Конспект по дисциплине «Администрирование корпоративных сетей»

Готовил и оформлял: студ. гр. 6111, Жмылёв Сергей Александрович (korg@cs.ifmo.ru).

Введение

Курс изучения администрирования корпоративных систем нацелен на изучение основ сетевого и системного администрирования систем. Курс предполагает повторение и закрепление знаний курса «Сети ЭВМ и телекоммуникации» в областях, требуемых для администрирования, а также изучение основ администрирования систем, построения корпоративных сетевых топологий и ознакомление с типовыми технологиями различных уровней модели OSI современных компьютерных сетей, такими как: DHCP, DNS, HTTP, TCP, UDP, и другими.

Физические аспекты

Физический уровень модели OSI включает в себя множество средств передачи данных и протоколы работы и доступа к этим средствам. Последние в данный курс не входят, а первые изучались в рамках дисциплины «Сети ЭВМ и телекоммуникации», но требуют уточнения.

Средства передачи данных можно разделить на:

- кабельные;
- беспроводные;
- оптические:
    - физическая топология:
        - кольцо;
        - звезда;
        - точка-точка;
    - способ передачи:
        - одномодовые – имеют высокую дальность передачи (50-60 км);
        - многомодовые – имеют сравнительно низкую стоимость;
    - количество используемых волокон.

Из-за сравнительно высоких скоростей при достаточно надёжной передаче, несмотря даже на широкое распространение 802.11ac с номинальной скоростью в 1300 Мбит/с, чаще всего в корпоративных сетях используются кабельные сети. Кабели, используемые для передачи данных в таких сетях, разделяются на:

- коаксиальные кабели, которые характеризуются:
    - топология: общая шина;
    - толщина: тонкие (5мм) и толстые (10мм);
    - коннектор: BNC (bayonet connector);
- кабели типа «витая пара», характеристики которых:
    - топология: звезда, общая шина, полносвязная и другие;
    - коннектор: 8P8C;

- кабели интерфейсов InfiniBand, SAS и других специализированных (в данном курсе не рассматриваются).

Тонкие коаксиальные кабели использовались для передачи на меньшие расстояния, т.к. имеют больший коэффициент затухания сигнала по сравнению с толстыми. Из преимуществ можно выделить два: гибкость, следовательно, простоту использования, а также простоту подключения к оборудованию за счёт использования BNC-коннекторов. В свою очередь, для подключения к толстому кабелю, требуется использования сложного и дорогого устройства, предназначенного для прокалывания оболочки и обеспечения контакта с основной жилой и с экраном. В настоящее время коаксиальные кабели не применяются из-за дороговизны самого кабеля, конечного оборудования и низких скоростей.

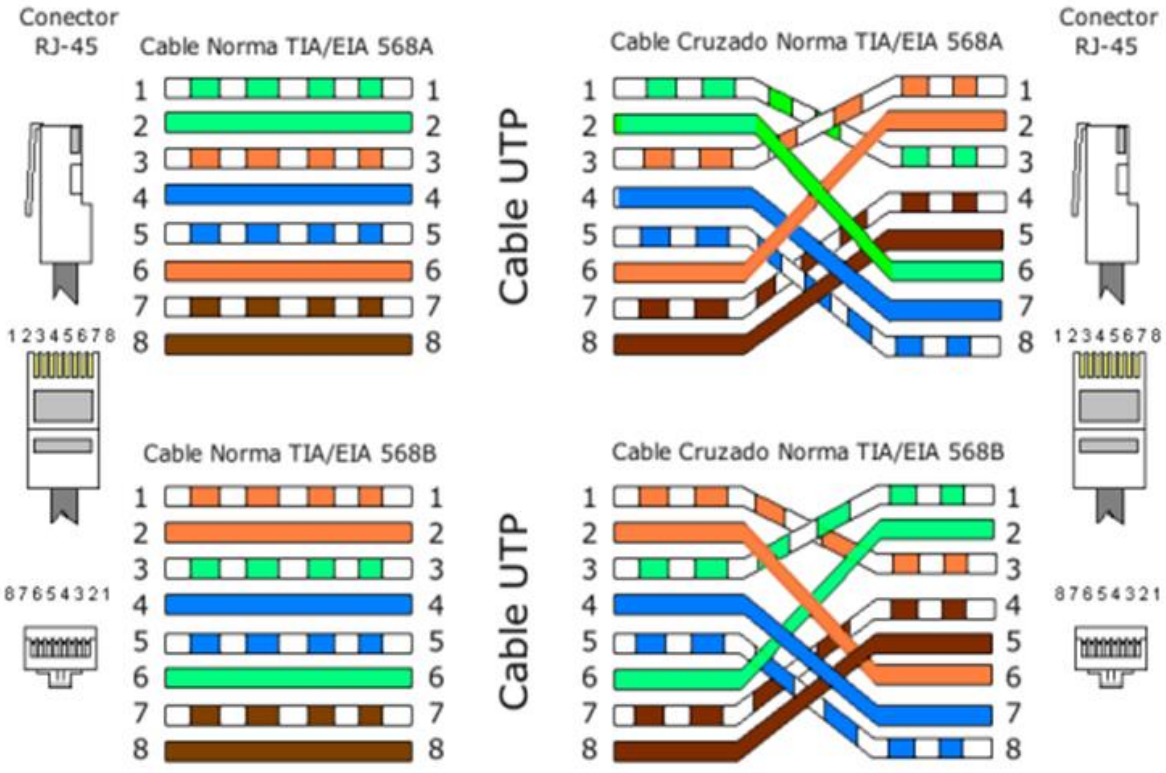
Кабели типа «витая пара» делятся на различные категории. Наибольшее распространение получили следующие категории, каждая из которых включает в себя четыре свитых пары проводников:

- категория 3. Скорость передачи до 100 Мбит/с на расстоянии до 100 метров;
- категория 5. Скорость передачи до 100 Мбит/с (при использовании двух пар) и до 1000 Мбит/с (при использовании четырёх);
- категория 5e. Скорость передачи – аналогично категории 5, имеет улучшенные и уточнённые (enhanced) спецификации;
- категория 6. Скорость передачи до 10 Гбит/с на расстоянии до 55 метров.

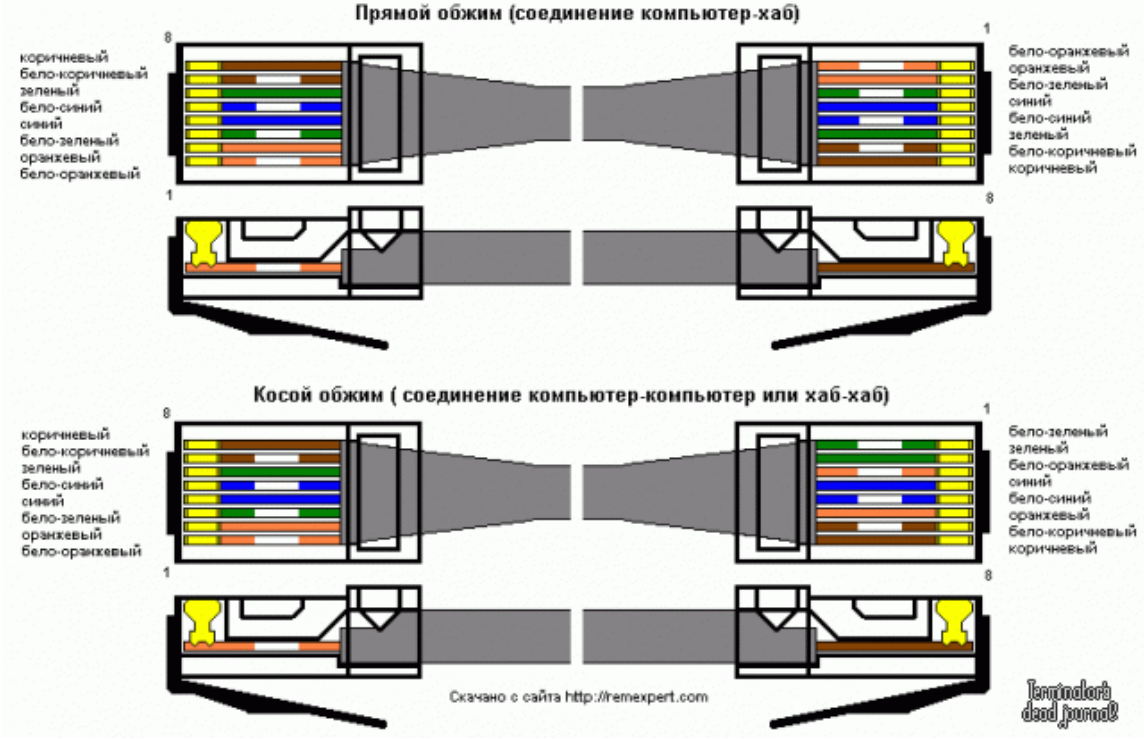
В Ethernet-адаптере используется дифференциальное кодирование сигналов, поэтому передача данных будет работать корректно если провода в одной паре поменять местами. Дифференциальное кодирование используется потому, что является помехоустойчивым. Если сигнал помехи наводится на провода в паре, потенциал обоих проводников изменяется одинаково, а дифференциал (разность) потенциалов остаётся прежним.

В современных Ethernet сетях существуют два стандарта обжатия коннекторов, чтобы сетевой монтажник мог корректно заделать коннектор, не имея информации о коннекторе с другой стороны – EIA/TIA-568B и EIA/TIA-568-A. Передающие устройства можно разделить на клиентские (например, компьютеры) и инфраструктурные (активное сетевое оборудование, например, коммутаторы). В сетях 10/100 Ethernet, при использовании витой пары, для передачи со стороны компьютера используются провода 1 и 2 (и называются TX), а для приёма – 3 и 6 (и называются RX). Соответственно, у активного сетевого оборудования RX и TX подключены наоборот. Провода 4 и 5 зарезервированы для совместимости с RJ-11. Исходя из строгого назначения TX и RX, для соединения двух хостов одного типа (PC-PC или Switch-Switch) между собой требуется «кроссировать» кабель, чтобы TX одной стороны попадал на RX второй и наоборот. Практически всё современное оборудование поддерживает технологию Auto MDI/MDI-X, благодаря которой, оборудование может само определить, к каким проводам подключён RX и TX удалённой стороны и, соответственно, принудительно работать по правильным парам. Несмотря на то, что в стандарте Gigabit Ethernet описан кроссированный кабель, в гигабитных Ethernet сетях кроссировать кабель не требуется, т.к. системы обязаны поддерживать технологию Auto MDI/MDI-X.

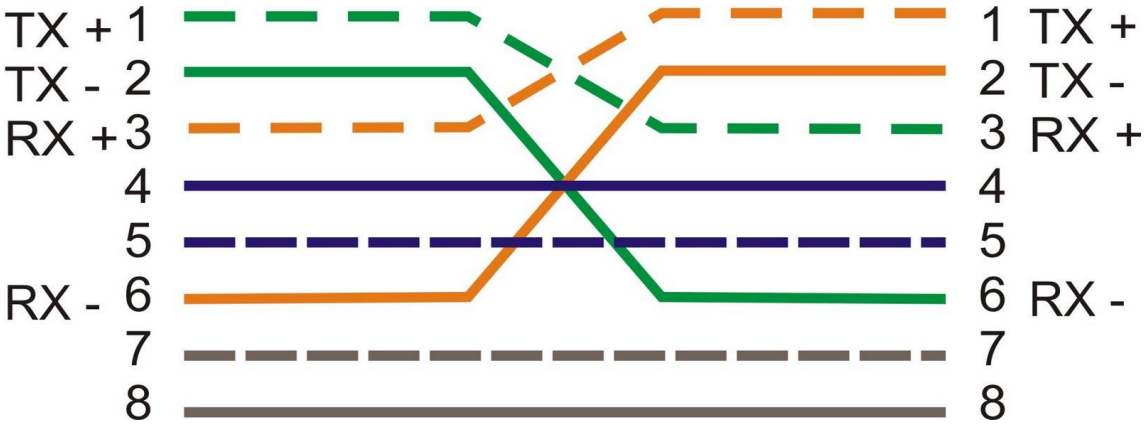
Схемы подключения (слева – прямой, справа – кросс):



Описанные ранее принципы кроссирования при подключении хостов одного и разного типов можно проиллюстрировать так:



А саму необходимость кроссирования (чтобы RX подключался к TX и наоборот) так:



К активному сетевому оборудованию можно отнести:

- повторитель (repeater);
- медиаконвертор (например, медь ↔ оптика);
- концентратор (hub)
    - физическая топология: звезда, логическая: общая шина;
    - практически не добавляет задержку;
- коммутатор (switch)
    - реализует работу полнодуплексном режиме.

Характерной чертой концентратора является низкая задержка передачи данных. Это обусловлено тем, что физически концентратор представляет собой аппаратный усилитель/разветвитель сигнала и передаваемые данные не обрабатываются. Таким образом, концентратор работает на первом уровне модели OSI, просто распространяя сигналы на все порты. Из преимуществ концентраторов перед коммутаторами выделяются только два: сравнительно низкая стоимость и удобство использования для анализа трафика между двумя хостами (нивелируется при наличии коммутатора с возможностью зеркалирования трафика – копирования трафика не в один, а в несколько портов).

В противовес концетратору, коммутатор работает на втором уровне модели OSI. В теории пропускная способность коммутатора равна количеству портов, умноженному на скорость передачи. Это обусловлено тем, что у коммутатора для каждого порта есть отдельный процессор ввода-вывода. Чаще всего, фреймы передаются из одного порта в другой и, в отличие от концентратора, аппаратно задействуют только эти два порта для передачи.

