширение, позволяющее серверу принимать несколько команд подряд, клиент передает серверу следующую команду только после получения ответа на предыдущую.

Ответы сервера SMTP

На каждую команду клиента сервер посылает ответ, состоящий из числового кода и отделенной от него пробелом текстовой строки. В большинстве случаев для правильной интерпретации ответа клиенту достаточно числового кода. Текстовая строка нужна для интерпретации ответа человеком. Исключение составляет ответ на команду EHLO, содержащий список расширений ESMTP, поддерживаемых сервером, а так же ответы на некоторые команды ESMTP. Согласно RFC 2821, код ответа состоит из трех цифр. Первая цифра кода может принимать следующие значения:

1 Предварительный положительный результат. Команда принята, но для ее выполнения сервер ожидает реакции клиента на посылаемую в этом ответе информацию. Клиент должен послать следующую команду для продолжения работы. В базовом протоколе SMTP не предусмотрено команд, требующих ответов такого типа.

2 Команда выполнена успешно.

3 Промежуточный положительный результат. Команда принята, но сервер ожидает от клиента дополнительные данные для завершения операции. Дополнительными данными может, например, быть текст сообщения в команде DATA.

4 Исполнение команды временно невозможно. Команда не может быть выполнена, но проблема может быть устранена. Клиенту следует попытаться повторить попытку через некоторое время.

5 Исполнение команды невозможно.

Вторая цифра может принимать следующие значения:

0 Синтаксическая ошибка, неправильное или недопустимое использование команды.

1 Ответ содержит запрошенную информацию.

2 Ответ о состоянии канала передачи.

5 Ответ информирует о состоянии принимающей почтовой системы.

Если ответ состоит из нескольких строк, то каждая из них начинается числовым кодом, который отделяется от сопровождающего текста не пробелом, а символом «дефис» (-). В последней строке цифровой код отделяется от текста пробелом. Каждая строка ответа заканчивается последовательностью CRLF.

Пример диалога SMTP

Команды клиента помечены буквой C, а ответы сервера – буквой S.

S	`220 foo.com Service Ready`	Сервер представляется как `foo.com` и сообщает о готовности к приему команд
C	`EHLO bar.com`	Клиент представляется как `bar.com`
S	`250-foo.com greets bar.com`	
S	`250-8 BITMIME`	Сервер сообщает о поддерживаемых расширениях ESMTP
S	`250-SIZE`	
S	`250-DSN`	
S	`250 HELP`	
C	`MAIL FROM:<Smith@bar.com>`	Адрес отправителя: `Smith@bar.com`
S	`250 OK`	
C	`RCPT TO:<Jones@foo.com>`	Адрес первого получателя: `Jones@foo.com`
S	`250 OK`	
C	`RCPT TO:<Green@foo.com>`	Адрес второго получателя: `Green@foo.com`
S	`550 No such user here`	Ошибка: ящик не существует
C	`RCPT TO:<Brown@foo.com>`	Адрес третьего получателя: `Brown@foo.com`
S	`250 OK`	
C	`DATA`	Адреса переданы, клиент готов передавать сообщение
S	`354 Start mail input; end with <CRLF>.<CRLF>`	Сервер готов к приему сообщения
C	Клиент передает сообщение	Сообщение заканчивается строкой, состоящей из одной точки
C	`.`	
S	`250 OK`	Сообщение принято
C	`QUIT`	Клиент завершает связь
S	`221 foo.com Service closing transmission channel`	Сервер подтверждает завершение связи

Расширения ESMTP

Механизм расширений ESMTP позволяет дополнять протокол SMTP новыми функциональными возможностями, не предусмотренными в RFC 2821. Расширения могут добавлять к протоколу SMTP новые функции или модифицировать существующие. При этом должна сохраняться обратная совместимость: функции базового протокола SMTP должны выполняться независимо от установленных расширений.

Многие расширения ESMTP описаны в документах RFC. Разработчики программного обеспечения также могут использовать в своих продуктах нестандартизованные расширения. Естественно, работать с ними могут только программы того же производителя. Чтобы не допустить ситуации, при которой новое стандартное расширение получит название, которое уже было использовано каким-либо производителем, названия таких расширений должны начинаться с буквы `X`. Название стандартного расширения с буквы `X` начинаться не может.

Расширения ESMTP могут добавлять новые команды, не предусмотренные базовым протоколом SMTP, а также вводить дополнительные параметры команд `MAIL` и `RCPT`. Формат дополнительных параметров:

```
Название_параметра=аргумент
```

Клиент узнает, какие именно расширения поддерживаются сервером, из ответа на команду `EHLO`. Каждая строка ответа может содержать ключевое слово, соответствующее названию поддерживаемого сервером расширения ESMTP, и, если необходимо, параметры этого расширения.

Протокол LMTP

Протокол LMTP (Local Mail Transfer Protocol)

Протокол LMTP применяется в основном для связи MTA с LDA, а также может использоваться для взаимодействия с MTA, не помещающими сообщения в исходящую очередь, и имеющими возможность немедленно ответить, возможна доставка или нет. Протокол LMTP определен в RFC 2033. LMTP работает аналогично SMTP, использует расширения ESMTP, но область его применения отличается от области применения SMTP, и он не должен использовать порт TCP 25.

Отличия LMTP от SMTP

- команды HELO и EHLO заменяются командой LHLO, идентичной по синтаксису и действию команде EHLO;
- команды DATA и, если используется расширение ESMTP CHUNKING, BDAT LAST возвращают не один ответ после окончания приема сообщения, а столько, сколько было успешно выполненных команд RCPT. Сервер передает клиенту результат доставки сообщения каждому получателю.

Пример диалога LMTP

Команды клиента помечены буквой C, а ответы сервера – буквой S.

```
S   220 foo.edu LMTP server ready
C   LHLO foo.edu
S   250-foo.edu
S   250-PIPELINING
S   250 SIZE
C   MAIL FROM:<chris@bar.com>
S   250 OK
C   RCPT TO:<pat@foo.edu>
S   250 OK
C   RCPT TO:<green@foo.edu>
S   250 OK
C   DATA
S   354 Start mail input; end
    with <CRLF>.<CRLF>
C   Передается сообщение
C   .
S   250 OK                          Сообщение успешно доставлено первому адресату: pat@foo.edu
S   452 <green@foo.edu> is          Сообщение не доставлено получателю green@foo.edu
    temporarily over quota
C   QUIT
S   221 foo.edu closing
    connection
```

Протокол POP3

Post Office Protocol — Version 3 (POP3)

Протокол POP3 предназначен для получения сообщений, находящихся в почтовом ящике пользователя на удаленном сервере электронной почты. Как правило, не целесообразно устанавливать серверы SMTP на рабочие станции, предназначенные для чтения писем. Сервер SMTP должен быть доступен постоянно, а рабочие станции обычно включают только на время работы пользователя, соединение с сервером они нередко устанавливают по коммутируемым линиям только для того, чтобы забрать накопившуюся почту.

По протоколу SMTP почта доставляются только в хранилище сообщений, откуда пользователь может ее забрать в удобное для него время.

Таким образом, в качестве клиента POP3 выступает MUA пользователя, а сервер должен иметь доступ к хранилищу сообщений. Информация по протоколу POP3 передается от сервера к клиенту.

Протокол POP был разработан в 1984 году, в 1985 году появилась вторая его версия, в 1988 году – третья, которая с существенными модификациями, сделанными в 1991, 1993, 1994 и 1996 годах, используется по сей день. На момент написания этого пособия, последняя модификация протокола POP3 описана в RFC 1939 и является стандартом Интернет STD 53.

POP3 прост в реализации и предоставляет минимальные необходимые возможности для работы с почтовым ящиком. Вопреки распространенному мнению, третья версия протокола POP дает возможность работать не только с ящиком в целом, но и с отдельными сообщениями, находящимися в нем, позволяя просматривать информацию о письмах, получать и удалять их по отдельности. К сожалению, не все существующие MUA используют эти возможности протокола POP3. Пользователь не всегда хочет скачивать с сервера все содержимое почтового ящика, и часто предпочел бы получать только некоторые сообщения, а другие сообщения, возможно, удалил бы, не получая. Все это можно сделать, используя протокол POP3. Более широкие возможности предоставляет протокол IMAP 4 , но в большом числе случаев возможностей протокола POP3 оказывается вполне достаточно.

Этапы (состояния) сеанса протокола POP3

Сеанс протокола POP3 делится на три этапа (состояния).

Сервер ожидает соединения по порту TCP 110.

После установления соединения сервер посылает клиенту строку приветствия, свидетельствующую о готовности к диалогу, и сеанс переходит в состояние авторизации (AUTHORIZATION State). На этом этапе выясняется, доступ к какому именно почтовому ящику запрашивает клиент и имеет ли он соответствующие права. Успешное прохождение авторизации необходимо для продолжения работы.

Если авторизация проходит успешно, то сеанс переходит в состояние транзакции (TRANSACTION State). На этом этапе клиент может проделывать все необходимые манипуляции с почтовым ящиком: он может просмотреть информацию о состоянии ящика и отдельных сообщений, получить выбранные сообщения и пометить письма, подлежащие удалению.

По окончании всех операций, клиент сообщает об окончании связи, и сеанс переходит в состояние обновления (UPDATE State). На этом этапе сервер стирает из ящика сообщения, помеченные на предыдущем этапе как подлежащие удалению, и закрывает соединение. Переход в состояние обновления в принципе возможен, только если клиент выходит из состояния транзакции по команде QUIT . Ни при каких других обстоятельствах, например, если сеанс связи прерывается по таймауту или из-за обрыва связи, переход в состояние обновления происходить не должен. То есть, если состояние транзакции прерывается не по команде QUIT, никакие удаления не должны производиться, пометки для удаления должны быть аннулированы. Как показывает практика, это требование выполняется не всегда.

Команды протокола POP3

В ходе сеанса клиент посылает серверу команды, а сервер сообщает о результате выполнения каждой из них. Ответ состоит из индикатора состояния (status indicator) и, если нужно, дополнительной информации, отделенной пробелом. Строка ответа может содержать до 512 символов, включая последовательность CRLF, обозначающую конец строки.
Предусмотрено два индикатора состояния: «`+OK`» – успешное завершение и «`-ERR`» – неуспешное завершение. Если строка ответа не содержит дополнительной информации, то после индикатора состояния сразу должна идти последовательность CRLF. Однако некоторые клиенты ожидают пробела после индикатора состояния. Это противоречит существующим стандартам, но наличие таких клиентов все же следует принимать во внимание (RFC 1957).
Если команда предусматривает многострочный ответ, то индикатор состояния передается только в первой строке, а последняя строка ответа должна состоять из одной точки. Эта строка не является частью ответа, а только обозначает его завершение. Чтобы сделать возможным использование строк, состоящих из одной точки, в ответах сервера, ко всем строкам ответа, начинающимся с точки, добавляется еще одна точка, аналогично тому, как это делается при передаче текста сообщения в команде DATA протокола SMTP. Если на приемном конце в ответе сервера обнаруживается строка, начинающаяся с точки, то, если непосредственно за этой точкой стоит последовательность CRLF, строка интерпретируется как конец ответа, если же за точкой следуют любые другие символы, то ведущая точка удаляется, а строка интерпретируется как часть ответа.
Каждая команда POP3 состоит из ключевого слова и, возможно, из аргументов, разделенных пробелами. Ключевые слова состоят из трех или четырех букв, передаваемых независимо от регистра. Аргументы могут содержать только символы ASCII. Каждый аргумент может состоять не более чем из сорока символов.
Кроме обязательных команд программное обеспечение, реализующее взаимодействие по протоколу POP3, поддерживает дополнительные возможности (capabilities), вводящие новые команды, влияющие на исполнение основных команд, облегчающие взаимодействие клиента и сервера, информирующие об особенностях реализации сервера и хранилища сообщений.
В число дополнительных возможностей входят, например, команды авторизации. Хотя бы один механизм авторизации должен быть реализован, так как доступ к почтовому ящику предоставляется только после аутентификации. Но, поскольку таких механизмов несколько, и их выбор оставляется на усмотрение разработчиков и администраторов, соответствующие команды не входят в число обязательных.

