Содержание

Физические основы

- Оптика
  - По количеству жил:
    - многомодовая
    - одномодовая
  - По направлению передачи
    - Парное волокно
    - Одинарное волокно
- Медный кабель
  - Витая пара
    - Категории:
  - Коаксикал

Стандарты обжимки кабеля

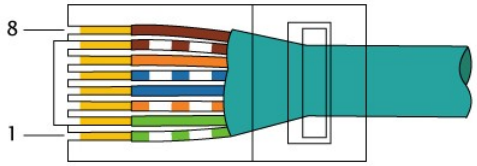
Разъем для стандарта Ethernet - 8P8C, часто ошибочно именуемый RJ-45. С формальной точки зрения RJ-45 - это витая пара с разъем 8P8C обжатая по стандарту B.

- A
  - БелоЗеленый Зеленый БелоЖелтый Синий БелоСиний Желтый БелоКоричневый Коричневый
  - Желтые идут через пропуск синего, потому что пара С-БС используется для телефонии и потенциально позволяет использовать один кабель для передачи данных Ethernet и телефонии одновременно.
- B
  - БелоЖелтый Желтый БелоЗеленый Синий БелоСиний Зеленый БелоКоричневый Коричневый

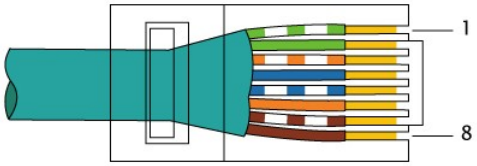
Для 10/100 Мбит/с сетей используются две пары из четырех - Зеленая и Желтая (в обоих стандартах обжимки) - одна для приема, другая для передачи данных, но фактически одновременно работает только одна пара. Иными словами, работа идет в режиме half-duplex.
Для 1 Гбит/с сетей используются все 4 пары (по каждой передается по 250 Мбит/с, дающие в сумме 1 Гбит/с). Поддерживают full-duplex.

При этом, выделяют два варианта обжимки кабеля:

- Для создания прямого кабеля (с обеих сторон A или с обеих сторон B) - для соединения порта сетевой карты с коммутатором или концентратором.
  - С обеих сторон A:

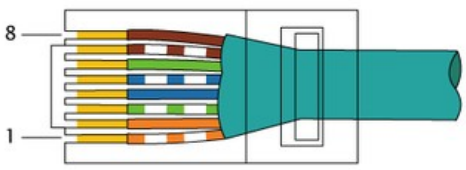


EIA/TIA-568A

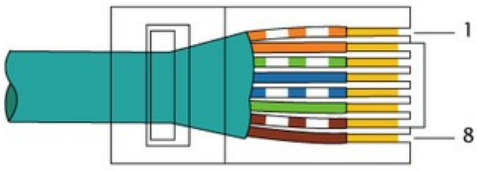


EIA/TIA-568A

  - с обеих сторон B:

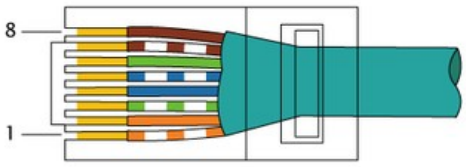


EIA/TIA-568B

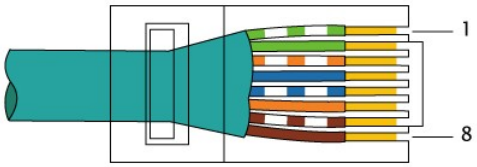


EIA/TIA-568B

- для создания перекрёстного (кроссированного) кабеля, имеющего инвертированную разводку контактов разъёма для соединения напрямую двух сетевых плат, установленных в компьютеры, а также для соединения некоторых старых моделей концентраторов или коммутаторов (uplink-порт).
    - 10/100 Мбит/с:

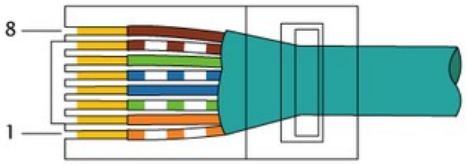


EIA/TIA-568B

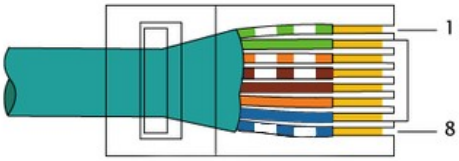


EIA/TIA-568A

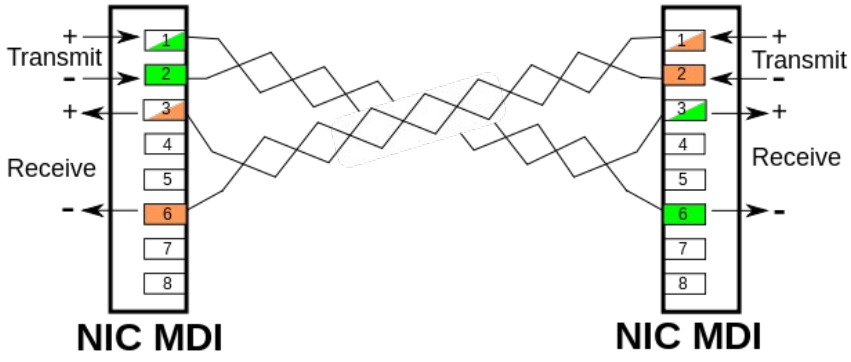
    - 1 Гбит/с:



EIA/TIA-568B



Цель кроссирования - правильное соединение пар предназначенных для приема/передачи данных:



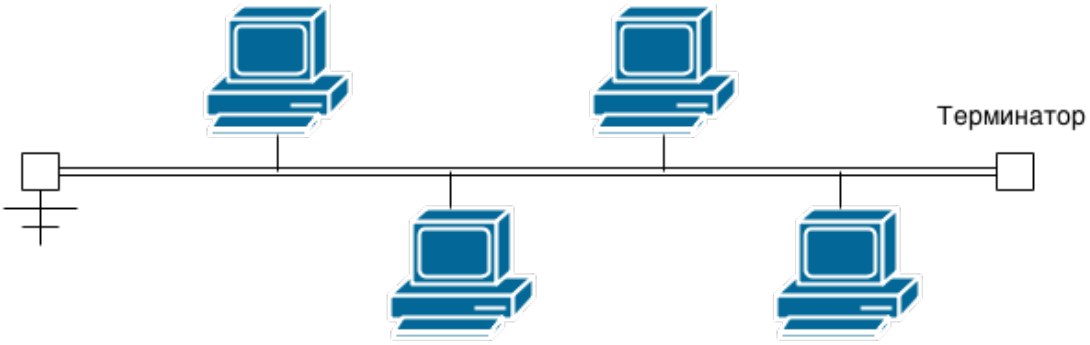
Практически все гигабитные устройства на данный момент не требуют кроссирования и умеют сами определять ориентацию кабеля.

NB!: необходимо понимать, что сетевой карте безразличен цвет кабеля - главное чтобы соответствующие приемо-передатчики были соединены правильно.

Лирическое отступление: благодаря тому, что в протоколе Ethernet используется дифференциальное кодирование, можно “безболезненно” поменять провода в паре.

Сетевое оборудование

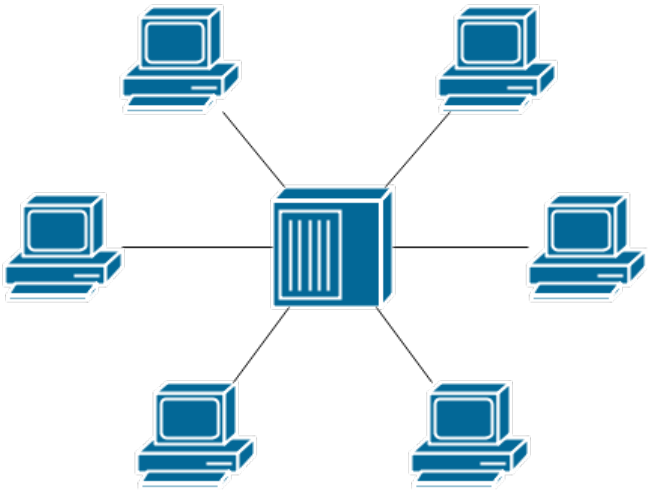
Лирическое отступление: на заре локальных сетей, для соединения компьютеров использовался коаксиальный кабель, позволявший соединить компьютеры с помощью топологии “шина” (“шина” в данном случае - и физическая и логическая топологии):



Логическая топология “шина” обладает очевидным недостатком: в каждый момент времени передавать данные может только один компьютер.

Концентратор

Концентратор повторяет любой входящий сигнал на все другие порты. Таким образом, несмотря на то, что физической топологией при использовании концентратора является “звезда” (см. рисунок), логической топологией является “шина“.
Пропускная способность концентратора равна минимальной скорости передачи данных одного из подключенных к нему устройств (По этой причине фактически концентраторы не используют для сетей выше 100 Мбит/с).



Коммутатор

В отличии от концентратора, логический топологией коммутатора является “звезда”. Передает пакет тому получателю, которому он нужен. Коммутатор таким образом может передавать несколько пакетов в разных направлениях, тем самым повышая свою суммарную пропускную способность (относительно концентратора).

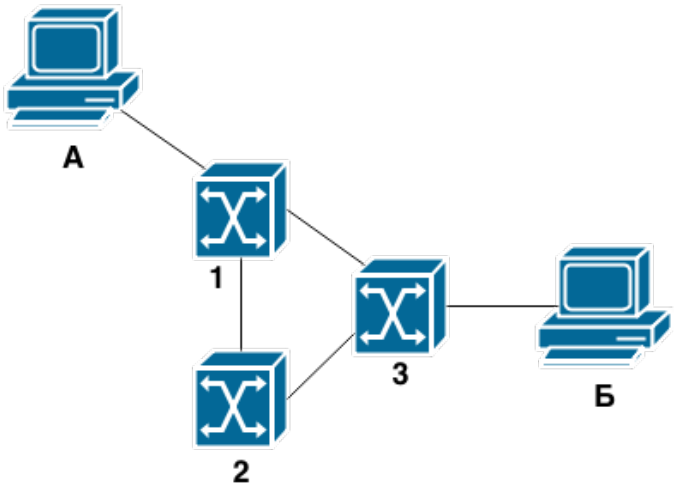
Необходимо помнить, что данные при этом не сразу передаются получателю. Минимальная задержка коммутатора равно времени, необходимому для получения Ethernet заголовка (для того чтобы узнать адрес получателя).
Задержка может быть больше, если порт получателя занят (образовывается очередь).
Также задержка может появится в случае рассогласования скоростей на портах.
Поддерживается одновременная передача и прием - **full-duplex** режим.

> *Лирическое отступление*. В связи с этим возможна проблема связанная с **согласованием дуплекса** - например у медиа-конверторов компании Билайн, не умеющих нормально согласовывать фулл-дуплекс, в следствии чего практически неработоспособна с сетевыми картами 1 Гбит/с (детектирующих правильную скорость (100 Мбит/с), но считающих, что работа идет в режим half-duplex).
> До тех пор, пока в каждый момент времени передает только одно устройство - все работает нормально. Но, если во время передачи медиаконвертера (устройства работающее в half-duplex), ему начинает отвечать сетевая карта (устройства работающее в full-duplex), то для устройства half-duplex это фактически означает *позднюю коллизию* (англ. late collision - коллизия, обнаруженная после передачи преамбулы), что вызывает прерывание передачи. Таким образом, пропускная способность такой сети резко падает.
>
> Аналогичная ситуация возможна при использовании активного сетевого оборудования (коммутатор, роутер), на котором системный администратор установил фиксировано: скорость работы 100 Мбит/с и режим дуплекса - full-duplex.

Для того, чтобы реализовать передачу каждого пакета только его получателю, необходимо использовать таблицы MAC адресов на портах. Коммутатор запоминает, что у него на определенном порту существует определенный MAC адрес. Если MAC адрес получателя ему не известен, то он рассылает этот пакет по всем портам, а далее по ответу, определяет, кто его “услышал”.

> *Лирическое отступление:* фильтрацией пакетов, направленных данному компьютеру занимается сетевая карта. В случае, если адрес получателя не совпадает с адресом сетевой карты, то драйвер сетевой карты даже не будет обращаться к ядру ОС с оповещением о пакете. Возможно также включение так называемого “неразборчивого” режима (англ. Promiscuous mode или *promisc mode*) сетевой карты, который позволяет принимать все пакеты независимо от того, кому они адресованы.
>
> Для прослушивания пакетов во всей сети, а не только в данном сегменте, можно искусственно создать ситуацию с переполнением таблицы MAC адресов коммутатора, что вынудит коммутатор слать каждый пакет по всем портам.

С вышеописанной логикой “рассылки пакета с неизвестным адресом получателя всем” связано понятие “сетевого шторма”.



В случае конфигурации, представленной на рисунке, Клиент А пытается переслать пакет Клиенту Б, при этом таблицы MAC адресов коммутаторов пусты (например их только что включили).

1. Коммутатор 1, не зная где находиться получатель, рассылает пакет по всем портам (в т.ч. 2 и 3 коммутатору).
2. Коммутатор 2 также не знает о клиенте Б и рассылает сообщения по всем портам.
3. Коммутатор 3 получив 2 пакета от 1 и 2 рассылает их по всем портам - в т.ч. 2 и 1.
4. Сеть не работает, потому что коммутаторы весело мигают лампочками, перебрасываясь пакетами :)

Для решения данной проблемы используется протокол STP (Spanning Tree Protocol - протокол связующего дерева). При подключении устройства к порту коммутатора, коммутатор вясняет, какие устройства к нему подключены на данном порту и с какими адресами, фактически выстраивая дерево сети. После чего, при возникновении вышеописанной ситуации выбирается линк, который отключается (“кладется”) (в нашем случае это может быть линк между коммутаторами 2 и 3). Выбор линка осуществляется исходя из максимальной удаленности от корневого коммутатора. У каждого коммутатора есть идентификатор для протокола STP. Используя этот идентификатор, а также на основании построенного дерева выбирается коммутатор, который будет считаться корневым (Т.е. от кого будет разворачиваться дерево). В случае же потери линка между коммутаторами 1 и 2, дерево будет перестроено и линк между 2 и 3 коммутатором будет включен (“поднят”).

Недостатками протокола STP являются:

- Большое время перестроения дерева. Время перестроения дерева также может достигать минуты и в течение этого времени пакетов во всей сети блокируется.
- Большое время поднятия линка. Так время поднятия линка для оконечного устройства составляет порядка минуты. Для решения этой проблемы в коммутаторе протоколу STP сообщается что к данному порту гарантировано подключаются только оконечные устройства (а не коммутаторы), путем перевода протокола STP на данном порту в специальный режим. Тем самым поднятие линка ускоряется.

  Лирическое отступление. Еще одной причиной не включать протокол STP на всех портах (кроме ускорения поднятия линка для оконечных устройств), является защита от несанкционированно подключенного коммутатора:
- Если протокол STP выключен, то коммутатор просто не будет работать.
- Если протокол STP включен и у подключенного коммутатора идентификатор STP больше, чем идентификаторы коммутаторов в сети, то он может стать корневым и “положить” сеть на некоторое время.

Существует несколько замен протокола STP (реализующий аналогичный функционал), от разных производителей, отличающиеся:

- более быстрым соединением
- защитой от несанкционированного подключения активного сетевого оборудования
- возможностью определения весовых коэффициентов для определенных VLAN’ов и линк будут отключаться не полностью, а только для определенных VLAN’ов. Это позволяет более равномерно нагрузить сеть.

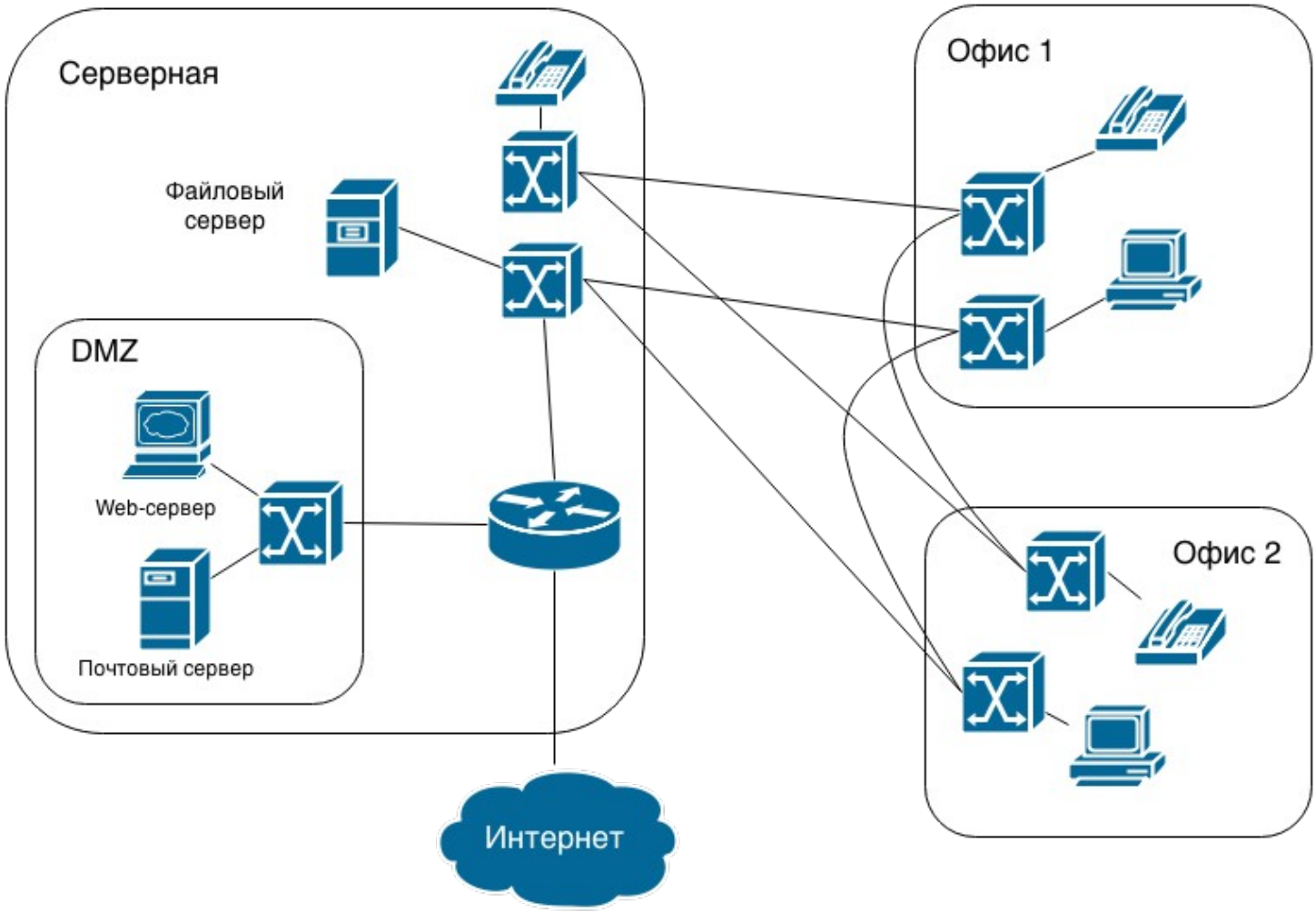
Соответственно, если технический парк состоит из оборудования различных производителей, придется использовать “старый добрый” STP.