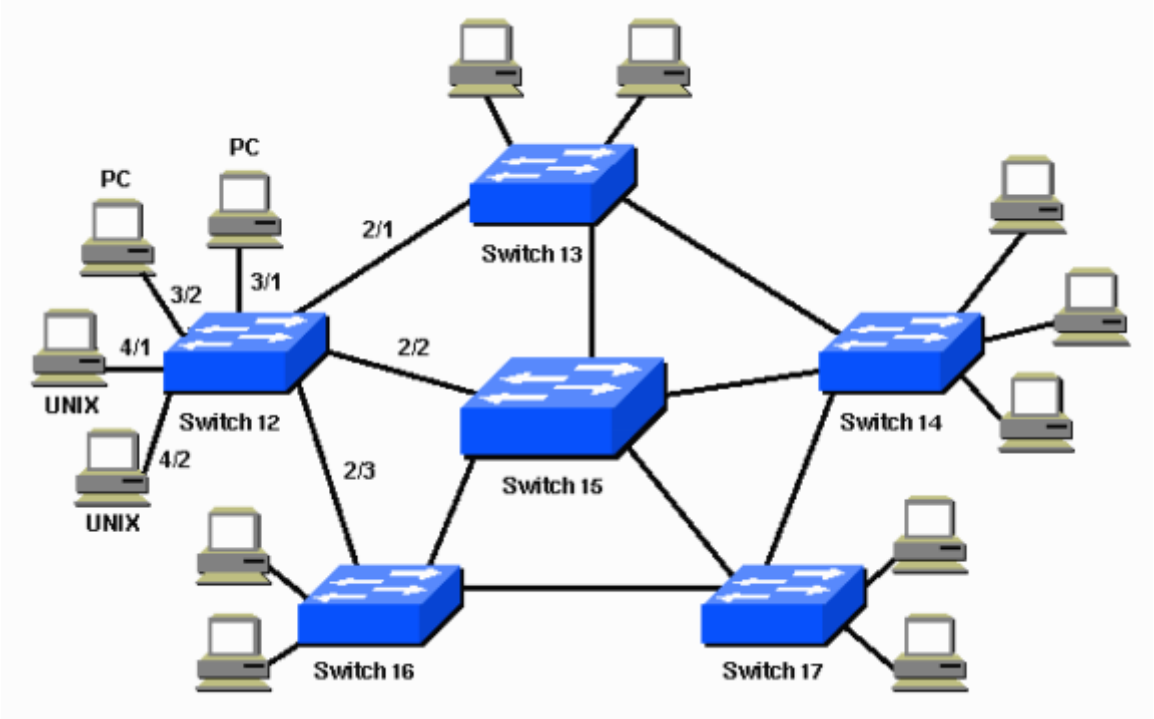
Практикой такая пропускная способность не подтверждается. Наблюдаются задержки, обусловленные необходимостью частичного декодирования кадра и определения адресата (в том числе, поиск по таблице MAC-адресов). Если адресат неизвестен (не содержится в таблице MAC-адресов коммутатора), фрейм отправляется всем. Поэтому существует угроза безопасности: если переполнить память MAC-адресов, то можно получать фреймы, адресованные другим устройствам. Решением данной проблемы является ограничение количества MAC-адресов на порт, либо отказ от использования общей памяти для хранения таблицы MAC-адресов. Также возможно использование статической привязки MAC-адреса к порту.

В случае, когда с одной стороны передача осуществляется в режиме half-duplex, а с другой – в full-duplex, возникает большое количество, так называемых, «поздних» коллизий. Это обусловлено тем, что система, работающая в full-duplex будет посылать фреймы в сеть во время приёма данных. Такая ситуация приводит к снижению скорости передачи данных. С помощью утилиты netstat (с ключом –i) можно проверять обнаруженные на интерфейсе коллизии:

```
$ netstat -i
Name  Mtu  Net/Dest      Address        Ipkts     Ierrs Opkts      Oerrs Collis Queue
lo0   8232 loopback      localhost      189876989 0     189876989  0     0      0
igb0  1500 helios        helios         927727420 4     347070     16    39354  0
```

В данном примере видно, что 16 исходящих пакетов было отправлено с ошибками. При этом, в сети было зафиксировано 39354 коллизии. Процент коллизий можно определить делением Collis на Opkts (общее число отправленных пакетов), а процент ошибок – делением Oerrs на Opkts. Если процент коллизий превышает 5-10 (в рассмотренном примере – 11), это означает, что в сети присутствуют проблемы и необходимо сверить режим дуплекса на интерфейсе и оборудовании, к которому он подключен. Высокий процент ошибок может говорить о том, что хост теряет пакеты (drop). В частности, в сети со скоростью 100 Мбит/с экспериментально было получено различие в 3,375 раза при использовании одним из хостов только full-duplex, а вторым – full- и half-duplex соответственно.

Если инфраструктура сети включает в себя закольцовывание свитчей, например, как показано на рисунке:



То возникают проблемы, связанные с запросами, отправляемыми на все порты (например, широковещательные или пакеты на неизвестный MAC-адрес). Такие запросы начинают курсировать по кольцу, создавая избыточную, увеличивающуюся нагрузку на каналы передачи. Эта ситуация носит название «broadcast storm». Для борьбы с таким поведением используется протокол STP (или различные его модификации). Данный протокол строит граф сети от «корневого коммутатора» и программно разрывает кольцо, переводя один из портов какого-либо коммутатора в специальное состояние (Blocking), таким образом, граф сети избавляется от цикличности. Если кольцо обрывается физически, то Blocking порт снова включается в работу.

Недостатки:

- большое время «перестройки» дерева;
- подключившись как «корневой коммутатор», можно отправлять в сеть некорректные запросы;

- новый хост не осуществляет передачу данных 60 секунд после подключения.

Решение:

- использование пропиретарных протоколов (требует использования оборудования одной фирмы);
- ограничение доступа в качестве «корневого коммутатор» по портам.

В качестве примера STP дерева можно рассмотреть вывод команды на одном из свитчей компании Cisco:

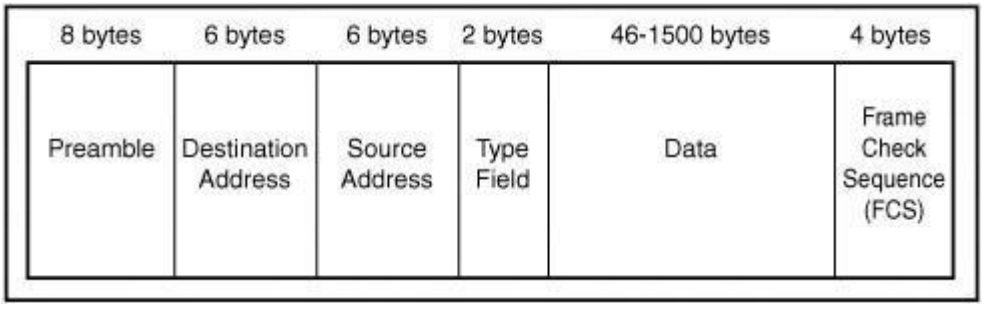
```
Distribution1#show spanning-tree mst 2

##### MST2    vlans mapped:   20,40,200
Bridge        address 0015.63f6.b700  priority      28674 (28672 sysid 2)
Root          address 0015.c6c1.3000  priority      24578 (24576 sysid 2)
              port    Gi1/0/24        cost          200000    rem hops 4

Interface        Role Sts Cost      Prio.Nbr Type
---------------- ---- --- --------- -------- --------------------------------
Gi1/0/1          Desg FWD 200000    128.1    P2p
Gi1/0/3          Desg FWD 200000    128.3    P2p
Gi1/0/23         Altn BLK 200000    128.23   P2p
Gi1/0/24         Root FWD 200000    128.24   P2p
```

Видно, что один из портов (а именно, Gi1/0/23) находится в состоянии BLK (Blocking), что говорит о том, что по данному порту фреймы передаваться не будут.

Канальные протоколы

Основы Ethernet

Наибольшее распространение в корпоративных сетях имеет протокол Ethernet версии 2. Передаваемые по протоколу Ethernet блоки данных называются фреймами и имеют следующую структуру:



Поле «Type» отвечает за тип полезных данных («Data») и может принимать такие значения, как: IP (0x8000) или IPX (0x8137).

Etheret-фрейм принимается системой в следующих случаях:

- совпадает MAC-адрес назначения (destination address) и MAC-адрес системы;

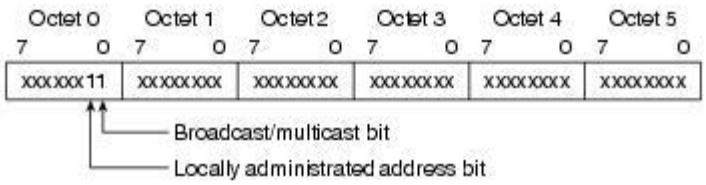
- когда фрейм является широковещательным (broadcast);
- когда сетевой адаптер переключен в «неразборчивый» (promiscious) режим (например, когда адаптер не поддерживает несколько MAC-адресов и включена виртуализация).

Ethernet-фрейм доставляется до системы в случае, когда активное сетевое оборудование по известному MAC-адресу (см. ранее) отправило в порт, к которому подключена целевая система этот фрейм. Это происходит если в таблице MAC адресов этот порт закреплён за этим MAC-адресом, либо если фрейм широковещательный, либо если MAC-адреса нету в таблице MAC-адресов и фрейм рассылается на все порты.

Неразборчивый режим работы адаптера – это такой режим, при котором адаптер принимает все фреймы, вне зависимости от адреса назначения. Принятые кадры передаются драйвером интефейса сетевому стеку ОС, что приводит к избыточной нагрузке на процессор. Поэтому, чаще всего неразборичвый режим отключен по-умолчанию. Этот режим предназначен для диагностики пакетов в текущем домене коллизий (ethernet-сегменте), но иногда используется для несанкционированного получения трафика соседних хостов.

Из-за того, что кадры Ethernet вносят довольно большую избыточность (18 байт на 1500), накладные расходы на обработку заголовков и, самое главное, из-за необходимости фрагментации больших объёмов данных, была создана технология «Jumbo Frame», которая позволяет увеличивать размер поля Data (теоретически, до 16 Кб, практически, системы поддерживают около 9 Кб). Для работы с ней, всё оборудование в одном сегменте обязано её поддерживать. Это обусловлено тем, что MTU конечных хостов неизвестно и отправляемые кадры будут формироваться с большим MTU. Если в сети присутствует система, не поддерживаются jumbo frame (в первую очередь, это устройства без поддержки 1000Base-T) или некорректно настроенная, то эта система будет создавать большое число коллизий, отправляя данные с маленьким MTU. В реальных системах эта технология встречается довольно редко. В гигабитной сети jumbo frame 9000 показал прирост около 93% при передаче большого файла (1 Гб) и набора небольших файлов (суммарно ~1.2 Гб). С результатами экспериментов можно ознакомиться по ссылке https://wiki.archlinux.org/index.php/jumbo_frames.

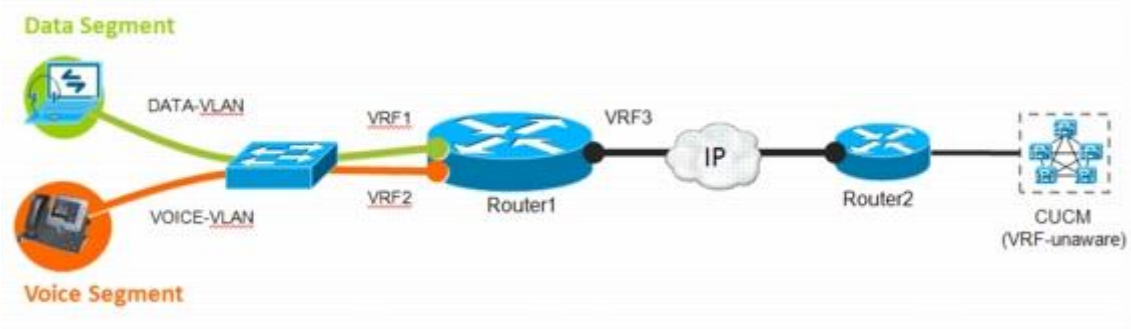
MAC-адрес состоит из шести байт (октетов) и имеет слудующую структуру:



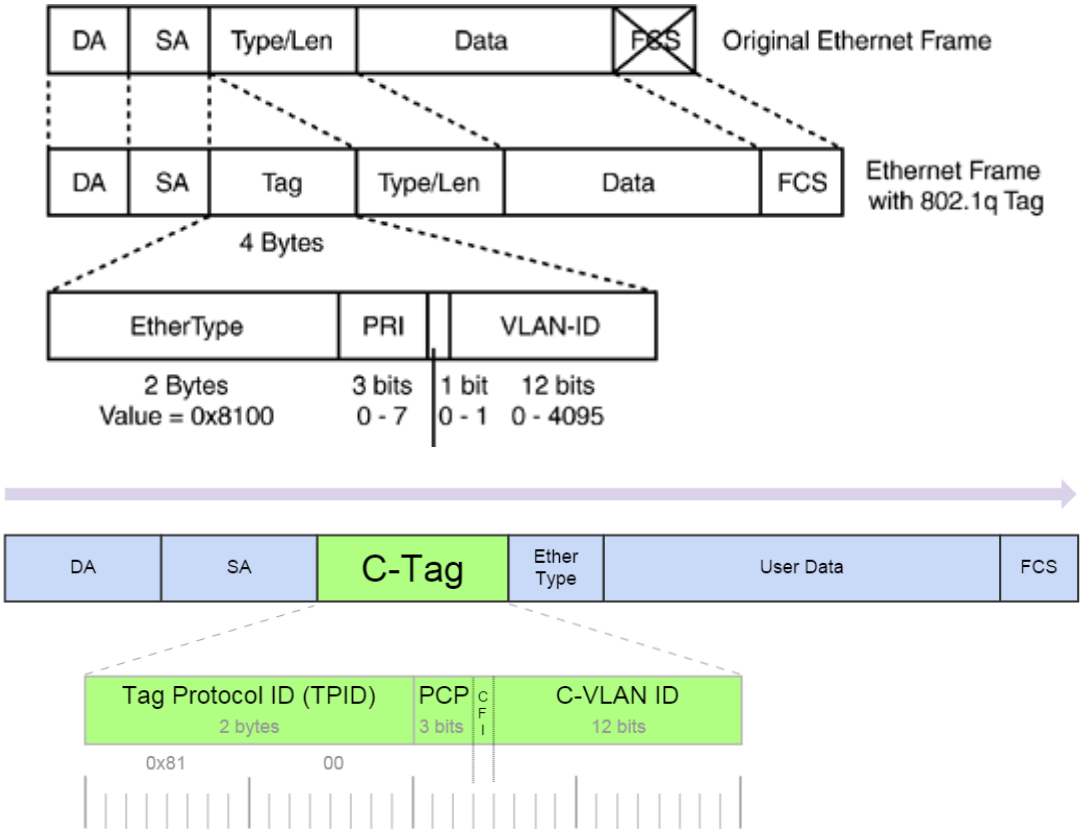
Интерес представляют два показанных на рисунке бита. Групповые адреса применяются для групповой рассылки (broadcast или multicast). Локально администрируемые адреса используются для указания того, что адрес отличается от установленного производителем и может быть настроен вручную. Естественно, уникальность таких адресов не гарантируется. Эта технология зачастую используется гипервизорами для виртуализации сетевых интерфейсов.