Пример сеанса POP3 — 1

S	+OK ready <6584.1077893295@myhost.ru>	В начальном приветствии сервера присутствует уникальный идентификатор, что свидетельствует о поддержке сервером команды APOP
C	CAPA	Запрос возможностей сервера
S	+OK Capability list follows	
S	TOP	Поддерживается команда TOP
S	USER	Поддерживается авторизация открытым паролем
S	LOGIN-DELAY 0	На сервере может быть установлен минимальный период времени между сеансами одного пользователя, но в настоящее время такого ограничения нет
S	EXPIRE 0	Прочитанная почта не хранится на сервере (если эта возможность правильно настроена)
S	UIDL	Поддерживается команда UIDL
S	RESP-CODES	Поддерживаются расширенные коды ответов
S	X-LOCALTIME Fri, 27 Feb 2004 15:48:46 +0100	Нестандартное сообщение, установленное разработчиком сервера
S	.	Конец ответа
C	USER lonk	Имя пользователя: lonk
S	+OK Password required for lonk.	Имя принято, ожидается ввод пароля
C	PASS my_passwd	Пароль пользователя lonk : my_passwd
S	+OK lonk has 3 visible messages in 223385 octets.	Доступ к почтовому ящику разрешен. Имеется 3 сообщения общим объемом 223385 октетов
C	CAPA	Повторный запрос возможностей сервера
S	+OK Capability list follows	
S	TOP	
S	USER	

Пример сеанса POP3 — 2

S	`LOGIN-DELAY 0`	
S	`EXPIRE 0`	
S	`UIDL`	
S	`RESP-CODES`	
S	`X-LOCALTIME Fri, 27 Feb 2004 15:49:04 +0100`	
S	`IMPLEMENTATION Qpopper-version-4.0.3`	Строка IMPLEMENTATION доступна только авторизованным пользователям
S	`.`	Конец ответа
C	`LIST`	Запрос списка сообщений
S	`+OK 3 visible messages (223385 octets)`	3 сообщения, 223385 октетов
S	`1 111293`	Размер первого сообщения: 111293 октета
S	`2 111285`	Размер второго сообщения: 111295 октетов
S	`3 807`	Размер третьего сообщения: 807 октетов
S	`.`	Конец списка
C	`UIDL`	Запрос списка идентификаторов сообщений
S	`+OK uidl command accepted.`	
S	`1 44b61790f4bb2`	Идентификаторы
S	`2 0e9c69b8c1feb`	
S	`3 e30d593d3af4c`	
S	`.`	Конец списка
C	`TOP 3 1`	Запрос первой строчки третьего сообщения
S	`+OK Message follows`	
S	`Return-Path: <lonk@pds.sut.ru>`	Передается заголовок сообщения. Пустая строка – конец заголовка
S	`...`	

Пример сеанса POP3 — 3

S	`Status: RO`	
S		
S	`Привет!`	Первая строка сообщения
S	`.`	Конец ответа
C	`DELE 3`	Удалить третье сообщение
S	`+OK Message 3 has been deleted.`	Сообщение удалено (на самом деле только помечено для удаления)
C	`STAT`	Запрос количества сообщений
S	`+ OK 2 222578`	Осталось два сообщения
C	`RETR 3`	Запрос третьего сообщения
S	`-ERR Message 3 has been deleted.`	Невозможно получить сообщение, помеченное для удаления
C	`RSET`	Отмена всех удалений
S	`+OK Maildrop has 3 messages (223385 octets)`	В ящике снова три сообщения
C	`RETR 3`	Запрос третьего сообщения
S	`+OK 807 octets`	
S	`Return-Path: <lonk@pds.sut.ru>`	Передается заголовок сообщения
S	`...`	
S	`Status: RO`	
S		
S	`Привет!`	Передается полный текст сообщения
S	`Это тестовое сообщение.`	
S	`.`	
C	`QUIT`	Конец работы
S	`+OK Pop server at myhost.ru signing off.`	

Протокол IMAP

Область применения протокола IMAP (Internet Message Access Protocol) аналогична области применения протокола POP3: он тоже предназначен для получения почты и используется на участке между MUA получателя и хранилищем сообщений. IMAP предоставляет более широкие возможности работы с почтовыми ящиками, чем POP3: он позволяет работать с несколькими почтовыми ящиками на одном или нескольких серверах IMAP как с файлами и каталогами на собственной машине пользователя. Обычно почтовые ящики сервера IMAP действительно представляют собой файлы в специальном каталоге сервера и его подкаталогах.
Сервер IMAP способен анализировать сообщение: выделять заданные поля заголовка и разбирать структуру тела сообщения.
В отличие от серверов POP3, серверы IMAP не должны блокировать ящик на время сеанса – несколько клиентов могут одновременно работать с одним и тем же ящиком. Множественный доступ к почтовым ящикам связан с рядом проблем, особенно, если информация в ящиках доступна для записи. Различные способы разрешения этих проблем описаны в RFC 2180.
Довольно часто IMAP используется в организациях, где пользователям нужно предоставить возможность совместно работать с одними и теми же почтовыми ящиками. Он удобен для работы с новостями USENET. Также протокол можно использовать для работы с личными каталогами и файлами пользователя, расположенными на сервере. Впрочем, для этой цели целесообразнее использовать протоколы, специально предназначенные для работы с каталогами на файловом сервере.

Хотя программное обеспечение, реализующее протокол IMAP, постоянно совершенствуется, IMAP менее защищен, чем POP3. Возможность хранить сообщения на сервере может стать причиной злоупотреблений со стороны пользователей, которые будут переполнять хранилище сообщений ненужной информацией.
Протокол IMAP предполагает в основном работу пользователей с почтовыми ящиками непосредственно на сервере, в отличие от протокола POP3, который ориентирован на то, что клиент забирает пришедшую почту и разбирает ее уже на своей машине (RFC 1733). Это делает IMAP неудобным для пользователей, подключающихся к сети кратковременно, только для того, чтобы получить или отослать почту. Во всяком случае, многие преимущества IMAP таким пользователям недоступны. При работе по протоколу IMAP клиенту желательно иметь доступ к сети все время, пока он работает с почтой.

Протокол IMAP

Протокол IMAP позволяет пользователю работать с множеством почтовых ящиков, расположенных, возможно, на разных серверах.
Допускается иерархическое расположение почтовых ящиков в каталогах и их подкаталогах, причем имена каталогов и почтовых ящиков сами по себе не различаются. Почтовый ящик может быть только конечным элементом иерархической структуры, он не может содержать никаких нижестоящих элементов. Каталог может содержать подкаталоги и почтовые ящики, но он не содержит сообщений и не может быть выбран командой `SELECT`.
Символ, используемый в качестве иерархического разделителя, может различаться в зависимости от используемого на сервере программного обеспечения:

- косая черта («/»), если сервер работает под управлением операционной системы, совместимой с UNIX;
- обратная косая черта («\») для операционной системы Windows:
- точка («.») для имен групп новостей USENET.

Чтобы использовать и различать разные пространства имен на одном сервере IMAP, имена, принадлежащие каждому из используемых пространств, должны начинаться с некоторого префикса, обычно начинающегося символом «#». Естественно, запросы, в которых путь к ящику начинается с одного префикса, будут давать отличные результаты от таких же запросов, начинающихся с другого префикса. Используемое по умолчанию пространство имен может префикса не иметь.
Клиент может выяснить, какие именно пространства имен для почтовых ящиков каких типов поддерживаются данным сервером IMAP, если сервер поддерживает расширение NAMESPACE. Префикс и иерархический разделитель конкретного имени почтового ящика или каталога можно выяснить при помощи команды `LIST`.

Большие возможности протокола IMAP создают большие сложности при разработке, настройке и эксплуатации серверов и клиентов. Некоторые рекомендации по этим вопросам даны в RFC 2683. В общем случае можно посоветовать использовать протокол IMAP только в том случае, если возможности протокола POP3 не достаточны для работы пользователей с их почтовыми ящиками.

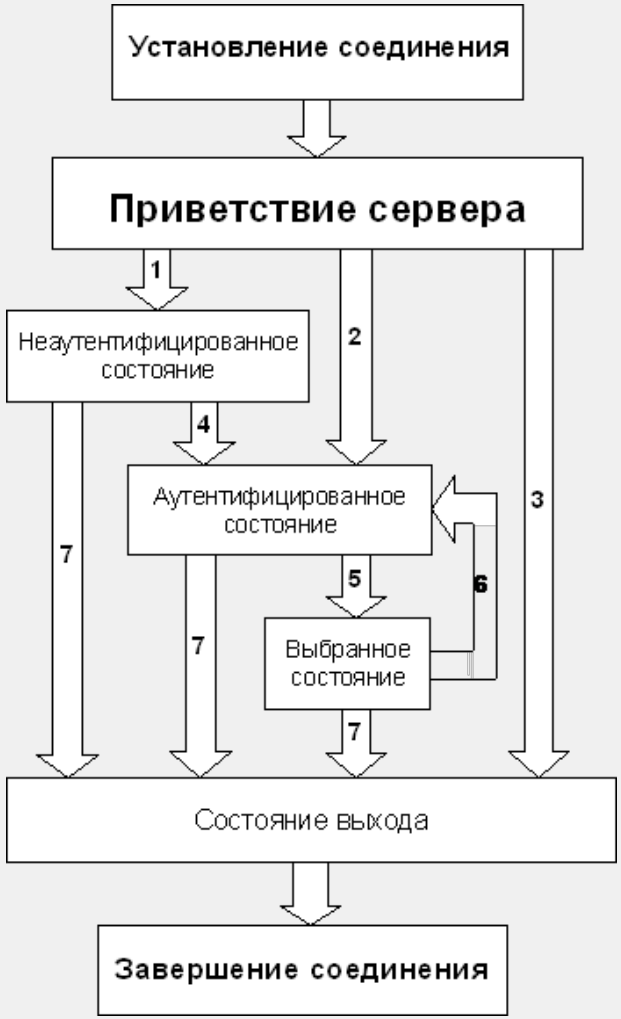
Сеанс протокола IMAP

Сервер IMAP ожидает соединения от клиентов на порту TCP 143. После установления соединения сервер посылает свое приветствие клиенту, и начинается диалог, в котором клиент посылает серверу команды, а сервер сообщает о результатах их выполнения или присылает затребованную клиентом информацию. Как и сеанс POP3, сеанс IMAP делится на несколько состояний (states). Допустимый набор команд зависит от текущего состояния сеанса. Сеанс может находиться в одном из следующих состояний:

1. Неаутентифицированное состояние (Not Authenticated State): клиент должен пройти процедуру аутентификации прежде, чем сможет выполнять большинство команд.
2. Аутентифицированное состояние (Authenticated State): клиент аутентифицирован и должен выбрать почтовый ящик, прежде чем сможет работать с отдельными сообщениями.
3. Выбранное состояние (Selected State): почтовый ящик выбран;
4. Состояние выхода (Logout State): сеанс завершается.