Ethernet frame

Размер фрейма (MTU - maximum transmission unit) для 10/100 Мбит/с - 1500 байт. Если использовать аналогичный размер фрейма для 10 Гбит/с сетей, то мы встаем перед необходимостью передавать ~833333 пакетов в секунду (10000000000 [бит/с] / (1500 [байт] * 8 [бит])) для обеспечения заявленной скорости. Обработка такого количества внешних запросов (без учета непосредственно обработки данных) является серьезной нагрузкой для центрального процессора. Для решения данной проблемы используется Jumbo Frame - фреймы, имеют размер, превышающий стандартный размер: от 1500 до 9000 байт. Использование Jumbo Frame ограничено сетевым оборудованием поддерживающим 1 Гбит/с.

VLAN

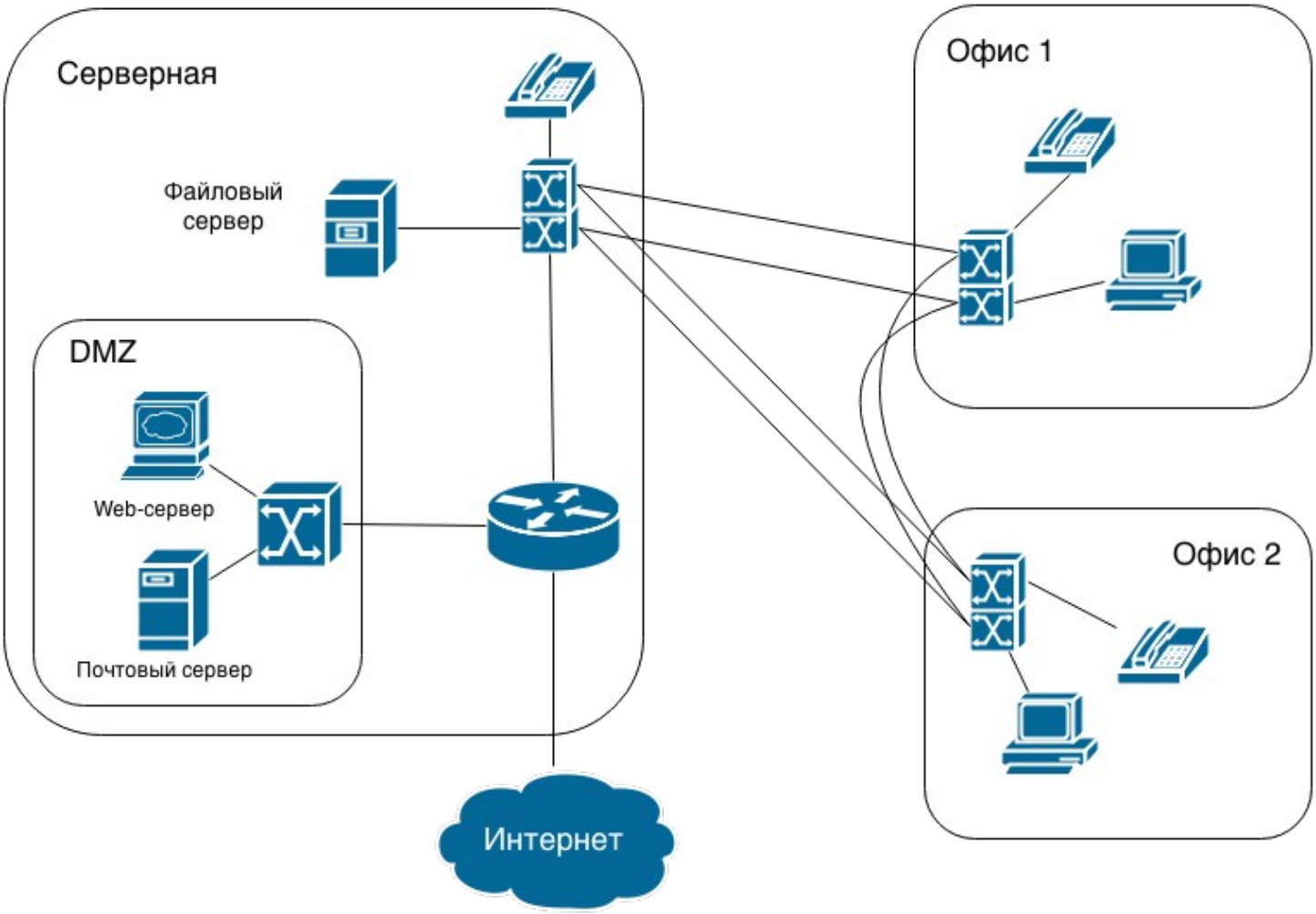
Ранее речь шла о простой ситуации, при которой к коммутатору подключено несколько устройств и они видят друг друга. Это удобно до тех пор, пока функции подключенных узлов не становятся различными. Например, на каждом рабочем месте находится IP-телефон и рабочая станция. В таком случае возникает желание вынести их в разные сети, обеспечить приоритизацию трафика и т.п., как следствие - необходимость разнести сегменты. Очевидное решение “в лоб” - поставить два коммутатора. Рассмотрим пример подобной организации сети:



При этом мы столкнемся с двумя проблемами:

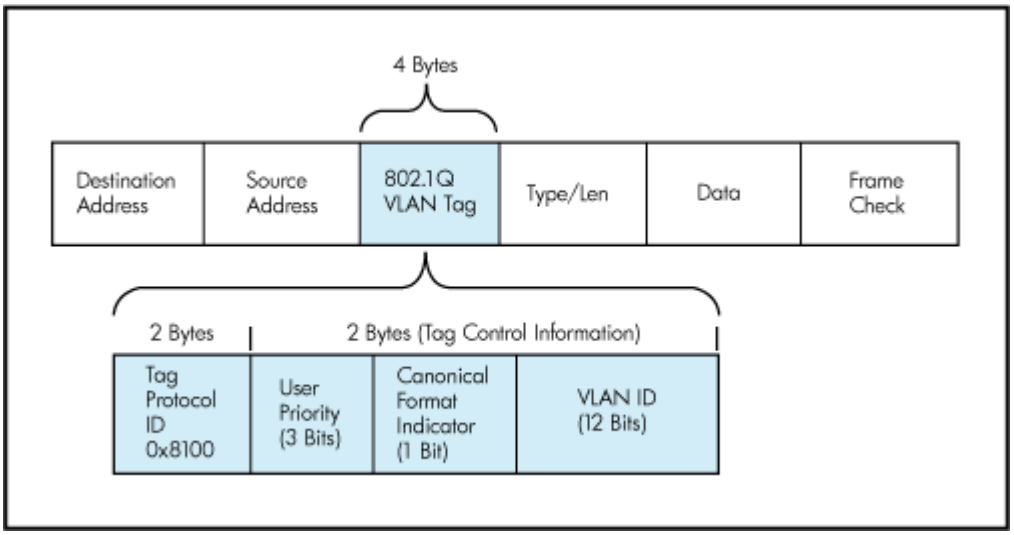
- подобная топология будет излишне сложной, а количество соединений станет слишком большим.
- ethernet-фрейм не содержит никакой информации о приоритете, приоритизация не будет работать.

Существуют коммутаторы, умеющие логически разбивать сеть. Часть их портов отводится под одну сеть, а часть - под другую. Таким образом, одно устройство обслуживает несколько сетей одновременно. Применим это решение к нашему примеру:



Удалось сократить количество коммутаторов, но остались продублированными каналы связи. Также, осталась нерешенной проблема приоритизации.

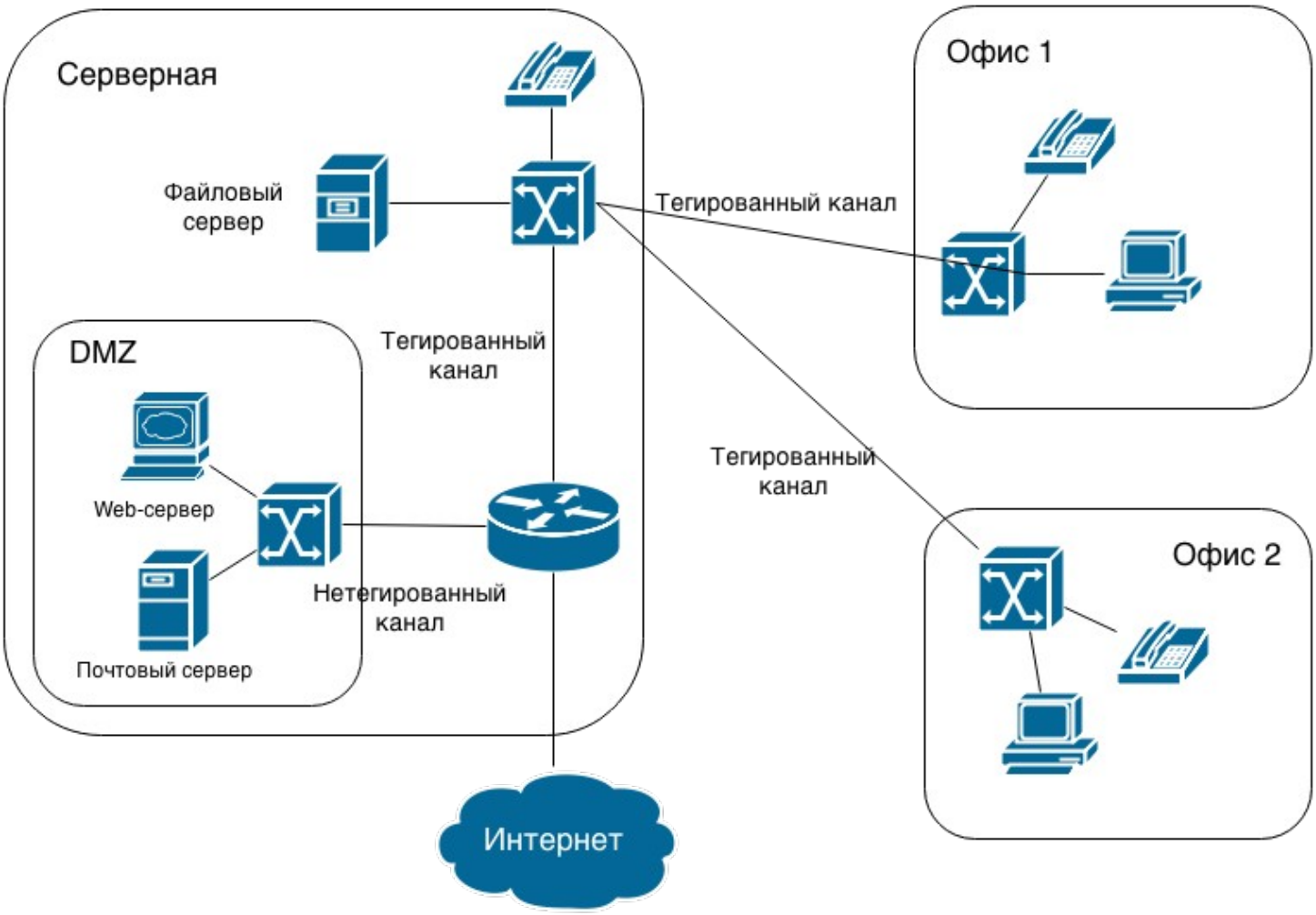
Для решения этих проблем был разработан стандарт IEEE 802.1Q. Этот стандарт описывает расширение ethernet-фрейма 4-мя байтами:



- **Tag Protocol Identifier (TPID)** — Идентификатор протокола тегирования. Указывает, какой протокол используется для тегирования. Для 802.1Q используется значение 0x8100.
- **Tag Control Information (TCI)**- поле, инкапсулирующее в себе поля приоритета, канонического формата и идентификатора VLAN:

- **Priority** — приоритет. Используется стандартом IEEE 802.1p для задания приоритета передаваемого трафика.
- **Canonical Format Indicator (CFI)** — Индикатор канонического формата. Указывает на формат MAC-адреса. 0 — канонический(Кадр Ethernet), 1 — не канонический(Кадр Token Ring,FDDI).
- **VLAN Identifier (VID)** — идентификатор VLAN'а. Указывает, какому VLAN'у принадлежит фрейм. Диапазон возможных значений VID от 0 до 4095. При этом значения 0 и 4095 зарезервированы.

Стандарт позволяет в рамках одного коммутатора логически организовать несколько сетей и таким образом разделить области видимости устройств, подключенных к одному физическому коммутатору. Устройства, находящиеся в разных VLAN не смогут обмениваться ethernet-фреймами.
С использованием 802.1Q пример изменится следующим образом:



При использовании VLAN как, например, на данной схеме, конечные узлы (рабочие станции и ip-телефоны) подключаются у незатегированным (untagged) портам, а вышестоящий коммутатор подключается а теггированному (tagged) порту. При этом коммутаторы совершают процессы тегирования и растегирования трафика. Тегированные порты используют 802.1Q, а нетегированые - стандартный ethernet frame. Пакеты, передаваемые между тегированными портами содержат теги, описывающие, что рабочие станции относятся к VLAN1, а ip-телефоны к VLAN2. Маршрутизатор видит все узлы, подключенные к разным VLAN и может перенаправлять пакеты между ними. Маршртуизация в таком случае происходит уже на уровне IP, а не ethernet. Отсюда следует интересный факт: возможно существование маршрутизатора с одним-единственным сетевым интерфейсом.
В основном все сетевые адаптеры умеют работать с тегированными пакетами.

При использовании стандарта 802.1Q может возникнуть следующая проблема. Параметр MTU описывает полностью весь фрейм, включая заголовки. Так как заголовок дополняется 4-мя байтами, то на эти же 4 байта сокращен объём полезных данных во фрейме. На некоторых старых ОС могут возникать проблемы, связанные с невозможностью правильно определить длину кадра.

IP

IP является самым распространенным протоколом 3-го уровня модели OSI. Найти организацию, в которой на этом месте был бы не он почти невозможно. Телефония и телевидение также постепенно переходят на этот протокол. На текущий момент протокол имеют две актуальные версии: IPv4 и IPv6. Основное отличие между ними заключается в количестве возможных адресов (около 4.2 млрд у IPv4 против примерно 200 млн адресов на каждого жителя Земли у IPv6).

Адрес

Несмотря на то, что IPv4 адреса формально кончились ещё в 2011 году, массовый перехода на IPv6 происходит медленно. Поэтому далее будет рассматриваться IPv4.
IP-адрес состоит из двух частей: адрес подсети и адрес узла в этой сети:

С формальной точки зрения подсеть подразумевает то, что внутри неё узлы могут непосредственно общаться, используя протоколы нижестоящего уровня.
В любой подсети зарезервировано два адреса:

- нулевой (состоящий целиком из нулей) - так называемый адрес сети
- последний (состоящий целиком из единиц) - broadcast-адрес для данной сети. Broadcast-адрес - это адрес, на который должны отвечать все.

Соответственно под адрес хоста должно быть отведено минимум 2 бита, что даст 4 возможных варианта, два из которых будут зарезервированы. Подобная конфигурация используется провайдерами при подключении абонентов в случае, когда необходима изоляция сегментов для разных абонентов. То есть абоненту предоставляется один внешний IP адрес, еще один адрес выделяется под адрес маршрутизатора со стороны провайдера. С формальной точки зрения возможна сеть с лишь одним битом в адресе хоста, но использование такой конфигурации не рекомендуется.

Маска

Разделение между битами, являющимися адресом сети, и битами, являющимися адресом хоста, осуществляется маской. Маска не является составной частью адреса, она лишь описывает где в адресе адрес узла, а где - адрес сети. Записать маску можно в двух видах:

- 111...111|000...00 - все единицы покрывают адрес сети, а нули покрывают адрес узла. Используется традиционно.
- 000...000|111..11 - наоборот. Встречается в различном сетевом оборудовании, например при описании ACL в коммутаторах и маршрутизаторах Cisco.