Технология VLAN

Зачастую возникает необходимость, при использовании одних физических каналов, разделить оборудование по разным сегментам (например, чтобы трафик VoIP «не мешал» трафику внутренней сети). Например, как это показано на рисунке:



Для этого была разработана технология VLAN. Эта технология добавляет в Ethernet кадр четыре байта, увеличивая его размер с 1518 до 1522 байт соответственно. В поле «Type» указывается значение 0x8100, означающее, что фрейм относится к кадрам 802.1q (VLAN), а сразу за полем «Type» следует структура: Priority (PCP) – приоритет [3 бита], CFI – тип кадра: Ethernet или иное (Token Ring, FDDI) [1 бит], Vlan tag – номер виртуального сегмента [12 бит]. Такое изменение структуры заголовка представлено на рисунке:

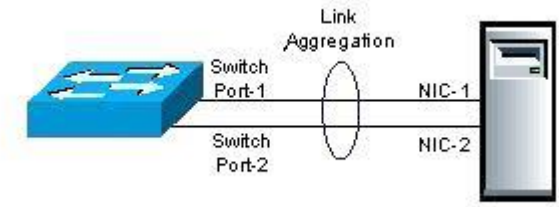


В современных сетях поле CFI утратило своё значение и используется как Drop eligible indicator (DEI) – поле, показывающее, можно ли отбрасывать кадр при перегрузке сети.

Номера сегментов 0 и 0x3FF зарезервированы, поэтому реально можно использовать только 4094 VLAN-а. Соответственно, для работы с VLAN, коммутаторы должны поддерживать эту технологию. Кроме базовой поддержки передачи таких фреймов, часто коммутаторы поддерживают настраиваемые политики приёма/передачи данных на основе VLAN тэга. В частности, можно осуществлять передачу данных из сегмента с VLAN-ами в обычный Ethernet сегмент.

Агрегирование каналов

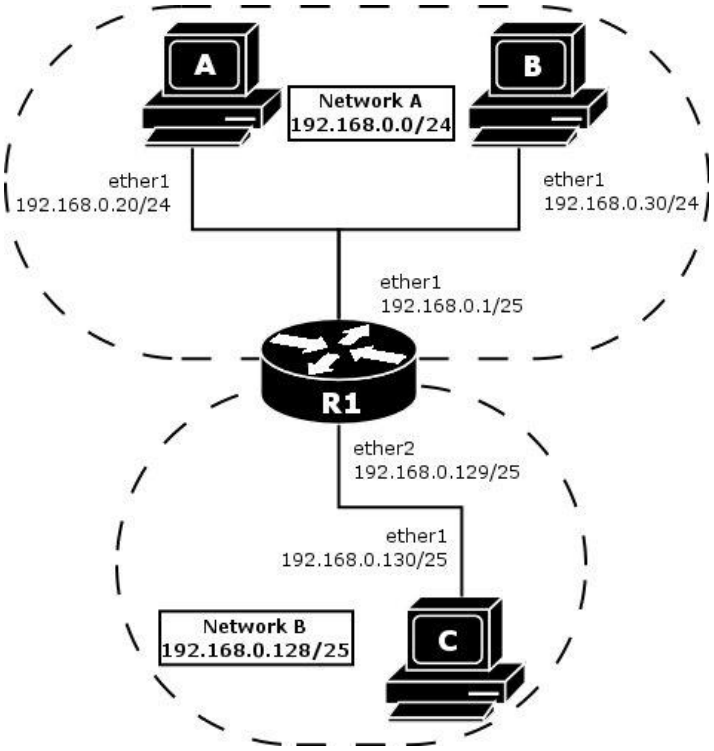
Другой, представляющей интерес технологией, широко распространённой в корпоративных сетях является технология объединения каналов передачи данных для повышения надёжности передачи. Эта технология называется Link Aggregation (наиболее часто встречается как имплементация стандарта 802.3ad), которая позволяет передавать данные одновременно через несколько каналов. Технологию можно проиллюстрировать так:



Соответственно, по двум физическим каналам можно передавать данные обеспечивая более высокую пропускную способность за счет горизонтального масштабирования и надёжность передачи за счёт избыточности каналов.

Сетевые мосты

Основываясь на использовании флага public, можно построить сетевой мост. Сетевые мосты называются ARP proxy. Рассмотрим пример:



В данном случае хост R1 имеет в таблице ARP публичные записи всех систем и отвечает на ARP запросы в соответствующие сегменты, практически объединяя subnet A и subnet B в один сегмент. Пример показывает, как можно объединять вложенные IP-подсети.

Максимально передаваемый блок данных (MTU)

В зависимости от типа, интерфейс может иметь такой параметр, как MTU – Maximum Transmission Unit. Следует помнить, что MTU – характеристика канального уровня, тогда как размер, например, пакетов – сетевого. Зачастую IP пакет (имея максимальную длину в 65536 байт) не

помещается в MTU, в таком случае происходит фрагментация – разбивка пакета на несколько. Очевидно, большее значение MTU приводит к ускорению реакции системы на маленькие пакеты и позволяет быстрее реагировать на потери, но при этом сильно увеличивается избыточность передаваемых по сети данных. Из рассмотренного ранее знаем, что для классического Ethernet MTU не может превышать 1500 (кроме описанного ранее Jumbo frame), поэтому довольно часто это значение и используется на множестве интерфейсов. Когда один хост отправляет другому фрейм с MTU превывающим MTU получателя, возникает проблема принятия такого пакета. Соответственно, данная проблема решается одним из способов:

- фрагментация на стороне отправителя;
- фрагментация на стороне сервера;
- если пакет нельзя фрагментировать (установлен IP флаг DF, do not fragment), отправка отправителю ICMP Fragmentation needed (см. сетевые протоколы - формат заголовка IP).

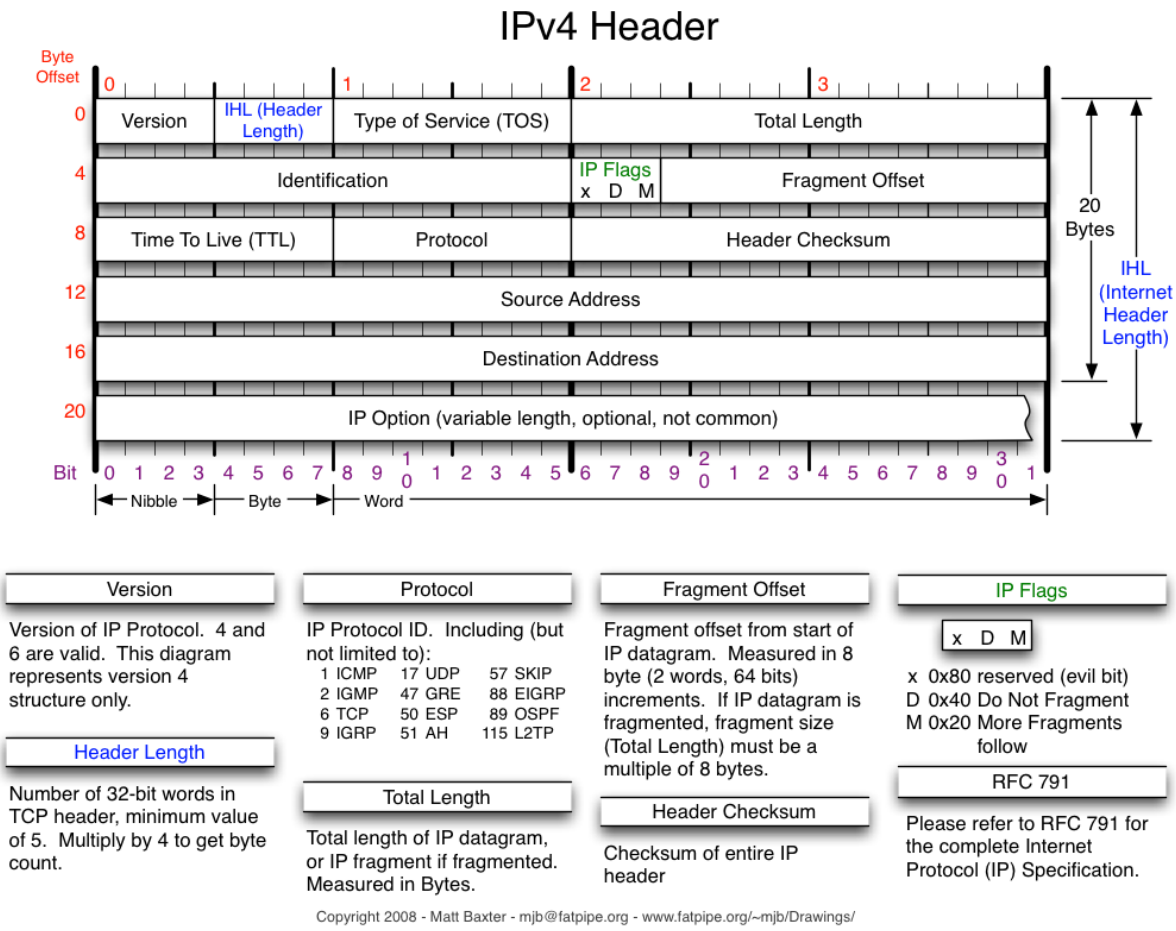
В случае пересборки пакетов промежуточным узлом, злоумышленник может переполнить буферы пересборщика, что приведёт к потерям пакетов и замедлению работы сети, поэтому пересборка используется не часто. Кроме того, иногда сетевой экран блокирует протокол ICMP, в таких случаях отправитель не узнает о том, что надо перефрагментировать пакет.

Существует спецификация, называемая PMTUD (Path MTU Discovery), которая описывает процесс определения минимального MTU между точками передачи. Этот процесс работает следующим образом: последовательно, с уменьшением, устанавливаются значения MTU, а также флаг DF, который (в случае слишком большого размера фрейма) приводит к отправке ICMP Fragmenation needed, и до тех пор, пока отправитель получает такие ICMP пакеты (пока на пути пакета есть каналы с меньшим MTU), MTU уменьшается.

Протоколы сетевого уровня

Протокол IPv4

С точки зрения модели OSI на сетевом уровне в настоящее время преобладает использование протокола IP. Широкое распространение получили две его версии: IPv4 и IPv6. IPv4 начала разрабатываться в 1970-х годах, а основной стандарт, описывающий этот протокол (RFC 791) был опубликован 1981 г. Протоколы IPv4 и IPv6 детально рассматриваться в этом курсе не будут, но описания требует формат заголовков и структура пакетов этих протоколов. Заголовок пакета, передаваемого по протоколу IPv4 выглядит следующим образом:



Основная часть заголовка – это пять 32-х битных слов, которые содержат в себе поля согласно рисунку. В заголовке присуствует указание версии протокола, длина пакета, протокол верхнего уровня (какие именно данные передаются в пакете, например TCP), адрес назначения и адрес отправителя, поля TTL и Checksum (описаны далее), поле TOS (описанное несколькими стандартами и в общем случае предназначенное для указания приоритета пакета, требований к надёжности передачи и к пропускной способности), а также поля для работы с фрагментами (указатель на текущий фрагмент и флаги фрагментации). Опционально в заголовке могут присутствовать специфические дополнительные поля. В свою очередь, заголовок IPv6 представляет собой аналогичную структуру. Отличие заключается в длине адресов и в меньших требованиях к обязательным полям.

Адрес IPv4 имеет длину 4 байта. Его можно записывать в различных видах:

- 192.168.0.1 – четыре октета, разделённые точкой;
- 3232235591 – как десятичное число (работает, например, в браузере);
- C0A80001 – в шестнадцатеричном виде.