Схема переходов между состояниями сеанса IMAP представлена на рисунке.
Переходы, обозначенные цифрами:

1. Соединение без предварительной аутентификации.
2. Соединение с предварительной аутентификацией.
3. Отвергнутое соединение.
4. Успешная аутентификация.
5. Успешное выполнение команды `SELECT` или `EXAMINE`.
6. Команда `CLOSE` или неудачное завершение команды `SELECT` или `EXAMINE`.
7. Команда `LOGOUT` или потеря связи.



Команды клиента и ответы сервера IMAP

Команда клиента состоит из идентификатора (ярлыка) – короткой строкой, состоящей из букв и цифр, не повторяющейся в других командах в течение всего сеанса. За ярлыком следует сама команда и ее аргументы. Регистр символов в названиях команд, как и в большинстве аргументов, как правило, не имеет значения.
Кроме стандартных команд, которые обязательно должны поддерживаться, имеются также дополнительные команды, описанные в стандартах и поддерживаемые серверами IMAP как элементы расширений. Разработчики также могут добавлять в своих реализациях новые команды. Названия таких нестандартизированных команд должны начинаться с буквы «`X`». Имена стандартных команд с буквы «`X`» начинаться не могут.
Все ответы сервера начинаются с метки, после которой следует отделенный пробелом текст.
В ответах сервера, сообщающих об исполнении команд, в качестве метки используется ярлык соответствующей команды. Это помеченные (tagged) ответы. За ним следует одно из ключевых слов:

- `OK` (успешное выполнение);
- `NO` (невыполнение);
- `BAD` (ошибка в команде).

Источники

- Материалы с сайта `https://wikipedia.org/`
- Telecommunication technologies — телекоммуникационные технологии / Ю. А. Семенов.
  URL: `http://book.itep.ru/`
- RFC 5321. Simple Mail Transfer Protocol.
- RFC 1939. Post Office Protocol - Version 3.
- RFC 3501. Internet Message Access Protocol - Version 4rev1.
- Электронный курс «Структура и протоколы электронной почты в INTERNET».

Сети связи
Протоколы, сервисы и услуги в Интернет и IP-сетях

Тема № 14
Протоколы WWW — HTTP и HTTPS

доц. каф. СС и ПД, к.т.н. С. С. Владимиров

2020 г.

Протокол HTTP

HyperText Transfer Protocol (HTTP)

Протокол передачи гипертекста. Используется службой WWW для передачи Web-страниц.

- RFC 1945 Hypertext Transfer Protocol - HTTP/1.0. May 1996.
- RFC 2616 Hypertext Transfer Protocol - HTTP/1.1. June 1999.
- RFC 7540 Hypertext Transfer Protocol Version 2 (HTTP/2). May 2015

Транспортным протоколом для HTTP является протокол TCP, причем сервер HTTP (сервер Web) находится в состоянии ожидания соединения со стороны клиента стандартно по порту 80 TCP, а клиент HTTP (браузер Web) является инициатором соединения.
Основным объектом манипуляции в HTTP является ресурс, на который указывает URI (Uniform Resource Identifier) в запросе клиента. В самом общем случае URI выглядит следующим образом:

```
protocol://user:password@host:port/path/file?parameters#fragment
```

- `protocol` — прикладной протокол, посредством которого получают доступ к ресурсу;
- `user` — пользователь, от имени которого получают доступ к ресурсу;
- `password` — пароль пользователя для аутентификации при доступе к ресурсу;
- `host` — IP-адрес или имя сервера, на котором расположен ресурс;
- `port` — номер порта, на котором работает сервер, предоставляющий доступ к ресурсу;
- `path` — путь к файлу, содержащему ресурс;
- `file` — файл, содержащий ресурс;
- `parameters` — параметры для обработки ресурсом-программой;
- `fragment` — точка в файле, начиная с которой следует отображать ресурс.

Сообщения HTTP

Взаимодействие между клиентом и сервером Web осуществляется путем обмена сообщениями. Сообщения HTTP делятся на *запросы* клиента серверу и *ответы* сервера клиенту. HTTP не сохраняет своего состояния. Это означает отсутствие сохранения промежуточного состояния между парами «запрос–ответ». Компоненты, использующие HTTP, могут самостоятельно осуществлять сохранение информации о состоянии, связанной с последними запросами и ответами (например, «куки» на стороне клиента, «сессии» на стороне сервера). Браузер, посылающий запросы, может отслеживать задержки ответов. Сервер может хранить IP-адреса и заголовки запросов последних клиентов. Однако сам протокол не осведомлён о предыдущих запросах и ответах, в нём не предусмотрена внутренняя поддержка состояния, к нему не предъявляются такие требования.

Общий формат сообщений HTTP

```
начальная строка
заголовок 1: значение
заголовок 2: значение
...
заголовок N: значение
CR LF (пустая строка)
тело сообщения (может
отсутствовать)
```

Формат начальной строки (start-line) клиента и сервера различаются.
Каждый заголовок состоит из названия, символа двоеточия ":"и значения.

Виды заголовков HTTP-сообщения

- **общие заголовки (general-headers)**, которые могут присутствовать как в запросе, так и в ответе;
- **заголовки запросов (request-headers)**, которые могут присутствовать только в запросе;
- **заголовки ответов (response-headers)**, которые могут присутствовать только в ответе;
- **заголовки объекта (entity-headers)**, которые относятся к телу сообщения и описывают его содержимое.

Сообщения HTTP. HTTP-запрос

Тело сообщения

В теле сообщения содержится собственно передаваемая информация. Тело сообщения представляет собой последовательность октетов (байтов). Тело сообщения может быть закодировано, например, для уменьшения объема передаваемой информации, при этом способ кодирования указывается в заголовке объекта `Content-Encoding`.

HTTP-запрос

Запрос от клиента к серверу состоит из строки запроса (request-line), заголовков (общих, запросов, объекта) и, возможно, тела сообщения.

Строка запроса:

```
<Метод HTTP> <Идентификатор запрашиваемого ресурса> <Версия HTTP>
```

Метод HTTP

Метод HTTP (или команда) — последовательность из любых символов, кроме управляющих и разделителей, указывающая на основную операцию над ресурсом. Обычно представляет собой короткое английское слово, записанное заглавными буквами. Название метода чувствительно к регистру.

Каждый сервер обязан поддерживать как минимум методы `GET` и `HEAD`. Если сервер не распознал указанный клиентом метод, то он должен вернуть статус 501 (Not Implemented). Если серверу метод известен, но он неприменим к конкретному ресурсу, то возвращается сообщение с кодом 405 (Method Not Allowed). В обоих случаях в сообщение ответа включается заголовок `Allow` со списком поддерживаемых методов.

Кроме методов `GET` и `HEAD`, часто применяется метод `POST`.

Методы HTTP

OPTIONS

Используется для определения возможностей веб-сервера или параметров соединения для конкретного ресурса. В ответ — заголовок Allow со списком поддерживаемых методов. Также может включаться информация о поддерживаемых расширениях.
Для того, чтобы узнать возможности всего сервера, клиент должен указать в URI звёздочку — «*». Запросы

```
OPTIONS * HTTP/1.1
```

могут также применяться для проверки работоспособности сервера (аналогично «пингованию») и тестирования на предмет поддержки сервером протокола HTTP версии 1.1.
Результат выполнения метода не кэшируется.

GET

Используется для запроса содержимого указанного ресурса. С помощью метода `GET` можно также начать какой-либо процесс. Можно передавать параметры выполнения запроса в URI целевого ресурса после символа «?»:

```
GET /path/resource?param1=value1&param2=value2 HTTP/1.1
```

Метод является идемпотентным: на одинаковый `GET` — одинаковый результат.

HEAD

Аналогичен `GET`, за исключением того, что в ответе сервера отсутствует тело. Запрос `HEAD` обычно применяется для извлечения метаданных, проверки наличия ресурса (валидация URL) и чтобы узнать, не изменился ли он с момента последнего обращения.

DELETE

Удаляет указанный ресурс.

Методы HTTP

POST

Применяется для передачи пользовательских данных заданному ресурсу. Аналогично с помощью метода POST обычно загружаются файлы на сервер. В отличие от метода GET, метод POST не считается идемпотентным — многократное повторение одних и тех же запросов POST может возвращать разные результаты.

PUT

Применяется для загрузки содержимого запроса на указанный в запросе URI.
Фундаментальное различие методов POST и PUT заключается в понимании предназначений URI ресурсов. Метод POST предполагает, что по указанному URI будет производиться обработка передаваемого клиентом содержимого. Используя PUT, клиент предполагает, что загружаемое содержимое соответствует находящемуся по данному URI ресурсу.

PATCH

Аналогично PUT, но применяется только к фрагменту ресурса.

TRACE

Возвращает полученный запрос так, что клиент может увидеть, какую информацию промежуточные серверы добавляют или изменяют в запросе.

CONNECT

Преобразует соединение запроса в прозрачный TCP/IP туннель, обычно чтобы содействовать установлению защищенного SSL соединения через нешифрованный прокси.

Сообщения HTTP. HTTP-ответ

Ответ HTTP-сервера клиенту

```
<Версия HTTP> <Код состояния> <Поясняющая фраза>
```

Код состояния (Status-Code) – это целочисленный трехразрядный код результата понимания и удовлетворения запроса.
Поясняющая фраза (Reason-Phrase) – короткое текстовое описание кода состояния.
Код состояния предназначен для обработки программным обеспечением, а поясняющая фраза предназначена для пользователей.

Код состояния

Первая цифра кода состояния определяет класс ответа. Последние две цифры не имеют определенной роли в классификации. Имеется 5 значений первой цифры:

`1xx` Информационные коды – запрос получен, продолжается обработка.

`2xx` Успешные коды – действие было успешно получено, понято и обработано.

`3xx` Коды перенаправления – для выполнения запроса должны быть предприняты дальнейшие действия.

`4xx` Коды ошибок клиента – запрос имеет ошибку синтаксиса или не может быть выполнен.

`5xx` Коды ошибок сервера – сервер не в состоянии выполнить допустимый запрос.

Особенности протокола HTTP

Хотя протокол HTTP разрабатывался как средство работы с ресурсами сервера, у него отсутствуют в явном виде средства навигации среди этих ресурсов. Например, клиент не может явным образом запросить список доступных файлов, как в протоколе FTP. Предполагалось, что конечный пользователь уже знает URI необходимого ему документа, закачав который, он будет производить навигацию благодаря гиперссылкам. Это вполне нормально и удобно для человека, но затруднительно, когда стоят задачи автоматической обработки и анализа всех ресурсов сервера без участия человека. Решение этой проблемы лежит полностью на плечах разработчиков приложений, использующих данный протокол.