Таким образом, имея маску и полный адрес и применив операцию конъюнкции, можно получить адрес сети и адрес хоста.
Разумеется, записывать маску в виде 32 бит не удобно. Существуют более короткие формы записи маски:

- разбиение сетевой маски по байтам и запись десятичного значения каждого байта, разделяя значения точками. Например, 255.255.255.240.
- аналогичная 16-тиричная запись. Например, FF.FF.FF.F0. Такую форму использует, в частности, Solaris.
- <ip-адрес>/<количество единиц в маске>. Например, 192.168.0.1/28. Данная форма была предложена Cisco. Большинство современных ОС понимают эту форму записи.

Маска вида 255.255.254.240 (в двоичным виде 11111111 11111111 1111111**0** 11110000) не корректна, потому что в ней часть, покрывающая адрес (единицы), разбита на несколько частей нулём.

Классы сети
Во время разработки протокола IP адреса были разделены на несколько классов:
A, B, C, D, E. Адреса классов A, B, C могут являться адресами конечных устройств. Класс D является классом, предназначеным для multicast, класс E зарезервирован.

Класс	Маска	Диапазон адресов	Число узлов	Зарезервированные подсети
A	/8	0.0.0.0 - 127.0.0.0	2^{24} -2	0. 10. - private 127. - loopback (127.0.0.1)
B	/16	128.0.0.0 - 191.255.0.0	2^{16} -2	172.16 - 172.31 - private
C	/24	192.0.0.0 - 223.255.255.0	2^{8} -2	192.168/16 - private
D	-	224.0.0.0 - 239.255.255.255	multicast	-
E	-	240.0.0.0 - 254.255.255.255	-	зарезервирован целиком

Операционные системы могут использовать классовое деление для определения маски сети в случае, если она не была указана явно.

Broadcast
Каждый компьютер с формальной точки зрения должен отзываться на два broadcast-адреса: Первый - это тот адрес, который относится к его подсети и второй - 255.255.255.255. Адрес 255.255.255.255 - немаршрутизируемый, т.е. используя его можно послать пакет только устройствам из своей подсети, потому что маршрутизатор не перешлет пакеты, отправленные на него, никуда выше. Пакет на broadcast-адрес, полученный из внешней сети, должен быть разослан всем узлам этой подсети (будет использован широковещательный MAC-адрес FF:FF:FF:FF:FF:FF).

IP over ...
Протокол IP подразумевает возможность пересылки не только через Ethernet, но и другие протоколы:
- Token Ring (не распространён)
- PPP (мобильный Интернет)
- IP/TCP/UDP (через туннель)

Маршрутизация

Таблица маршрутизации
Таблица маршрутизации имеется на любом устройстве, реализующим IP-стек. Выглядит она следующим образом:

```
>netstat -rnv

IRE Table: IPv4
  Destination             Mask         Gateway       Device  MTU  Ref Flg  Out  In/Fwd
------------------- --------------- ------------------- ------ ----- --- --- ----- ------
default             0.0.0.0         192.168.11.121    z122cs11  1500   2 UG      25     0
127.0.0.1           255.255.255.255 127.0.0.1         lo0    8232   2 UH   948    948
74.125.143.103      255.255.255.255 192.168.11.1                    0   2 UHD   1      0
128.8.10.90         255.255.255.255 192.168.11.1                    0   2 UHD   9289   19
128.63.2.52         255.255.255.255 192.168.11.1                    0   2 UHD  10626   19
128.63.2.52         255.255.255.254 192.168.11.104                  0   2 UHD  9251    0
192.5.5.241         255.255.255.255 192.168.11.1                    0   2 UHD  10690   22
192.33.4.12         255.255.255.255 192.168.11.1                    0   2 UHD   9237   16
192.168.11.0        255.255.255.0   192.168.11.122   z122cs11  1500  36 U       15196  338
192.168.103.0       255.255.255.0   192.168.11.104   z122cs11  1500   1 UG      0      0
```

```
192.168.105.0       255.255.255.0   192.168.11.104    z122cs11  1500   1 UG      0      0
192.168.115.0       255.255.255.0   192.168.11.115    z122cs11  1500   1 UG      0      0
192.168.116.0       255.255.255.0   192.168.11.115    z122cs11  1500   1 UG      0      0
192.168.118.0       255.255.255.0   192.168.11.118    z122cs11  1500   1 UG      0      0
192.168.122.0       255.255.255.0   192.168.122.1     z122cs122  1500   8 U      22201   7653
192.168.125.0       255.255.255.0   192.168.11.125    z122cs11  1500   1 UG      0      0
192.168.126.0       255.255.255.0   192.168.11.126    z122cs11  1500   1 UG      0      0
192.168.133.0       255.255.255.0   192.168.11.133    z122cs11  1500   1 UG      0      0
192.168.136.0       255.255.255.0   192.168.11.136    z122cs11  1500   1 UG      0      0
192.168.137.0       255.255.255.0   192.168.11.137    z122cs11  1500   1 UG      0      0
192.168.153.0       255.255.255.0   192.168.11.153    z122cs11  1500   1 UG      0      0
```

Ключи команды netstat:

- r - выводить таблицу маршрутизации
- n - не преобразовывать адреса в имена в соответствии с DNS и т.п.
- v - показ сетевой маски, которая используется для данного маршрута.

Формат вывода этой таблицы может отличаться. Например, Solaris 11 также добавляет в вывод маршруты для ipv6-сетей, а BSD и Linux не имеют ключа v, поскольку всегда показывают сетевую маску.

Описание полей таблицы маршрутизации

- Destination - адрес сети, к которой относится данный маршрут. В некоторых системах в качестве destination выводится default, под которым на самом деле скрывается адрес 0.0.0.0
- Mask - сетевая маска, которая может быть наложена на адрес сети, чтобы определить, какие маршруты покрывает данный маршрут.
- Gateway - шлюз, выход, то есть то, через что нужно отправить пакет.
- Flg (Flags) - Флаги:
    - U (UP) - рабочий, “живой”
    - G (Gateway) - чтобы воспользоваться данным маршрутом, необходимо повторить поиск по таблице маршрутизации, но используя в качестве назначения не первоначальный адрес назначения, а адрес того gateway, который указан в маршруте. Этого флага нет у маршрута, если он относится к той подсети, к которой мы подключены.
    - H (Host) - маршрут до конкретного хоста. Использовался в старых ОС для указания маршрута для хоста вместо использования сетевой маски /32. Сейчас этот флаг фактически deprecated и устанавливается автоматически.
    - D (Dynamic) - маршрут построен демоном динамического построения таблицы маршрутизации. (включается *routeadm -e ipv4-forwarding*). Наличие таких маршрутов нежелательно и это повод задуматься.
- Interface - интерфейс, через который будет послан пакет по этому маршруту

Маршрут по умолчанию (default route)

Маршрут по умолчанию - это маршрут, у которого в качестве Destination указано 0.0.0.0, а маска равна 0.0.0.0. Если наложить на любой IP-адрес данную маску, то получится 0. Таким образом, маршрут по умолчанию подразумевает, что если мы не нашли других вхождений, то при наличии соответственно маршрута по умолчанию он будет использован, потому что нахождение маски 0 гарантированно даёт 0 и мы попадём в данный маршрут. Данный маршрут тоже находится в таблице маршрутизации. Наличие маршрута по умолчанию не является обязательным. Тогда, если в таблице маршрутизации не будет найден подходящий маршрут, отправителю будет отправлен ICMP-пакет с сообщением "host unreachable".

В маршрутизаторах уровня провайдера и иных крупных сетях используются протоколы глобальной маршрутизации. При этом маршрутизаторы строят свои таблицы маршрутизации на основании данных этих протоколов. У такого маршрутизатора в том или ином виде есть маршруты для всего Интернета, но как такового маршрута по умолчанию у него нет. Если сеть закрытая, то в таком случае можно прописать все маршруты напрямую и не указывать маршруты по умолчанию.

Поиск по таблице маршрутизации

Пусть нам дан адрес получателя, которому требуется отправить пакет (это адрес назначения в заголовке IP-пакета).

1. На этот адрес накладывается сетевая маска /32 (маска из всех единиц). Получается тот же самый адрес.

2. Пара (получившийся адрес, сетевая маска) ищется в таблице маршрутизации.
3. В таблице маршрутизации может быть маршрут непосредственно для конкретного хоста. В таком случае он будет сразу найден.
4. Маска арифметически сдвигается влево на один бит (и становится /31).
5. Маска вновь накладывается на адрес и производится повторный поиск.
6. Шаги 4-5 повторяются, пока не будет найдено совпадение.
7. Последней будет наложена маска /0, гарантированно получится адрес 0.0.0.0. В большинстве случаев при этом будет использован маршрут по умолчанию.

В таблице маршрутизации могут находиться маршруты с одинаковыми адресами, но разными масками. Рассмотрим примеры, в которых будет произведен поиск по таблице маршрутизации, выведенной выше.

Пример 1. Адрес 128.63.2.52/32

Маска	Адрес с наложенной маской	Комментарий
/32	128.63.2.52/32	В таблице маршрутизации найдена запись с совпадающими адресом 128.63.2.52 и маской /32. Таким образом, поиск по таблице маршрутизации окончен. Обратите внимание, что в таблице маршрутизации также присутствует вторая запись с адресом 128.63.2.52, но с маской /31. В данном примере эта запись не участвует, т.к. поиск осуществляется до первого совпадения пары маски и адреса.

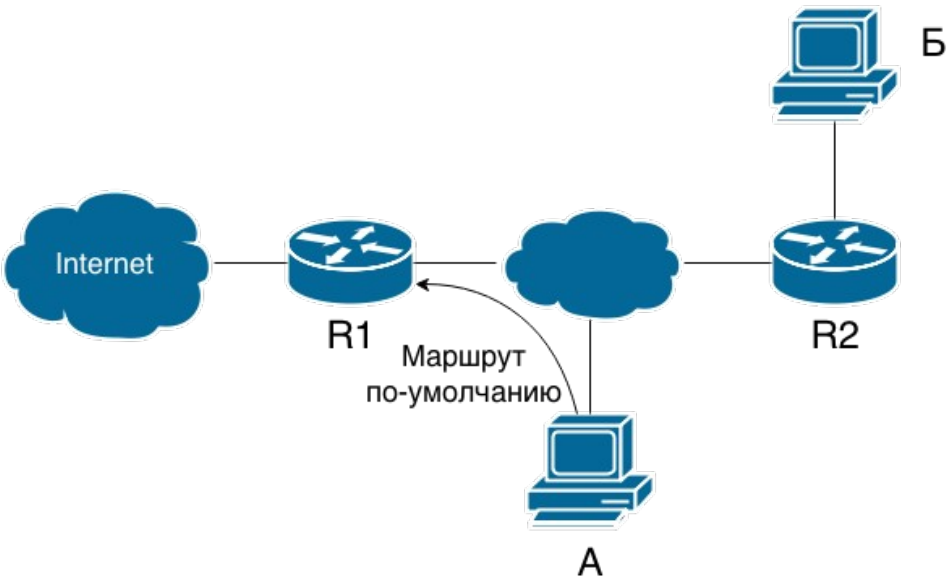
Пример 2. Адрес 192.168.117.1/32.

Маска	Адрес с наложенной маской	Комментарий
/32	192.168.117.1/32	В таблице маршрутизации нет совпадающей пары маски и адреса. Маска сдвигается.
/31	192.168.117.0/31	Маска /31 затерла младшую единицу в адресе. В таблице маршрутизации нет совпадающей пары маски и адреса. Маска сдвигается.
...	...	...
/23	192.168.116.0/23	Маска /23 затерла единицу в изначальном адресе, сделав из него адрес 192.168.116.0. В таблице маршрутизации есть запись с адресом 192.168.116.0, однако в этой записи указана маска /24, а на данном шаге используется маска /23. Таким образом, совпадающей пары маски и адреса нет, маска сдвигается дальше.
...	...	...
/1	128.0.0.0/1	Маска /1 затирает все единицы в адресе, кроме последней, сделав таким образом адрес 128.0.0.0. В таблице маршрутизации нет совпадающей пары маски и адреса. Маска сдвигается.
/0	0.0.0.0/0	Маска /0 затерла все единицы в адресе. В таблице маршрутизации есть маршрут 0.0.0.0/0, являющийся маршрутом по умолчанию. Он помечен как Gateway. Это означает, что нужно повторить поиск по

		таблице маршрутизации, используя указанный в записи адрес (192.168.11.121) для поиска маршрута.
/32	192.168.11.121/32	Поиск повторяется сначала с использованием адреса из маршрута по умолчанию. В таблице маршрутизации нет совпадающей пары маски и адреса. Маска сдвигается.
...	...	...
/24	192.168.11.0/24	Маска /24 затерла младший байт адреса. В таблице маршрутизации присутствует запись с такими же адресом и маской. Эта запись указывает на адрес 192.168.11.122. Поиск по таблице маршрутизации окончен. Теперь известен интерфейс (z122cs11), через который следует перенаправить пакет. В качестве источника будет использован адрес Gateway (192.168.11.122). Далее отправкой занимаются более низкоуровневые протоколы.

Динамическая маршрутизация

Рассмотрим следующую конфигурацию сети, где у узла А настроен только маршрут по умолчанию:



При отправке пакета из узла А в узел Б пакет будет направлен согласно таблице маршрутизации на маршрутизатор R1. Так как данный маршрутизатор не отвечает за перенаправление пакета в узел Б, то он пошлет обратно узлу А ICMP-дейтаграмму “Redirect” с указанием правильного маршрута (через R2). Получив её, узел А добавит в свою таблицу маршрутизации динамическую запись о маршруте в узел Б через R2, после чего успешно осуществит передачу. Однако, через некоторое время динамическая запись будет удалена из таблицы маршрутизации и процесс повторится вновь.

При использовании statefull-файервола, пропускающего только пакеты, инициализирующие соединения, ситуация усугубляется. Первый пакет, устанавливающий соединение, будет передан аналогично описанному выше случаю и какое-то время передача данных от узла А к узлу Б будет идти нормально. Однако, когда динамическая запись удалится из таблицы маршрутизации узла А, тот отправит пакет с данными вновь на маршрутизатор R1. Так как R1 не знает о том, что соединение между А и Б было установлено, пакет будет уничтожен и передача данных прекратится.

Ещё один негативный момент связан с тем, что передача ICMP-дейтаграмм с динамическими маршрутами ничем не регулируется и злоумышленник может совершать атаки, рассылая

соответствующие сообщения. Поэтому хорошей практикой является фильтрация таких пакетов и правильная настройка маршрутизации.

ARP

В подавляющем большинстве случаев протокол IP работает поверх Ethernet, поэтому далее будем рассматривать именно эту пару протоколов. Для отправки пакета необходимо преобразовать IP-адрес, по которому требуется доставить пакет в MAC-адрес получателя.
Протокол ARP (Address Resolution Protocol) предназначен для получения MAC-адрес по известному IP-адресу. В то же время это название таблицы, в которой хранится эта информация за время работы. Отправка ARP-запроса осуществляется в тот момент, когда происходит попытка отправить пакет на адрес, для которого нет записи в ARP-таблице. Соответственно, если запись уже есть, то она и будет использована.. Стоит отметить, что команда ARP не отправляет ARP-запрос, а лишь происходит поиск по указанному адресу в ARP-таблице.

ARP-таблица

```
>arp -an

Net to Media Table: IPv4
Device   IP Address               Mask      Flags      Phys Addr
------ -------------------- --------------- -------- ---------------
cs10   192.168.10.1         255.255.255.255          00:14:4f:a2:ef:2c
cs10   192.168.10.10        255.255.255.255          00:25:90:98:33:be
cs10   192.168.10.11        255.255.255.255          02:08:20:33:d0:ec
cs10   192.168.10.12        255.255.255.255 SPLA     02:08:20:71:e1:ab
cs10   192.168.10.188       255.255.255.255          00:11:d8:c8:ec:8b
igb0   224.0.0.0           240.0.0.9       SM        01:00:5e:00:00:00
```

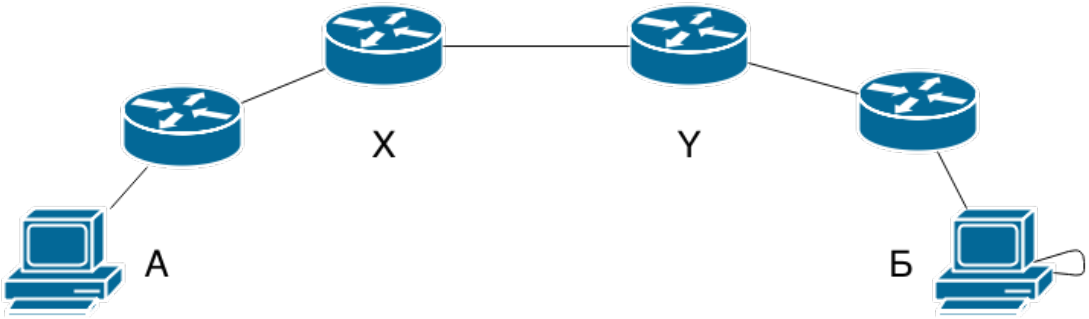
Описание полей:

- Device - сетевой интерфейс, к которому привязана данная запись
- IP Address и Mask - соответственно IP адрес и сетевая маска
- Flags (флаги)
    - S (Static) - данная запись является статической. То есть:
        - не была получена с использованием протокола ARP
        - не является динамической, то есть не будет удалена по истечении какого-то времени.
          Все динамические записи в таблице ARP периодически удаляются и обновляются, используя протокол ARP
    - P (Public) - эта конкретная система будет отвечать на ARP-запросы для данного конкретного адреса.
      Для получения ответа на ARP-запрос не обязательно иметь в наличии в сети интересующую нас систему, этот ответ может быть получен от любой другой системы, которой известен MAC-адрес запрашиваемой системы и которая имеет в своей ARP-таблице запись о ней с ключом P
    - L (Local) - этот адрес назначен на логическом интерфейсе данной системы
    - A (Authority) - данная запись является авторитетной, ARP не будет принимать обновления этой записи от других систем
    - U (Unresolved) - на ARP-запрос по данному адресу не ответила ни одна система. По истечении короткого времени эта запись будет удалена.
    - M (Multicast) - означает, что для данной записи используется специальное преобразование, при котором часть IP-адреса отображается в MAC-адреса.
- Phys Addr - собственно MAC-адрес устройства.