Для разделения IP-сети на различные подсети, чтобы хосты знали адреса компьютеров локальной сети и могли отличать локальное пространство сети от остального, используется маска подсети. Она состоит из непрерывной последовательности единиц, которые дополняются нулями до формата IP-адреса. Записывать маску можно в двух форматах:

- 255.255.240.0
- /19

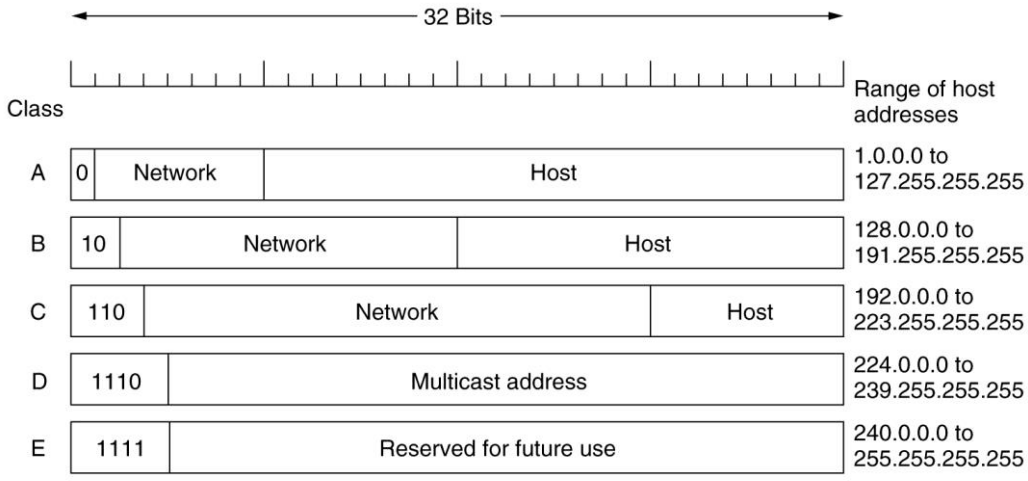
Оба формата подразумевают в старших разрядах единицы, а в младших нули. Адрес подсети можно получить выполнив коньюнкцию маски с IPv4 адресом:

	Network Number	Host ID
IP Addr	192 . 1 . 1	. 0
IP Addr. Binary	**1100 0000 . 0000 0001 . 0000 0001**	. 0000 0000
Mask	255 . 255 . 255	. 0
Mask Binary	1111 1111 . 1111 1111 . 1111 1111	. 0000 0000

Отсюда и формируются зарезервированные общий (предназначенный для маршрутизации) адрес сети и широковещательный адрес сети – указанием 0x0 и всех единиц в качестве Host ID соответственно.

В реальности иногда используется инветированная маска (0.0.255.255), при которой единицы отвечают за адрес хоста. Такая маска получила название wildcard mask и её можно увидеть, например, в оборудовании комании Cisco (в списках контроля доступа, в конфигурации OSPF).

Помимо бесклассовой адресации, при которой используется маска подсети, существует также классовая адресация. Разделение на классы основано на старших битах IP адреса. Стандартные классы приведены ниже:



Из рисунка видно, что классы A, B и C имеют маску /8, /16 и /24 соответственно. Адреса класса D используются для multicast-рассылок, а класса E – зарезервированы.

В классах A, B и C выделены специальные диапазоны, используемые для частных сетей. Пакеты из этих диапазонов адресов принято не маршрутизировать (см. далее) в глобальную сеть. Эти диапазоны приведены в таблице:

Класс	**Диапазон**
A	10.0.0.0 /8
B	172.16.0.0 /12
C	192.168.0.0 /16

Кроме указанных диапазонов, в IP сетях существует довольно много других зарезервированных диапазонов адресов. Наиболее часто встречаются:

Диапазон
0.0.0.0/8
127.0.0.0/8

Последний зарезервирован под адреса обратной петли (loopback). На традиционных UNIX системах, однако, по-умолчанию как адрес обратной петли используется 127.0.0.1/32, поэтому иногда (в случаях, когда требуется больше одного адреса обратной петли, то есть почти никогда) приходится вручную менять таблицу маршрутизации (см. таблицу маршрутизации). Системы Microsoft Windows избавлены от этой проблемы и в стандартной инсталляции корректно работают со всеми адресами 127.0.0.0/8.

Протокол ARP

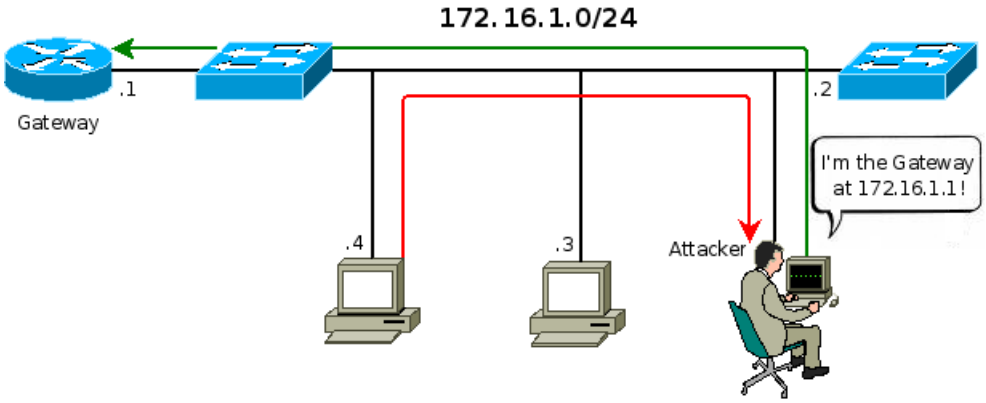
Строго говоря, протокол ARP не относится к протоколам сетевого уровня, а занимает место где-то между сетевым и канальным. Протокол Ethernet, при работе в IP-сетях, для отправки данных требует MAC-адрес системы назначения. Для поиска такого адреса используется протокол ARP (Address Resolution Protocol) и таблица ARP, которая используется в качестве кеша. В ARP таблице есть следующие поля: IP, маска, MAC адрес, устройство и флаги. Соответственно, когда система решила отправить данные на IP адрес 192.168.10.178, в ARP таблице находится соответствующая запись и становится ясным, с какого (из Device) и на какой (из Physical address) адреса отправлять:

```
$ arp –an
Device   IP Address          Mask        Flags      Phys Addr
------ ----------------- --------------- -------- ------------
igb0   192.168.10.162    255.255.255.255 SPLA     00:25:90:98:33:be
igb0   192.168.10.178    255.255.255.255 o        00:10:e0:0d:c2:72
igb0   192.168.10.102    255.255.255.255 SP       00:25:90:98:33:be
```

Если в ARP таблице нет соответсвующей искомому IP адресу записи, система с не отключенным ARP (на сетевом устройстве не установлен флаг noarp) отправляет широковещательный ARP запрос. На этот запрос ответят системы, в ARP таблице которых есть соответствующая запись с установленным флагом P (public). В показанном примере, система ответит на запрос адреса 192.168.10.102. Поле Flags может принимать следующие значения:

- d Непроверенный. Система проверяет адрес на дублирование и до окончания проверки отвечать на него не будет;
- o Устаревший. Адрес сохранен для отслеживания изменения MAC-адреса удалённого хоста;
- y Отложенный. Система откладывает проверку адреса на дублирование из-за внутреннего ограничения;
- A Авторитетный. Система не будет принимать уведомления об изменении MAC для этого адреса;
- L Локальный. Система будет проверять, чтобы другой хост не захватил этот адрес;
- M Мультикаст. Используеся для мультикаст рассылок;
- P Публичный. Система будет отвечать на ARP-запросы этого адреса;
- S Статический. Указан вручную и не требует ARP-запросов. Не устаревает;
- U Неизвестен. Система ожидает ответ на ARP-запрос.

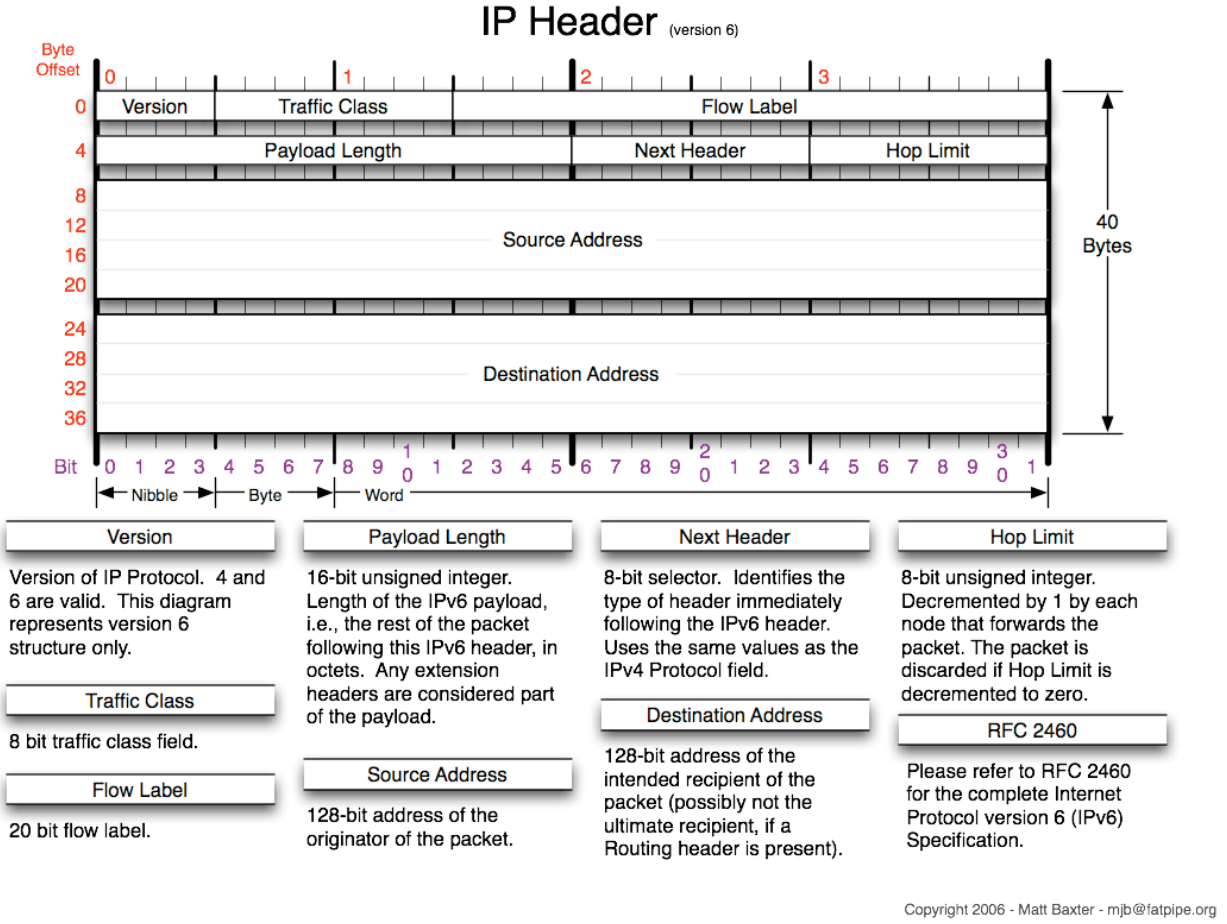
В таком подходе к разрешению адресов, существует проблема перехвата чужих адресов, описанная ниже:



Злоумышленник быстрее шлюза успевает ответить хосту 172.16.1.4, что MAC адрес шлюза – система злоумышленника и перехватывает все пакеты. Такая атака относится к виду (MITM) man-in-the-middle. Проблему решают различными способами: использование статических ARP записей, использование PPPoE и др.

Протокол IPv6

Шестая версия протокола IP начала разрабатываться в 1994 г., но широкое распространение получила в 2005 г. Не смотря на то, что процесс полного перехода на IPv6 сильно затянулся, многие клиентские системы (особенно, семейство ОС Microsoft Windows) используют Link-Local адреса даже не подозревая об этом. В протоколе IPv6 большинство опций, при использовании стандартных значений, не требуют явного указания и могут быть перечислены после заголовка. Формат заголовка IPv6 представлен на рисунке:



IPv6 адрес имеет длину 16 байт. Протокол имеет ряд отличий от IPv4. Некоторые описаны далее. Как было отмечено ранее, в заголовке IPv4 есть также поле «Checksum» - контрольная сумма заголовка, которое пересчитывается для каждого пакета при изменении значений полей заголовка. Однако, пересчитывать контрольную сумму на сетевом уровне практически всегда бессмысленно (контрольную сумму данных проверяют протоколы транспортного уровня, а передачи – канального), поэтому в IPv6 отказались от этого поля и вносимых им накладных расходов.

Существенным отличием является оказ от фрагментации пакетов, разобранной ранее. Протокол IPv6 обязывает передающую сторону использовать рассмотренный ранее механизм Path MTU discovery.

В IPv6 сетях предусмотрена автоконфигурация. Во-первых, новое устройство автоматически получает адрес основанный на её MAC-адресе (поэтому ARP для IPv6 не требуется), а во-вторых, используются протоколы neighbour discovery и router solicitation, предназначенные для определения хостов и маршрутизаторов своей сети соответственно, используя групповую рассылку запросов. Для ускорения процесса маршрутизации (см. далее) используется алгоритм, схожий с механизмом коммутации по меткам, рассмотренном в курсе «Сети ЭВМ и телекоммуникации».

Маршрутизация

В IPv4 сетях для определения назначения пакета используется маршрутизация. Маршрутизация – это процесс определения маршрута и параметров пакета по специальной таблице.