Большинство протоколов предусматривают установление TCP-сессии, в ходе которой один раз происходит авторизация, и дальнейшие действия выполняются в контексте этой авторизации. HTTP же устанавливает отдельную TCP-сессию на каждый запрос; в более поздних версиях HTTP было разрешено делать несколько запросов в ходе одной TCP-сессии, но браузеры обычно запрашивают только страницу и включённые в неё объекты (картинки, каскадные стили и т. п.), а затем сразу разрывают TCP-сессию. Для поддержки авторизованного (неанонимного) доступа в HTTP используются cookies; причём такой способ авторизации позволяет сохранить сессию даже после перезагрузки клиента и сервера.

При доступе к данным по FTP или по файловым протоколам тип файла (точнее, тип содержащихся в нём данных) определяется по расширению имени файла, что не всегда удобно. HTTP перед тем, как передать сами данные, передаёт заголовок «`Content-Type: тип/подтип`», позволяющую клиенту однозначно определить, каким образом обрабатывать присланные данные. Это особенно важно при работе с CGI-скриптами, когда расширение имени файла указывает не на тип присылаемых клиенту данных, а на необходимость запуска данного файла на сервере и отправки клиенту результатов работы программы, записанной в этом файле (при этом один и тот же файл в зависимости от аргументов запроса и своих собственных соображений может порождать ответы разных типов — в простейшем случае картинки в разных форматах).
Кроме того, HTTP позволяет клиенту прислать на сервер параметры, которые будут переданы запускаемому CGI-скрипту. Для этого же в HTML были введены формы.

Примеры диалогов HTTP. Перенаправление на другой домен

Предположим, что у вымышленной компании Example Corp. есть основной сайт по адресу `http://example.com` и домен-псевдоним `example.org`. Клиент посылает запрос страницы «О компании» на вторичный домен (часть заголовков опущена):

```
GET /about.html HTTP/1.1
Host: example.org
User-Agent: MyLonelyBrowser/5.0
```

Так как домен `example.org` не является основным и компания не собирается в будущем его использовать в других целях, их сервер вернёт код для постоянного перенаправления, указав в заголовке Location целевой URL:

```
HTTP/1.x 301 Moved Permanently
Location: http://example.com/about.html#contacts
Date: Thu, 19 Feb 2009 11:08:01 GMT
Server: Apache/2.2.4
Content-Type: text/html; charset=windows-1251
Content-Length: 110
(пустая строка)
<html><body><a href=''http://example.com/about.html#contacts''>Click
here</a></body></html>
```

В заголовке Location можно указывать фрагменты как в данном примере. Браузер не указал фрагмент в запросе, так как его интересует весь документ. Но он автоматически прокрутит страницу до фрагмента «`contacts`», как только загрузит её. В тело ответа также был помещён короткий HTML-документ со ссылкой, с помощью которой посетитель попадёт на целевую страницу, если браузер не перейдёт на неё автоматически. Заголовок Content-Type содержит характеристики именно этого HTML-пояснения, а не документа, который находится по целевому URI.

Примеры диалогов HTTP. Перенаправление на региональный домен

Допустим, эта же компания Example Corp. имеет несколько региональных представительств по всему миру. И для каждого представительства у них есть сайт с соответствующим ccTLD. Запрос главной страницы основного сайта `example.com` может выглядеть так:

```
GET / HTTP/1.1
Host: example.com
User-Agent: MyLonelyBrowser/5.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: ru,en-us;q=0.7,en;q=0.3
Accept-Charset: windows-1251,utf-8;q=0.7,*;q=0.7
```

Сервер принял во внимание заголовок Accept-Language и сформировал ответ с временным перенаправлением на российский сервер example.ru, указав его адрес в заголовке Location:

```
HTTP/1.x 302 Found
Location: http://example.ru/
Cache-Control: private
Date: Thu, 19 Feb 2009 11:08:01 GMT
Server: Apache/2.2.6
Content-Type: text/html; charset=windows-1251
Content-Length: 82
(пустая строка)
<html><body><a href="http://example.ru»Example Corp.</a></body></html>
```

Обратите внимание на заголовок Cache-Control. Значение «private» сообщает остальным серверам (в первую очередь прокси) что ответ может кэшироваться только на стороне клиента. В противном случае не исключено, что следующие посетители из других стран будут переходить всё время не в своё представительство.

HTTP cookie

HTTP cookie

Куки — небольшой фрагмент данных, отправленный веб-сервером и хранимый на компьютере пользователя. Веб-клиент всякий раз при попытке открыть страницу соответствующего сайта пересылает этот фрагмент данных веб-серверу в составе HTTP-запроса. Применяется для сохранения данных на стороне пользователя, на практике обычно используется для:

- аутентификации пользователя;
- хранения персональных предпочтений и настроек пользователя;
- отслеживания состояния сеанса пользователя;
- ведения статистики о пользователях.

HTTP cookie. Установка куки

Установка куки

Запрашивая страницу, браузер отправляет веб-серверу короткий текст с HTTP-запросом. Например, для доступа к странице `http://www.example.org/index.html`, браузер отправляет на сервер `www.example.org` следующий запрос:

```
GET /index.html HTTP/1.1
Host: www.example.org
```

Сервер отвечает, отправляя запрашиваемую страницу вместе с текстом, содержащим HTTP-ответ. Там может содержаться указание браузеру сохранить куки:

```
HTTP/1.1 200 OK
Content-type: text/html
Set-Cookie: name=value
```

Строка `Set-cookie` отправляется лишь тогда, когда сервер желает, чтобы браузер сохранил куки. В этом случае, если куки поддерживаются браузером и их приём включён, браузер запоминает строку `name=value` (имя = значение) и отправляет её обратно серверу с каждым последующим запросом. Например, при запросе следующей страницы `http://www.example.org/spec.html` браузер пошлёт серверу `www.example.org` следующий запрос:

```
GET /spec.html HTTP/1.1
Host: www.example.org
Cookie: name=value
Accept: */*
```

Этот запрос отличается от первого запроса тем, что содержит строку, которую сервер отправил браузеру ранее. Таким образом, сервер узнает, что этот запрос связан с предыдущим. Сервер отвечает, отправляя запрашиваемую страницу и, возможно, добавив новые куки.
Значение куки может быть изменено сервером путём отправления новых строк `Set-Cookie: name=newvalue`. После этого браузер заменяет старое куки с тем же name на новую строку.

HTTP cookie. Атрибуты куки

Атрибуты куки

Кроме пары имя/значение, куки может содержать срок действия, путь и доменное имя. RFC 2965 также предусматривает, что куки должны обязательно иметь номер версии, но это используется редко. Эти атрибуты должны идти после пары `name=newvalue` и разделяться точкой с запятой. Например:

```
Set-Cookie: name=newvalue; expires=date; path=/; domain=.example.org.
```

Домен и путь говорят браузеру, что куки должна быть отправлена обратно на сервер при запросах URL для указанного домена и пути. Если они не указаны, используются домен и путь запрошенной страницы.
Фактически, куки определяются тройкой параметров имя-домен-путь. Иными словами, куки с разными путями или доменами являются разными куки, даже если имеют одинаковые имена. Соответственно, куки меняется на новое, только если новое куки имеет те же имя, путь и домен.
Дата истечения указывает браузеру, когда удалить куки. Если срок истечения не указан, куки удаляется по окончании пользовательского сеанса, то есть с закрытием браузера. Если же указана дата истечения срока хранения, куки становится постоянной до указанной даты.
Срок хранения куки истекает в следующих случаях:

- В конце сеанса (например, когда браузер закрывается), если куки не являются постоянными.
- Дата истечения была указана, и срок хранения вышел.
- Браузер удалил куки по запросу пользователя.

Сервер может узнать, когда истекают сроки хранения куки, только когда браузер отправляет на сервер эту информацию.

HTTP cookie. Аутентификация

Аутентификация с использованием куки

Куки могут использоваться сервером для опознания ранее аутентифицированных пользователей. Это происходит следующим образом:
Пользователь вводит имя пользователя и пароль в текстовых полях страницы входа и отправляет их на сервер.
Сервер получает имя пользователя и пароль, проверяет их и, при их правильности, отправляет страницу успешного входа, прикрепив куки с неким идентификатором сессии. Эта куки может быть действительна не только для текущей сессии браузера, но может быть настроена и на длительное хранение.
Каждый раз, когда пользователь запрашивает страницу с сервера, браузер автоматически отправляет куки с идентификатором сессии серверу. Сервер проверяет идентификатор по своей базе идентификаторов и, при наличии в базе такого идентификатора, «узнаёт» пользователя.

Протокол HTTPS

HTTPS (HyperText Transfer Protocol Secure)

Это расширение протокола HTTP, поддерживающее шифрование. Данные, передаваемые по протоколу HTTP, «упаковываются» в криптографический протокол SSL или TLS. В отличие от HTTP, для HTTPS по умолчанию используется TCP-порт 443. Фактически, HTTPS не является отдельным протоколом. Это обычный HTTP, работающий через шифрованные транспортные механизмы SSL и TLS.

Протокол был разработан компанией Netscape Communications для браузера Netscape Navigator в 1994 году. HTTPS широко используется в мире веб и поддерживается всеми популярными браузерами.

Чтобы подготовить веб-сервер для обработки https-соединений, администратор должен получить и установить в систему сертификат для этого веб-сервера. Сертификат состоит из 2 частей (2 ключей) — public и private. Public-часть сертификата используется для зашифрования трафика от клиента к серверу в защищённом соединении, private-часть — для расшифрования полученного от клиента зашифрованного трафика на сервере. После того как пара ключей приватный/публичный сгенерированы, на основе публичного ключа формируется запрос на сертификат в Центр сертификации, в ответ на который ЦС высылает подписанный сертификат. ЦС, при подписывании проверяет клиента, что позволяет ему гарантировать что держатель сертификата является тем, за кого себя выдаёт (обычно это платная услуга).

Существует возможность создать такой сертификат, не обращаясь в ЦС. Такие сертификаты могут быть созданы для серверов, работающих под Unix. Подписываются такие сертификаты этим же сертификатом и называются самоподписанными (self-signed).

Эта система также может использоваться для аутентификации клиента, чтобы обеспечить доступ к серверу только авторизованным пользователям. Для этого администратор обычно создаёт сертификаты для каждого пользователя и загружает их в браузер каждого пользователя. Также будут приниматься все сертификаты, подписанные организациями, которым доверяет сервер. Такой сертификат обычно содержит имя и адрес электронной почты авторизованного пользователя, которые проверяются при каждом соединении, чтобы проверить личность пользователя без ввода пароля.

В HTTPS для шифрования используются ключи длиной 40, 56, 128 или 256 бит. Многие современные сайты требуют использования новых версий браузеров, поддерживающих шифрование с длиной ключа 128 бит, с целью обеспечить достаточный уровень безопасности. Такое шифрование значительно затрудняет злоумышленнику поиск паролей и другой личной информации.

Протокол HTTP/2

Протокол HTTP/2 разработан рабочей группой IETF «Hypertext Transfer Protocol working group» на основе ранее представленного компанией Google протокола SPDY. Спецификация HTTP/2 опубликована как RFC 7540 в мае 2015. Протокол HTTP/2 является бинарным. По сравнению с предыдущим стандартом изменены способы разбиения данных на фрагменты и транспортирования их между сервером и клиентом.
В HTTP/2 сервер имеет право послать то содержимое, которое ещё не было запрошено клиентом. Это позволит серверу сразу выслать дополнительные файлы, которые потребуются браузеру для отображения страниц, без необходимости анализа браузером основной страницы и запрашивания необходимых дополнений.

