Теория 2

Протоколы электронной почты

Основные характеристики протоколов электронной почты:

- Синтаксис - определяет формат данных и способы их представления.
- Семантика - определяет значения данных и основные операции, выполняемые над ними.
- Согласованность - обеспечивает согласованность и последовательной действий между участниками коммуникации
- Порты - каждый протокол работает на своем порте.
  - POP3 - 110;
  - IMAP - 143;
  - SMTP - 25;
  - HTTP - 80;
  - SSMTP - 465;
  - IMAP4-SSL - 585;
  - IMAPS - 993;
  - SSL-POP3 - 995.

Протокол SMTP

Протокол передачи электронной почты в сетях TCP/IP.
SMTP можно использовать только для отправки электронных писем, а не для их получения.
Впервые был описан в RFC 821.
Последнее обновление в RFC 5321.

Команды:

- MAIL FORM - Указание обратного адреса. Адрес для недоставленных писем;
- RCPT TO - Указание адреса получателя. Может быть использована многократно для массовой рассылки;
- DATA - Содержание письма. Состоит из заголовка и тела сообщения, разделённых пустой строкой.
  Сервер отвечает два раза на эту команду. Первый раз о готовности принимать текст и второй о решении принять или отклонить всё письмо;

Ответы:

Каждый ответ содержит свой код (на месте 'x' могут быть другие цифры).

- 2xx - Положительный (команда принята);
- 3xx - Ожидаются дополнительные данные от клиента;
- 4xx - Временный отказ;
- 5xx - Отказ выполнять команду.

Протокол POP3

Протокол передачи электронной почты.
Обеспечивает доступ к почтовому ящику и загрузки сообщений на ПК.
При использовании POP3 все сообщения электронной почты будут загружены с почтового сервера на локальную машину. Это хорошо, потому что можно взаимодействовать с письмами без подключения к сети, после загрузки, но плохо тем, что загружаются ВСЕ письма, что может быть просто объёмным или иметь нежелательное содержание.
Описан в RFC 1939.

Сеанс может быть в 3 состоянях:

1. Авторизация - клиент проходит процедуру аутентификации.
2. Транзакция - Клиент получает информацию о состоянии почтового ящика, принимает и удаляет почту.
3. Обновление - Сервер удаляет выбранные письма и закрывает соединение.

Команды:

Имя	Аргументы	Описание	Ограничения	Возможные ответы
APOP	[имя] [digest]	Команда служит для передачи серверу имени пользователя и хэша пароля (digest)	Её поддержка не является обязательной	+OK maildrop has n message -ERR password supplied for [имя] is incorrect
USER	[имя]	Передаёт серверу имя пользователя	Нет ограничений	+OK name is a valid mailbox -ERR never heard of mailbox name
PASS	[пароль]	Передаёт серверу пароль почтового ящика	Работает после успешной передачи имени почтового ящика	+OK maildrop locked and ready -ERR invalid password -ERR unable

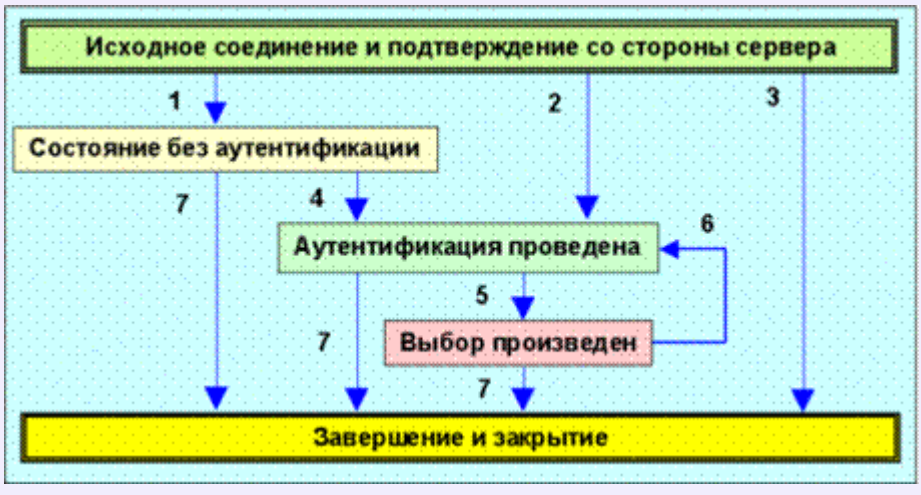
Имя	Аргументы	Описание	Ограничения	Возможные ответы
				to lock maildrop
DELE	[сообщение]	Сервер помечает указанное сообщение для удаления. Сообщения, помеченные на удаление, реально удаляются только после закрытия транзакции (закрытие транзакций происходит обычно после посыла команды QUIT, кроме этого, например, на серверах закрытие транзакций может происходить по истечении определённого времени, установленного сервером)	Доступна после успешной аутентификации	+OK message deleted -ERR no such message
LIST	[сообщение]	Если был передан аргумент, то сервер выдаёт информацию об указанном сообщении. Если аргумент не был передан, то сервер выдаёт информацию обо всех сообщениях, находящихся в почтовом ящике. Сообщения, помеченные для удаления, не перечисляются	Доступна после успешной аутентификации	+OK scan listing follows -ERR no such message
NOOP	Нет аргументов	Сервер ничего не делает, всегда отвечает положительно	Доступна после успешной аутентификации	+OK
RETR	[сообщение]	Сервер передаёт сообщение с указанным номером	Доступна после успешной аутентификации	+OK message follows -ERR no such message
RSET	Нет аргументов	Этой командой производится откат транзакций внутри сессии. Например, если пользователь случайно пометил на удаление какие-либо сообщения, он может убрать эти	Доступна после успешной аутентификации	+OK

Имя	Аргументы	Описание	Ограничения	Возможные ответы
		пометки, отправив эту команду		
STAT	Нет аргументов	Сервер возвращает количество сообщений в почтовом ящике и размер почтового ящика в октетах. Сообщения, помеченные как удалённые, при этом не учитываются	Доступна после успешной аутентификации	+OK a b
TOP	[сообщение] [количество строк]	Сервер возвращает заголовки указанного сообщения, пустую строку и указанное количество первых строк тела сообщения	Доступна после успешной аутентификации	+OK n octets -ERR no such message
QUIT	Нет аргументов	Закрытие соединения	Нет ограничений	+OK

Протокол IMAP

Протокол доступа к электронной почте.
Обеспечивает работу с письмами без необходимости скачивать каждое письмо.
Описан в RFC 3501

Состояния сеанса:



1. Соединение без предварительной аутентификации.
2. Соединение с предварительной аутентификации.
3. Соединение отвергнуто.
4. Условное завершение команды `LOGIN` или `AUTHENTICATE`.
5. Успешное завершение команды `SELECT` или `EXAMINE`.
6. Выполнение команды `CLOSE` или неудачная команда `SELECT` или `EXAMINE`.

7. Выполнение команды `LOGOUT`, закрытие сервера или прерывание соединения.