Таблица маршрутизации изменяется или настраивается в следующих случаях:

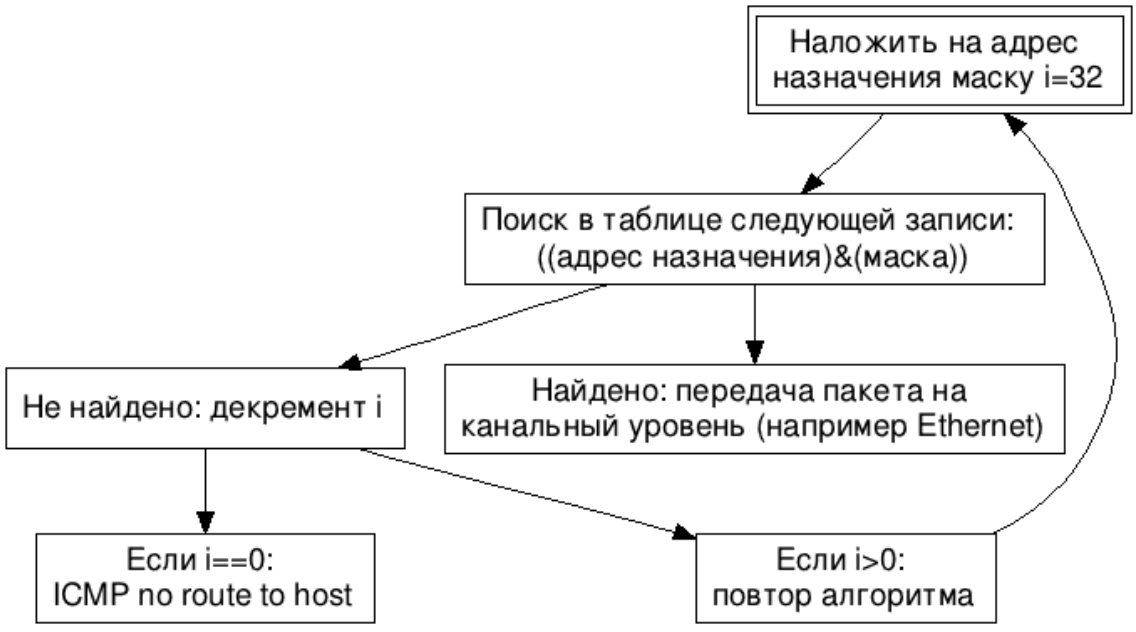
- при добавлении интерфейса;
- при выполнении команды route add;

- при использовании виртуальных частных сетей (VPN) или стороннего ПО;
- при использовании протокола DHCP (см. лабораторные работы);
- при использовании динамической маршрутизации;
- при работе механизма ICMP redirect (см. транспортные протоколы).

В ОС Solaris эту таблицу можно посмотреть, например, командой netstat –rnv:

```
$ netstat -rnv
Destination     Mask            Gateway          Device  Flg
--------------- --------------- ---------------- ------- ---
default         0.0.0.0         192.168.10.1             UG
192.168.10.0    255.255.255.0   192.168.10.10    igb0    U
192.169.8.0     255.255.252.0   192.168.32.1             UG
192.168.32.0    255.255.255.0   192.168.32.17    igb1    U
224.0.0.0       240.0.0.0       192.168.10.10    igb0    U
192.169.12.126 255.255.255.255 192.168.32.96            UGH
192.168.0.0     255.255.254.0   192.168.10.127           UGD
127.0.0.1       255.255.255.255 127.0.0.1        lo0     UH
```

Специальный адрес назначения 0.0.0.0 иногда называется default (см. далее). Упрощённый (без gateway, чтобы не загромождать рисунок) алгоритм работы таблицы маршрутизации представлен на граф-схеме:



Рассмотрим принцип работ таблицы маршрутизации при отправке пакета на адрес 8.8.8.8 с использованием gateway:

```
Запись 8.8.8.8/32 = 8.8.8.8 в таблице не найдена; i--; continue
Запись 8.8.8.8/31 = 8.8.8.8 в таблице не найдена; i--; continue
Запись 8.8.8.8/30 = 8.8.8.8 в таблице не найдена; i--; continue
Запись 8.8.8.8/29 = 8.8.8.8 в таблице не найдена; i--; continue
Запись 8.8.8.8/28 = 8.8.8.0 в таблице не найдена; i--; continue
```

Запись 8.8.8.8/27 = 8.8.8.0 в таблице не найдена; i--; continue

…

Запись 8.8.8.8/0 = 0.0.0.0 в таблице найдена и имеет флаг G, получаем адрес шлюза и выполяем над ним алгоритм сначала.

Запись 192.168.10.1/32 = 192.168.10.1 в таблице не найдена; i--; continue

Запись 192.168.10.1/31 = 192.168.10.0 в таблице не найдена (не совпадает маска); i--; continue

…

Запись 192.168.10.1/24 = 192.168.10.0 в таблице найдена; формируем пакет с IP-адресом назначения 8.8.8.8, передаём пакет на канальный уровень, указывая в качестве назначения канального фрейма адрес 192.168.10.1. На канальном уровне (в случае Ethernet) ARP запросит MAC адрес, соответствующий IP 192.168.10.1 и укажет его в качестве destination MAC.

Соответственно, запись вида 0.0.0.0/0 применяется в самом конце и, следовательно, является маршрутом по умолчанию (default gateway). В рассмотренном примере пакет будет отправлен именно на default gatway.

В ранних версиях операционных систем использовался другой алгоритм поиска маршрутов даже без учёта маски подсети. Одна из его модификаий заключается в том, что последовательно срвнивается адрес назначения с каждой записью в таблице маршрутизации. Если совпадения не обнаружено, на основе класса адреса вычисляется адрес сети и снова последовательно сравнивается с записями в таблице маршрутизации. Если совпадения не обнаружено, пакет отправляется по маршруту default, либо, если таковой отсутствует, отбрасывается.

После определения маршрута ядро обращается к драйверу интерфейса: для ppp (point-to-point) – в туннель, для ethernet – в ARP таблицу и т.д. Если в процессе поиска по таблице маршрутизации назначение пакета выявлено не было, то отправителю посылается ICMP – No route to host.

Поле флаги может принимать например такие значения:

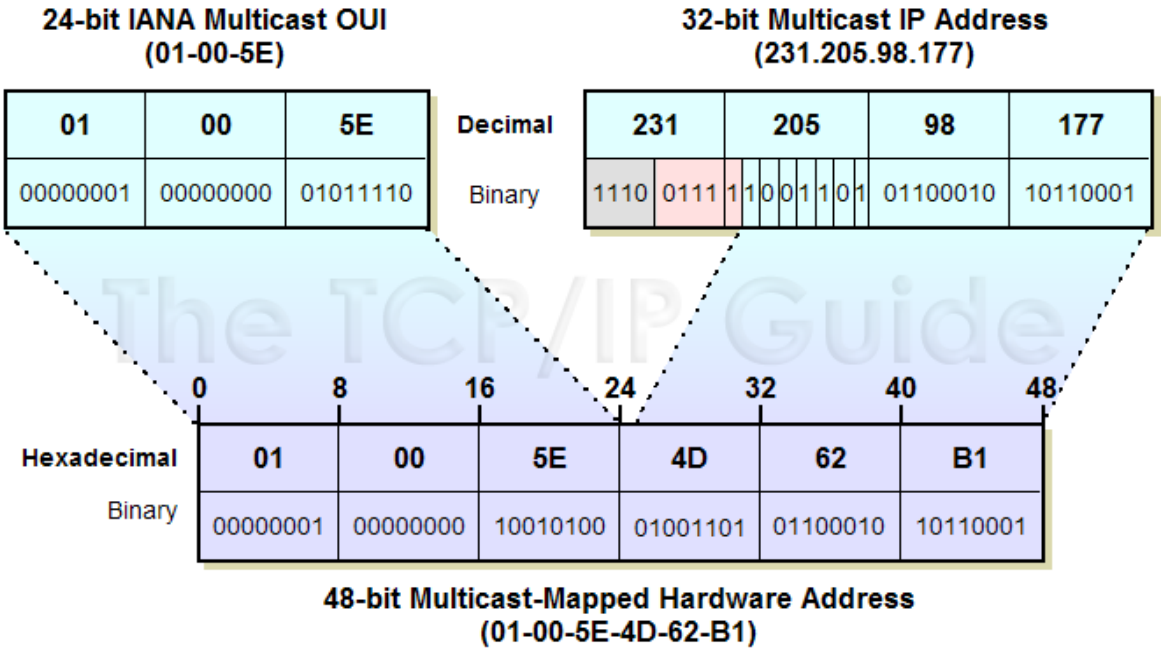
- D – dynamic – запись была создана динамически (например, см. далее ICMP redirect);
- H – host – запись указывает на хост, а не подсеть (в настоящее время, устарел. Для обозначения хоста используется маска /32);
- U – up – маршрут включён;
- G – gateway. Означает, что нужно взять адрес из столбца gateway и выполнить рассмотренный выше алгоритм для него.

В случае, когда система не знает, какой адрес указывать в поле заголовка IP «Source address» (например при отправке в свой Ethernet-сегмент), в него заносится значение столбца «Gateway».

Помимо прочего, существует особенность работы механизма маршрутизации в широковещательных рассылках. Если для широковещательной рассылки используется адрес 255.255.255.255, то такая рассылка работает только в пределах Ethernet сегмента, если же используется адрес сети и широковещательный адрес хоста (в двоичном виде – все единицы), например 192.168.10.255, то на запрос ответят системы даже через глобальную сеть. Однако в реальных системах, широковещательные запросы из глобальной сети запрещаются для защиты от атак вида DDoS Amplification.

В IP сетях, исходя из того, что пакет может обходить большие множества хостов, возможно зацикливание передачи, что приведёт к избыточной нагрузке на сеть из-за неверной конфигурации. Для решения этой проблемы используется механизм TTL (Time-to-Live), который, фактически, реализуется определённым числовым полем в IP заголовке. Стандартное значение TTL для многих систем – 64. Суть работы механизма заключается в следующем: каждый узел, при получении пакета, уменьшает значение поля TTL. Если TTL приняло значение равное нулю, формируется и отправляется в ответ на такой пакет специальный пакет протокола ICMP – TTL expired, что говорит отправителю о том, что пакет не был доставлен адресату. Эту особенность можно использовать для трассировки маршрута до хоста (route tracing).

Групповая рассылка организуется с помощью использования групповых IP адресов. В процессе маршрутизации, из таблицы маршрутизации выбирается исходящее устройство. В рассмотренном ранее примере соответствующая запись есть. Если это устройство – ethernet адаптер, то в общем случае multicast работает следующим образом: если в ARP таблице для этого IP установлен флаг M (multicast), то младшая часть IP адреса помещается в отведённые для Ethernet multicast 23 бита MAC адреса назначения и сформированный таким образом фрейм отправляется в сегмент. Формирование MAC адреса можно проиллюстрировать так:



На уровне IP, чтобы получать multicast запросы, необходимо подписаться на multicast группу. Классическими механизмами маршрутизация multicast не осуществляется, для маршрутизации групповой рассылки используются специальные маршрутизаторы, отслеживающие, на какие multicast группы подписаны абоненты и не создающие избыточного трафика в сети.

Туннелирование

Под туннелированием понимается технология, позволяющая пакеты одного протокола отправлять поверх какой-либо транспортной сети используя другой протокол для их передачи. Строго говоря, туннелирующие протоколы занимают отдельную нишу в сетевом стеке и их нельзя отнести к какому-либо конкретному уровню OSI. На данный момент популярностью пользуются различные туннелирующие протоколы, работающие на разных уровнях модели OSI. Например, GRE (generic routing encapsulation), PPTP (point-to-point tunneling protocol), L2TP (layer-2 tunneling protocol). Также туннели могут быть построены используя прикладные протоколы, такие как DNS, SSH и HTTP или транспортные, например, ICMP.

Транспортные протоколы

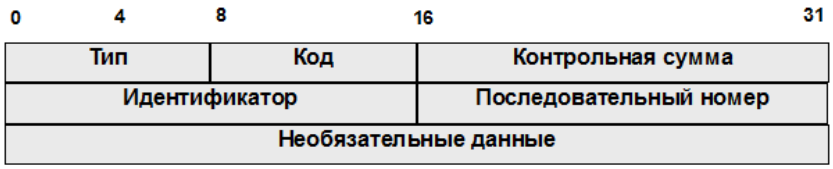
Поверх протокола IP работают различные протоколы: TCP, UDP, ICMP и другие, такие как GRE, AH, ESP, SCTP.

Протокол ICMP

Классически, описание транспортных протоколов начинается с описания TCP или UDP. В данном курсе сначала будет рассмотрен ICMP, чтобы завершить тему маршрутизации и только потом будут рассмотрены TCP и UDP. Протокол ICMP – Internet Control Message Protocol – предназначен для отправки служебных (управляющих) сообщений в IP сетях. Ранее рассматривались наиболее часто встречаемые сообщения: ICMP fragmentation needed, ICMP TTL expired и ICMP no route to host. Кроме рассмотренных сообщений, можно встретить сообщение ICMP redirect, предназначенное для работы механизма перенаправления маршрутов используя протокол ICMP.