Поддержка веб-браузерами

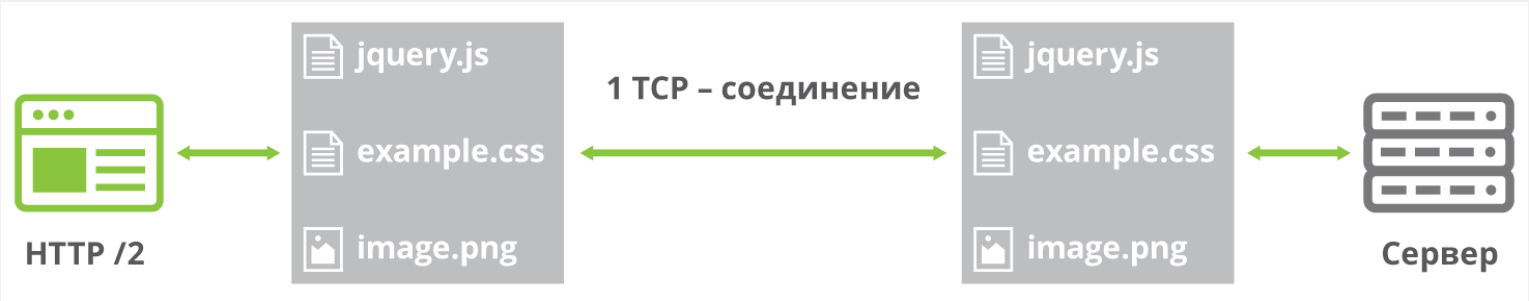
- Chrome (и произв.) — с 2014. Поддерживается только режим HTTP/2 поверх TLS.
- Firefox — с версии 36. Поддерживается только режим HTTP/2 поверх TLS.
- Internet Explorer — с версии 11 для Windows 8.1. Поддерживается только режим HTTP/2 поверх TLS.
- Microsoft Edge.
- Safari — с версии 9.

Поддержка веб-серверами

- IIS — в Windows 10 и Windows Server 2016.
- Apache — с версии 2.4.17 (mod_http2 module).
- nginx — с версии 1.9.5.

Механизмы HTTP/2

Мультиплексирование — множество запросов в рамках одного TCP-соединения



Приоритизация — каждому запросу можно назначить приоритет

Существует два подхода к назначению приоритетов:

1. На основе веса. Каждый поток получает определённый вес. Потом на основе веса сервер распределяет нагрузку между потоками. (Унаследовано от SPDY.)
2. На основе зависимостей. Браузер просит сервер загружать определённые элементы контента в первую очередь. Например, браузер может попросить сервер сначала загрузить CSS-файлы или JavaScript, а уже потом — HTML или изображения.

В HTTP/2 приоритизация является не обязательным, а желательным методом. Однако мультиплексирование без неё работать должным образом не будет. Скорость загрузки может быть даже ниже, чем HTTP/1.1. Ресурсы с более низким приоритетом будут занимать полосу, что приведёт снижению производительности.

Механизмы HTTP/2

Server Push

Позволяет серверу сразу же, не дожидаясь ответа веб-браузера, добавить нужные по его мнению файлы в кэш для быстрой выдачи.

Компрессия заголовков HPACK

В HTTP/2 заголовки передаются в сжатом виде. Благодаря этому уменьшается количество информации, которой обмениваются между собой сервер и браузер. Вместо алгоритмов gzip/deflate используется HPACK.

Шифрование

В HTTP/2 оно обязательного характера не имеет. Однако разработчики браузеров приняли решение внедрить новый протокол только для TLS(HTTPS)-соединений.

Оптимизация HTTP/2

Главная оптимизация HTTP/2 по сравнению с HTTP 1.1 — отключение или модификация многих оптимизаций прошлой версии протокола.

- Отказаться от распределения множества файлов по различным доменам и CDN. Для HTTP 1.1 это решает проблему параллельных соединений. В случае с HTTP/2 такое решение ухудшает производительность и нивелирует приоритизацию трафика
- Отказ от объединения изображений в одно большое.
- Отказ от объединения файлов (в основном CSS и JavaScript).

Протоколы HTTP/3 и QUIC

В настоящий момент под HTTP/3 понимают так называемый протокол HTTP-over-QUIC.

Протокол QUIC — Quick UDP Internet Connections

QUIC представляет собой замену TCP, которая работает поверх UDP. Изначально создан компанией Google. В первое время QUIC именовали “HTTP/2-encrypted-over-UDP” и был фактически связкой нового транспортного протокола с HTTP/2. В IETF новый протокол разделили на две части: транспорт, сохранивший название QUIC, и HTTP, ныне HTTP/3. Разделение связано с тем, что транспортный протокол можно использовать для передачи данных любых прикладных протоколов. Разработкой транспортного протокола занимается рабочая группа QUIC Working Group в IETF. Последняя версия черновика RFC датирована 17 декабря 2018. В то же время Google продолжила работу над своей собственной реализацией и внедрила её в браузер Chrome.

К преимуществам QUIC относят отсутствие установления соединения (реально соединение есть, но оно происходит на уровне приложений, а не на транспортном) и отсутствие очередей обработки пакетов.

К недостаткам, особенно реализации от Google, относят меньшую производительность в сетях, не гарантирующих порядок доставки пакетов. Также отмечалось, что в ряде случаев при потере пакетов, согласующих шифрование, при перепосылке реализация может случайно шифровать пакеты. Также отмечается повышение нагрузки на процессор из-за необходимости разбиения данных на датаграммы на уровне пользователя.

Особенности QUIC

Превентивная коррекция ошибок

Каждый пересылаемый пакет содержит в себе данные других пакетов, что позволяет воссоздать любой потерянный пакет по данным в его соседях, без перезапроса потерянного пакета. Недостаток решения: каждый пакет становится больше — на 10% в текущей реализации. Таким образом, при доле потерь пакетов не более 0.1, перезапросов не потребуется. На практике выигрыш был неочевиден (не доказан экспериментально) и данный механизм исключили из спецификации.

Возобновление сессии и параллельные загрузки

QUIC вводит понятие идентификатора соединения — Connection UUID. Появляется возможность перейти с одного канала ПД на другой с сохранением Connection UUID, избежав затрат на пересоздание соединения. Соответственно, можно использовать несколько каналов параллельно.

Источники

- Материалы с сайта `https://wikipedia.org/`
- Telecommunication technologies — телекоммуникационные технологии / Ю. А. Семенов.
  URL: `http://book.itep.ru/`
- Что такое cookies и как с ними работать / А. А. Аликберов.
  URL: `http://citforum.ru/internet/html/cookie.shtml`
- RFC 7540. Hypertext Transfer Protocol Version 2 (HTTP/2).
  URL: `https://tools.ietf.org/html/rfc7540`
- HTTP/2: готовимся к переходу / А. Емельянов, комп. Selectel // Хабр.
  URL: `https://habr.com/company/selectel/blog/278167/`
- Оптимизация в HTTP/2 // Ruhighload.com
  URL: https://ruhighload.com/Оптимизация+в+HTTP/2
- Протокол QUIC: переход Web от TCP к UDP // Хабр.
  URL: `https://habr.com/company/infopulse/blog/315172/`
- Протокол HTTP-over-QUIC официально становится HTTP/3 // Хабр.
  URL: `https://habr.com/company/globalsign/blog/429820/`
- QUIC, TLS 1.3, DNS-over-HTTPS, далее везде // Хабр.
  URL: `https://habr.com/company/qrator/blog/416633/`
- QUIC: A UDP-Based Multiplexed and Secure Transport / J. Iyengar, M. Thomson // QUIC Working Group
  URL: `https://quicwg.org/base-drafts/draft-ietf-quic-transport.html`

Сети связи
Протоколы, сервисы и услуги в Интернет и IP-сетях

Тема № 15
Протокол DHCP

доц. каф. СС и ПД, к.т.н. С. С. Владимиров

2020 г.

Протокол DHCP

DHCP (Dynamic Host Configuration Protocol)

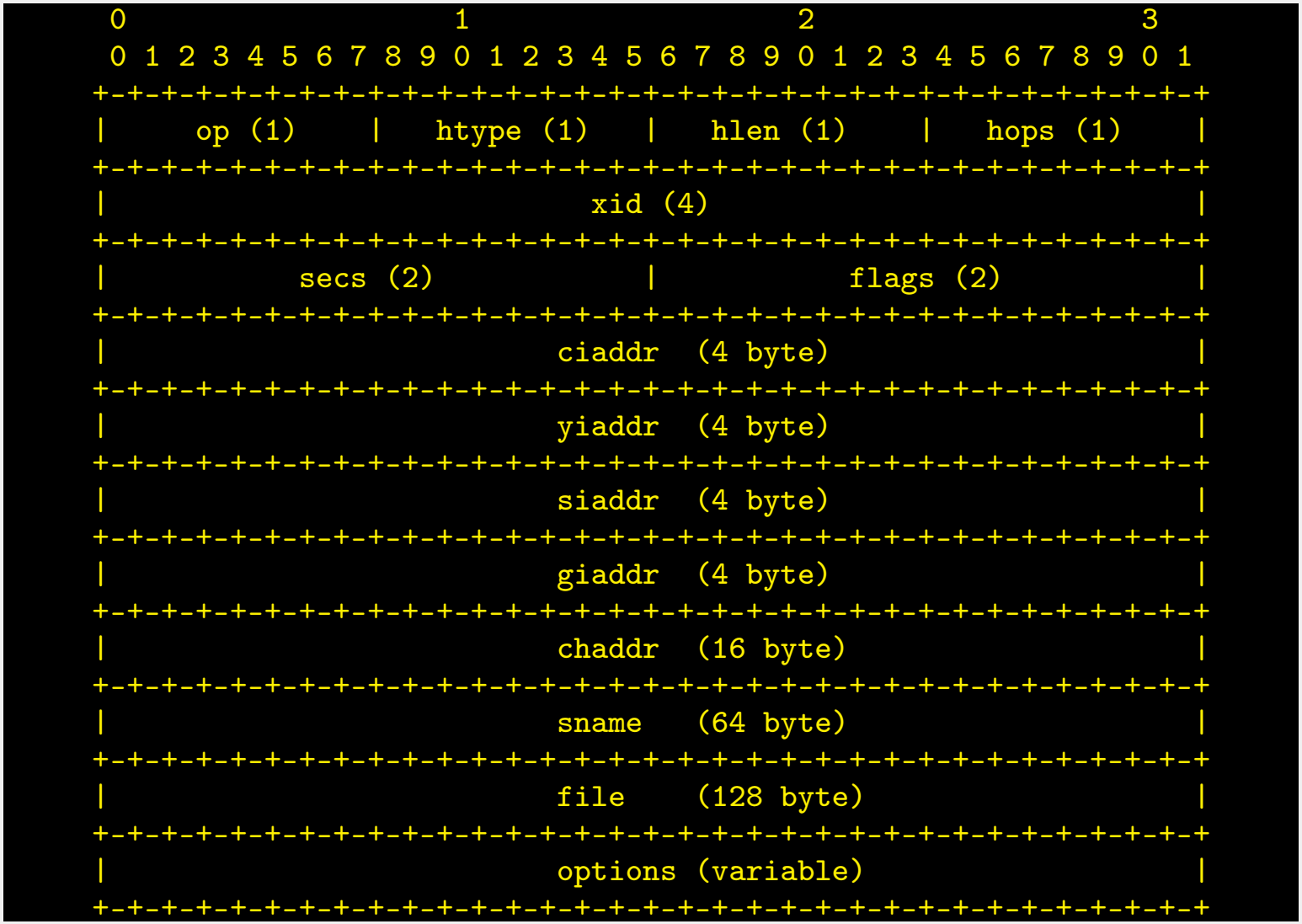
Протокол динамической настройки узла — сетевой протокол прикладного уровня, позволяющий компьютерам автоматически получать IP-адрес и другие параметры, необходимые для работы в сети TCP/IP. Работает по модели «клиент-сервер». Для автоматической конфигурации компьютер-клиент на этапе конфигурации сетевого устройства обращается к так называемому серверу DHCP, и получает от него нужные параметры.
Фактически является расширением (и заменой) протокола BOOTP, использовавшегося ранее для обеспечения бездисковых рабочих станций IP-адресами при их загрузке. Сохраняет обратную совместимость с BOOTP.
В качестве протокола транспортного уровня использует протокол UDP. Порт 67 для клиента и порт 68 для сервера.
Стандарт протокола был принят в октябре 1993 года. Действующая версия протокола (март 1997 года) описана в RFC 2131. Новая версия DHCP, предназначенная для использования в среде IPv6, носит название DHCPv6 и определена в RFC 3315 (июль 2003 года). DHCPv6 использует порты UDP 546 (клиент) и 547 (сервер).