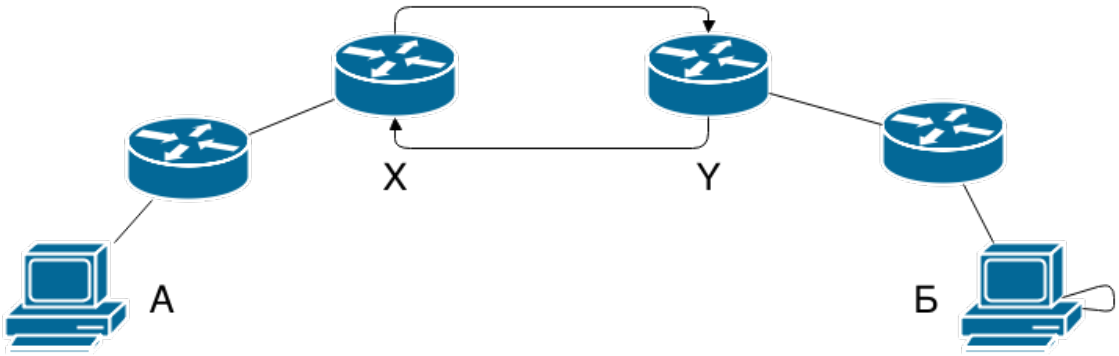
Поля заголовка IP пакета

TTL

Предположим, имеются две системы, соединенные цепью маршрутизаторов в одну сеть.



Теперь рассмотрим случай, когда на маршрутизаторе X маршрут к узлу Б указывает на маршрутизатор Y, а на маршрутизаторе Y маршрут к узлу Б указывает назад на маршрутизатор X:



Когда из узла А в узел Б будет отправлен пакет, то согласно таблицам маршрутизации он начнет двигаться циклически между маршрутизаторами X и Y. Чтобы подобное перемещение пакета могло завершиться, в IP-заголовке присутствует поле TTL (Time To Live) - время жизни. Во времена разработки протокола IP данное поле действительно было временем жизни пакета в секундах. На современных же скоростях такая семантика стала нерациональной и поле стало отвечать за количество хопов (Hop), то есть устройств на пути следования пакета. Каждый маршрутизатор, через который проходит IP-пакет, уменьшает значение TTL на единицу. Когда после уменьшения TTL становится равным нулю, узел уничтожает пакет. При этом также отправителю пакета шлется ICMP-дейтаграмма “Host unreacheble. TTL expired.”

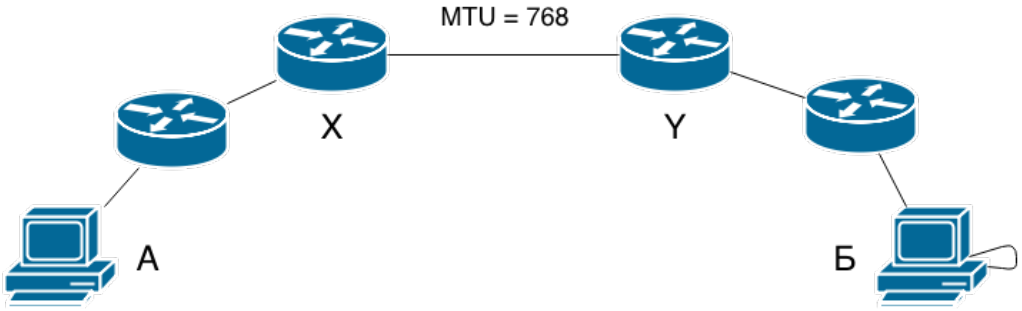
На основе вышеописанного поведения работает команда traceroute. Traceroute делает следующее:

1. Отправляет UDP (по умолчанию в UNIX-системах) пакет с TTL = 1 на указанный IP-адрес
2. Получает ICMP “Host unreacheble. TTL expired.” от первого в цепочке маршрутизатора
3. Увеличивает TTL на единицу
4. Повторяет п.1-3, получая ICMP-ответы от последующих в цепи маршрутизаторов до тех пор, пока пакет не достигнет адресата или не будет достигнут предел TTL.
5. Traceroute использует порт 33434 (по умолчанию), который не рекомендуется использовать для иных целей. Соответственно, этот порт никем не слушается и при достижении UDP-пакетом адресата ядро ОС сформирует ICMP-ответ “Unreachable”, что будет являться сигналом к тому, что трассировка завершена.

MTU

MTU (Maximum Transmission Unit) - максимальный размер передаваемого пакета в байтах. Является характеристикой сетевого интерфейса, так как эта величина зависит от используемого протокола 2-го уровня OSI. Для протокола Ethernet MTU = 1500.

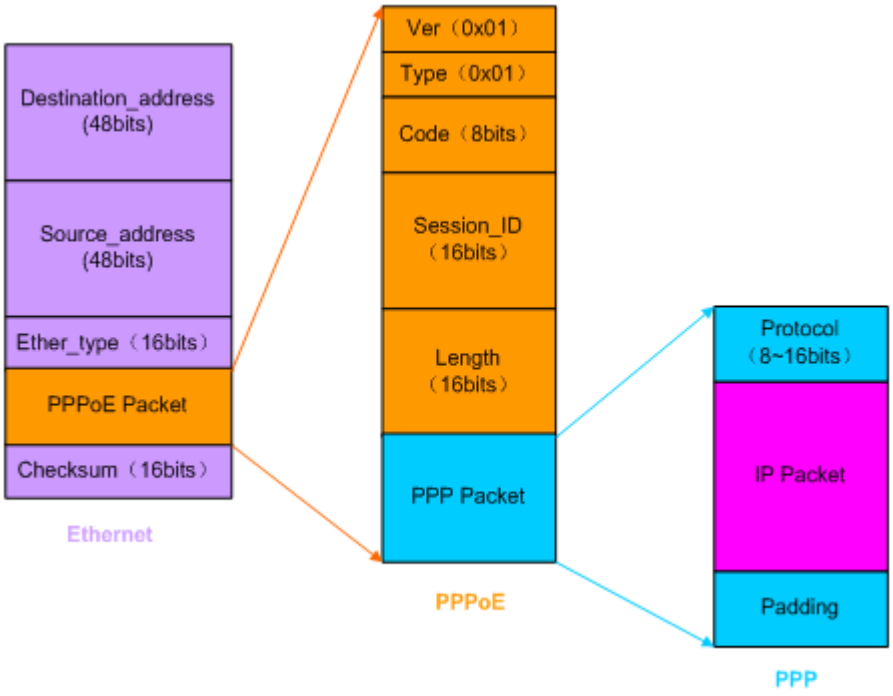
Существуют сети, в которых используются различные физические каналы передачи, разные протоколы, туннели и, как следствие, разное значение MTU на разных участках сети. Рассмотрим пример, в котором MTU между узлами X и Y меньше 1500 и равно 768.



Пусть из узла А в узел Б отправлен IP-пакет длиной 1500 байт.
Стоит напомнить, что в заголовке IP-пакета есть флаг DNF - “не фрагментировать”
Если этот флаг установлен, то дойдя до маршрутизатора X, пакет будет уничтожен.
Если флаг не установлен, то маршрутизатор X может поступить двумя способами:

1. Самостоятельно произвести фрагментацию и отправить пакет по кусочкам маршрутизатору Y. Однако это дополнительная нагрузка на маршрутизаторы.
2. Уничтожить пакет и послать отправителю ICMP-дейтаграмму “Fragmentation needed” с указанеием требуемого MTU. Соответственно хост А получит это сообщение и повторит передачу пакетами поменьше.

Аналогичная потребность в перефрагментировании возникает при использовании, например, протокола PPPoE, инкапсулирующего IP-пакет. За счет заголовков максимальная длина инкапсулированного пакета оказывается меньше 1500 байт.



Протоколы поверх IP

Поверх IP работают протоколы TCP, UDP, ICMP, SCTP (совмещающий достоинства TCP и UDP) и прочие.
Протоколы UDP и ICMP являются дейтаграммными и обладают следующими особенностями:

- работают без установления соединения
- негарантиуемость доставки
- негарантиуемость порядка

ICMP

Протокол ICMP является в свою очередь служебным для UDP и для IP. Напрямую используется, например, в команде ping, посылающей запрос ICMP “echo request” и получающей ICMP “echo response”. Одна из проблем этого протокола в том, что никем не регулируются адреса отправителя и получателя. Это позволяет злоумышленнику посылать ICMP “echo request” с чужим адресом отправителя. Все, получившие данный пакет ответят ICMP “echo response”, что выльется в DDOS атаку, причем источник будет невозможно определить.

UDP

Используется в следующих случаях:

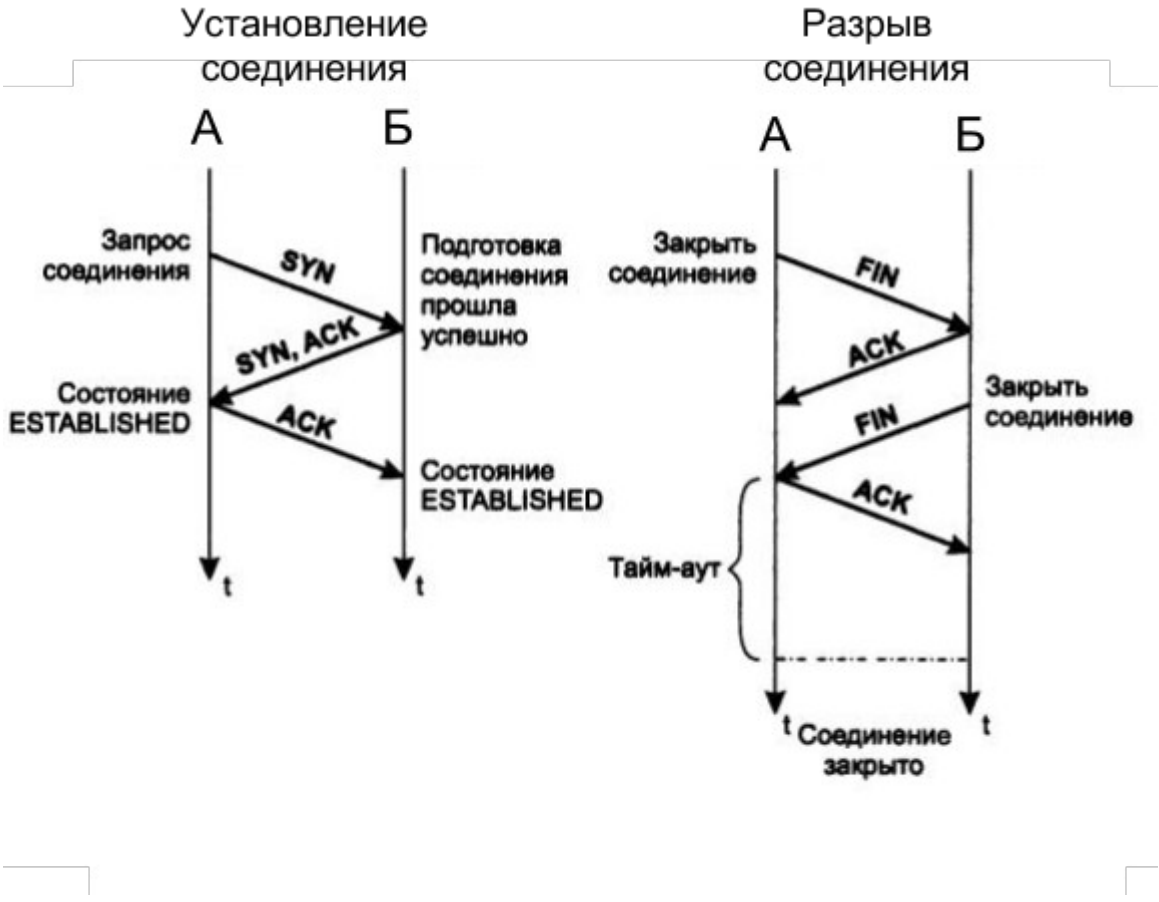
- нет необходимости в установлении и разрыве соединения
- скорость доставки важнее гарантированности доставки (например, при передаче телефонного разговора)

Типичный пример использования этого протокола - в DNS. При отсутствии ответа ничего не стоит отправить запрос повторно.

TCP

Подразумевает подтверждение доставки каждого пакета и таким образом условно гарантирует доставку. Если пакет не был доставлен, то будет осуществлена повторная передача с некоторым таймаутом. Все последующие пакеты будут ожидать этот таймаут.
Протокол использует механизм плавающего окна (подробнее см. *Компьютерные сети. Таненбаум Э.*)

Установление и разрыв соединения



Установление соединения происходит следующим образом:

1. Узел А запрашивает установление соединение с узлом Б
2. Узел Б подтверждает соединение
3. Узел А окончательно подтвержает установление соединения. Этот шаг необходим, чтобы узел Б тоже убедился в установлении соединения, так как SYN мог придти от поддельного IP-адреса. Чтобы этого не произошло, передатчик должен подтвердить подтверждение.

Разрыв соединения происходит аналогичным образом.
Итого, для отправки одного пакета с полезными данными по TCP потребуется передать минимум 8 пакетов.
MSS (Maximum Segment Size) - максимальный размер TCP сегмента, который будет использован для передачи. Фактически, это подмена MTU, т.к. в нём содержится минимальный MTU между узлами. Делается это для того, что избежать ICMP ответов “Fragmentation needed”, пакеты будут заранее фрагментироваться под минимальный MTU.
TSO (TCP Segment Offload) Формально отслеживание MSS, фрагментация пакетов и прочая нагрузка - задача ядра ОС. С современными скоростями эта нагрузка слишком велика. TSO переносит эту работу на уровень сетевого адаптера.

Порты

Исторически порт - это расширитель IP-адреса. Под порт в заголовке отводится 2 байта, следовательно диапазон возможных значений порта 0-65535.
Некоторым портам (и UDP и TCP) присвоены имена, фиксированием которых занимается организация IANA. Соответственно, существуют файлы с соответствиями имен сервисов и портов. В UNIX системах эта информация находится в файле /etc/services.
Формат файла может отличаться в зависимости от ОС; в Solaris он следующий:

```
ftp-data        20/tcp
ftp             21/tcp
ssh             22/tcp                          # Secure Shell
telnet          23/tcp
smtp            25/tcp          mail
time            37/tcp          timserver
time            37/udp          timserver
name            42/udp          nameserver
```

Первый столбец - имя сервиса, затем номер порта и протокол, для которого этот порт указан, далее может быть указан псевдоним (некторые порты могут иметь несколько имён)

Сервисы

Echo

Echo – сервис, отвечающий на порту 7, доступен как по протоколу TCP так и по протоколу UDP. Используя данный сервис можно послать пакет и получить обратно его абсолютную копию (отсюда и его название - “эхо”). Сервис полезен для проверки пересылки какой-либо специфичной последовательности байт в обе стороны. Так же может использоваться в качестве DDOS атак.

Discard

Discard – сервис на порту 9, позволяет открыть соединение TCP и все получаемые пакеты игнорируются, а для UDP позволяет принимать пакеты, но в ответ ни чего не отправляя.

Daytime

Daytime – сервис на порту 13, использовался для синхронизации времени с сервером.

CHARGEN

CHARGEN (Character Generator Protocol) - сервис, отвечающий на порту 19, доступен по протоколам TCP и UDP. При открытии TCP соединения начинает пересылать случайные символы до момента закрытия соединения. На UDP запрос отвечает UDP дейтаграммой, содержащей случайное число (от 0 до 512). Все входящие данные игнорируются.

Telnet

Telnet **–** протокол подразумевает под собой передачу служебной информации и символов. Не путать с утилитой telnet, реализующей клиент для этого протокола. При подключении по протоколу, в рамках служебного обмена информацией (не видимого пользователю), в первую очередь передается тип терминала, используемый с стороны клиента (для определения поддерживаемых управляющих последовательностей, чтобы пересылаемая информация правильно отображалась).

> Лирическое отступление: необходимо понимать, что под словом “терминал” в большинстве случаев подразумевается эмулятор терминала, эмулирующий поведение аппаратного терминала, которые настоящее время практически не используются. Ввиду того, что существовало много видов аппаратных терминалов, многие из них отличались поддерживаемыми управляющими последовательностями.

Как уже упоминалась выше, для работы с протоколом telnet может быть использована одноименная утилита, в качестве параметра которой передается адрес хоста.

```
> telnet helios
```

Недостатки протокола telnet:

- Отсутствие буферизации (каждый вводимый символ передается отдельным пакетом). Это может плохо сказаться в сетях с высокой задержкой передачи данных (например GPRS/EDGE, не гарантирующий время доставки пакета).
- Плохо работает в сетях с низкой скоростью передачи данных (не путать с вышеописанным случаем сетей с большими задержками).
- Отсутствует шифрование.

В настоящее время протокол telnet практически полностью вытеснен протоколом SSH.

Утилита telnet может работать также с “сырым” TCP соединением (помимо протокола telnet) к удаленному порту.

```
> telnet helios 21 # или, вместо номера порта, имя сервиса согласно /etc/services
```

При этом будет включена буферизация - символы будут отправляться только после нажатия “Enter”. Это позволяет проверять работу других сервисов, таких как HTTP, FTP и других human-readable протоколов.

rlogin и rsh

Другими способами войти в систему удаленно являются протоколы (и одноименные утилиты-клиенты) rlogin (англ. Remote login) и rsh (англ. Remote shell).
Отличие команды rlogin от telnet:

- По умолчанию у пользователя не спросят логин, а сразу предложат ввести пароль.
- Для rlogin возможно описать некоторый файл со списком хостов, с которых разрешен вход на сервер без пароля.