Флаги:

- \seen — сообщение прочитано
- \answered — на сообщение отправлен ответ
- \flagged — сообщение отмечено как «важное»
- \deleted — сообщение отмечено как удалённое
- \draft — сообщение отмечено как черновик
- \recent — недавнее сообщение (впервые появилось в ящике в ходе текущей сессии)

Команды:

- **LOGIN** - Позволяет использовать при регистрации идентификатор пользователя и пароль в текстовом виде.
- **AUTHENTICATE** - Позволяет клиенту использовать при регистрации альтернативные методы проверки подлинности. Индивидуальная проверка подлинности пользователей не является обязательной и поддерживается не всеми серверами IMAP. Реализации такой проверки могут различаться в зависимости от сервера.
- **CLOSE** - Закрывает почтовый ящик. Когда почтовый ящик закрыт с помощью этой команды, то сообщения, помеченные флагом `\DELETED`, удаляются из него. Не имеет параметров.
- **LOGOUT** - Завершает сеанс для текущего идентификатора пользователя.
- **CREATE** - Создаёт новый почтовый ящик. Имя и местоположение новых почтовых ящиков определяются в соответствии с общими спецификациями сервера.
- **DELETE** - Применяется к почтовым ящикам. Сервер IMAP при получении этой команды попытается удалить почтовый ящик с именем, указанным в качестве аргумента команды. Сообщения удаляются вместе с ящиками и восстановлению не подлежат.
- **RENAME** - Изменяет имя почтового ящика. Эта команда имеет два параметра — имя почтового ящика, который требуется переименовать, и новое имя почтового ящика.
- **SUBSCRIBE** - Добавляет почтовый ящик в список активных ящиков клиента. В этой команде используется только один параметр — имя почтового ящика, который нужно внести в список. Почтовый ящик не обязательно должен существовать, чтобы его можно было добавить в список активных ящиков — это позволяет добавлять в список активных ящиков ящики, которые ещё не созданы, или удалять их, если они пусты.
- **UNSUBSCRIBE** - Удаляет почтовые ящики из списка активных. В ней так же используется один параметр — имя почтового ящика, который удаляется из списка

активных ящиков клиента. При этом сам по себе почтовый ящик не удаляется.

- **LIST** - Получить список всех почтовых ящиков клиента.
- **LSUB** - В отличие от команды `LIST` используется для получения списка ящиков, активизированных командой `SUBSCRIBE`.
- **STATUS** - Формирует запрос о текущем состоянии почтового ящика. Первым параметром для этой команды является имя почтового ящика, к которому она применяется. Второй параметр — это список критериев, по которым клиент хочет получить информацию. Команда `STATUS` может использоваться для получения информации о состоянии почтового ящика без его открытия с помощью команд `SELECT` или `EXAMINE`.
  Пользователь может получить информацию по критериям:
  - **MESSAGES** — общее число сообщений в почтовом ящике
  - **RECENT** — число сообщений с флагом `\recent`
  - **UIDNEXT** — идентификатор UID, который будет назначен новому сообщению
  - **UIDVALIDITY** — уникальный идентификатор почтового ящика
  - **UNSEEN** — число сообщений без флага `\seen`
- **APPEND** - Добавляет сообщение в конец указанного почтового ящика. В качестве аргументов указываются имя ящика, флаги сообщения (не обязательно), метка времени (не обязательно) и само сообщение — заголовок и тело.
  Если в команде указаны флаги, то они устанавливаются для добавляемого сообщения. В любом случае для сообщения устанавливается флаг `\Recent`.
  Если в команде задана метка времени, то это время будет установлено в качестве времени создания сообщения, в противном случае за время создания принимается текущее время.
  Поскольку сообщение состоит не из одной строки, используются литералы (параметр - константа, в данном случае количество символов в команде)
  Пример:

```
C       A003 APPEND saved-messages (\Seen) {247}
S       + Ready for literal data
C       Date: Mon, 7 Feb 1994 21:52:25 -0800 (PST)
C       From: Fred Foobar <foobar@Blurdybloop.COM>
C       Subject: afternoon meeting
C       To: mooch@owatagu.siam.edu
C       Message-Id: <B27397-0100000@Blurdybloop.COM>
C
C       Hello Joe, do you think we can meet at 3:30 tomorrow?
S       A003 OK APPEND completed
```

Расширение `MULTIAPPEND`, описанное в RFC 3502, позволяет одной командой добавлять в почтовый ящик несколько сообщений.

- **CHECK** - станавливает контрольную точку в почтовом ящике. Любые операции, такие, например, как запись данных из памяти сервера на его жёсткий диск, должны выполняться при соответствующем состоянии почтового ящика. Именно

для проверки целостности почтового ящика после дисковых и других подобных им операций и применяется команда `CHECK`. Эта команда используется без параметров.

- **EXPUNGE** - Удаляет из почтового ящика все сообщения, помеченные флагом `\DELETED`, при этом почтовый ящик не закрывается. Ответ сервера на команду `EXPUNGE` представляет собой отчёт о новом состоянии почтового ящика.
- **SEARCH** - Поиск сообщений по критериям в активном почтовом ящике с последующим отображением результатов в виде номера сообщения. Возможен поиск сообщений, в теле которых имеется определённая текстовая строка, или имеющих определённый флаг, или полученных до определённой даты и т. д.
- **FETCH** - Получить текст почтового сообщения. Команда применяется только для отображения сообщений. В отличие от POP3, клиент IMAP не сохраняет копию сообщения на клиентском ПК.
- **STORE** - Изменяет информацию о сообщении.
- **COPY** - Копирует сообщения из одного почтового ящика в другой.
- **UID** - Используется в связке с командами `FETCH`, `COPY`, `STORE` или `SEARCH`. С её помощью в этих командах можно использовать реальные идентификационные номера UID вместо последовательности чисел из диапазона номеров сообщений.
- **CAPABILITY** - Запрос у сервера IMAP информации о его возможностях.
- **NOOP** - Команда ничего не делает. Ответ сервера на команду `NOOP` всегда должен быть положительным.

DNS

DNS - компьютерная распределенная система для получения информации о доменах. Чаще всего используется для получения IP-адреса по имени хоста (компьютера или устройства), получения информации о маршрутизации почты и/или обслуживающих узлах для протоколов в домене.