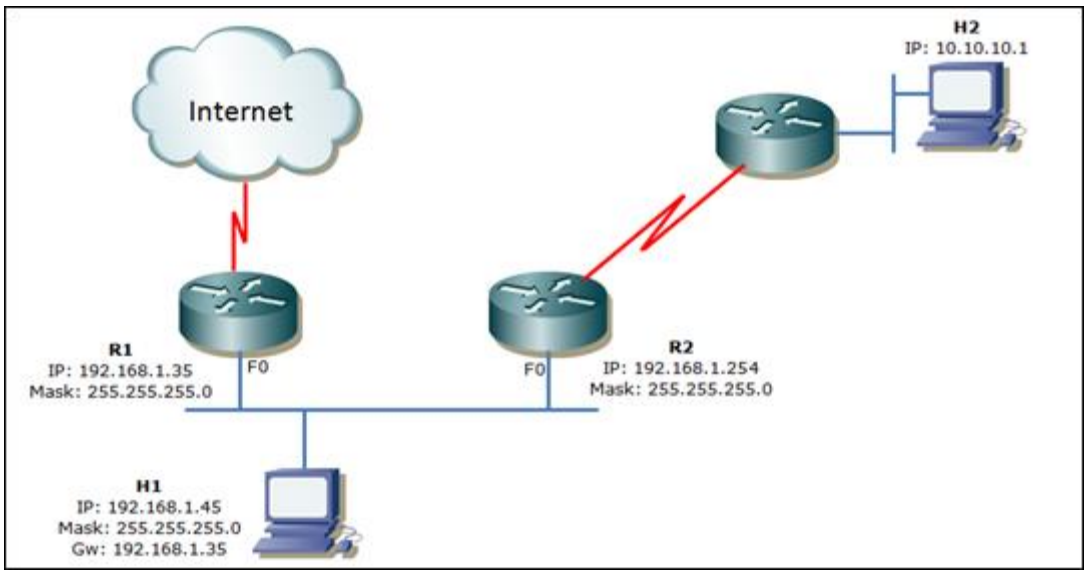
Некоторые возможные значения поля «Тип» и полный формат пакета ICMP представлен ниже:

Тип	Содержание	Тип	Содержание
0	Ответ на эхо	11	Превышено время жизни пакета
3	Получатель недостижим	12	Ошибка параметров в пакете
4	Подавление источника	17	Запрос маски адреса
5	Изменение маршрута	18	Ответ на запрос маски адреса
8	Запрос эха		



Как видно, в пакете есть поле для данных, которое иногда используется для построения ICMP-туннелей через echo request-ы (ping).

Протокол ICMP может использоваться для динамического управления маршрутизацией. В частности, для перенаправления маршрутов. Это реализовано механизмом ICMP redirect, рассметренном ниже:



Хост H1 отправляет пакет с адресом назначения H2 (10.10.10.1). В качестве маршрутизатора по умолчанию в таблице маршрутизации H1 указан R1 (192.168.1.35). При получении такого пакета от H1, R1 может поступить двумя следующими способами:

- осуществить стандартную маршрутизацию – переслать пакет на R2;
- отправить хосту H1 ICMP redirect вида «10.10.10.1/32 -> 192.168.1.254».

Получив такой ICMP redirect, H1 модифицирует свою таблицу маршрутизации и в последствии будет отправлять пакеты сразу на R2.

В таком подходе существует ряд проблем:

- ICMP redirect добавляет не сеть, а конкретные хосты, что приводит к насыщению таблицы маршрутизации большим количеством похожих записей;
- по сети передаётся избыточный трафик по протоколу ICMP;
- возникнут проблемы с Keep-state (см. TCP) соединениями, когда dynamic route перейдёт в состояние «expired»;
- с помощью редиректов злоумышленник может легко «украсть» чужой трафик.

В связи с последней проблемой, ICMP redirect на внешних интерфейсах отклоняется.

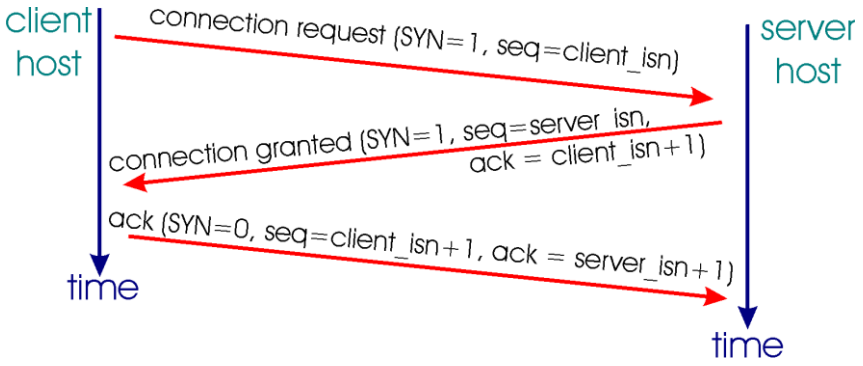
Кроме указанных, у ICMP есть ещё ряд проблем, например, широко известная утилита ping(1) работает с пакетами ICMP Echo request и ICMP Echo reply. Протокол ICMP подразумевает поле данных. В ICMP Echo request/reply, для сохранения маршрута пакета, это поле используется редко, поэтому существуют технологии, позволяющие построить ICMP-туннель. Многие администраторы разрешают ICMP Echo request/reply во внешнюю сеть, что с учётом выше сказанного создаёт реальную угрозу безопасности.

Протокол TCP

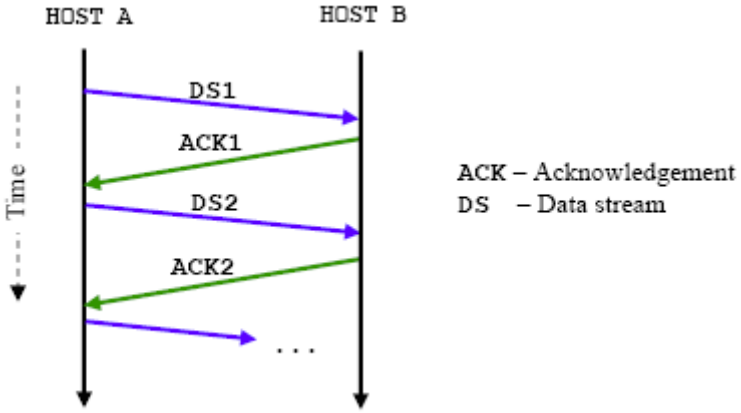
Для протокола TCP существует такое понятие, как соединение. Фактически, соединением называют договорённость сторон о параметрах передачи данных. Процесс установления договорённости называют, соответственно, установлением соедниения. Адресация в протоколе TCP основывается на паре IP:port. При попытке соединения на порт, который никто не слушает, целевая система отправляет отказ в установлении соединения. К особенностям TCP можно отнести:

- установка соединения;
- гарантия передачи;
- корректный порядок принятых данных (FIFO).

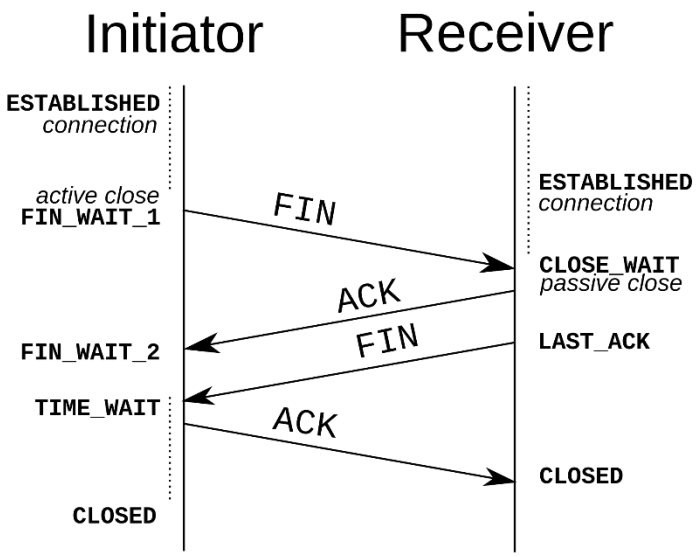
Процесс установления соединения в протоколе TCP выглядит следующим образом:



Передачи данных в рамках установленного соединения:



А закрытия, в свою очередь:

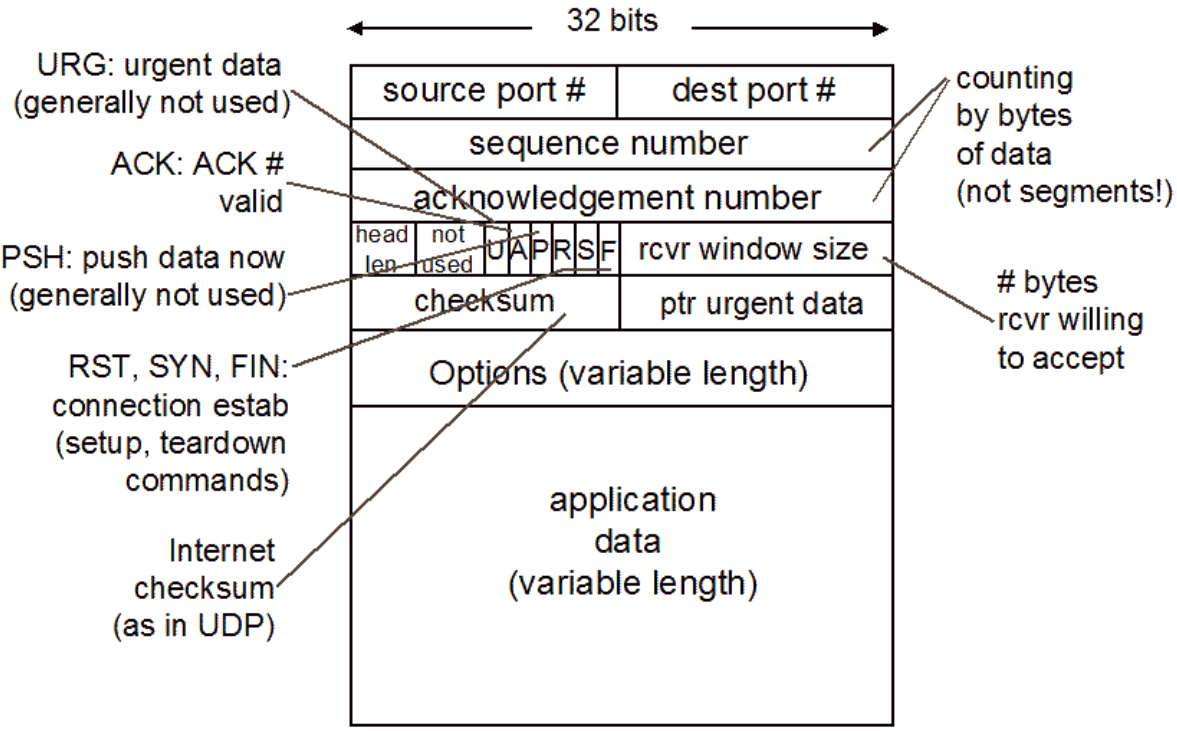


Для корректной работы TCP с рассмотренным ранее механизмом MTU, в протокол TCP был добавлен механизм MSS (Maximum Segment Size), который определяется по следующей формуле:

MSS = MTU – размер заголовка TCP – размер заголовка IP

Напоминаем, что для IPv4 размер заголовка составляет 20 байт, а для IPv6 – недетерминирован, но как минимум составляет 40 байт.

Отправляемые по TCP блоки данных называются TCP-сегментами (либо, окном передачи). Сегменты имеют следующий формат:



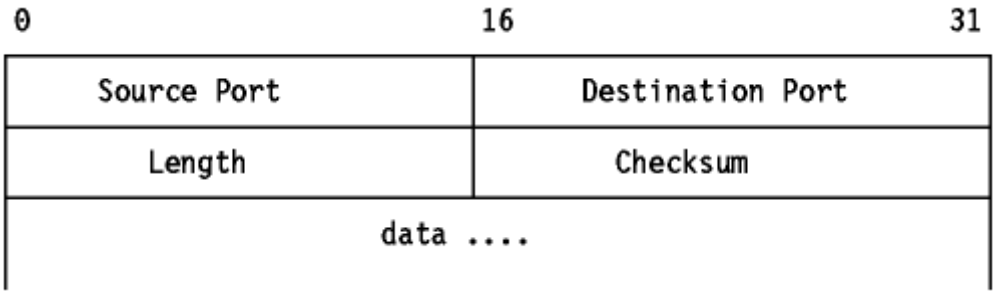
Протокол рассматривался в курсе «Сети ЭВМ и телекоммуникации», а также детально рассматривался в курсе «Сетевые протоколы», поэтому пояснения требуют только некоторые поля. В частности, rcvr window size – это количетство байт, которые может принять отправитель в ответ на свой запрос. Это поле используется для обеспечения механизма плавающего окна.

На каждое окно отправляется ACK. Для уменьшения накладных расходов в протоколе TCP есть механизм TCP Windows Scale Option, позволяющий увеличить размер окна максимально до 1 Гб.

Кроме того, для обеспечения быстродействия при обработке протокола TCP, существует механизм называемый TSO (TCP segmentation offloading). Этот механизм позволяет обрабатывать сегменты на стороне сетевого адаптера.

Протокол UDP

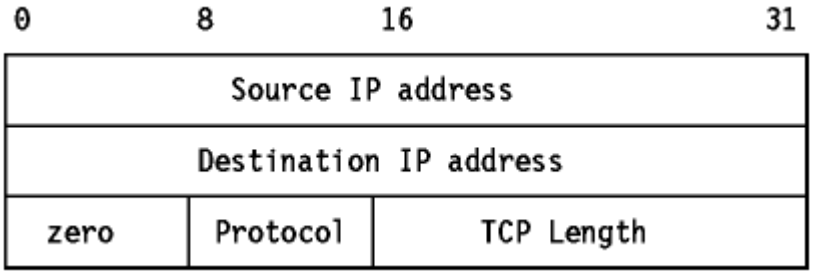
В свою очередь, протокол UDP ориентирован на передачу дейтаграмм. Протокол не предусматривает наличие соединения, не гарантирует доставку данных вообще и порядок доставки данных, в частности. Из преимуществ можно выделить простоту реализации, работы с протоколом, легковесность и поддержку широковещательной рассылки. Формат сегмента UDP прост и представлен на рисунке:



В отличие от TCP, если пакет отправляется по протоколу UDP, а порт никто не слушает, целевая система формирует ICMP Port unreachable, обозначающий, что порт никто не слушает.