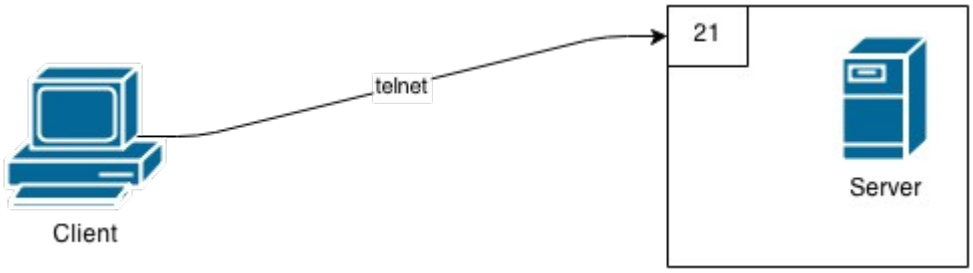
rsh может использоваться как rlogin, но впервую очередь предназначен для выполнения команд на удаленном сервере. Интерфейс команды ssh разрабатывался по образу и подобию интерфейса команды rsh.

FTP

FTP (*англ. File Transfer Protocol*) фактически первый протокол, созданный для передачи файлов в IP сетях.

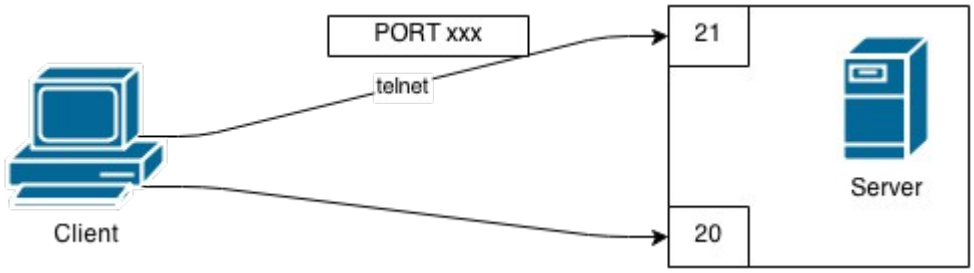
Порядок работает протокола FTP:

1. Клиент инициирует управляющее соединение на 21 порт по протоколу telnet. Протокол является human-readable, что в принципе позволяет используя утилиту telnet подключиться к FTP-серверу и управлять передачей файлов вручную. Набор поддерживаемых комманд в протоколе FTP зависит от реализации и как правило может быть получен коммандой HELP.

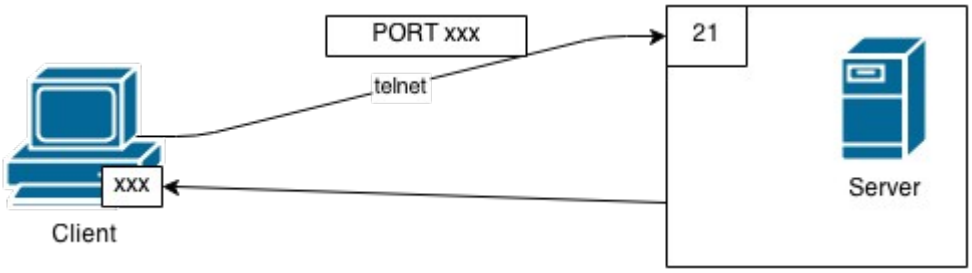


2. Для инициации передачи данных (файлов, списки файлов каталогов и т.п.), клиент передает серверу определенную команду по управляющему соединению, фактически описывающую каким образом передать эти данные клиенту. Передача может вестись в двух режимах:

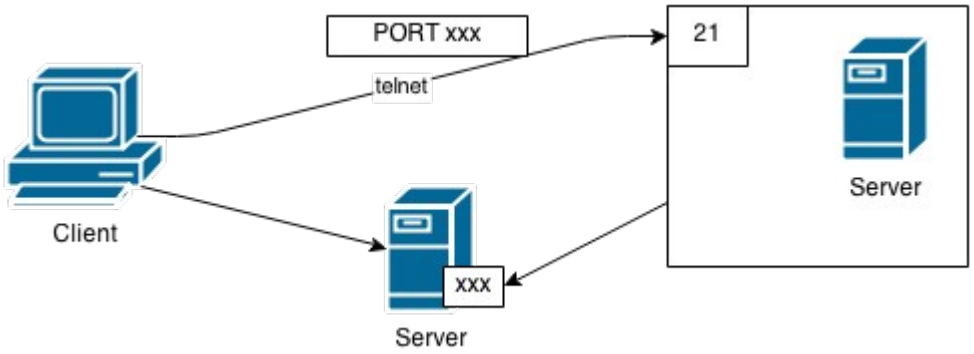
- Пассивном: клиент инициирует TCP соединение на 20 порт сервера и передает по нему нужную информацию.



- Активном: сервер инициирует TCP соединение на определенный порт клиента (указанный в команде переданной клиентом) и передает по нему файлы. Был создан в связи с наличием в старых системах ограничения на количество соединений на один порт.



Формально, используя пассивный режим, клиент также может организовать следующую схему передачи:



Недостатки активного режима:

- Активный режим проблематично использовать, если клиент “сидит за” NAT (хотя “умный” NAT умеет обрабатывать даже активное соединение).
- Активный режим не имеет смысла использовать, если сервер “сидит за” NAT, так как это фактически означает, что серверу необходимо предоставить разрешение на соединение с любым адресом по любому порту.
- Для каждого передаваемого файла создается новое TCP соединение (выглядит как DoS атака).

Поскольку управляющее соединение работает по протоколу telnet, передаваемые по нему пароль (необходимый для аутентификации) не шифруется, в следствии чего большинство имеющихся на данный момент FTP серверов используются для раздачи публичной информации, для доступа к которой используется “анонимный пользователь” с парой логин/пароль anonymous/anonymous или ftp/ftp.

Одной из особенностей протокола является тот факт, что FTP работает в текстовом режиме только с ASCII кодировкой.

> Лирическое отступление: в связи с этим, возникают проблемы, при использовании других кодировок. Так например в кодировке Windows-1251 (CP1251) символ с кодом 255 является значащим (незарезервированным) и значит символ “я” (я маленькое). А поскольку для управляющего соединения используется протокол telnet, в котором 255 символ является управляющим, FTP сервер увидев в команде этот символ, убирает его и следующий за ним символ. Так создание через FTP каталога с именем “Новая папка” завершится наличием каталога с именем “Новапапка” (убран символ “я” и следующим за ним символ пробела).

Протокол FTP умеет передавать данные в следующих режимах:

- ASCII. Для него FTP сервер автоматически преобразовывает символы переводы строки используемые на клиенте в те, что используются на сервере. В этой связи передача бинарных файлов в ASCII режиме может привести к “битым” файлам.
- бинарном.

Для использования FTP, конечно же, не стоит работать через утилиту telnet. Гораздо лучше воспользоваться каким либо клиентом (например одноименным клиентом ftp).

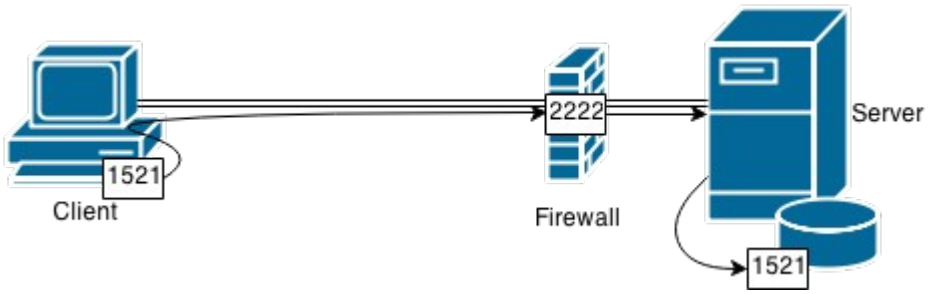
Преимущества FTP:
- Существует много разновидностей FTP-серверов.
- Малые накладные расходы на передачу данных (файлы передаются по TCP соединению в чистом виде без какой-либо модификации, проверки и т.п.).

TODO example

SSH

Протокол

SSH – сетевой протокол прикладного уровня, поддерживающий в отличии от вышеописанных протоколов шифрование и позволяющий производить удалённое управление операционной системой (аналогично telnet, rlogin и rsh) и туннелирование TCP-соединений.
Рассмотрим следующий пример проброса TCP-соединения.



Пусть удаленный компьютер желает подключить SQL developer к базе данных Oracle, расположенной на сервере, скрытом firewall'ом. На firewall'е настроен проброс TCP-соединений с портом 2222 к серверу. Oracle на сервере слушает на порту 1521.

```
> ssh -L 1521:localhost:1521 -p 2222 username@server
```

Ключ `-L` означает переназначение локального TCP порта. В данном случае SSH начнет слушать локальный порт 1521. При попытке открыть TCP соединение с ним, SSH через свой туннель перебросит его на localhost:1521 (относительно сервера) - т.е. на порт который слушает Oracle.
После этого достаточно указать SQL developer в качестве адреса базы данных localhost:1521 (относительно клиента).

Обратный пример: проброс TCP-соединения на удаленный порт 1521 на ваш локальный порт 1521:

```
> ssh -R 1521:localhost:1521 -p 2222 username@server
```

Другие особенности протокола SSH:
- SSH позволяет в рамках одного соединения открыть несколько терминальных сессий.
- С помощью протокола SSH можно передавать файлы, используя утилиту scp.
  ```
  > scp aqua:/etc/services ./ # копируем файл /etc/services с aqua в текущий каталог
  ```
- Утилита sftp имитирует интерфейс ftp, но передача файлов ведется все тем же SSH.
- SSH позволяет пробрасывать туннели для соединения с X-сервером.
- SSH поддерживает компрессию.

Главным недостатком SSH являются большие накладные расходы (в плане процессорного времени), в связи с тем, что ssh является шифрованным протоколом и шифрует все отправляемые данные.

Клиент

Интерфейс команды ssh частично унаследован от команды rsh.

Удаленное выполнение команды:

```
> ssh aqua ls
```

Ввод имени пользователя:

```
> ssh –l {имя пользователя} {имя сервер} {команда}
```

В большинстве случаев используют сокращенный вариант:

```
> ssh {имя пользователя}@{имя сервер}
```

У ssh также существуют Escape-последовательность: ~. (тильда точка). Данная последовательность разрывает ssh соединение с сервером. Escape-последовательность модет быть изменена с помощью соответствующего ключа:

```
> ssh –e {escape символ} {имя пользователя}@{имя сервер}
```

NAT

(0:1:07@3)

Inetd

Для того чтобы при прослушивании множества портов (каждым сервисом) минимизировать накладные расходы (используемую память и процессорное время) был создан демон inetd. Демон inetd, в соответствии со своим конфигурационным файлом, слушает определенные TCP и UDP порты и при появлении соединения на некотором порту запускает нужную программу. Таким образом, постоянно в системе с целью прослушивания портов работает только один демон (inetd), а процессы обслуживающие появляющиеся соединения будут появляться по мере необходимости.

Конфигурационный файл находиться в /etc/inetd.conf.
Пример записи в конфигурационном файле:

```
telnet  stream  tcp6    nowait  root    /usr/sbin/telnetd      telnetd -a
```

Где столбцы слева на право содержат:

1. Имя сервиса, описанное в /etc/services
2. Тип сокета (указаны только наиболее часто использующиеся):
    - stream - для протоколов ориентированных на TCP соединение
    - dgram - для UDP сервисов
3. Используемое семейство протоколов (tcp, udp, tcp6 или udp6 - последние два для IPv6)
4. wait/nowait
    - wait - inetd дожидается окончания работы этого сервера, чтобы обслужить следующее соединение. Используется для дейтаграммных протоколов (подразумевается, что поступивший по UDP пакет послужит началом обмена некоторым количеством пакетов между сервером и клиентом. Соответственно завершать процесс сервера после обработки каждого пакета не имеет смысла, поскольку с высокой долей вероятности его придется снова запускать и тратить на это ресурсы системы);
    - nowait - inetd запустит обслуживающий сервер в фоновом режиме и, не дожидаясь завершения его работы, вернется к обслуживанию следующих соединений. Используется для TCP соединение (подразумевается, что после установления соединения сервер работает с ним продолжительное время).
5. user[:group] - пользователь (и опционально группа) от имени которой будет запущена программа сервер.
6. Полный путь к исполняемому файлу программы сервера.
7. Перечисление аргументов, которые буду переданы программе (включая нулевой аргумент - имя программы - для системного вызова exec(). Иногда может отличаться от стандартного, например: /bin/sh с аргументом “-sh”, который говорит системе что необходимо войти в shell с выполнением всех профилей).

При появлении соединения выполняет:

1. системный вызов vfork()
2. устанавливает в качестве stdin и stdout для нового процесса устанавливается появившееся соединение (т.е. фактически программа может ничего не знать о TCP и UDP, поскольку просто работает со стандартным входом и выходом).
3. системный вызов exec()

inetd является потенциальной уязвимостью в системе, поскольку по умолчанию процессы серверов запускаются от root’а, а одной из программ умеющих работать с stdin и stdout является sh.
Недостатками inetd является тот факт, что на каждое новое соединение выполняет два “тяжелых” системных вызова - vfork() и exec(). Поэтому при наличии высокой нагрузки на какой-либо сервис inetd будет только нагружать систему лишними системными вызовами.

DNS

Введение

Помимо IP-адресов, хосты идентифицируются доменными именами, более легкими для запоминания и отражающими логическое структурирование сети и часто функциональное назначение того или иного хоста. Первоначально для преобразования символьных имен в IP адреса использовался файл `/etc/hosts`.

```
#
# Internet host table
#
127.0.0.1     localhost
192.168.10.10    helios  loghost helios.cs.ifmo.ru
```

Комментарий начинается с решетки (`#`). Запись начинается с IP адреса, после чего через пробел следуют имена для данного IP адреса. В большинстве ОС сначала проверяется этот файл, а потом уже опрашивается DNS. Это является источником ряда уязвимостей (Легко направить пользователя на ненастоящий сайт добавив в файл запись с соответствующим именем и IP адресом с вредным контентом).

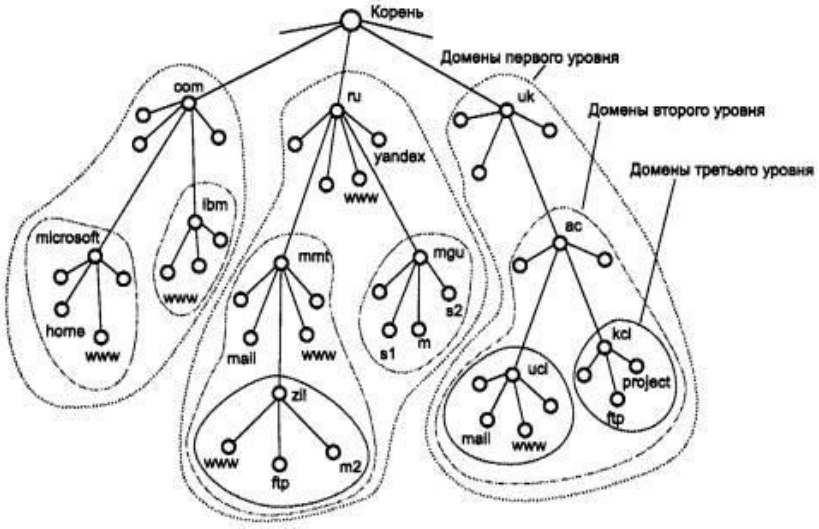
Распространение этого файла по сети весьма проблематично (особенно во времена зарождения Интернета), что стало причино появления системы *DNS (Domain Name System)*, занимающейся преобразованием доменных имён в IP-адреса и обратно. Помимо вышеуказанных преобразований DNS также выполняет ряд второстепенных функций (например, получения информации о маршрутизации почты, об обслуживающих узлах для протоколов в домене и др.) Протокол DNS использует для работы TCP- или UDP-порт 53 для ответов на запросы. Традиционно запросы и ответы отправляются в виде одной UDP дейтаграммы.

Особенности DNS

К первой особенности DNS можно отнести *децентрализованность управления.* Это значит, что почти каждый пользователь может зарегистрировать домен, который будет находиться на его личном сервере имён, при этом этот сервер имён будет обслуживать эту зону и в то же время станет частью глобальной службы DNS.

Второй особенностью является *ориентированность на отказоустойчивость,* то есть изначально предусмотрено наличие дублирования и резервирования данных серверов. Крайне редко администратор будет разворачивать лишь один сервер имён. Для обслуживания зоны их поднимают, как минимум, два, так что выход одного из серверов имён из строя почти никак не скажется на работе пользователей.

Структура DNS



Представление иерархии доменных имён в виде дерева.

DNS имеет древовидную структуру. Корневой домен обозначается зарезервированным пустым (нулевой длинны) именем. Поскольку каждый домен отделяется точкой, корневой домен выглядит как отдельно стоящая точка (**.**).

Иерархия доменных имен аналогична иерархии имен файлов, принятой во многих файловых системах. Каждому узлу дерева соответствует текстовая метка, длина которой не может превышать 63 символа, причем использование символа точки недопустимо. Как мы уже сказал, пустая (нулевой длины) метка зарезервирована для корневого домена. *Полное доменное имя* произвольного узла дерева - это последовательность меток в пути от этого узла до корня.