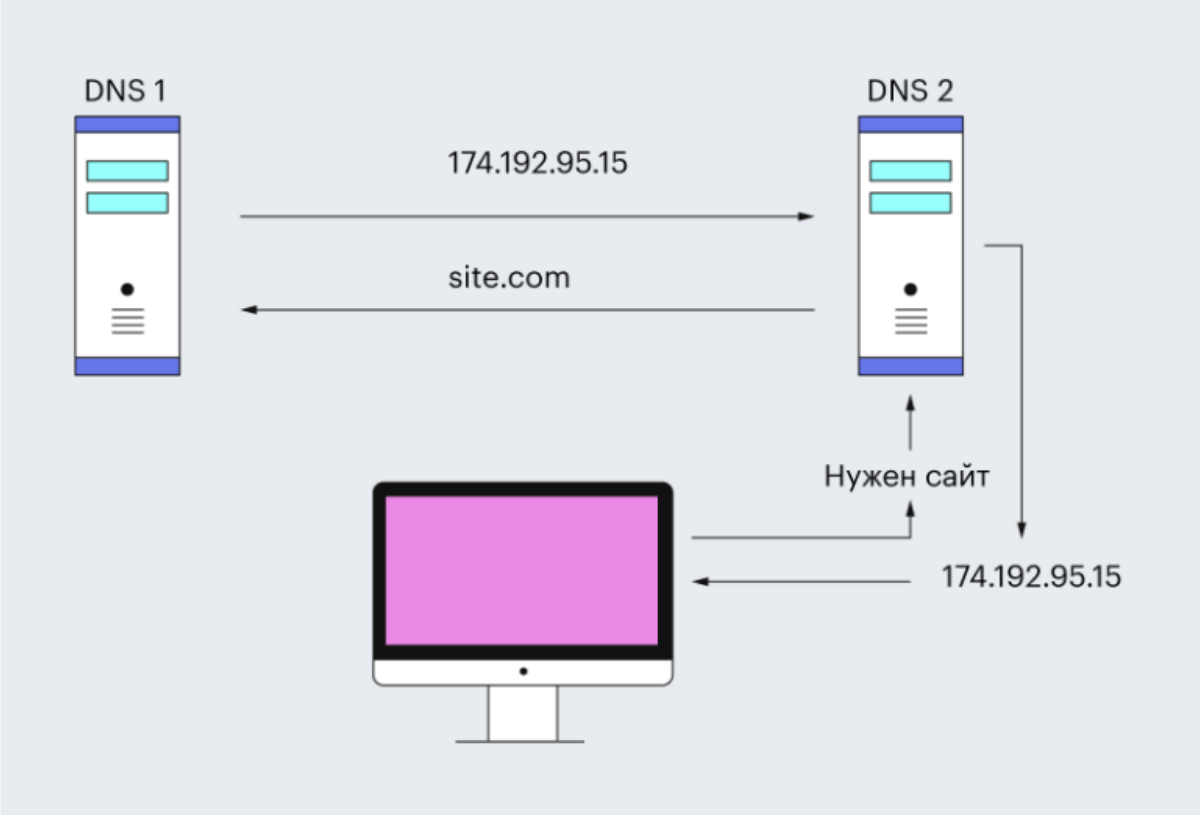
ⓘ Небольшие сноски:

- Домен - узел в дереве имён, вместе со всеми подчинёнными ему узлами (если таковые имеются), то есть именованная ветвь или поддерево в дереве имён. 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): вверху находится корневой домен (имеющий идентификатор «`.`»(точка)), ниже идут домены первого уровня (доменные зоны), затем — домены второго уровня, третьего и т. д. (например, для адреса `ru.wikipedia.org.` домен первого уровня — `org`, второго — `wikipedia`, третьего — `ru`). DNS позволяет не указывать точку корневого домена. Домены нужны, так как их проще запомнить, чем набор цифр.

Основой DNS является представление об иерархической структуре имени и зонах. Каждый сервер, отвечающий за имя, может передать 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.

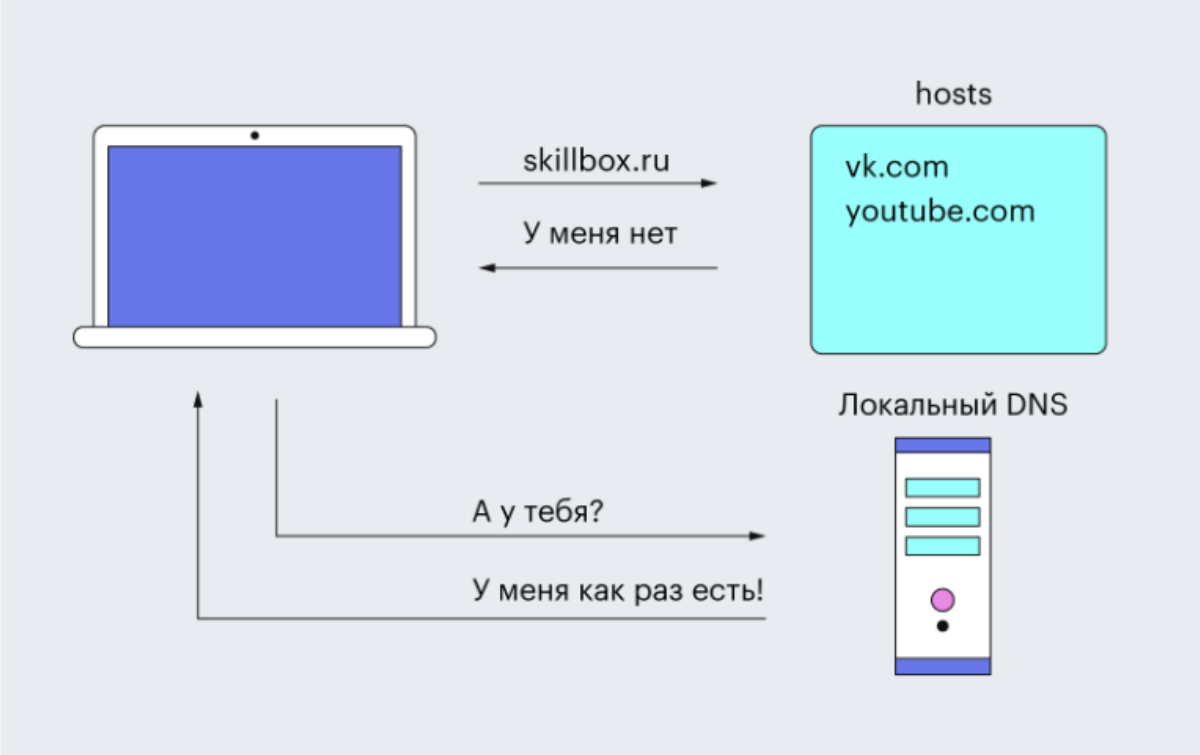
Ключевые характеристики DNS:

- 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.
- Кэширование 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.