Псевдозаголовок TCP/UDP

Для корректной работы транспортных протоколов TCP и UDP в IP сетях существует, так называемый, псевдозаголовок. Это структура данных, физически отсутствующая в пакете и формируемая сетевым стеком операционной системы при работе с TCP и UDP поверх IP. Смысл механизма заключается в том, что с точки зрения TCP/UDP, сегмент адресуется портом назначения, а IP-адрес в адрес сегмента не входит. Такой подход приводит к проблеме, связанной с тем, что приложение, работающее с несколькими адресами (например, со всеми) не может определить, на какой адрес клиент отправлял данные. Для решения проблемы ядро формирует дополнительную структуру, которая позволяет получить IP адреса назначения и отправителя. Формат структуры представлен на рисунке:



Поле zero – зарезервированное поле, заполненное нулями. Protocol и длина сегмента (TCP/UDP length), а также адреса назначения и отправителя заносятся в структуру из расмотренного ранее заголовка IP.

Сетевые экраны и раздел пакетные фильтры

Традиционно этот помещают в описание сетевых протоколов и программ, работающих на сетевом уровне. Однако, с позиций администрирования корпоративных сетей, уместнее давать описание работы firewall говоря именно о транспортных протоколах, т.к. в современных корпоративных сетях сетевые экраны работают как-раз на транспортном уровне.

Используя информацию для протокола транспортного уровня, такие сетевые экраны и пакетные фильтры позволяют применять правила в зависимости от состояния соединения конечных точек передачи. Эти сетевые экраны называют «stateful» (от англ. state – состояние). Получив пакет установления соединения, такой экран запоминает, какой именно клиент и с каким хостом устанавливал соединение. Ответы, пришедшие от целевого хоста в ответ на клиентские будут обработаны корректно без явного указания в конфигурационном файле сетевого экрана. А часто (например, при динамическом выборе портов) указать их вручную вообще не представляется возможным, поэтому единственный способ работы с такими подключениями – использование stateful firewall.

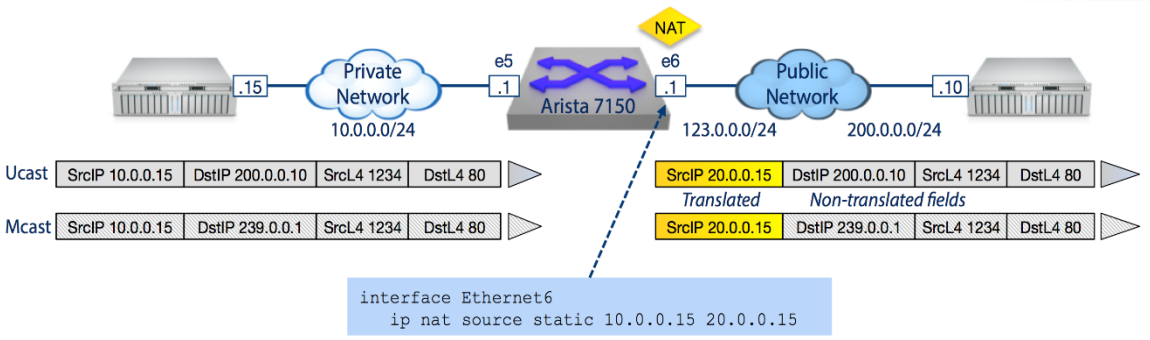
Описывая работу сетевых экранов, необходимо дать описание такого широко используемого механизма, как NAT (network address translation). Данный механизм выполняет изменение адресов и портов отправителя и адресата по заданным правилам. Наиболее частые случаи NAT:

- обычный NAT (или source NAT) – трансляция source адресов, чтобы обеспечить доступ, например, локальной сети в глобальную;
- destination NAT – позволяет предоставить доступ к внутренним системам из глобальной сети. Наиболее встречаемая разновидность – port mapping (или port

forwarding), при которой порты сетевого экрана отображаются на разные адреса и разные порты внутренних систем;

- source NAT + destination NAT – это комбинация из двух предыдущих механизмов трансляции адресов.

Проиллюстрируем на примере классический NAT:



Клиент отправляет пакеты на адрес сервера 200.0.0.10; в ходе маршрутизации выясняется, что для отправки на этот адрес используется шлюз 10.0.0.1 и IP пакеты с адресом назначения 200.0.0.10 отправляются в Ethernet фреймах с адресом шлюза в качестве назначения. Шлюз, получив такой фрейм производит декапсуляцию и видит, что source IP = 10.0.0.1, а destination IP = 200.0.0.10 и выполняет трансляцию адресов по правилу «от всех хостов из сети 10.0.0.1/24 -> в адрес 20.0.0.15», одновременно добавляя в таблицу трансляции соответствующую запись. Поэтому до конечного сервера пакеты доходят уже с source IP 20.0.0.15. Сервер, отвечая на эти пакеты, формирует IP пакеты с адресом назначения 20.0.0.15, получив которые маршрутизатор пройдет по таблице трансляции, изменит адрес на 10.0.0.5 и отправит клиенту.

Прикладные программы и протоколы

Серверное и клиентское программное обеспечение чаще всего использует транспортный уровень для обмена данными по сети. Существуют устоявшиеся номера портов, закреплённые за различными службами. Некоторые порты описаны стандартами организации IANA (например, 22 для сервера ssh), некоторые – устоявшиеся договорённости (например, 3128 и 8080 для http proxy). Относительно полный список таких портов можно посмотреть в файле /etc/services (для UNIX) систем и %WINDIR%\system32\drivers\etc\services (для Windows). Этот файл имеет следующую структуру:

```
имя_службы  порт/протокол     дополнительные имена     #комментарий
```

В частности, этот файл содержит такие записи как:

```
ftp             21/sctp   #File Transfer [Control]
ftp             21/tcp    #File Transfer [Control]
ftp             21/udp    #File Transfer [Control]
ssh             22/sctp   #Secure Shell Login
ssh             22/tcp    #Secure Shell Login
ssh             22/udp    #Secure Shell Login
domain          53/tcp    #Domain Name Server
domain          53/udp    #Domain Name Server
http            80/sctp   www www-http #World Wide Web HTTP
http            80/tcp    www www-http #World Wide Web HTTP
```

```
http            80/udp     www www-http #World Wide Web HTTP
```

В первом столбце указано название службы, которая работает с этим портом, во втором – порт и протокол, в третьем и остальных – дополнительные названия служб.

Данный файл можно использовать с любым программным обеспечением, которое использует системную функцию формирования сетевого адреса. Например, можно использовать рассматриваемую далее программу telnet для установления соединения на порт web-сервера без явного указания его номера:

```
$ telnet 127.1 www
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
```

В данном примере, программа успешно установила соединение на порт 80 системы с адресом 127.0.0.1.

Как уже говорилось, большинству программ для работы в сети требуется связка адрес:порт. В силу того, что философия UNIX предполагает использование простых текстовых потоков для передачи данных и чтение/запись в stdin/stdout, было бы разумным реализовать программу, которая связывает сетевое соединение с программами-фильтрами. Такой программой стал супердемон inetd. Эта программа слушает несколько сокетов и как только на один из сокетов приходит попытка подключения, она выбирает сервис, обслуживающий этот сокет и запускает его связывая стандартный ввод и вывод с подключением. После завершения работы программы, inetd продолжает слушать сокет. Этот подход позволил довольно просто писать сетевые программы в UNIX на таких языках, как язык интерпретатора bourne shell.

Протокол TELNET

Программа telnet предназначена для работы с другим хостом по протоколу TELNET (RFC854). Назначение этого протокола – обеспечение возможности двунаправленной передачи восьмибитных данных между хостами. Протокол задумывался как протокол связи process-process или terminal-terminal, но широкое распространение получил только во втором применении. Протокол подразумевает существование так называемого «сетевого виртуального терминала», который избавляет клиента и сервера от необходимости учитывать специфику работы реальных, физических терминалов. Этот терминал поддерживает некоторые типовое функции, такие как interrupt process, are you there, erase character, erase line и другие. По умолчанию, протокол TELNET подразумевает посимвольную передачу данных. Один символ зарезервирован под escape-символ «forward backspace» – это символ с кодом 0xff. В частности, в кодировке CP1251, этот код соответствует русскому символу «я», что приводит к проблемам использования протокола для передачи русскоязычных данных в этой кодировке. Данный протокол в настоящее время не используется, т.к. имеет ряд проблем. Например, протокол не поддерживает шифрование данных, следовательно, использовать его по прямому назначению не безопасно.

Для работы с этим протоколом были разработаны программы telnet (клиент) и telnetd (демон), реализующие соответствующие функции. Программа telnet имеет особенность, что при подключении на нестандартный порт (не на 23/tcp), клиент не выполняет все стандартные TELNET инициализации, а просто устанавливает TCP соединение на указанный порт. Эту особенность удобно использовать, чтобы протестировать возможность установления TCP соединений и даже произвести отладку plaintext протоколов, например HTTP (см. далее).

Протокол FTP

Данный протокол (File Transfer Protocol) предназначен для передачи файлов по сети. Протокол подразумевает наличие управляющего соединения и нескольких соединений для передачи данных (control и data, соответственно). В рамках каждого data соединения происходит передача одного файла.

Кроме этого, существуют два режима работы FTP: активный и пассивный. По умолчанию используется активный режим. В этом режиме клиент подключается к серверу и запрашивает обратное подключение для передачи. для этого используется команда PORT. Например:

```
PORT 10,1,242,250,180,46
```

Эта команда просит сервер подключиться к клиентской системе с IP адресом 10.1.242.250 на порт 46126. Значение порта рассчитывается по формуле: 256 * a + b, в нашем случае a = 180, a b = 46.

В пассивном режиме, соединения для передачи данных инициируются клиентом. Очевидно, существует проблема, связанная с работой ftp в условиях NAT. Если клиент находится в не маршрутизируемой сети (как в рассмотренном примере) и получает доступ к серверу через NAT, по умолчанию работать FTP не сможет. Решением данной проблемы является использование пассивного режима. Возникает также проблема, если и сервер, и клиент находятся за NAT. Обобщением проблемы можно назвать firewall. Протокол ftp простой протокол и решение проблем с сетевым экраном простое: добавление поддержки ftp на сетевой экран, который при фильтрации будет осуществлять необходимые модификации пакетов, чтобы обеспечить взаимодействие сторон по ftp.

Кроме этого, существуют два режима работы ftp – ASCII и binary. Первый используется по умолчанию и предназначен для передачи символов в кодировке ASCII. Работая в ASCII режиме, ftp вносит коррективы в EOL последовательности (это последовательности, обозначающие конец строки). Как известно, в Windows в качестве end-of-line последовательности используется "\r\n", в Macintosh – "\r", а в UNIX – "\n". В процессе работы, перед передачей, клиент ftp перекодирует текст приводя его к сетевому виду (UNIX), а перед предоставлением пользователю – к виду пользовательской системы. Поэтому при передаче не ASCII текстов или бинарных файлов возникают проблемы. Для их решения был разработан режим binary, передающий данные без изменения, осуществляя только преобразование в network endian.

Пример работы с ftp-клиентом:

```
$ ftp ftp.freebsd.org

Trying 77.88.40.106:21 ...

Connected to ftp.geo.freebsd.org.

220 This is ftp0.ydx.freebsd.org - hosted at Yandex.

Name (ftp.freebsd.org:server): Anonymous

331 Please specify the password.

Password:

230-

230-This is ftp0.isc.FreeBSD.org, graciously hosted by Yandex.

230-

230-FreeBSD files can be found in the /pub/FreeBSD directory.
```

```
230-
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> ls
229 Entering Extended Passive Mode (|||60441|).
150 Here comes the directory listing.
-rw-r--r--    1 ftp      ftp          5430 Jul 19 04:50 favicon.ico
-rw-r--r--    1 ftp      ftp           665 Jul 19 04:43 index.html
drwxr-xr-x    3 ftp      ftp             3 Jul 19 04:34 pub
226 Directory send OK.
ftp> cd pub
250 Directory successfully changed.
ftp> pwd
Remote directory: /pub
ftp> bin
200 Switching to Binary mode.
ftp> get favicon.ico
local: favicon.ico remote: favicon.ico
229 Entering Extended Passive Mode (|||51303|).
150 Opening BINARY mode data connection for favicon.ico (5430 bytes).
100% |*********************************|  5430       6.89 MiB/s    00:00
ETA
226 Transfer complete.
5430 bytes received in 00:00 (386.10 KiB/s)
```

Было осуществлено анонимное подключение на сервер, в качестве пароля используется e-mail (либо, любая строка с символом ‘@’). На сервере были выполнены команды получения листинга файлов (ls), смены директории (change directory), получения текущего имени каталога (pwd), перехода в binary режим (bin) и получения файла (get).

Протокол FTP является plaintext протоколом и позволяет прямое взаимодйствие:

```
$ telnet ftp.freebsd.org 21
Trying 77.88.40.106...
Connected to ftp.geo.freebsd.org.
Escape character is '^]'.
220 This is ftp0.ydx.freebsd.org - hosted at Yandex.
USER Anonymous
331 Please specify the password.
```

```
PASS admin@cs.ifmo.ru
230-
230-This is ftp0.isc.FreeBSD.org, graciously hosted by Yandex.
230-
230-FreeBSD files can be found in the /pub/FreeBSD directory.
230-
230 Login successful.
HELP
214-The following commands are recognized.
 ABOR ACCT ALLO APPE CDUP CWD  DELE EPRT EPSV FEAT HELP LIST MDTM MKD
 MODE NLST NOOP OPTS PASS PASV PORT PWD  QUIT REIN REST RETR RMD  RNFR
 RNTO SITE SIZE SMNT STAT STOR STOU STRU SYST TYPE USER XCUP XCWD XMKD
 XPWD XRMD
214 Help OK.
CWD pub
250 Directory successfully changed.
PWD
257 "/pub"
SIZE favicon.ico
213 5430
```

Была выполнена аутентификация, запрос возможностей сервера, смена директории в /pub, запрос имени текущей директории и запрос размера файла favicon.ico.