Полное доменное имя известно также под названием *абсолютного доменного имени*, обозначаемого аббревиатурой *FQDN* (*fully qualified domain name*). Оно записывается относительно корня и однозначно определяет расположение узла в иерархии. Имена без завершающей точки иногда интерпретируются относительно некоторого доменного имени (не обязательно корневого) точно так же, как имена каталогов, не начинающиеся с символа « / », часто интерпретируются относительно текущего каталога.

Доменные имена всегда читаются от собственно узла к корню («вверх» по дереву), причем метки разделяются точкой. Теоретически такое деление может достигать глубины 127 уровней. Если метка корневого узла должна быть отображена в доменном имени, она записывается как символ точки, например, так: **www.ifmo.ru.** (На самом деле имя заканчивается точкой-разделителем и пустой меткой корневого узла.) В результате некоторые программы интерпретируют имена доменов, заканчивающиеся точкой, как абсолютные.

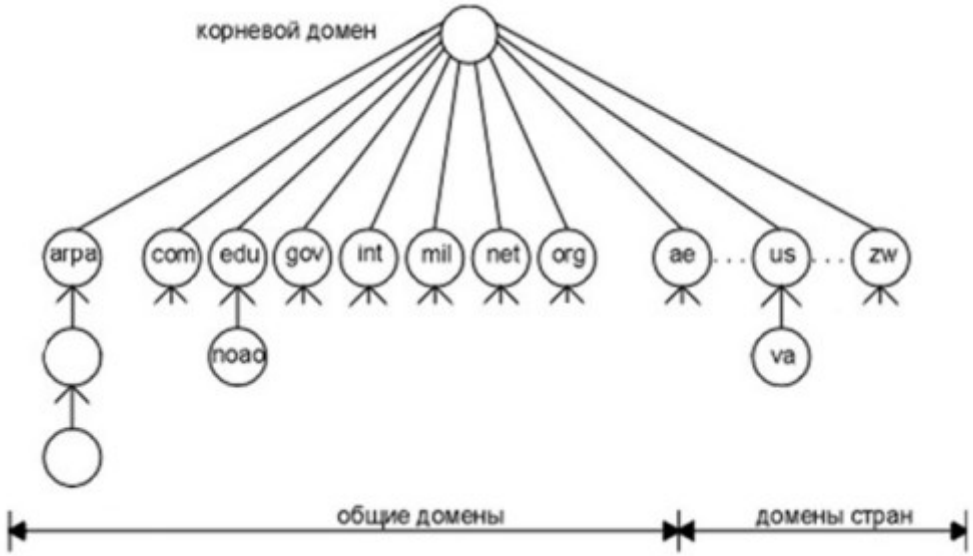
Совокупность имен, у которых несколько старших составных частей совпадают, образует домен имен (domain). Например, имена www.ifmo.ru, www.cs.ifmo.ru, yandex.ru входят в домен ru, так как все эти имена имеют одну общую старшую часть - имя ru.

Если один домен входит в другой домен как его составная часть, то такой домен могут называть поддоменом (subdomain), хотя название домен за ним также остается. Обычно поддомен называют по имени той его старшей составляющей, которая отличает его от других поддоменов. Например, поддомен mmt.ru обычно называют поддоменом (или доменом) mmt. Имя поддомену назначает администратор вышестоящего домена.

В DNS узлы, имеющие общего родителя, должны иметь разные метки. Такое ограничение гарантирует, что доменное имя единственно возможным образом идентифицирует отдельный узел дерева. Это ограничение на практике не является ограничением, поскольку метки должны быть уникальными только для братских узлов одного уровня, но не для всех узлов дерева.

В Internet корневой домен управляется центром InterNIC. Домены верхнего уровня назначаются для каждой страны, а также на организационной основе. Имена этих доменов должны следовать международному стандарту ISO 3166. Например, для различных типов организаций используются следующие названия доменов первого уровня:

com - коммерческие организации (например, microsoft.com);
edu - образовательные (например, mit.edu);
gov - правительственные организации (например, nsf.gov);
org - некоммерческие организации (например, fidonet.org);
net - организации, поддерживающие сети (например, nsf.net).



Каждый домен администрируется отдельной организацией, которая разбивает свой домен на поддомены и передает функции администрирования этих поддоменов другим организациям.

DNS-сервер

DNS-сервер - специализированное ПО для обслуживания DNS, а также компьютер, на котором это ПО выполняется. DNS-сервер может быть ответственным за некоторые зоны и/или может перенаправлять запросы вышестоящим серверам.

Для каждого домена имен создается свой DNS-сервер. Этот сервер может хранить отображения «доменное имя - IP-адрес» для всего домена, включая все его поддомены. Однако такое решение плохо масштабируется, так как при добавлении новых поддоменов нагрузка на этот сервер может превысить его возможности. Чаще сервер домена хранит только имена, которые заканчиваются уровнем ниже по сравнению с именем домена (аналогично каталогу файловой системы, который содержит записи о файлах и подкаталогах, непосредственно в него «входящих»).

Именно при такой организации службы DNS нагрузка по разрешению имен распределяется более-менее равномерно между всеми DNS-серверами сети. Например, в первом случае DNS-сервер домена ifmo.ru будет хранить отображения для всех имен, заканчивающихся на ifmo.ru: www.cs.ifmo.ru и др. Во втором случае этот сервер хранит отображения только имен типа www.ifmo.ru, а все остальные отображения должны храниться на DNS-сервере поддомена cs.

Каждый DNS-сервер кроме таблицы отображений имен содержит ссылки на DNS-сервера своих поддоменов. Эти ссылки связывают отдельные DNS-сервера в единую службу DNS. Ссылки представляют собой IP-адреса соответствующих серверов. Для обслуживания корневого домена выделено несколько дублирующих друг друга DNS-серверов, IP-адреса которых являются широко известными (их можно узнать, например, в InterNIC).

Клиенты DNS (resolvers) - позволяют осуществлять доступ к DNS-серверам. Программы, которым требуется информация из пространства доменных имен, используют DNS-клиент, решающий следующие задачи:

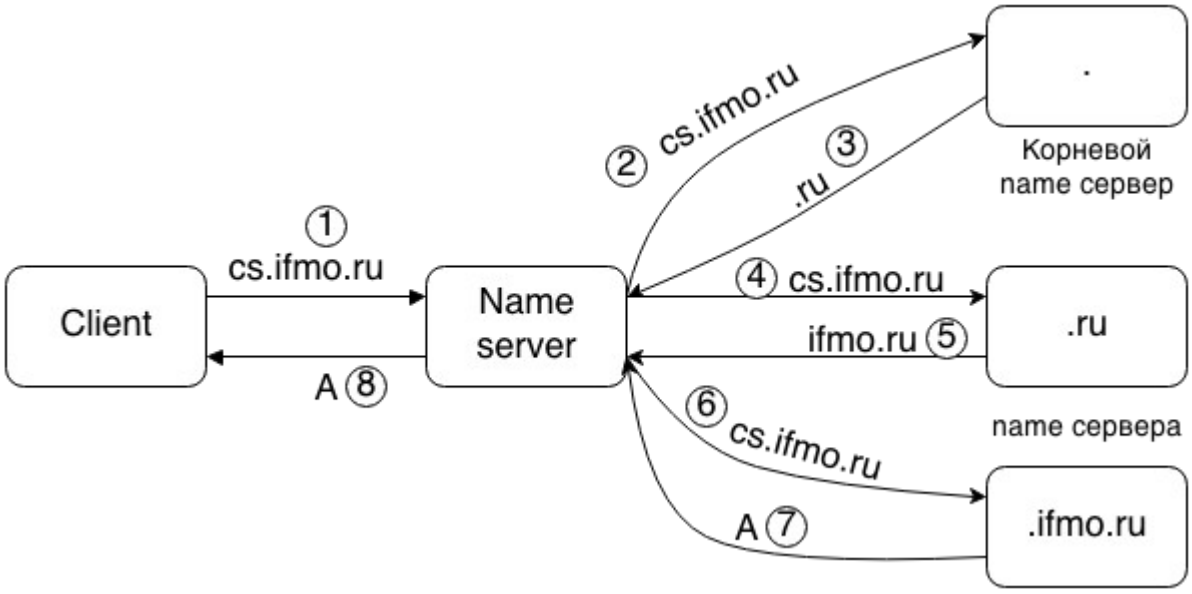
- Опрашивание DNS-серверов
- Интерпретация полученных ответов (Resource Records-записей или сообщений об ошибках)
- Возврат информации в программу, которая ее запросила

Схемы разрешения DNS-имен

Существуют две основные схемы разрешения DNS-имен - *рекурсивная* и *итеративная (нерекурсивная)*.

Рекурсивная схема

При использовании этой схемы клиент перекладывает работу по получению информации на DNS сервер. DNS сервер при этом работает в роли клиента и отправляет серию итерационных запросов на другие сервера DNS.

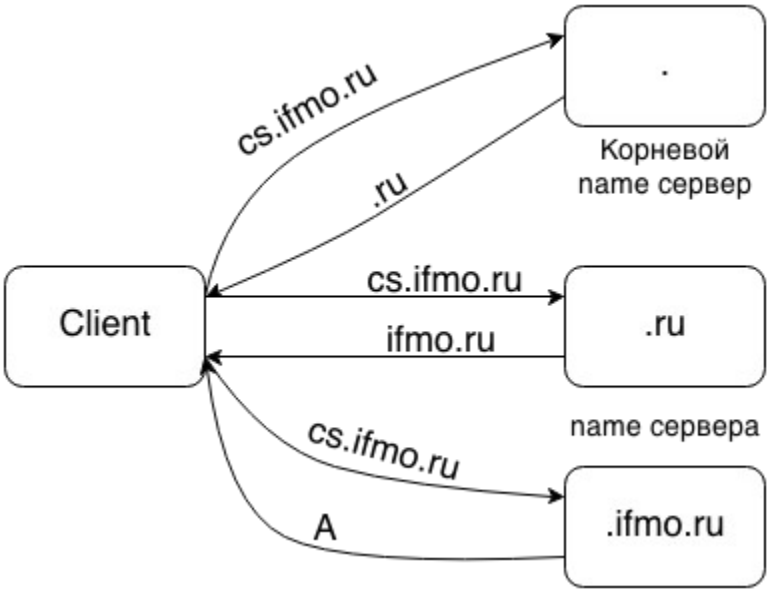


Допустим, пользователь ввел в качестве URL адрес cs.ifmo.ru. Тогда произойдет следующая цепочка событий:

1. Запрос на преобразование названия домена cs.ifmo.ru передается на предпочтительный сервер DNS (preferred DNS server), заданный на рабочей станции. Скорее всего в кэше предпочтительного сервера DNS уже содержится запрашиваемая запись и она сразу же вернется. Рассмотрим ситуацию, когда на сервере DNS нет информации относительно домена cs.ifmo.ru. Итак, предпочтительный сервер DNS на рабочей станции не знает IP адреса cs.ifmo.ru и IP адрес сервера DNS, который отвечает за домен cs.ifmo.ru. Ему известен лишь IP адрес DNS сервера корневого уровня (благодаря файлу с корневыми подсказками).
2. Предпочтительный сервер DNS передает запрос корневому серверу DNS (root DNS server).
3. Корневой сервер DNS также не знает IP адрес веб-сервера *www.cs.ifmo.ru*. Он знает IP адрес сервера DNS, который отвечает за домен *.ru*. Корневой сервер root DNS server передает IP адрес сервера DNS, который отвечает за домен *.ru* предпочтительному серверу DNS.
4. Предпочтительный сервер DNS после этого передает клиентский запрос на сервер DNS домена .ru.
5. Сервер DNS домена *.ru* не знает IP адрес сайта *www.ifmo.ru*, но он знает IP адрес сервера DNS, который отвечает за домен *ifmo.ru*. Сервер домена *.ru* возвращает IP адрес сервера DNS, который отвечает за домен *ifmo.ru* предпочтительному серверу DNS.
6. Предпочтительный сервер DNS клиента, затем посылает запрос DNS серверу домена *ifmo.ru*.
7. у DNS сервера домена ifmo.ru есть запись про cs.ifmo.ru, он возвращает соответствующий IP адрес.
8. Полученный адрес передается клиенту, который его запросил.

В этом примере необходимо обратить внимание на две вещи. Во-первых, клиент совершает лишь один запрос DNS. Он абсолютно ничего не знает об итеративных запросах сервера DNS. Во-вторых, сервер DNS, который отвечает за домен cs.ifmo.ru вовсе не обязательно принадлежит ifmo. Этот сервер DNS принадлежит компании, занимающейся веб-хостингом (Web hosting company) и отвечает за все сайты, принадлежащие этой компании. Именно поэтому предпочтительный сервер DNS не может пропустить один шаг и сразу передать клиенту адрес DNS сервера, который отвечает за домен, по крайней мере, в нашем случае.
Если сервер DNS не поддерживает рекурсивные очереди (recursive queries), то клиент по умолчанию будет выполнять итеративные запросы (iterative queries).
Для ускорения поиска IP-адресов DNS-сервера широко применяют процедуру кэширования проходящих через них ответов. Чтобы служба DNS могла оперативно отрабатывать изменения, происходящие в сети, ответы кэшируются на определенное время - обычно от нескольких часов до нескольких дней.

Итеративная схема
В этом варианте работу по поиску IP-адреса координирует DNS-клиент, который сам итеративно выполняет последовательность запросов к разным серверам имен:



- DNS-клиент обращается к корневому DNS-серверу с указанием полного доменного имени;
- DNS-сервер отвечает, указывая адрес следующего DNS-сервера, обслуживающего домен верхнего уровня, заданный в старшей части запрошенного имени;
- DNS-клиент делает запрос следующего DNS-сервера, который отсылает его к DNS-серверу нужного поддомена, и т. д., пока не будет найден DNS-сервер, в котором хранится соответствие запрошенного имени IP-адресу. Этот сервер дает окончательный ответ клиенту.

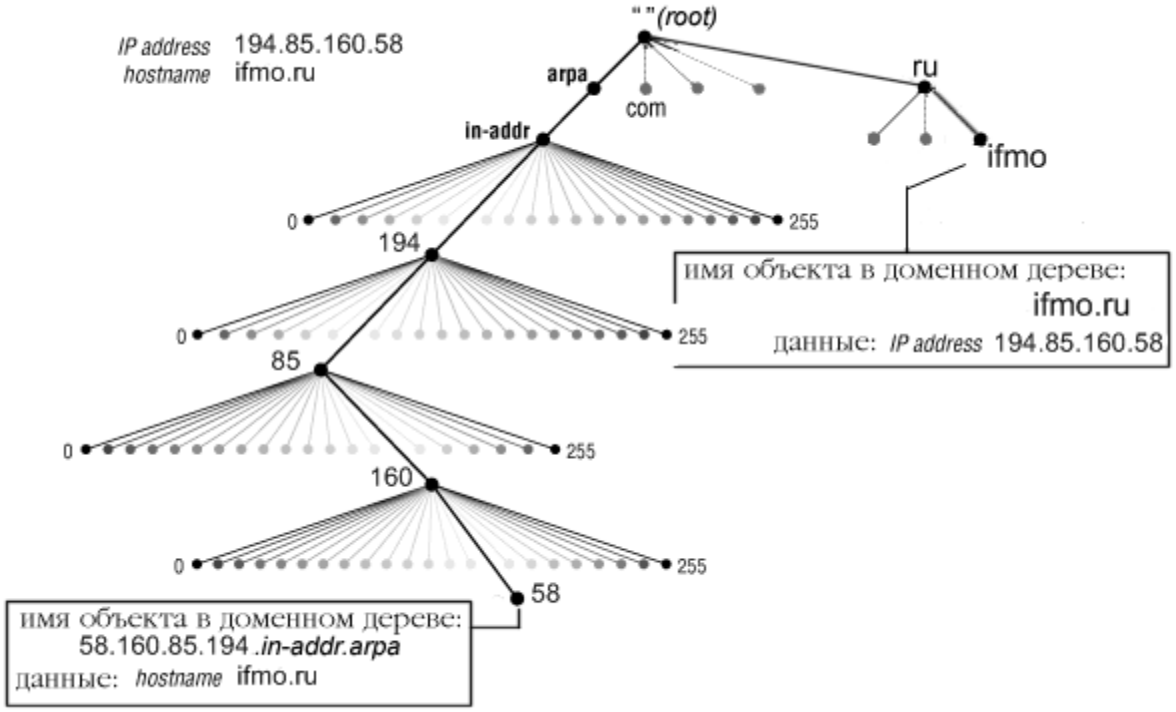
Так как эта схема загружает клиента достаточно сложной работой, она применяется редко.

Обратное преобразование

Технология этого преобразования, использующего специальный домен in-addr.arpa, следующая. Заметим, что всякое DNS-преобразование представляет собой поиск узла в дереве и возврат информации, связанной с этим узлом. Деревом здесь является иерархия доменных имен, причем о смысле и виде того или иного имени не делается никаких предположений (например, имя может не относиться ни к какому физическому хосту, а обозначать некое множество почтовых адресов).

Заметим далее, что пространство IP-адресов, записанных в десятично-точечной нотации, также представляет собой дерево (точнее, лес из 256 деревьев). Отличие от иерархии доменных имен одно: в доменном имени узлы, более близкие к корню, записываются справа, а в IP-адресе - слева. Иными словами, направление от общего к частному в первом случае справа налево, а во втором - слева направо.

Следовательно, можно взаимно однозначно отобразить пространство IP-адресов на специально образованное поддерево дерева доменных имен. Таким поддеревом является домен in-addr.arpa, в который в качестве поддоменов входят значения старшего октета IP-адреса. В каждом из этих поддоменов вида X.in-addr.arpa, где X=0..255, находится 256 поддоменов следующего уровня, представляющие значения второго справа октета IP-адреса - и так далее. Таким образом, IP-адрес 194.85.160.58 представлен узлом в доменном дереве, имеющем имя 58.160.85.194.in-addr.arpa, а сеть 194.85.160.0- доменом 160.85.194.in-addr.arpa (см. рис.).