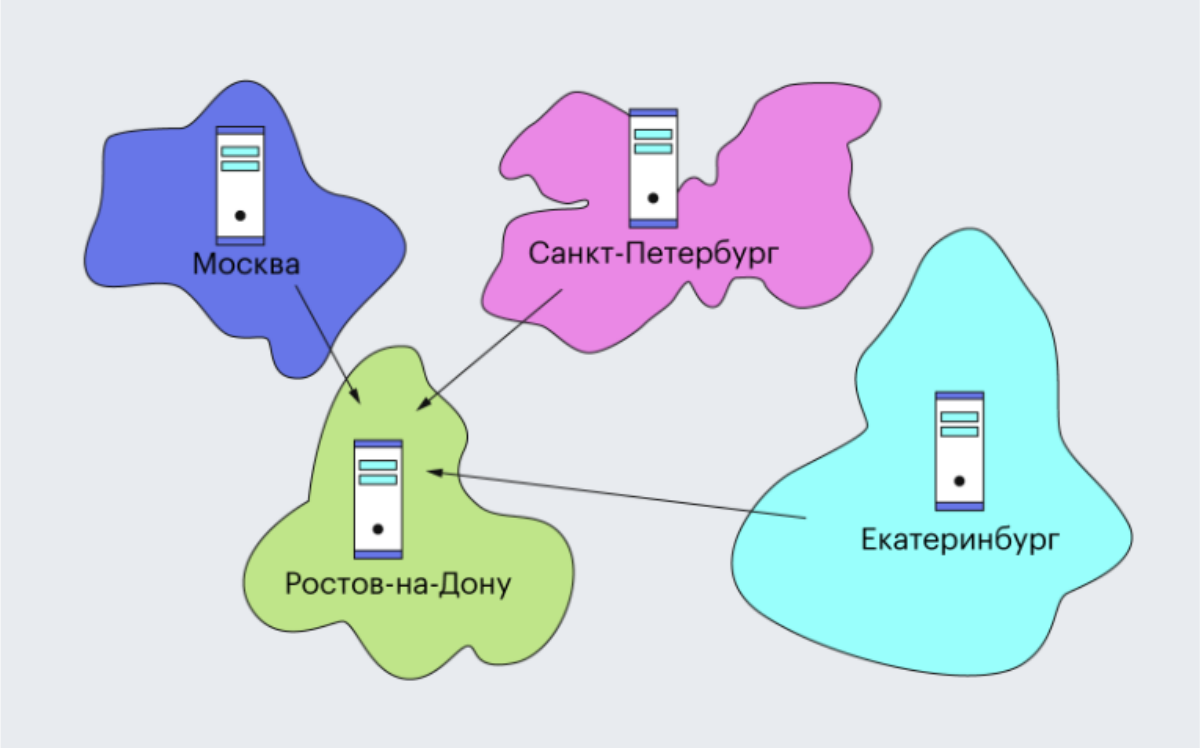


Когда пользователь вводит имя сайта в адресную строку браузера, компьютер сначала обращается к локальному файлу настроек DNS — файлу hosts. В нём содержатся IP-адреса всех сайтов, на которые пользователь заходил с этого устройства. Если нужного адреса там

нет, компьютер направляет запрос на локальный сервер интернет-провайдера пользователя.



На локальном сервере происходит взаимодействие между другими серверами из региона, в котором находится запрошенный сайт. Локальный сервер несколько раз обращается к разным региональным и наконец получает информацию о сайте, а затем отправляет его обратно пользователю.



Чтобы локальный сервер облегчил себе жизнь в будущем и меньше обращался к другим, при получении нужного IP-адреса он сохраняет его у себя в памяти. Это и называется кэшированием. Со временем кэш очищается.

Главные DNS - серверы

Всего существует 13 главных DNS-серверов. Они называются корневыми, и в них содержится информация обо всех сайтах интернета. Чтобы обезопасить Сеть и не потерять все данные разом из-за катаклизмов или чего-то ещё, у этих изначальных корневых есть полные копии, которые размещены в разных странах мира. Вместе с копиями количество корневых выросло до 123.

- 40 - в Северной Америке;
- 35 - в Европе;
- 6 - в Южной Америке;
- 3 - в Африке;
- 39 - в остальных странах.
  Такое расположение связано с интенсивностью обращений к ним по регионам.
  В России находится пять копий корневых — в Москве, Санкт-Петербурге, Ростове-на-Дону, Новосибирске и Екатеринбурге.

Dns как протокол

DNS принадлежит к семейству интернет-протоколов и модели OSI (а именно прикладному уровню). Он по умолчанию содержится во всех операционных системах, чтобы компьютер мог общаться с другими машинами по сети.

Записи DNS

В DNS есть специальные файлы, которые хранят связи между доменами и их IP-адресами. Также они хранят там информацию о поддоменах и почтовых серверах, если те есть у сайта. И всё это называется DNS-зоной.
Зоны содержат разные виды записей и помогают серверам понять, что за домен перед ними. Основные типы записей:

- **A** — IP-адрес обычного сайта;
- **MX** — адрес почтового сервера;
- **CNAME** — запись для поддоменов, которая указывает, к каким адресам прикреплён основной домен;
- **NS** — адрес сервера, где находятся все ресурсные записи;
- **TXT** — текстовая информация, которая относится к конкретному домену;
- **SPF** — список серверов, которые могут отправлять сообщения от имени указанного домена;
- **SOA** — главная запись зоны, где указаны все сведения о сервере.

Каждая запись состоит из следующих полей:

- имя (NAME) — доменное имя, к которому привязана или которому «принадлежит» данная ресурсная запись,

- тип (TYPE) ресурсной записи — определяет формат и назначение данной ресурсной записи,
- класс (CLASS) 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,
- TTL (Time To Live) — допустимое время хранения данной ресурсной записи в кэше неответственного DNS-сервера,
- длина поля данных (RDLEN),
- поле данных (RDATA), формат и содержание которого зависит от типа записи.

Небольшие сноски:

- Поддомен - подчинённый домен (например, `wikipedia.org` - поддомен домена `org`, а `ru.wikipedia.org` - домена `wikipedia.org`).

Запросы DNS

Помимо рекурсивного и нерекурсивного запросов есть также рекрусивные (умеющие производить рекурсивный поиск) и неркурсивные (не умеющие производить рекурсивный поиск) сервера.

Рекурсивный

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).
Рассмотрим на примере работу всей системы.
Предположим, мы набрали в браузере адрес `ru.wikipedia.org`. Браузер ищет соответствие этого адреса IP-адресу в файле hosts. Если файл не содержит соответствия, то далее браузер спрашивает у сервера DNS: «какой IP-адрес у `ru.wikipedia.org`»? Однако сервер DNS может ничего не знать не только о запрошенном имени, но и даже обо всём домене `wikipedia.org`. В этом случае сервер обращается к корневому серверу - например, 198.41.0.4. Этот сервер сообщает - «У меня нет информации о данном адресе, но я знаю, что 204.74.112.1 является ответственным за зону `org`.» Тогда сервер DNS направляет свой запрос к 204.74.112.1, но тот отвечает «У меня нет информации о данном сервере, но я знаю, что 207.142.131.234 является ответственным за зону `wikipedia.org`.» Наконец, тот же запрос отправляется к третьему DNS-серверу и получает ответ — IP-адрес, который и передаётся клиенту — браузеру.

В данном случае при разрешении имени, то есть в процессе поиска IP по имени:

- браузер отправил известному ему DNS-серверу *рекурсивный* запрос — в ответ на такой тип запроса сервер обязан вернуть «готовый результат», то есть IP-адрес, либо пустой ответ и код ошибки NXDOMAIN;
- DNS-сервер, получивший запрос от браузера, последовательно отправлял нерекурсивные запросы, на которые получал от других DNS-серверов ответы, пока не получил ответ от сервера, ответственного за запрошенную зону;
- остальные упоминавшиеся DNS-серверы обрабатывали запросы нерекурсивно (и, скорее всего, не стали бы обрабатывать запросы рекурсивно, даже если бы такое требование стояло в запросе).