Распределение IP-адресов

- **Ручное распределение.** При этом способе сетевой администратор сопоставляет аппаратному адресу (для Ethernet сетей это MAC-адрес) каждого клиентского компьютера определённый IP-адрес. Фактически, данный способ распределения адресов отличается от ручной настройки каждого компьютера лишь тем, что сведения об адресах хранятся централизованно (на сервере DHCP), и потому их проще изменять при необходимости.
- **Автоматическое распределение.** При данном способе каждому компьютеру на постоянное использование выделяется произвольный свободный IP-адрес из определённого администратором диапазона.
- **Динамическое распределение.** Этот способ аналогичен автоматическому распределению, за исключением того, что адрес выдаётся компьютеру не на постоянное пользование, а на определённый срок. Это называется арендой адреса. По истечении срока аренды IP-адрес вновь считается свободным, и клиент обязан запросить новый (он может оказаться тем же самым). Кроме того, клиент сам может отказаться от полученного адреса.

Помимо IP-адреса, DHCP также может сообщать клиенту дополнительные параметры (опции DHCP), необходимые для нормальной работы в сети: IP-адрес маршрутизатора по умолчанию; маска подсети; адреса серверов DNS; имя домена DNS.

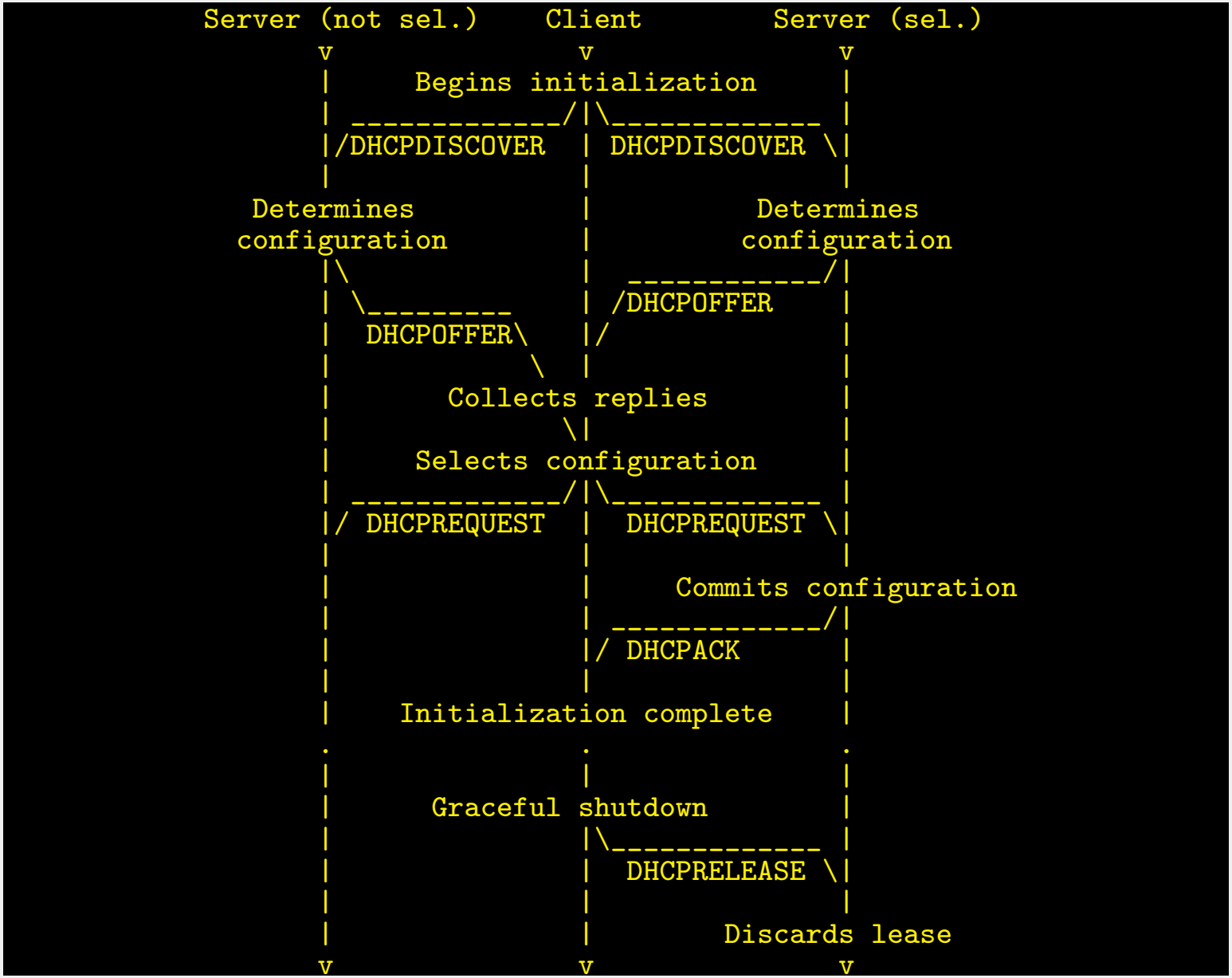
Формат кадра DHCP



Формат кадра DHCP. Описание полей

Поле	Описание	Длина
`op`	Тип сообщения. Например может принимать значения: BOOTREQUEST (1, запрос от клиента к серверу) и BOOTREPLY (2, ответ от сервера к клиенту).	1
`htype`	Тип аппаратного адреса. Допустимые значения этого поля определены в RFC1700 «Assigned Numbers». Например, для MAC-адреса Ethernet 10 Мбит/с это поле принимает значение 1.	1
`hlen`	Длина аппаратного адреса в байтах. Для MAC-адреса Ethernet — 6.	1
`hops`	Количество промежуточных маршрутизаторов (так называемых агентов ретрансляции DHCP), через которые прошло сообщение. Клиент устанавливает это поле в 0.	1
`xid`	Уникальный ID транзакции, генерируемый клиентом в начале процесса получения адреса.	4
`secs`	Время в секундах с момента начала процесса получения адреса. Может не использоваться (в этом случае оно устанавливается в 0).	2
`flags`	Поле для флагов — специальных параметров протокола DHCP.	2
`ciaddr`	IP-адрес клиента. Заполняется только в том случае, если клиент уже имеет собственный IP-адрес и способен отвечать на запросы ARP (это возможно, если клиент выполняет процедуру обновления адреса по истечении срока аренды).	4
`yiaddr`	Новый IP-адрес клиента, предложенный сервером.	4
`siaddr`	IP-адрес сервера. Возвращается в предложении DHCP.	4
`giaddr`	IP-адрес агента ретрансляции, если таковой участвовал в доставке сообщения DHCP до сервера.	4
`chaddr`	Аппаратный адрес (обычно MAC-адрес) клиента.	16
`sname`	Необязательное имя сервера в виде нуль-терминированной строки.	64
`file`	Необязательное имя файла на сервере, используемое бездисковыми рабочими станциями при удалённой загрузке. Как и sname, представлено в виде нуль-терминированной строки.	128
`options`	Поле опций DHCP. Здесь указываются различные дополнительные параметры конфигурации. В начале этого поля указываются четыре особых байта со значениями 99, 130, 83, 99 («волшебные числа»), позволяющие серверу определить наличие этого поля. Поле имеет переменную длину, однако DHCP-клиент должен быть готов принять DHCP-сообщение длиной в 576 байт (в этом сообщении поле options имеет длину 340 байт).	перем.

Пример процесса получения адреса



Пример процесса получения адреса

Процесс получения адреса состоит из четырёх этапов.

1. Обнаружение DHCP

В начале клиент выполняет широковещательный запрос по всей физической сети с целью обнаружить доступные DHCP-серверы. Он отправляет сообщение типа `DHCPDISCOVER`, при этом в качестве IP-адреса источника указывается `0.0.0.0` (так как компьютер ещё не имеет собственного IP-адреса), а в качестве адреса назначения — широковещательный адрес `255.255.255.255`.

```
Frame 34 (342 bytes on wire, 342 bytes captured) Ethernet II,
Src: 02:00:4c:4f:4f:50,
Dst: Broadcast (ff:ff:ff:ff:ff:ff) # MAC-адрес получателя широковещательный
Internet Protocol,
Src: 0.0.0.0,
Dst: 255.255.255.255 # IP-адрес также широковещательный
User Datagram Protocol,
Src Port: bootpc (68), Dst Port: bootps (67) # UDP-порты 68 и 67 - клиент и сервер
Client IP address: 0.0.0.0 # текущий адрес клиента, может содержать не нулевое значение если,
например, у клиента есть IP-адрес и он продлевает время его аренды
Your (client) IP address: 0.0.0.0 # адрес, выдаваемый DHCP-сервером при ответе
Next server IP address: 0.0.0.0 # адрес самого DHCP-сервера
Relay agent IP address: 0.0.0.0 # адрес Relay-агента, если имеется
Client MAC address: 02:00:4c:4f:4f:50 # MAC-адрес клиента
```

В поле `xid` помещается уникальный идентификатор транзакции, который позволяет отличать данный процесс получения IP-адреса от других, протекающих в то же время.