Прямое и обратное преобразование в системе DNS

Смысл описанной схемы состоит в том, что таким образом обратное DNS-преобразование "адрес в имя" не является каким-то особенным преобразованием, а представляет собой обычное прямое преобразование в одном из поддоменов домена in-addr.arpa, при этом данными для каждого узла (т.е., фактически, IP-адреса) в этом домене выступает собственно доменное имя хоста. Например, данными для объекта 194.85.160.58.in-addr.arpa является строка "ifmo.ru", и для выполнения преобразования IP-адреса 58.160.85.194 в доменное имя сервер DNS выполняет поиск данных для "доменного имени" 194.85.160.58.in-addr.arpa и возвращает полученный результат.
Поиск производится по обычному алгоритму: сначала запрашивается корневой сервер, который, если у него в кэше нет готового ответа, возвращает адрес сервера, отвечающего за in-addr.arpa. Далее сервер in-addr.arpa (при отсутствии готового ответа) возвращает адрес сервера, отвечающего за 58.in-addr.arpa - и так далее.

Механизм зон

Деление доменного пространства имен между DNS-серверами осуществляется посредством механизма *зон (zone)*. Зона представляет собой базу данных, в которой содержатся записи о соответствии некоторого множества доменных имен IP-адресам. Каждая зона представляет собой фрагмент доменного пространства имен. Следует рассматривать зоны как основной административный элемент, на уровне которого происходит как управление пространством имен в целом, так и управление процессом разрешения имен. Зоны, используемые для размещения содержимого обратных доменов, называются зонами *обратного просмотра (reverse lookup zone).*
Границы зоны не определяются доменной структурой. Одна зона может включать в себя несколько доменов, в то время как объекты, принадлежащие к одному домену, могут быть размещены в

нескольких зонах. Осуществляя разделение доменного пространства имен на зоны, необходимо исходить, в первую очередь, из удобства администрирования.
Зону можно разместить на нескольких серверах. На каждом из вовлеченных DNS-серверов размещается отдельная копия зоны. Для поддержания этих копий в согласованном состоянии используется *модель репликации с одним основным участником (single-master replication)*. Один из DNS-серверов выступает в качестве *основного носителя зоны (primary zone)*. Только основной носитель зоны обладает возможностью вносить изменения в ее содержимое (имеет право на запись). Остальные DNS-сервера располагают копией зоны, доступной только для чтения. Эти сервера называются *дополнительными носителями зоны (secondary zone)*. Изменения, произведенные в копии зоны основным носителем, реплицируются на дополнительные носители.
Использование нескольких носителей зоны позволяет, с одной стороны, распределить нагрузку между несколькими серверами, а с другой стороны, реализовать некоторый уровень отказоустойчивости. В случае выхода из строя одного из носителей зоны разрешение имен будет осуществляться остальными носителями зоны. На каждом DNS-сервере может быть размещено несколько зон. В этом случае каждая зона конфигурируется отдельно. Один и тот же сервер может выступать как основным, так и дополнительным носителем для различных зон.
Для того чтобы связать файлы данных зоны, DNS-серверу требуется файл настройки - он обычно называется named.conf. Формат файлов данных зоны одинаков во всех реализациях DNS и называется форматом мастер-файла. Однако формат файлов настройки является специфичным для реализации DNS-сервера.

Ответственность за зоны

Ответственность - признак размещения зоны на DNS-сервере. Ответы DNS-сервера могут быть двух типов: ответственные (когда сервер заявляет, что сам отвечает за зону) и неответственные (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 кэширования).

Делегирование

Делегирование - операция передачи ответственности за часть дерева доменных имен другому лицу или организации. Организация, получившая домен таким путем, несет ответственность за работу с данными этого поддомена. Она может свободно изменять эти данные, разделять поддомен на несколько более мелких поддоменов, а те, в свою очередь, снова делегировать. Родительский домен сохраняет только указатели на источники данных для поддоменов в целях перенаправления запросов по соответствующим адресам.

При делегирование поддоменов происходит назначение различных DNS-серверов в качестве авторитетных в делегируемых поддоменах. Хранимая зоной информация после делегирования включает уже не информацию по делегированному поддомену, а информацию о серверах имен, являющихся для этого поддомена авторитетными. В таком случае, если у DNS-сервера родительского домена запрашиваются данные для поддомена, в ответ предоставляется список DNS-серверов, которые обладают соответствующей информацией.
За счет делегирования в DNS обеспечивается распределенность администрирования и хранения.

Записи ресурсов

Информация, связанная с доменными именами содержится в записях ресурсов (RRs, resource records). Записи разделяются на классы, каждый из которых определяет тип сети или программного обеспечения. В пределах класса записи делятся на типы, которые соответствуют различным видам данных, хранимых в пространстве доменных имен. В различных классах определяются различные типы записей, хотя некоторые типы могут являться общими для нескольких классов. Так, практически в каждом классе определен тип адрес. Каждый тип записи в конкретном классе определяет формат, который должны соблюдать все RR-записи, принадлежащие этому классу и имеющие данный тип.

Основные типы DNS-записей

- **A** (address record) – запись адреса связывает имя хоста с адресом IPv4. Например, запрос A-записи на имя referrals.icann.org вернет его IPv4 адрес — 192.0.34.164.
- **AAAA** (IPv6 address record) – связывает имя хоста с его адресом по протоколу IPv6.

- **CNAME** (canonical name record) – каноническая запись имени указывает псевдоним для официального имени хоста. Когда DNS клиент запрашивает IP адрес этого типа, то он получает IP адрес, прописанный в записи, к которой сделана привязка.
- **MX** (mail exchange) или почтовый обменник указывает сервер(ы) обмена почтой для данного домена.
- **NS** (name server) указывает на DNS-сервер для данного домена.
- **PTR** (pointer) запись указателя связывает IP хоста с его каноническим именем. Запрос в домене in-addr.arpa на IP хоста в reverse форме вернёт имя (FQDN) данного хоста. Например, для IP адреса 192.0.34.164: запрос записи PTR 164.34.0.192.in-addr.arpa вернет его каноническое имя referrals.icann.org.
- **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) кэширования зонной информации и взаимодействия DNS-серверов.
- **SRV** (server selection) указывает на сервера для сервисов, используется, в частности, для Jabber и Active Directory.

WHOIS

WHOIS - это сетевой протокол, основанный на TCP, который широко используется для запросов к базам данных WHOIS. С его помощью можно проверить занятость доменного имени или IP-адреса и получить общедоступные сведения о его владельце.
WHOIS может поставлять информацию о сетях, о структуре доменов и т.д. Главная база данных, относящихся к сетям, поддерживается Регистрационной службой Интернет (InterNic).

Обращение к местному клиент-серверу производится по форме:

```
WHOIS <-h имя_сети> идентификатор
```

где имя_сети - адрес домена, куда вы собираетесь послать запрос (например, whois.internic.net); идентификатор - фамилия человека, название сети или домена, IP-адрес. С идентификатором могут использоваться специальные символы, определяющие тип поиска.

В общем виде обращение к серверу WHOIS может иметь вид:

```
whois [-aAbdgiIlmQrR6] [-c country-code | -h host] [-p port] name …
```

где буквы в квадратных скобках являются обозначениями возможных опций.

Примеры:
Запрос о доменном имени:

```
>whois ifmo.ru
```

Ответ:

```
domain:        IFMO.RU
nserver:       ns2.ifmo.ru. 194.85.160.54
nserver:       ns.ifmo.ru. 194.85.160.51
nserver:       ns.runnet.ru.
state:         REGISTERED, DELEGATED, VERIFIED
org:              FGBU OUVPO "Saint-Petersburg State University of Information Technologies,
Mechanics and Optics"
registrar:     RU-CENTER-REG-RIPN
admin-contact: https://www.nic.ru/whois
created:       1997.09.29
paid-till:     2014.10.01
free-date:     2014.11.01
source:        TCI

Last updated on 2014.01.25 22:31:36 MSK
```

Для получения актуальных данных рекомендуется использовать официальный whois-сервер для зоны, например:

```
whois -h whois.nic.ru ifmo.ru
```

nslookup

nslookup – утилита для диагностики служб DNS. Утилита проверяет работоспособность DNS-сервера путём отправки ему некоторых запросов.

При возврате записи отображается один из двух типов ответов сервера преобразования имен:

- Авторитетный ответ — сервер DNS содержит у себя запись для этого узла
- Неавторитетный ответ — сервер DNS получил информацию об этой записи от другого сервера

Синтаксис:

```
nslookup [-<подкоманда>] [узел] [-<сервер имён>]
```

где подкоманда (используются в интерактивном режиме работы):

- `exit` – выход из nslookup
- `ls` – отображение содержимого определённой зоны на сервере DNS. Если зона не настроена на передачу данных системе, утилита выдаёт ответ «refused» .
- `lserver` – позволяет подключиться к указанному DNS-серверу
- `server` – то же самое, что lserver, только для поиска адреса нового сервера DNS используется текущий принятый по умолчанию DNS-сервер
- `view` – позволяет смотреть файл, в который был направлен вывод nslookup.
- `set` – позволяет настроить способ запроса и получения записей для текущего сеанса nslookup

Пример:

```
>nslookup ifmo.ru

Server:         192.168.10.1
Address:        192.168.10.1#53

Non-authoritative answer:
Name:   ifmo.ru
Address: 194.85.160.58
```

Запрос:

```
nslookup 194.85.160.58
```

Ответ:

```
Server:         192.168.10.1
Address:        192.168.10.1#53

Non-authoritative answer:
58.160.85.194.in-addr.arpa      name = forest.ifmo.ru.

Authoritative answers can be found from:
160.85.194.in-addr.arpa nameserver = ns2.ifmo.ru.
160.85.194.in-addr.arpa nameserver = ns.runnet.ru.
160.85.194.in-addr.arpa nameserver = ns.ifmo.ru.
ns.ifmo.ru      internet address = 194.85.160.51
ns.runnet.ru    internet address = 194.85.32.18
ns2.ifmo.ru     internet address = 194.85.160.54
```

Конфигурационные файлы

/etc/resolv.conf

Файл resolv.conf определяет настройки DNS-клиента. Может содержать поля нескольких типов:

- **nameserver** – адрес удалённого DNS-сервера. Возможно несколько записей. В случае отсутствия таких записей, все обращения будут поступать на локальный сервер.
- **domain** – домен, в котором находится данная машина.

- **search** – список доменов для поиска. Если имя, которое требует разрешения, не является FQDN (Fully Qualified Domain Name), то домены из search будут подставляться к нему в качестве окончаний.

Пример файла:

```
search cs.ifmo.ru
nameserver 147.11.1.11
nameserver 147.11.100.30
```

/etc/named.conf

Конфигурационный (загрузочный) файл named.conf содержит информацию обо всех зонах, для которых DNS-сервер является первичным или вторичным. Помимо имени зоны, в первом случае указывается файл, в котором содержится база данных DNS для данной зоны, во втором - адрес первичного сервера и файл временного хранения базы данных, полученной от первичного сервера.

В число зон входят:

- прямая зона
- обратная зона
- зона локальной петли

Также в конфигурационном файле указывается файл инициализации кэша и каталог, в который помещены все файлы с данными.

Разделы

acl

acl (access control list) - позволяет задать именованный список сетей.
Формат раздела:

```
acl "имя_сети" {ip; ip; ip; };
```

Options задает глобальные параметры конфигурационного файла, управляющие всеми зонами. Данный раздел имеет формат: options {операторы_раздела_Options};.
Options может быть "вложен" в раздел Zone, при этом он переопределяет глобальные параметры. Часто используемые операторы options:

- allow-query [список_ip] - Разрешает ответы на запросы только из список_ip. При отсутствии - сервер отвечает на все запросы.
- allow-recursion [список_ip] - На запросы из список_ip будут выполняться рекурсивные запросы. Для остальных - итеративные. Если не задан параметр, то сервер выполняет рекурсивные запросы для всех сетей.
- allow-transfer [список_ip] - Указывает список серверов, которым разрешено брать зону с сервера (в основном тут указывают slave сервера)
- directory /path/to/work/dir - указывает абсолютный путь к рабочему каталогу сервера. Этот оператор допустим только в разделе options.
- forwarders [ip порт, ip порт…] - указывает адреса хостов и если нужно, порты, куда переадресовывать запросы (обычно тут указываются DNS провайдеров ISP).
- forward [ONLY|FIRST] - параметр first указывает, DNS-серверу пытаться разрешать имена с помощью DNS-серверов, указанных в параметре forwarders, и лишь в случае, если разрешить имя с помощью данных серверов не удалось, то будет осуществлять попытки разрешения имени самостоятельно.
- notify [YES|NO] - YES - уведомлять slave сервера об изменениях в зоне, NO - не уведомлять.
- recursion [YES|NO] - YES - выполнять рекурсивные запросы, если просит клиент, NO - не выполнять (только итеративные запросы). Если ответ найден в кэше, то возвращается из кэша (может использоваться только в разделе Options).

Zone определяет описание зон(ы).
Формат раздела: zone {операторы_раздела_zone};
Операторы, которые наиболее часто используются:

- allow-update {список_ip} - указывает системы, которым разрешено динамически обновлять данную зону.
- file "имя_файла" - указывает путь файла параметров зоны (должен быть расположен в каталоге, определенном в разделе options оператором directory)
- masters {список_ip} -указывает список мастер-серверов. (допустим только в подчиненных зонах)
- type "тип_зоны" - указывает тип зоны, описываемой в текущем разделе, тип_зоны может принимать следующие значения:
    - forward - указывает зону переадресации, которая переадресовывает запросы, пришедшие в эту зону.
    - hint - указывает вспомогательную зону (данный тип содержит информацию о корневых серверах, к которым сервер будет обращаться в случае невозможности найти ответ в кэше)
    - master - указывает работать в качестве мастер сервера для текущей зоны.
    - slave - указывает работать в качестве подчиненного сервера для текущей зоны.

Дополнительные параметры конфигурации

Значения времени в файлах зон по умолчанию указывается в секундах, если за ними не стоит одна из следующих букв: S - секунды, M - минуты, H- часы, D - дни, W - недели. Соответственно, запись 2h20m5sбудет иметь значение 2 часа 20 минут 5 секунд и соответствовать 8405 секунд.

Любое имя хоста/записи, не оканчивающиеся точкой считается не FQDN именем и будет дополнено именем текущей зоны. Например, запись domen в файле зоны examle.com будет развернуто в FQDN-имя domen.examle.com. .

В конфигурационных файлах DNS-серверов (например, BIND) могут применяться следующие директивы:

- $TTL - определяет TTL по умолчанию для всех записей в текущей зоне.
- $ORIGIN - изменяет имя зоны с указанного в файле named.conf. При этом, область действия данной директивы не распространяется "выше" (то есть если файл включен директивой $INCLUDE, то область действия $ORIGN не распространяется на родительский)
- $INCLUDE - включает указанный файл как часть файла зоны.

Пример файла:

```
options {
 directory "/usr/named";
};

zone "vvsu.ru" {
   type master;
   file "vvsu.hosts";
};

zone "195.16.212.IN-ADDR.ARPA" {
    type master;
    file "vvsu-195.rev";
};

zone "197.16.212.IN-ADDR.ARPA" {
    type master;
    file "vvsu-197.rev";
};

zone "vsue.ru" {
    type slave;
    masters { 212.16.198.4; };
```

```
    file "vsue.hosts";
};

zone "198.16.212.IN-ADDR.ARPA" {
    type slave;
    masters { 212.16.198.4; };
    file "vsue198.rev";
};

zone "0.0.127.in-addr.arpa" {
    type master;
    file "local.rev";
};

zone "." {
    type hint;
    file "root.cache";
};

// однострочный комментарий
/* многострочный
     комментарий        */
```

Файл зоны

Зона должна быть описана в файле. Один и тот же файл зоны можно использовать для нескольких зон. Такое решение может быть полезно, например, при регистрации нескольких созвучных доменов для одного и того же ресурса во избежание фишинга. Содержимое файлов зон будет одинаковое, поэтому необходимость в разных файлах отпадает. Несущественный недостаток в наличии лишних записей, которые не будут спрашиваться (объяснено ниже). Более важный недостаток - при таком описании зон нельзя использовать механизмы динамического обновления зон, это будет приводить к потере данных. Вот пример такого файла:

```
$TTL 1H

@     IN    SOA   ns.cs.ifmo.ru. noc.cs.ifmo.ru. (
                  20140129
                  3600
                  600
                  1209600
                  600 )

            NS    ns.cs.ifmo.ru.
            NS    ns2.cs.ifmo.ru.

            MX    10 mx.cs.ifmo.ru.
            A     192.168.10.1

ns          A     192.168.10.2
2           PTR   ns.cs.ifmo.ru.

ns2         A     192.168.10.3
3           PTR   ns2.cs.ifmo.ru.

www         A     192.168.10.1
1           PTR   www.cs.ifmo.ru.

mx          A     192.168.10.4
4           PTR   mx.cs.ifmo.ru.

$GENERATE  128-254    dhcp-10-$  A     192.168.10.$
$GENERATE  128-254    $          PTR   dhcp-10-$.cs.ifmo.ru.
```

Рассмотрим подробнее каждую часть файла.

Директива TTL

```
$TTL 1H
```

Эта директива указывает name server'у, что по умолчанию для записей использовать TTL (time to live - то есть на сколько данная запись кешируется кеширующими name server'ами) 1 час. Значение выбирается в стремлении компромисса между временем обновления кешей у клиентов и, с другой стороны, нагрузкой на name server.