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 идёт дальше кэша сервера, обращения к другим серверам не происходит вообще.

✓ Итеративный

При ответе на нерекурсивный запрос, а также при неумении или запрете выполнять рекурсивные запросы, DNS-сервер либо возвращает данные о зоне, за которую он ответственен, либо возвращает ошибку.

✓ Обратный

Запрос не адреса по домену, а наоборот, домена по адресу.

? Применение MX DNS запроса

При отправке электронной почты, сервер-отправитель запрашивает у DNS-сервера MX-запись домена получателя электронного сообщения. В результате запроса возвращается список имён хостов почтовых серверов, принимающих входящую почту для данного домена, и номеров предпочтения для каждого из них

? Round robin DNS

Round robin DNS - один из методов распределения нагрузки, или отказоустойчивости за счёт избыточности количества серверов, с помощью управления ответами DNS-сервера в соответствии с некой статистической моделью.

В простейшем случае Round robin DNS работает, отвечая на запросы не только одним IP-адресом] а списком из нескольких адресов серверов, предоставляющих идентичный сервис. Порядок, в котором возвращаются IP-адреса из списка, основан на алгоритме round-robin. С каждым ответом последовательность ip-адресов меняется. Как правило, простые клиенты пытаются устанавливать соединения с первым адресом из списка, таким образом разным клиентам будут выданы адреса разных серверов, что распределит общую нагрузку между серверами.

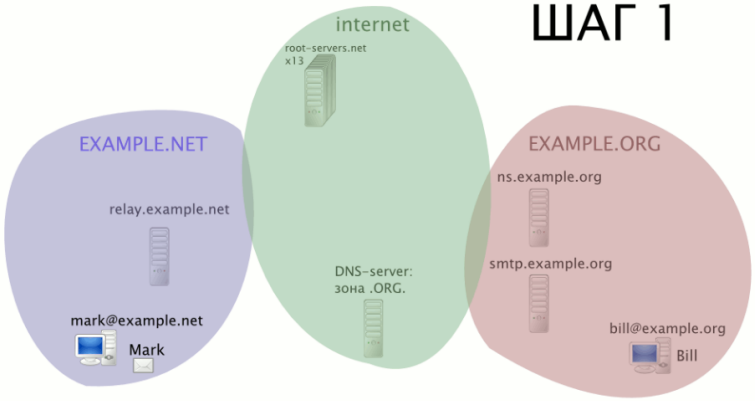
> **Алгоритм Round robin**
>
> Пусть имеется *N* объектов, способных выполнить заданное действие, и *M* задач, которые должны быть выполнены этими объектами. Подразумевается, что объекты *n* равны по своим свойствам между собой, задачи *m* имеют равный приоритет. Тогда первая задача (*m* = 1) назначается для выполнения первому объекту (*n* = 1), вторая — второму и т. д., до достижения последнего объекта (*m* = *N*). Тогда следующая задача (*m* = *N*+1) будет назначена снова первому объекту и т. п. Проще говоря, происходит перебор выполняющих задания объектов по циклу, или по кругу (round), и по достижении последнего объекта следующая задача будет также назначена первому объекту.

Взаимодействие почтовых серверов

1. Письмо пишется.
2. Письмо отправляется на почтовый сервер автора (ПС 1).
3. ПС 1 ищет данные о dns-zone указанного домена первого уровня
4. ПС 1 ищет данные о зоне домена второго уровня.
5. ПС 1 узнает IP адреса почтового сервера получателя (ПС 2).
6. ПС 1 соединяется с ПС 2 и передает письмо.
7. ПС 2 удостоверяется, что письмо для локального пользователя и помещает его в почтовый ящик.
8. Получатель обращается к ПС 2.
9. Получатель получил письмо.

Простейшая схема отправки почты

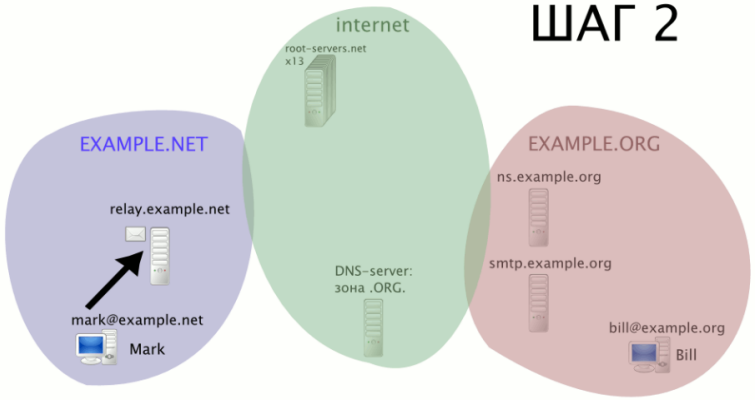
ШАГ 1



1. Марк решает отправить почту на bill@example.org, он пишет его в почтовой программе
2. Почтовая программа пересылает письмо на почтовый сервер Марка (relay.example.net)
3. Сервер relay.example.net ищет данные о DNS-зоне org
4. relay.example.net ищет данные о зоне example.org
5. Он узнаёт у ns.example.org, что почту надо слать на smtp.example.org и узнаёт его IP-адрес
6. Сервер relay.example.net соединяется с сервером smtp.example.org и передаёт письмо
7. smtp.example.org видит, что письмо для локального пользователя и помещает его в почтовый ящик
8. Билл приходит, включает компьютер, запускает почтовую программу
9. Почтовая программа обращается к серверу smtp.example.org
10. Программа находит письмо я ящике, скачивает его – письмо доставлено Биллу

Простейшая схема отправки почты

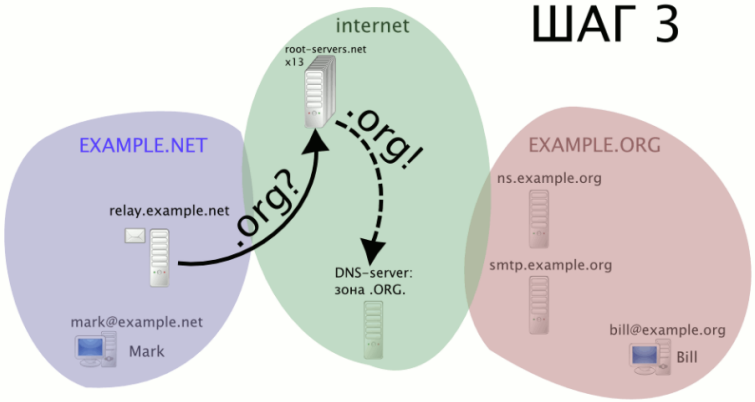
ШАГ 2



1. Марк решает отправить почту на bill@example.org, он пишет его в почтовой программе
2. Почтовая программа пересылает письмо на почтовый сервер Марка (relay.example.net)
3. Сервер relay.example.net ищет данные о DNS-зоне org
4. relay.example.net ищет данные о зоне example.org
5. Он узнаёт у ns.example.org, что почту надо слать на smtp.example.org и узнаёт его IP-адрес
6. Сервер relay.example.net соединяется с сервером smtp.example.org и передаёт письмо
7. smtp.example.org видит, что письмо для локального пользователя и помещает его в почтовый ящик
8. Билл приходит, включает компьютер, запускает почтовую программу
9. Почтовая программа обращается к серверу smtp.example.org
10. Программа находит письмо я ящике, скачивает его – письмо доставлено Биллу