Пример процесса получения адреса

1. Обнаружение DHCP. Опции

Дальше идет поле опций, номера опций могут быть в диапазоне от 0 до 255, каждая опция имеет свое назначение:

```
Option: (t=50,l=4) Requested IP Address = 192.168.13.2
# опция 50 имеет длину 4 байта, в ней указывается IP-адрес, который хотел бы получить клиент
по возможности (последний известный клиенту IP)
Option: (t=12,l=8) Host Name = "MainHost"
# опция 12 имеет длину 8 байт, в ней указывается текущее имя хоста, которое может быть
изменено после конфигурации
Option: (t=55,l=11) Parameter Request List
# опция 55, в ней содержится список запрашиваемых клиентом параметров, в данной ситуации
клиент запрашивает 11 параметров, каждому из которых соответствует номер опции
```

Сообщение `DHCPDISCOVER` может быть распространено за пределы локальной физической сети при помощи специально настроенных агентов ретрансляции DHCP, перенаправляющих поступающие от клиентов сообщения DHCP-серверам в других подсетях.
Не всегда процесс получения IP-адреса начинается с `DHCPDISCOVER`. В случае, если клиент ранее уже получал IP-адрес и срок его аренды ещё не прошёл — клиент может пропустить стадию `DHCPDISCOVER`, начав с запроса `DHCPREQUEST`, отправляемого с идентификатором сервера, который выдал адрес в прошлый раз. В случае же отсутствия ответа от DHCP-сервера, выдавшего настройки в прошлый раз, клиент отправляет `DHCPDISCOVER`. Тем самым, клиент начинает процесс получения с начала, обращаясь уже ко всем DHCP-серверам в сегменте сети.

Пример процесса получения адреса

2. Предложение DHCP

Получив сообщение от клиента, сервер определяет требуемую конфигурацию клиента в соответствии с указанными сетевым администратором настройками. Сервер отправляет ему ответ `DHCPOFFER`, в котором предлагает конфигурацию. Предлагаемый клиенту IP-адрес указывается в поле yiaddr. Прочие параметры (такие, как адреса маршрутизаторов и DNS-серверов) указываются в виде опций в соответствующем поле.
Это сообщение DHCP-сервер отправляет хосту, пославшему `DHCPDISCOVER`, на его MAC, при определенных обстоятельствах сообщение может распространяться как широковещательная рассылка. Клиент может получить несколько различных предложений DHCP от разных серверов; из них он должен выбрать то, которое его «устраивает».

3. Запрос DHCP

Выбрав одну из конфигураций, предложенных DHCP-серверами, клиент отправляет запрос DHCP (`DHCPREQUEST`). Он рассылается широковещательно; при этом к опциям, указанным клиентом в сообщении `DHCPDISCOVER`, добавляется специальная опция — идентификатор сервера — указывающая адрес DHCP-сервера, выбранного клиентом. Таким образом клиент говорит всем серверам в широковещательном домене, какому из них он отдал предпочтение.

```
Option: (t=54,l=4) DHCP Server Identifier = 192.168.13.1
```

4. Подтверждение DHCP

Наконец, сервер подтверждает запрос и также широковещательно направляет это подтверждение (`DHCPACK`) клиенту. В теле сообщения явным образом указывает MAC-адрес клиента.

```
Client MAC address: 02:00:4c:4f:4f:50 (02:00:4c:4f:4f:50)
```

После этого клиент должен настроить свой сетевой интерфейс, используя предоставленные опции.

Проверка уникальности адресов

При назначении адресов и клиент и сервер проверяют их уникальность. Предположим, на сервере сконфигурирован пул адресов, который начинается с адреса `192.168.13.2`. Первый адрес пула назначен вручную одним из пользователей сети. При назначении такого адреса по DHCP произойдет конфликт, поэтому, для развязания конфликтов существует следующий механизм:

1	0.000000	0.0.0.0	255.255.255.255	DHCP	DHCP Discover - Transaction ID 0xe864e6be
2	0.018000	c0:00:0e:a4:00:00	Broadcast	ARP	Who has 192.168.13.2? Tell 192.168.13.1
3	0.160000	c0:01:0e:a4:00:00	c0:00:0e:a4:00:00	ARP	192.168.13.2 is at c0:01:0e:a4:00:00
4	0.825000	192.168.13.1	192.168.13.2	ICMP	Echo (ping) request
5	0.919000	192.168.13.2	192.168.13.1	ICMP	Echo (ping) reply
6	0.969000	c0:00:0e:a4:00:00	Broadcast	ARP	Who has 192.168.13.3? Tell 192.168.13.1
7	14.723000	0.0.0.0	255.255.255.255	DHCP	DHCP Request - Transaction ID 0xe864e6be
8	15.296000	192.168.13.1	255.255.255.255	DHCP	DHCP Offer - Transaction ID 0xe864e6be
9	15.923000	192.168.13.1	255.255.255.255	DHCP	DHCP ACK - Transaction ID 0xe864e6be
10	17.199000	02:00:4c:4f:4f:50	Broadcast	ARP	Gratuitous ARP for 192.168.13.3 (Request)
11	18.165000	02:00:4c:4f:4f:50	Broadcast	ARP	Gratuitous ARP for 192.168.13.3 (Request)
12	18.179000	02:00:4c:4f:4f:50	Broadcast	ARP	Gratuitous ARP for 192.168.13.3 (Request)

После получения сообщения `DISCOVER` (строка 1), сервер выбирает первый адрес из пула (в данном случае `192.168.13.2`) и отправляет на него ARP-запрос (строка 2)

Так как компьютер с таким адресом существует в сети, сервер получает ответ (строка 3).

Чтобы убедиться в наличии в сети узла с адресом `192.168.13.2` сервер отправляет на этот адрес `Echo-Request` (строка 4) и получает ответ (строка 5).

В таком случае, сервер берет следующий свободный адрес из пула (в данном случае `192.168.13.3`) и отправляет на него ARP-запрос (строка 6)

Не дождавшись ответа (прошло почти 15 сек.) сервер считает адрес свободным и предлагает его клиенту в сообщении `REQUEST` (строка 7).

Клиент, подтвердив получение адреса (строка 8) и дождавшись подтверждения от сервера (строка 9), также проверяет, не занят ли кем-то выданный адрес.

Это делается путем отправки ARP-запросов клиентом (строки 10–12), если ответ на запрос не пришел, клиент назначает себе на интерфейс полученный адрес.

Прочие сообщения DHCP

Помимо сообщений, необходимых для первоначального получения IP-адреса клиентом, DHCP предусматривает несколько дополнительных сообщений для выполнения иных задач.

Отказ DHCP

Если после получения подтверждения (`DHCPACK`) от сервера клиент обнаруживает, что указанный сервером адрес уже используется в сети, он рассылает широковещательное сообщение отказа DHCP (`DHCPDECLINE`), после чего процедура получения IP-адреса повторяется. Использование IP-адреса другим клиентом можно обнаружить, выполнив запрос ARP.

Отмена DHCP

Если по каким-то причинам сервер не может предоставить клиенту запрошенный IP-адрес, или если аренда адреса удаляется администратором, сервер рассылает широковещательное сообщение отмены DHCP (`DHCPNAK`). При получении такого сообщения соответствующий клиент должен повторить процедуру получения адреса.

Освобождение DHCP

Клиент может явным образом прекратить аренду IP-адреса. Для этого он отправляет сообщение освобождения DHCP (`DHCPRELEASE`) тому серверу, который предоставил ему адрес в аренду. В отличие от других сообщений DHCP, `DHCPRELEASE` не рассылается широковещательно.

Информация DHCP

Сообщение информации DHCP (`DHCPINFORM`) предназначено для определения дополнительных параметров TCP/IP (например, адреса маршрутизатора по умолчанию, DNS-серверов и т. п.) теми клиентами, которым не нужен динамический IP-адрес (то есть адрес которых настроен вручную). Серверы отвечают на такой запрос сообщением подтверждения (`DHCPACK`) без выделения IP-адреса.

Настройка сервера DHCP (на FreeBSD)

`dhcpd.conf` состоит из деклараций относительно подсетей и хостов, и проще всего описывается на примере:

```
option domain-name "example.com";                #(1)
option domain-name-servers 192.168.4.100;        #(2)
option subnet-mask 255.255.255.0;                #(3)
default-lease-time 3600;  #(4)
max-lease-time 86400;     #(5)
ddns-update-style none;   #(6)
subnet 192.168.4.0 netmask 255.255.255.0 {
  range 192.168.4.129 192.168.4.254;           #(7)
  option routers 192.168.4.1;                  #(8)
}
host mailhost {
  hardware ethernet 02:03:04:05:06:07;         #(9)
  fixed-address mailhost.example.com;          #(10)
}
```

1. Домен, который будет выдаваться клиентам в качестве домена, используемого по умолчанию при поиске.
2. Список разделённых запятыми серверов DNS, которые должен использовать клиент.
3. Маска сети, которая будет выдаваться клиентам.
4. Клиент может запросить определённое время, которое будет действовать выданная информация. В противном случае сервер выдаст настройки с этим сроком (в секундах).
5. Максимальное время, на которое сервер будет выдавать конфигурацию. Если клиент запросит больший срок, он будет подтверждён, но будет действовать только max-lease-time секунд.
6. Задаёт, будет ли сервер DHCP пытаться обновить DNS при выдаче или освобождении конфигурационной информации. В реализации ISC этот параметр является обязательным.
7. Определение того, какие IP-адреса должны использоваться в качестве резерва для выдачи клиентам (включительно).
8. Объявление маршрутизатора, используемого по умолчанию, который будет выдаваться клиентам.
9. Аппаратный MAC-адрес хоста (чтобы сервер DHCP мог распознать хост, когда тот делает запрос).
10. Определение того, что хосту всегда будет выдаваться один и тот же IP-адрес. Указание здесь имени хоста корректно, так как сервер DHCP будет разрешать имя хоста самостоятельно до того, как выдать конфигурационную информацию.

Настройка IP-адресов в IPv6. Протокол DHCPv6

Методы настройки IP-адресов в IPv6

- *Автоматическая настройка адресов без сохранения состояния* применяется для настройки локальных адресов канала и нелокальных адресов путем обмена сообщениями Router Solicitation (RS) и Router Advertisement (RA) с соседними маршрутизаторами.
- *Автоматическая настройка адресов с сохранением состояния* применяется для настройки нелокальных адресов с помощью протокола настройки, например DHCP.

Настройка адресов без сохранения состояния

Настройку адресов без сохранения состояния узел IPv6 выполняет автоматически. При этом он задействует протокол настройки (DHCPv6), основываясь на флагах из сообщения RA, полученного от соседнего маршрутизатора:

- **Флаг управляемой настройки адресов (флаг M).** Если данный флаг имеет значение 1, это указывает на то, что узел должен получить адреса с сохранением состояния посредством протокола настройки.
- **Флаг других параметров настройки с сохранением состояния (флаг O).** Если данный флаг имеет значение 1, это указывает на то, что узел должен обратиться к услугам протокола настройки для получения других параметров.