Запись SOA

Служит для описания основной информации о зоне.

```
@ IN SOA ns.cs.ifmo.ru. noc.cs.ifmo.ru. (
     20140129;  Serial number
     3600;      Refresh time
     600;       Retry time
     1209600;   Expire time
     600 );     NACK TTL
```

где:

- **@** - имя текущей зоны (если файл читается для cs.ifmo.ru, то будет подставлено cs.ifmo.ru. если файл будет читаться для зоны 10.168.192.in-addr-.arpa, то будет подставлено имя этой обратной зоны). Таким образом решается проблема описания SOA записи для двух разных зон.
- **IN** - класс записи. Возможные варианты:
  - IN - Интернет
  - CS - CSNET
  - CH - CHAOS
  - HS - Hesiod
- **SOA** - тип записи
- **ns.cs.ifmo.ru.** - доменное имя primary master сервера зоны. Точка в конце обязательна, иначе имя будет преобразовано в "ns.cs.ifmo.ru.cs.ifmo.ru." Использовать просто "ns" в данном случае нельзя, так как для прямой зоны преобразование будет произведено корректно, но для обратной зоны получится "ns.10.168.192.in-addr-.arpa."
- **noc.cs.ifmo.ru** - почтовый адрес лица, осуществляющего администрирование зоны, в данном случае noc@cs.ifmo.ru. Так как символ "@" имеет особый смысл при описании зоны, то вместо этого символа в почтовом адресе используется символ "."
- **Serial number** - серийный номер зоны, который должен увеличиться при её изменении. В данном примере составлен в соответствии с рекомендацией формата ГГГГММДДВВ, однако это всего лишь рекомендация. Если бы здесь предполагалось использование механизма динамического обновления зон, то даже значение "0" являлось бы корректным, для корректной работы важно лишь увеличение серийного номера при изменении зоны.
- **Refresh time** - период опроса вторичным name server'ом для обновления информации. В примере время указано в секундах, однако здесь и далее можно использовать и иные форматы, например, 1h (1 час).
- **Retry time** - период опроса вторичным name server'ом в случае отсутствия ответа
- **Expire time** - интервал времени, после которого вторичный name server должен прекратить обслуживание запросов к зоне, если он не смог в течение этого времени верифицировать описание зоны, используя информацию с master сервера.
- **NACK TTL** - время кэширования негативных ответов, которые утверждают, что установить соответствие между доменным именем и IP-адресом нельзя. В отсутствии директивы TTL это значение используется в качестве TTL и сопровождается предупреждениями в логах.

Запись NS

Эта запись указывает name server данной зоны.

```
NS ns.cs.ifmo.ru.
```

В этой строке опущены имя записи и класс записи. При этом будут использованы значения из предыдущей строки. В примере используются 2 name server'а, поэтому и записей тоже две.

Запись MX

MX-записи для данного домена указывают серверы, на которые нужно отправлять электронную почту, предназначенную для адресов в данном домене.

```
MX 10 mx.cs.ifmo.ru.
```
где 10 - приоритет данного почтового сервера.

Записи A и PTR

Запись типа A указывает на IP-адрес ресурса по его имени, а запись типа PTR на имя ресурса по его IP-адресу.

```
ns A 192.168.10.2
2 PTR ns.cs.ifmo.ru.
```
Важно обратить внимание на то, что для записей типа A используется короткое имя (так как оно будет корректно дополнено именем домена до ns.cs.ifmo.ru), а для записей типа PTR всегда полное имя (иначе оно будет некорректно дополнено до ns.cs.ifmo.ru.10.168.192.in-addr.arpa.)

Преимущество данного способа записи - все записи находятся “в парах”, четко прослеживается соответствие между именем и IP-адресом. С другой стороны, в примере данный файл используется для двух зон. Тогда в обратной зоне для имени 10.168.192.in-addr.arpa будет запись типа A, указывающая на некий IP-адрес. Аналогично произойдёт с записью PTR. Это не является проблемой, так как к этим записям фактически не будут происходить обращения.

```
A 192.168.10.1
```
В этой записи имя пустое, поэтому после подстановки имени домена получится запись, говорящая о том, что корень домена cs.ifmo.ru располагается по адресу 192.168.10.1

Директива GENERATE

```
$GENERATE 128-254 dhcp-10-$ A 192.168.10.$
$GENERATE 128-254 $ PTR dhcp-10-$.cs.ifmo.ru.
```
Цель директивы GENERATE в упрощении файла зоны и избавления от однотипных записей (например, при использовании DHCP-сервера). Данная директива генерирует строки, в которых вместо знака $ подставляется число из указанного диапазона. В этом примере эквивалентом использования GENERATE будут являться строки

```
dhcp-10-128 A 192.168.10.128
dhcp-10-129 A 192.168.10.129
…
dhcp-10-254 A 192.168.10.254
128 PTR dhcp-10-128.cs.ifmo.ru
129 PTR dhcp-10-129.cs.ifmo.ru
...
254 PTR dhcp-10-254.cs.ifmo.ru
```

Со другими типами записей можно ознакомиться, к примеру, в Википедии.

HTTP

Hyper Text Transfer Protocol был создан в 1992 году и изначально предназначался для передачи гипертекстов по модели клиент-сервер. Со временем по этому протоколу стали передаваться изображения и практически вся иная информация в сети, так что название протокола утратило свою актуальность.

На данный момент существуют 3 версии протокола - 0.9, 1.0 и 1.1., Все современные браузеры используют версию 1.1, а web-серверы поддерживают все версии.

HTTP 0.9

HTTP версии 0.9 поддерживает запросы только типа GET, заголовки отсутствуют.

Простейший запрос по протоколу 0.9:

```
GET /index.html
<!DOCTYPE html>
<html>
<head>
<title>Example page</title>
</head>
<body>
<h1>Example page</h1>
</body>
```

```
</html>
Connection to 192.168.11.104 closed by foreign host.
```

Так как протокол не поддерживает кодов состояний, то единственный способ узнать об ошибке - прочитать об этом на вернувшейся с сервера странице:

```
GET /wrongaddress.html
<html>
<head><title>404 Not Found</title></head>
<body bgcolor="white">
<center><h1>404 Not Found</h1></center>
<hr><center>nginx/1.4.3</center>
</body>
</html>
Connection to 192.168.11.104 closed by foreign host.
```

Единственный способ передать параметры в запросе - через параметры GET. Например, при отправке следующей формы:

```
<form              action="http://www.example.com"              method="GET">
          <input     type="hidden"     name="a"     value="1"     />
          <input     type="hidden"     name="b"     value="2"     />
          <input     type="hidden"     name="c"     value="3"     />
                    <input          type="submit"          />
</form>
```

Будет сформирован GET запрос http://www.example.com?a=1&b=2&c=3. При этом значения полей на языках, отличных от английского, должны пройти через процедуру URL encoding, поскольку в URL допускаются буквы только латинского алфавита.

Недостатки этой версии протокола:

- из ответа не понятен статус - произошла ли ошибка?
- невозможность создать виртуальные хосты (возможность размещения нескольких ресурсов на одном ip-адресе)
- неясен тип ответа (текст или же картинка или же нечто иное)

HTTP 1.0

Версия 1.0 была опубликована в 1996 году. В этой версии введены новые типы запросов и понятия заголовков запроса и ответа.
Пример запроса по протоколу 1.0:

```
GET /index.html HTTP/1.0
host: cs.ifmo.ru

HTTP/1.1 200 OK
Server: nginx/1.4.3
Date: Wed, 29 Jan 2014 21:36:27 GMT
Content-Type: text/html
Content-Length: 612
Last-Modified: Wed, 29 Jan 2014 20:30:12 GMT
Connection: close
ETag: "52e964d4-264"
Accept-Ranges: bytes

<!DOCTYPE html>
<html>
<head>
<title>Example page</title>
</head>
```

```
<body>
<h1>Example page</h1>
</body>
</html>
Connection to 192.168.11.104 closed by foreign host.
```

В запросе:

- В первой строке указан тип запроса и адрес ресурса, затем в явном виде версия протокола.
- На следующих строках указаны заголовки запроса. В HTTP/1.0 заголовок host допустим, но не является обязательным, в отличие от HTTP/1.1
- После чего посылается пустая строка в качестве тела запроса.

В данном случае web-сервер решил ответить, используя протокол HTTP/1.1, что допустимо (форматы ответов одинаковы).

В ответе:

- В первой строке ответа указан код ответа, что позволяет автоматически реагировать на возникшие ошибки. Затем написано краткое объяснение этого статуса в human-readable формате. Со списком возможных кодов состояний можно ознакомиться в Википедии.
- Затем следуют строки заголовка ответа в формате “<ключ> : <значение>”, где значением может быть достаточно длинным и содержать двоеточия. В данном примере:
  - **Server** - версия и имя сервера;
  - **Date** - дата запроса в GMT;
  - **Content-Type** - тип данных ответа (текст, изображение и т.п.); В этом примере это html-текст в кодировке utf-8
  - **Content-Length** - длина ответа в байтах (что полезно при скачивании больших файлов для отображения прогресса загрузки).
  - **Connection: close** - соединение будет закрыто после передачи ответа.
  - **Last-Modified** - когда данный документ был в последний раз изменен. Позволяет кешировать контент и не перекачивать вновь данные с сервера. При последующих запросах клиенту следует указать заголовок **If-Modified-Since** с этим временем и, если изменений не было, сервер вернет ответ со статусом 304 (Not Modified).
  - В Википедии можно ознакомиться со списком всех стандартных заголовков.
- Затем следует обязательная пустая строка и далее сам ответ.

Новые типы запросов, появившиеся в HTTP/1.0:

POST

Служит для передачи дополнительных данных в теле запроса для обработки web-сервером. Например:

```
POST                          /path/script.cgi                          HTTP/1.0
User-Agent:                                                          HTTPTool/1.0
Content-Type:                                  application/x-www-form-urlencoded
Content-Length:                                                                32

home=Cosby&favorite+flavor=flies
```

HEAD

Аналог GET, но web-сервер возвращает только заголовки ответа, не возвращая сам контент.

PUT

Изначально разрабатывался как метод для заливки файлов, в реальной жизни почти не используется.

HTTP 1.1

Был опубликован в 1999-м коду. Имеет два отличия от HTTP/1.0. Пример такого запроса:

```
GET /index.html HTTP/1.1
host: cs.ifmo.ru

HTTP/1.1 200 OK
Server: nginx/1.4.3
Date: Wed, 29 Jan 2014 21:32:29 GMT
Content-Type: text/html
Content-Length: 612
Last-Modified: Wed, 29 Jan 2014 20:30:12 GMT
```

```
Connection: keep-alive
ETag: "52e964d4-264"
Accept-Ranges: bytes

<!DOCTYPE html>
<html>
<head>
<title>Example page</title>
</head>
<body>
<h1>Example page</h1>
</body>
</html>
```

Существенныхотличий от HTTP/1.0 два. Заголовок Host является обязательным, его указание сообщает серверу информацию о том, с какого ресурса желает получить клиент, что позволяет размещать на одном IP-адресе несколько ресурсов, отличающихся именем (так называемые named-виртуальные хосты).
Второе отличие состоит в том, что по умолчанию, если иное не указано в заголовках запроса, используется keep-alive соединение, при котором web-сервер ожидает следующий запрос в рамках того же соединения. В большинстве случаев вместе с html-документом необходимо скачать CSS, javascript и изображения, причем на каждый файл нужен отельный запрос. В случае HTTP/1.0 на каждый файл пришлось бы установить соединение (3 пакета), совершить передачу и закрыть соединение (ещё 3 пакета) - итого 6 пакетов лишней нагрузки на каждую составляющую документа. В случае с keep-alive всё происходит в рамках одного соединения, лишняя нагрузка значительно меньше.

В Википедии можно ознакомиться с полным списком типов запросов.

Web-серверы

Web-сервер - это сервер, принимающий HTTP-запросы от клиентов и выдающий им HTTP-ответы (обычно html-страницы, изображения и т.п.)
Примеры web-серверов:

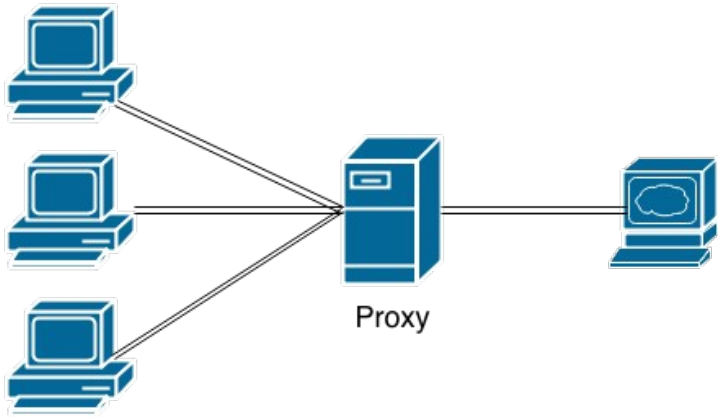
- nginx
- Apache HTTPD
- lighttpd

Все эти сервера умеют выдавать статический контент. Apache HTTPD сразу после установки способен генерировать динамический контент следующими способами:
SSI (Server Side Includes) - язык для динамической «сборки» веб-страниц на сервере из отдельных составных частей и выдачи клиенту полученного HTML-документа. Позволяет включать в текст страницы другие SSI-страницы, вызывать внешние CGI-скрипты, реализовывать условные операции. Файлы документов SSI имеют расширение .shtml.
CGI (Common Gateway Interface) - возможность сервера вызывать скрипты и программы на различных языках программирования, работающих со стандартным вводом-выводом и выдавать контент. При этом на стандартный ввод передается тело запроса, а заголовки записываются в переменные окружения. Главный недостаток CGI - на каждый запрос запускается новый процесс (что означает два ресурсоемких системных вызова - fork и exec).
Fast CGI - отдельный протокол взаимодействия с программами, запущенными внутри Fast CGI сервера, на который передаются запросы.
Также Apache поддерживает mod_perl и т.д. - подключаемые модули, позволяющие использовать PHP, Perl и иные языки.

Прокси

Слово прокси переводится как представитель кого-то перед чем-то. Типичным примером прокси является NAT. Обычно же под прокси подразумевается http-прокси. Существуют 2 режима работы http-прокси: прямое проксирование и обратное проксирование.

Прямое проксирование



Прямое проксирование позволяет осуществлять косвенные запросы к сетевым службам. Клиент подключается к прокси-сервер и шлёт запрос на него. Прокси сервер перенаправляет (при необходимости изменяя) запрос на указанный ресурс. За счет этого клиент сохраняет анонимность, его права ограничены, а уровень безопасности выше.
Чтобы использовать http-прокси для работы с https, необходимо воспользоваться командой CONNECT:

```
>telnet proxy 3128
Trying 192.168.10.1...
Connected to sunrise.cs.ifmo.ru.
Escape character is '^]'.
CONNECT google.com:443 HTTP/1.0

HTTP/1.0 200 Connection established
```

Далее https-прокси будет использоваться как передатчик информации между клиентом и сервером и он не будет вмешиваться в передаваемый трафик. Ограничение такого подхода в том, что прокси-сервер никак не анализирует проходящие через него данные (можно поднять SSH-туннель, что является дырой в безопасности). Поэтому следует настраивать https-прокси так, чтобы она работала только с 443-м портом. С другой стороны, если вэб-сервис работает не на 443-м порту, то клиенты, находящиеся за прокси, не смогут им воспользоваться.

Transparent-proxy

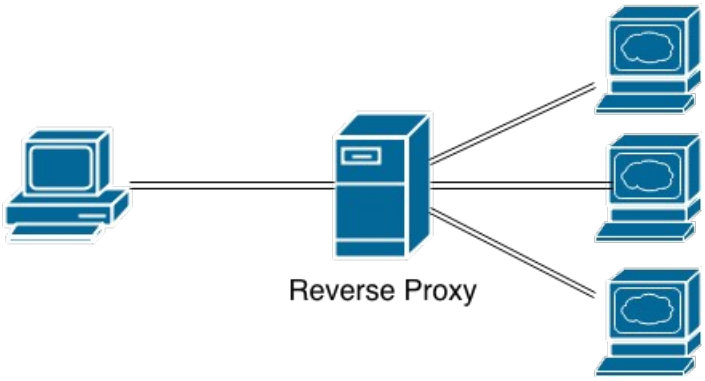
В обычной ситуации клиент в явном виде настраивает прямое проксирование. Transparent-proxy можно применять, если требуется

- кеширование запросов
- контроль доступа

Все запросы от клиента в Интернет на 80-й порт средствами пакетного фильтра заворачиваются в прокси. Так как все современные браузеры используют HTTP 1.1, то они в явном виде указывают заголовок host. По этой информации сервер может восстановить, куда было обращение клиента и перенаправить его куда следует.
Ограничения transparent-proxy:

- на прокси можно отправлять соединения только на 80-й порт. Если web-сервер работает не на 80-м порту, то пакет не будет обработан прокси-сервером и отфильтруется файерволом.
- невозможность использовать https, поскольку информация об адресате запроса передается уже после момента зашифрованного соединения.

Обратное проксирование



При обратном проксировании клиенты шлют запрос на прокси-сервер, который перенаправляет запрос на нужный web-сервер. При этом с точки зрения всех web-серверов все запросы приходят от одного-единственного прокси-сервера.
Обратное проксирование служит для:

- не было прямого доступа извне к серверам, выполняющим бизнес-задачи, базам данных
- балансировка нагрузки (load balancing)
- использование разных web-серверов для одного и того же ресурса

В случае прямого прокси клиент обычно знает, что используется прокси. В случае обратной прокси клиенту и серверу это неизвестно. В качестве прокси-сервера могут работать Apache с mod_proxy, nginx, lighttpd.