Простейшая схема отправки почты

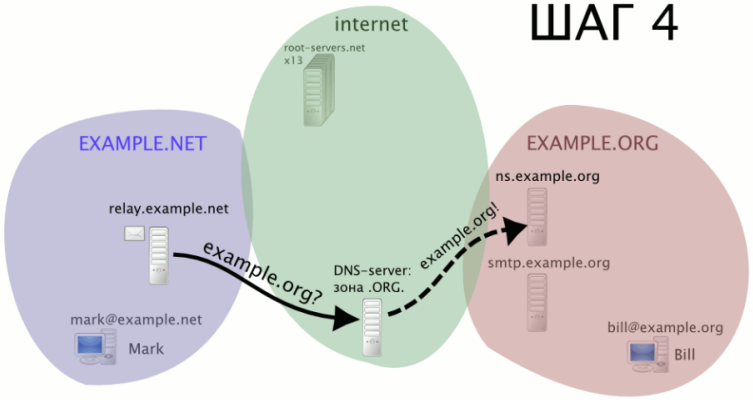
ШАГ 3



1. Марк решает отправить почту на bill@example.org, он пишет его в почтовой программе
2. Почтовая программа пересылает письмо на почтовый сервер Марка (relay.example.net)
3. Сервер relay.example.net ищет данные о DNS-зоне org
4. relay.example.net ищет данные о зоне example.org
5. Он узнаёт у ns.example.org, что почту надо слать на smtp.example.org и узнаёт его IP-адрес
6. Сервер relay.example.net соединяется с сервером smtp.example.org и передаёт письмо
7. smtp.example.org видит, что письмо для локального пользователя и помещает его в почтовый ящик
8. Билл приходит, включает компьютер, запускает почтовую программу
9. Почтовая программа обращается к серверу smtp.example.org
10. Программа находит письмо я ящике, скачивает его – письмо доставлено Биллу

Простейшая схема отправки почты

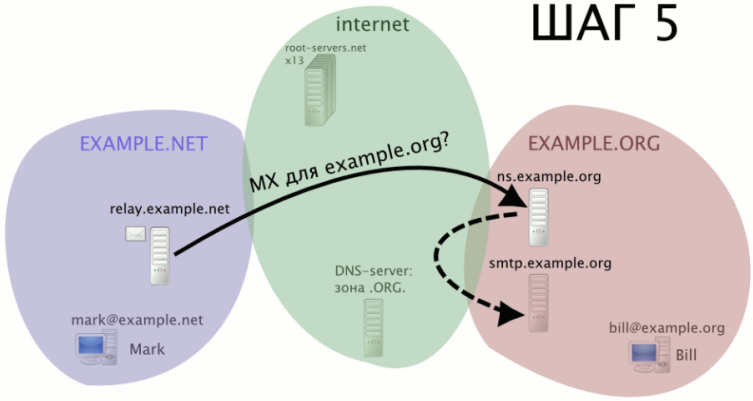
ШАГ 4



1. Марк решает отправить почту на bill@example.org, он пишет его в почтовой программе
2. Почтовая программа пересылает письмо на почтовый сервер Марка (relay.example.net)
3. Сервер relay.example.net ищет данные о DNS-зоне org
4. relay.example.net ищет данные о зоне example.org
5. Он узнаёт у ns.example.org, что почту надо слать на smtp.example.org и узнаёт его IP-адрес
6. Сервер relay.example.net соединяется с сервером smtp.example.org и передаёт письмо
7. smtp.example.org видит, что письмо для локального пользователя и помещает его в почтовый ящик
8. Билл приходит, включает компьютер, запускает почтовую программу
9. Почтовая программа обращается к серверу smtp.example.org
10. Программа находит письмо я ящике, скачивает его – письмо доставлено Биллу

Простейшая схема отправки почты

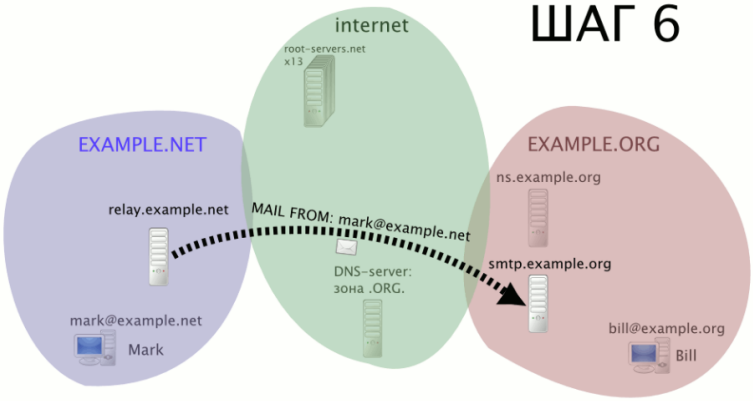
ШАГ 5



1. Марк решает отправить почту на bill@example.org, он пишет его в почтовой программе
2. Почтовая программа пересылает письмо на почтовый сервер Марка (relay.example.net)
3. Сервер relay.example.net ищет данные о DNS-зоне org
4. relay.example.net ищет данные о зоне example.org
5. Он узнаёт у ns.example.org, что почту надо слать на smtp.example.org и узнаёт его IP-адрес
6. Сервер relay.example.net соединяется с сервером smtp.example.org и передаёт письмо
7. smtp.example.org видит, что письмо для локального пользователя и помещает его в почтовый ящик
8. Билл приходит, включает компьютер, запускает почтовую программу
9. Почтовая программа обращается к серверу smtp.example.org
10. Программа находит письмо я ящике, скачивает его – письмо доставлено Биллу