Возможные варианты сочетания флагов M и O

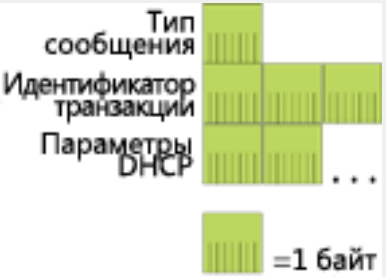
- **M=0 и O=0.** Это сочетание соответствует сети без инфраструктуры DHCPv6. При настройке нелокальных адресов хосты основываются на объявлениях маршрутизатора; остальные параметры настраиваются другими методами (например вручную).
- **M=1 и O=1.** Адреса и другие параметры настраиваются при помощи протокола DHCPv6. Это сочетание называется «DHCPv6 с сохранением состояния»; при этом протокол DHCPv6 назначает узлам IPv6 адреса с сохранением состояния.
- **M=0 и O=1.** Протокол DHCPv6 не участвует в назначении адресов, но применяется для присвоения других параметров настройки. Соседние маршрутизаторы объявляют префиксы нелокальных адресов, на основании которых узлы IPv6 формируют адреса без сохранения состояния. Это сочетание называется «DHCPv6 без сохранения состояния». Протокол DHCPv6 распределяет между узлами IPv6 не адреса с сохранением состояния, а параметры настройки без сохранения состояния.
- **M=1 и O=0.** В этом сочетании протокол DHCPv6 задействуется для настройки адресов, но не участвует в определении других параметров. Так как необходимость в настройке других параметров (например IPv6-адресов DNS-серверов) на узлах IPv6 возникает в большинстве случаев, подобное сочетание встречается редко.

Инфраструктура DHCPv6 аналогична DHCP. Она состоит из клиентов DHCPv6, запрашивающих настройки, серверов DHCPv6, предоставляющих настройки, и агентов ретрансляции DHCPv6, которые выступают посредниками при обмене сообщениями между серверами и клиентами, когда последние располагаются в подсетях без сервера DHCPv6.

Сообщения протокола DHCPv6

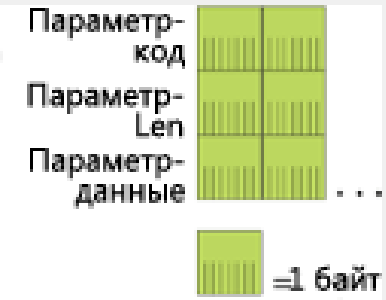
Как и DHCP, протокол DHCPv6 использует сообщения протокола UDP. Клиенты DHCPv6 прослушивают на предмет сообщений DHCP порт 546 протокола UDP. Серверы и агенты ретрансляции DHCPv6 прослушивают в ожидании сообщений DHCPv6 порт 547 протокола UDP. По структуре сообщения DHCPv6 значительно проще своих аналогов протокола DHCP для IPv4, который в целях обеспечения поддержки бездисковых рабочих станций был основан на протоколе BOOTP.

Структура сообщений DHCPv6, которыми обмениваются клиент и сервер



В поле `Msg-Type` размером 1 байт указывается тип сообщения DHCPv6. Поле `Transaction-ID` (3 байта), определяется клиентом и предназначено для группировки сообщений протокола DHCPv6, которыми обмениваются стороны. После поля `Transaction-ID` следуют параметры протокола DHCPv6, содержащие идентификационные данные клиента и сервера, адреса и другие настройки. Перечень стандартных параметров DHCPv6 содержится в RFC 3315.

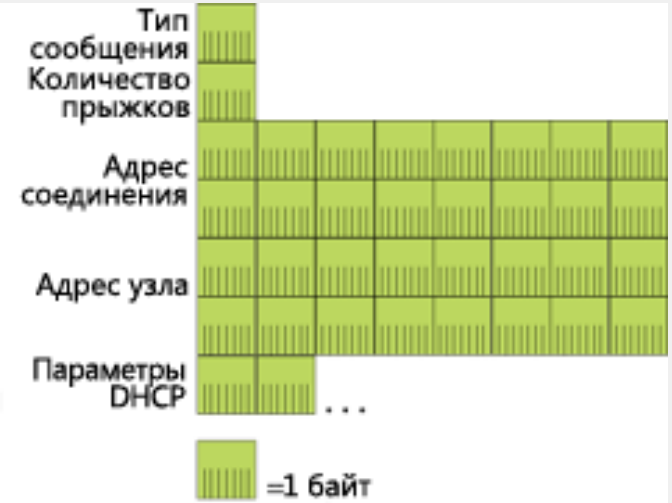
Структура параметров DHCPv6



Параметры протокола DHCPv6 имеют формат TLV (type–length–value — тип–длина–значение).
Поле `Option-Code` (2 байта) предназначено для идентификации параметра. В поле `Option-Len` (2 байта) указывается длина поля `Option-Data` в байтах. Поле `Option-Data` содержит данные параметра.

Сообщения протокола DHCPv6

Сообщения для записи дополнительных сведений



Сообщения, которыми агенты ретрансляции и серверы обмениваются для записи дополнительных сведений.

В поле `Hop-Count` (1 байт) указывается число агентов ретрансляции, получивших данное сообщение. Если число прыжков, указанное в сообщении, превышает максимально допустимую величину, принимающий агент ретрансляции может отказаться от него. В поле `Link-Address` (16 байт) содержится нелокальный адрес, назначаемый интерфейсу, подключенному к подсети клиента. Значение поля `Link-Address` позволяет серверу правильно определить область, из которой следует назначить адрес. Поле `Peer-Address` (16 байт) содержит IPv6-адрес клиента, являющегося исходным отправителем сообщения, или предыдущего агента ретрансляции. За полем `Peer-Address` следуют параметры протокола DHCPv6, в том числе параметр `Relay Message`, содержащий ретранслируемое сообщение, и другие. Параметр `Relay Message` обеспечивает инкапсуляцию сообщений, которыми обмениваются клиент и сервер.

В протоколе IPv6 не предусмотрены широковещательные адреса. По этой причине ограниченный широковещательный адрес, применявшийся при передаче некоторых сообщений протокола DHCPv4, в DHCPv6 заменен адресом `All_DHCP_Relay_Agents_and_Servers` (все агенты ретрансляции и серверы протокола DHCP), имеющим значение `FF02::1:2`. К примеру, клиент DHCPv6, пытающийся определить местонахождение сервера DHCPv6 в сети, должен отправить со своего локального адреса канала по адресу `FF02::1:2` сообщение `Solicit` (обращение). Если в подсети узла есть сервер DHCPv6, он получает это сообщение и отправляет соответствующий ответ. Впрочем, как правило, сообщение `Solicit` (обращение) получает агент ретрансляции протокола DHCPv6, находящийся в подсети узла, после чего оно передается на сервер DHCPv6.

Обмен сообщениями в DHCPv6

Обмен сообщениями с сохранением состояния

Обмен сообщениями протокола DHCPv6 с сохранением состояния, предназначенный для получения IPv6-адресов и параметров настройки (когда флаги M и O в полученном объявлении маршрутизатора имеют значение 1), обычно выглядит следующим образом:

- для обнаружения серверов клиент отправляет сообщение `Solicit` (обращение);
- сервер отправляет сообщение `Advertise` (объявление), подтверждая этим возможность предоставления адресов и параметров настройки;
- клиент запрашивает адреса и параметры настройки у конкретного сервера путем отправки сообщения `Request` (запрос);
- сервер в ответ отправляет сообщение `Reply` (ответ) с запрошенными адресами и параметрами настройки.

Если в качестве посредника между клиентом и сервером выступает агент ретрансляции, он передает серверу сообщения `Relay-Forward` (ретрансляция-пересылка) с инкапсулированными сообщениями `Solicit` (обращение) и `Request` (запрос), поступающими от клиента. Сервер передает агенту ретрансляции сообщения `Relay-Reply` (ретрансляция-ответ) с инкапсулированными сообщениями `Advertise` (объявление) и `Reply` (ответ), предназначенными для клиента.

Обмен сообщениями без сохранения состояния

Обмен сообщениями протокола DHCPv6 без сохранения состояния, предназначенный для получения исключительно параметров настройки (когда флаг M в полученном объявлении маршрутизатора имеет значение 0, а флаг O — значение 1), обычно выглядит следующим образом: клиент DHCPv6 отправляет сообщение Information-Request (запрос данных), запрашивая у сервера параметры настройки, а тот передает сообщение Reply (ответ) с запрошенными параметрами.

В сети IPv6 с маршрутизаторами, настройки которых позволяют назначать узлам IPv6 префиксы адресов без сохранения состояния, подобная схема обмена протокола DHCPv6, состоящая из двух сообщений, позволяет назначать DNS-серверы, доменные имена службы DNS и другие параметры настройки, которые не указываются в объявлении маршрутизатора.

Полный перечень сообщений DHCPv6

Solicit (обращение)

Отравляется клиентом локальным серверам. Аналог DHCPDiscover.

Advertise (объявление)

Отправляется сервером в ответ на сообщение Solicit (обращение) в качестве свидетельства доступности. Аналог DHCPOffer.

Request (запрос)

Отправляется клиентом с целью запросить у конкретного сервера адреса или параметры настройки. Аналог DHCPRequest.

Confirm (подтверждение)

Отправляется клиентом всем серверам с целью проверки соответствия конфигурации клиента требованиям установленного канала связи. Аналог DHCPRequest.

Renew (обновление)

Отравляется клиентом определенному серверу для продления срока аренды выделенных адресов и получения обновленных параметров настройки. Аналог DHCPRequest.

Rebind (повторная привязка)

Отправляется клиентом любому серверу, если ответ на ранее отправленное сообщение Renew (обновление) не получен. Аналог DHCPRequest.

Полный перечень сообщений DHCPv6

Reply (ответ)

Отправляется сервером конкретному клиенту в ответ на сообщения Solicit (обращение), Request (запрос), Renew (обновление), Rebind (повторная привязка), Information-Request (запрос данных), Confirm (подтверждение), Release (освобождение) и Decline (отклонение). Аналог DHCPAck.

Release (освобождение)

Отправляется клиентом в связи с отказом от применения ранее назначенного адреса. Аналог DHCPRelease.

Decline (отклонение)

Отправляется клиентом конкретному серверу в качестве оповещения о том, что выделенный адрес уже используется. Аналог DHCPDecline.

Reconfigure (перенастройка)

Отправляется сервером клиенту в качестве оповещения о том, что на сервере появились новые или обновленные параметры настройки. После этого клиент отправляет одно из двух сообщений: Renew (обновление) или Information-Request (запрос данных).

Information-Request (запрос данных)

Отправляется клиентом для запроса параметров настройки (без адреса). Аналог DHCPInform.

Полный перечень сообщений DHCPv6

Relay-Forward (ретрансляция-пересылка)

Отправляется агентом ретрансляции с целью пересылки клиентского сообщения серверу. Внутри сообщения Relay-Forward (ретрансляция-пересылка) в качестве параметра Relay-Message протокола DHCPv6 размещается инкапсулированное сообщение клиента.

Relay-Reply (ретрансляция-ответ)

Отправляется сервером для передачи клиенту сообщения через агента ретрансляции. Внутри сообщения Relay-Reply (ретрансляция-ответ) в качестве параметра Relay-Message протокола DHCPv6 содержится инкапсулированное сообщение сервера.

Источники

- Материалы с сайта `https://wikipedia.org/`
- Telecommunication technologies — телекоммуникационные технологии / Ю. А. Семенов.
  URL: `http://book.itep.ru/`
- RFC 2131. Dynamic Host Configuration Protocol.
- RFC 3315. Dynamic Host Configuration Protocol for IPv6 (DHCPv6).
- Особенности работы и настройки DHCP на маршрутизаторах Cisco / А. Коваленко. // Хабрахабр.
  URL: `https://habrahabr.ru/post/87920/`