Простейшая схема отправки почты

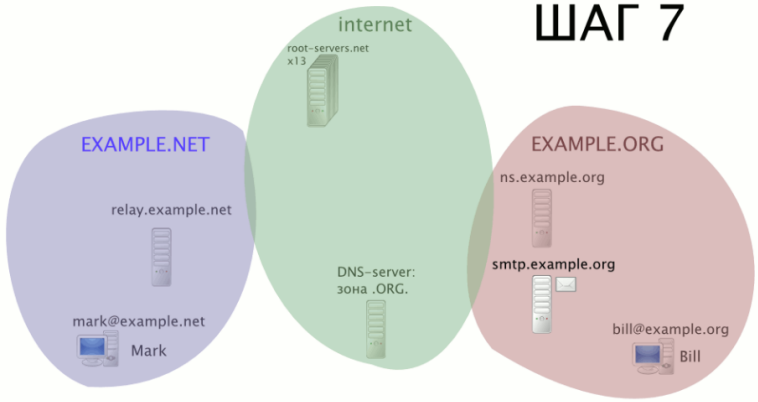
ШАГ 6



1. Марк решает отправить почту на bill@example.org, он пишет его в почтовой программе
2. Почтовая программа пересылает письмо на почтовый сервер Марка (relay.example.net)
3. Сервер relay.example.net ищет данные о DNS-зоне org
4. relay.example.net ищет данные о зоне example.org
5. Он узнаёт у ns.example.org, что почту надо слать на smtp.example.org и узнаёт его IP-адрес
6. Сервер relay.example.net соединяется с сервером smtp.example.org и передаёт письмо
7. smtp.example.org видит, что письмо для локального пользователя и помещает его в почтовый ящик
8. Билл приходит, включает компьютер, запускает почтовую программу
9. Почтовая программа обращается к серверу smtp.example.org
10. Программа находит письмо я ящике, скачивает его – письмо доставлено Биллу

Простейшая схема отправки почты

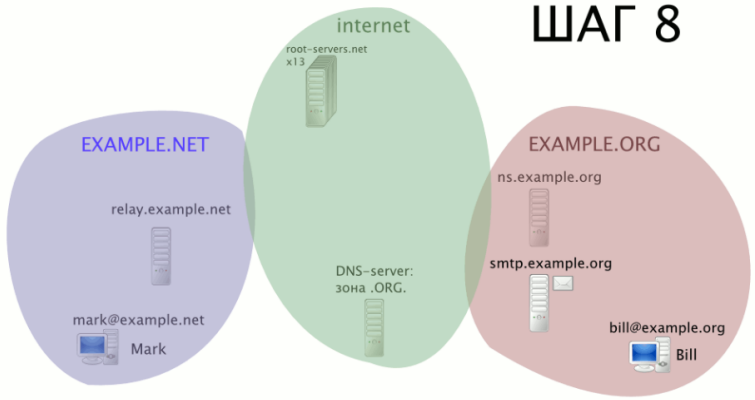
ШАГ 7



1. Марк решает отправить почту на bill@example.org, он пишет его в почтовой программе
2. Почтовая программа пересылает письмо на почтовый сервер Марка (relay.example.net)
3. Сервер relay.example.net ищет данные о DNS-зоне org
4. relay.example.net ищет данные о зоне example.org
5. Он узнаёт у ns.example.org, что почту надо слать на smtp.example.org и узнаёт его IP-адрес
6. Сервер relay.example.net соединяется с сервером smtp.example.org и передаёт письмо
7. **smtp.example.org видит, что письмо для локального пользователя и помещает его в почтовый ящик**
8. Билл приходит, включает компьютер, запускает почтовую программу
9. Почтовая программа обращается к серверу smtp.example.org
10. Программа находит письмо я ящике, скачивает его – письмо доставлено Биллу

Простейшая схема отправки почты

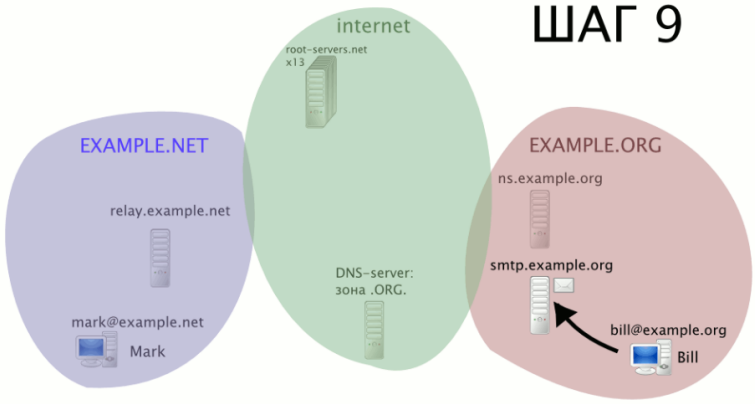
ШАГ 8



1. Марк решает отправить почту на bill@example.org, он пишет его в почтовой программе
2. Почтовая программа пересылает письмо на почтовый сервер Марка (relay.example.net)
3. Сервер relay.example.net ищет данные о DNS-зоне org
4. relay.example.net ищет данные о зоне example.org
5. Он узнаёт у ns.example.org, что почту надо слать на smtp.example.org и узнаёт его IP-адрес
6. Сервер relay.example.net соединяется с сервером smtp.example.org и передаёт письмо
7. smtp.example.org видит, что письмо для локального пользователя и помещает его в почтовый ящик
8. **Билл приходит, включает компьютер, запускает почтовую программу**
9. Почтовая программа обращается к серверу smtp.example.org
10. Программа находит письмо я ящике, скачивает его – письмо доставлено Биллу

Простейшая схема отправки почты

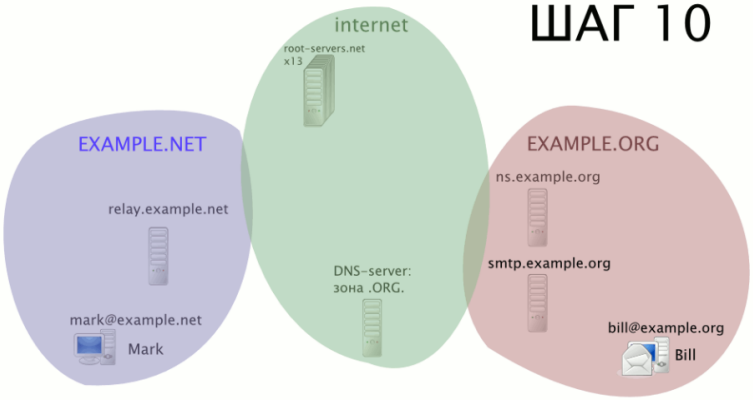
ШАГ 9



1. Марк решает отправить почту на bill@example.org, он пишет его в почтовой программе
2. Почтовая программа пересылает письмо на почтовый сервер Марка (relay.example.net)
3. Сервер relay.example.net ищет данные о DNS-зоне org
4. relay.example.net ищет данные о зоне example.org
5. Он узнаёт у ns.example.org, что почту надо слать на smtp.example.org и узнаёт его IP-адрес
6. Сервер relay.example.net соединяется с сервером smtp.example.org и передаёт письмо
7. smtp.example.org видит, что письмо для локального пользователя и помещает его в почтовый ящик
8. Билл приходит, включает компьютер, запускает почтовую программу
9. Почтовая программа обращается к серверу smtp.example.org
10. Программа находит письмо я ящике, скачивает его – письмо доставлено Биллу

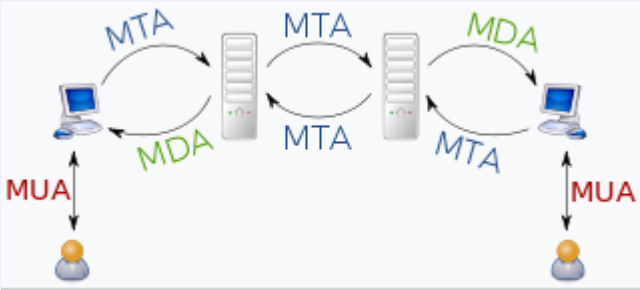
Простейшая схема отправки почты

ШАГ 10



1. Марк решает отправить почту на bill@example.org, он пишет его в почтовой программе
2. Почтовая программа пересылает письмо на почтовый сервер Марка (relay.example.net)
3. Сервер relay.example.net ищет данные о DNS-зоне org
4. relay.example.net ищет данные о зоне example.org
5. Он узнаёт у ns.example.org, что почту надо слать на smtp.example.org и узнаёт его IP-адрес
6. Сервер relay.example.net соединяется с сервером smtp.example.org и передаёт письмо
7. smtp.example.org видит, что письмо для локального пользователя и помещает его в почтовый ящик
8. Билл приходит, включает компьютер, запускает почтовую программу
9. Почтовая программа обращается к серверу smtp.example.org
10. Программа находит письмо я ящике, скачивает его – письмо доставлено Биллу

Схема взаимоотношения почтовых агентов и серверов



MTA - агент пересылки почты.
MDA - агент доставки почты.
MUA - почтовый агент пользователя (почтовый клиент).
MRA - почтовый сервер.

MIME

MIME - формат передачи данных, содержание которых выходит за рамки ASCII.
Формат MIME поддерживает передачу нескольких сущностей в пределах одного сообщения. Причём сущности могут передаваться не только в виде одноуровневой последовательности, но и в виде иерархии с вложением элементов друг в друга. Для обозначения множественного содержимого используются медиа-типы `multipart/*`. Работа с такими типами осуществляется по общим правилам, описанным в RFC 2046) (если иное не определено конкретным медиа-типом). Если получателю неизвестно как работать с типом, то он обрабатывает его так же, как `multipart/mixed`.

Для передачи множественного сообщения в заголовок `Content-Type` добавляется параметр `boundary` (граница), который обозначает последовательность символов, разделяющих части сообщения. Граница может состоять из цифр, букв и символов `'()+_,-./:=?`. При использовании специальных символов (не цифр и букв) значение параметра `boundary` следует заключать в двойные кавычки `"`. Максимальная длина границы — 70 символов.

Начало каждой части сообщения обозначается строкой `--boundary`. Конец последнего сообщения обозначается строкой `--boundary--`. Самые первые символы переноса строки CRLF (коды 13 и 10), которыми начинаются и заканчиваются пограничные строки, не входят в содержимое самой части. Если за ними следуют ещё переносы строк, то они уже принадлежат включаемой части.

Перед первой частью и после последней может быть дополнительный текст. Он называется преамбулой и эпилогом, соответственно. В протоколе HTTP эти элементы игнорируются. В сообщении электронной почты преамбула может содержать текст, выводимый клиентами электронной почты, не понимающими формата MIME.

В самом начале включаемой части располагаются заголовки, описывающие её содержимое (`Content-Type`, `Content-Length` и т. п.). Перед непосредственно телом части обязательно должна быть пустая строка, даже если заголовки отсутствуют. Если не определён `Content-Type`, то он берётся по умолчанию — `text/plain`.

Пример:

```
Content-Type: multipart/mixed; boundary="===============BOUNDARY=="
MIME-Version: 1.0
To: Qweru2003@gmail.com
From: 20gurulyov2003203@gmail.com
Subject: =?utf-8?b?0JA=?=

--===============HOMEP BOUNDARY==
Content-Type: text/plain; charset="utf-8"
MIME-Version: 1.0
Content-Transfer-Encoding: base64

0JA=

--===============BOUNDARY==
Content-Type: application/octet-stream
MIME-Version: 1.0
Content-Transfer-Encoding: base64
Content-Disposition: attachment; filename="text.txt"

VGhpcyBpcyBhIHRleHQgZmlsZSBmb3IgbGFiIG9mIFdhVCdzLiBTb210aGluZyBjb250ZW50OgpC
TEFCTEFCTEFCTEFCTEFCTEFCTEFCTEFCTEFCTEFCTEFCTEFCTEFCTEFCTEFCTEFCTEFCTEFCTEFC
TEE=

--===============BOUNDARY==--
```

Кодировка base64

Base64— стандарт кодирования двоичных данных при помощи только 64 символов ASCII. Алфавит кодирования содержит латинские символы A-Z, a-z, цифры 0-9 (всего 62 знака) и 2 дополнительных символа, зависящих от системы реализации.
В MIME base64 дополняется символами `+` и `/`.

Для того, чтобы преобразовать данные в Base64, первый байт помещается в самые старшие восемь бит 24-битного буфера, следующий — в средние восемь и третий — в младшие восемь бит. Если кодируется менее чем три байта, то соответствующие биты буфера устанавливаются в ноль. Далее каждые шесть бит буфера, начиная с самых старших, используются как индексы строки «ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/», и её символы, на которые указывают индексы, помещаются в выходную строку. Если кодируется только один или два байта, в результате получаются только первые два или три символа строки, а выходная строка дополняется двумя или одним знаком `=`. Это предотвращает добавление дополнительных битов к восстановленным данным. Процесс повторяется над оставшимися входными данными. При кодировании Base64 размер сообщения увеличивается приблизительно на 33 %. Это надо учитывать, если есть ограничения на размер конечного сообщения. Так при максимально допустимом размере 64 МБ реальный размер передаваемого сообщения должен быть не более 48 МБ.

Благодаря Base64 в html-документы можно включать бинарный контент, создавая единый документ без отдельно расположенных картинок и прочих дополнительных файлов. Таким образом html-документ с включённой в него графикой, аудио, видео, программами, стилями и прочими дополнениями становится прекрасной альтернативой другим форматам сложнооформленных документов типа doc, docx, pdf.

nslookup

nslookup - утилита, позволяющая обращаться к системе DNS.
Для MX-запроса, необходимо ввести следующую команду:

```
nslookup -type=mx domain.com
```

Где domain.com - домен, для которого выполняется запрос.
Пример:

```
 C:\Users\Qwery>nslookup -type=mx gmail.com
Сервер:  csp1.zte.com.cn
Address:  192.168.0.1

Ќ Ґ § б«г¦Ёў ойЁ© ¤®ўҐаЁп ®вўҐв:
gmail.com       MX preference = 5, mail exchanger = gmail-smtp-in.l.google.com
gmail.com       MX preference = 40, mail exchanger = alt4.gmail-smtp-in.l.google.com
gmail.com       MX preference = 10, mail exchanger = alt1.gmail-smtp-in.l.google.com
gmail.com       MX preference = 30, mail exchanger = alt3.gmail-smtp-in.l.google.com
gmail.com       MX preference = 20, mail exchanger = alt2.gmail-smtp-in.l.google.com
```