Программы удалённого управления rlogin, rsh и ssh

Для удалённого управления хостами были разработаны более специализированные, чем telnet, программы.

Программа rlogin представляет из себя login на удалённую систему. Перед началом работы требуется произвести аутентификацию пользователя. Фактически, программа открывает удалённую терминальную сессию. Программа поддерживает escape-последовательности, например, "~." для выхода и "~^Z" для приостановки удалённого сеанса. Для разграничения прав подключения с помощью rlogin существует файлы /etc/hosts.equiv и ~/.rlogin, в которых перечисляются хосты, с которых можно осуществлять подключение, а также возможно указание пользователя или группы пользователей, которыми можно будет зайти в систему. Программу rlogin специальными ключами можно настроить так, чтобы escape-последовательности игнорировались и предоставлялось просто TCP подключение, аналогично рассмотренной утилите telnet.

Программа rsh предназначена для исполнения команды, переданной в аргументах на удалённом хосте:

```
$ rsh host cat file1 '>>' file2
```

Примечательно, что команда запускается в полноценном шелле на удалённой системе. Корректно обработав перенаправление потоков, данная команда выполнит на хосте host

добавление содержимого файла file1 к содержимому файла file2. Если программе rsh не указать аргументы, поведение будет аналогично утилите rlogin. К недостаткам этих утилит можно отнести отсутствие шифрования.

В современных сетях наиболее широкое распространение получил протокол SSH (secure shell), который имеет большой функционал для удалённого управления системами и избавлен от проблем рассмотренных ранее протоколов. Протокол SSH имеет две (SSH 1.x и SSH 2.x) несовместимых между собой ветки версий, первая из которых сильно упрощена. Достоинством второй ветки является улучшение алгоритмов безопасности, добавление обмена ключами по Диффи-Хеллману, несколько каналов передачи в рамках одного соединения и другие.

Устаревшая форма запуска ssh клиента подразумевает использование ключа –l для указания имени пользователя. В современном виде, имя пользователя указывается перед именем хоста и отделяется от него символом '@':

```
$ ssh -l user hostname
```

```
$ ssh user@hostname
```

В домашнем каталоге пользователя есть подкаталог ~/.ssh, в котором интерес представляют файлы id_rsa, id_rsa.pub, authorized_keys и known_hosts. Первые два – ключи шифрования (приватный и публичный, соответственно), сгенерированных используя RSA. Кроме RSA, SSHv2 поддерживает и другие типы шифрования, например, DSA. Третий файл содержит в себе публичные ключи, используя которые можно подключиться к данной системе, а четвёртый файл содержит fingerprint-ы хостов, к которым уже осуществлялось подключение. Эти «отпечатки» используются для защиты от атак типа man-in-the-middle, запрещая подключение, если указываемый сервером fingerprint не совпадает с имеющимся у клиента.

Клиент ssh имеет также возможность запуска команды, указанной в аргументах, аналогично rsh:

```
$ ssh host cmd
```

Помимо этого, существует ещё много полезных функций. Среди них выделяются:

- сжатие потока передаваемых данных:
  - `$ ssh -C host`
- передача протокола X11 и протокола аутентификации X11, которые можно комбинировать:
  - `$ ssh -X host`
  - `$ ssh -Y host`
  - далее достаточно установить правильно переменную DISPLAY=host:display или DISPLAY=host:display.screen и приложения, использующие X11 будут выводить на наш X-сервер;
- форвардинг портов через SSH-туннель:
  - локальный форвардинг – соединения на локальный порт 1488 будут отправляться в SSH-туннель и удалённая система будет подключаться к хосту 1.4.8.8 на порт 666:
    `$ ssh -L 1488:1.4.8.8:666 host`
  - удалённый форвардинг – соединения на удалённый порт 2007 будут отправляться в SSH-туннель и локальная (наша) система будет подключаться к emo.ru на порт 80:
    `$ ssh -R 2007:emo.ru:80`

Кроме этого есть возможность копирования файлов через SSH:

```
$ scp host:file file
```

Также поддерживается эмуляция FTP средствами sftp или эмуляция файловой системы средствами sshfs, что позволяет обеспечить удобный доступ к файлам удалённой системы.

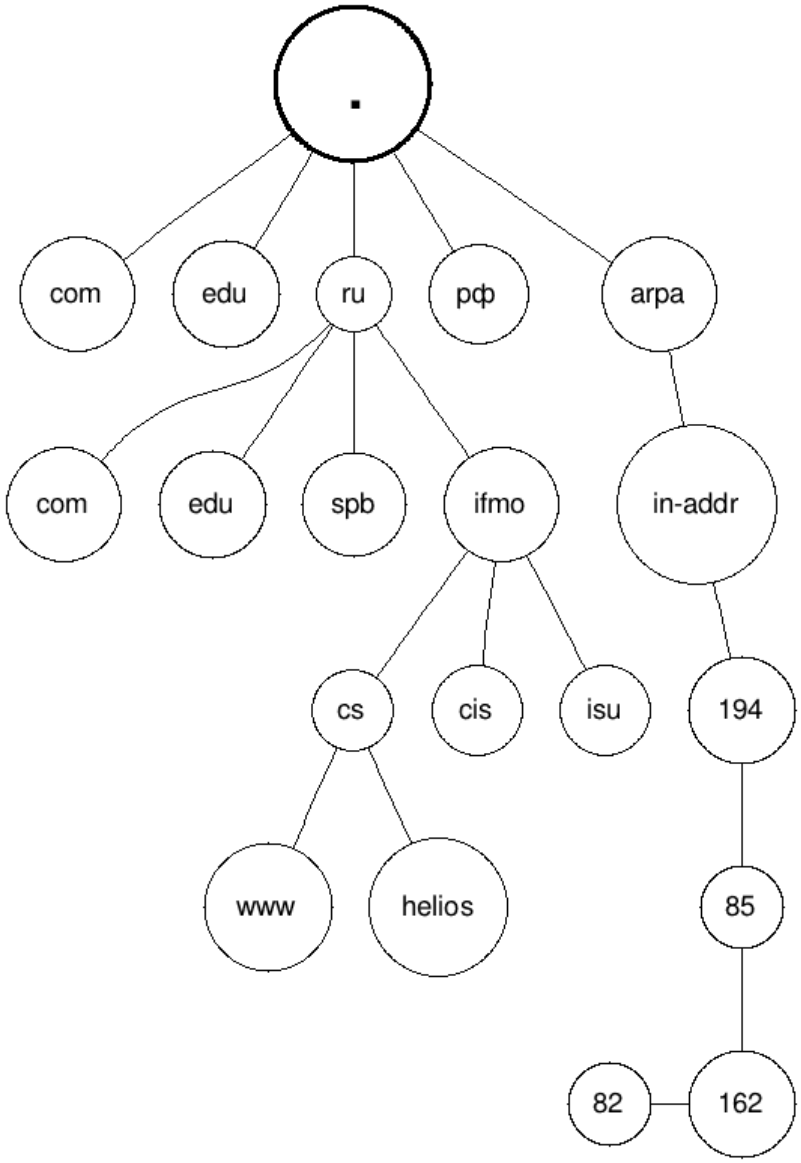
Распределённая система доменных имён (DNS)

Для удобства обращения к компьютерам в сети и для идентификации хостов по именам традиционно используется механизм разрешения имён hosts. Механизм заключается в том, что в специальном файле (для UNIX - /etc/hosts, для Windows - %WINDIR%\system32\drivers\etc\hosts) содержится список соответствий IP-адресов понятным именам. Например, строка вида

```
127.0.0.1 localhost myserver
```

означает, что при обращении к имени localhost или myserver, ОС будет обращаться к хосту с адресом 127.0.0.1.

Однако, такой подход не подходит для использования в корпоративных сетях – требовалось бы много затрат на поддержание когерентности таких локальных файлов на всех корпоративных узлах. И, тем более, не подходит для использования в глобальной сети. Поэтому была разработана распределённая система доменных имён DNS (distributed name system). Система предполагает, что отдельные домены имеют строгую иерархию, но администрируются и предоставляются различными серверами. Это позволяет, во-первых, снизить нагрузку с центральных серверов этой системы, во-вторых, обеспечить простоту управления собственным доменом (не требуется согласовывать изменения с администраторами центральных серверов, достаточно изменить свою доменную зону и дождаться автоматического обновления кешей). Корневой домен «.» (точка) обслуживается центральными серверами. Большинство этих центральных (или корневых) серверов также распределены используя технологию IP anycast, что позволяет построить отказоустойчивое решение и лучше распределить нагрузку. Иерархия доменов выглядит следующим образом:



Как видно из картинки, домены на разных уровнях могут повторяться. Интуитивно понятно, что www.vk.com и www.zhmylove.ru – разные хосты.

Доменные имена разделяются точкой. Причем, могут быть как полные доменные имена (заканчиваются корневым доменом и называются Fully Qualified Domain Names или FQDN), например, «helios.cs.ifmo.ru.». И могут быть относительные: «helios.cs.ifmo.ru». Полные доменные имена всегда обозначают конкретный хост, а относительные теоретически могут обозначать множество хостов. Относительное доменное имя преобразуется в FQDN добавлением точки в конец, либо добавлением одного или нескольких доменов поиска. На клиентской операционной системе есть механизм разрешения (resolving) имён. В Windows он настраивается в свойствах каждого сетевого адаптера, а в UNIX – в файле /etc/resolv.conf. Если резолвер имеет домен поиска, то относительные имена он также будет искать в нём. Пусть файл /etc/resolv.conf имеет такое содержимое:

```
search ifmo.ru
nameserver ns.ifmo.ru
nameserver ns2.ifmo.ru
```

В файле указан домен поиска – ifmo.ru и два сервера доменных имён, к которым будут отправляться запросы. Директиве search можно указать несколько доменов для поиска. В

устаревших системах директива search не поддерживалась, вместо неё использовалась директива domain, которая поддерживает только один поисковый домен.

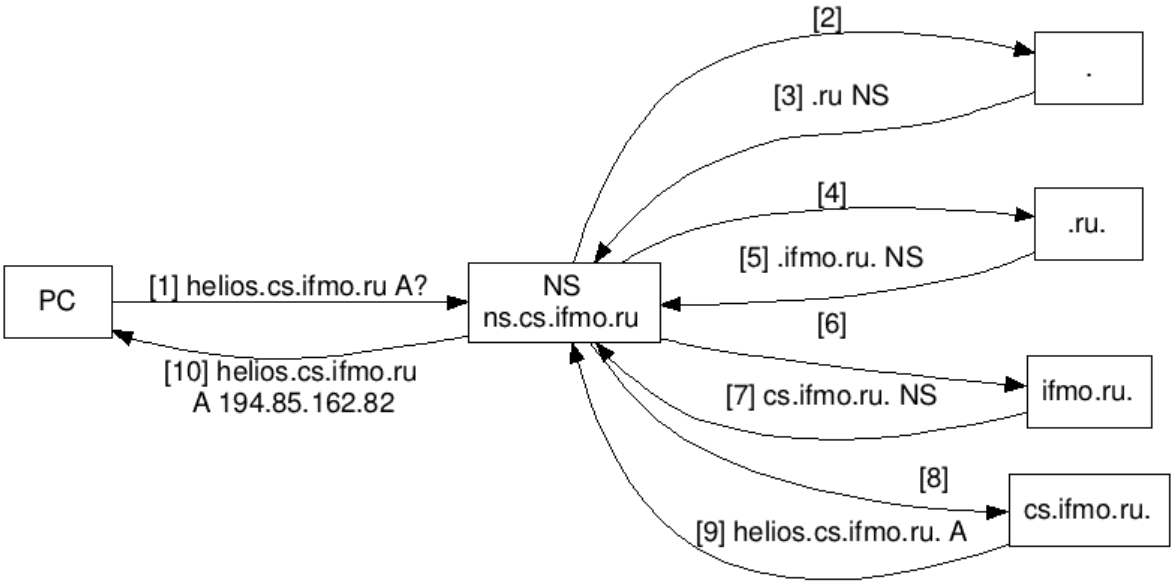
В этом случае при обращении к «ya.ru.», будет запрошен только один конкретный домен, а при обращении к helios.cs будет запрошено два: «helios.cs.» и «helios.cs.ifmo.ru.».

Существуют различные типы запросов DNS. Наиболее часто запрашивается соответствие IP адреса некоторому имени (для IPv4 такой запрос называется A, для IPv6 - AAAA), есть также обратный запрос – запрос имени по известному IP адресу. Такой запрос называется PTR. Его работа основывается на специальном домене «in-addr.arpa.». Запрашиваемый IP адрес разворачивается, т.к. младшие октеты IP адреса соответствуют меньшей сети, а в DNS меньшей единице хостов соответствуют старшие домены. Это можно наглядно продемонстрировать: ясно, что 192.168.0.2 и 192.168.0.4 почти наверняка принадлежат одной сети; в то же время aqua.cs.ifmo.ru и terra.cs.ifmo.ru – тоже. Развёрнутый IP адрес объединяется через стандартный разделитель – точку и к нему добавляется суффикс in-addr.arpa. Пусть необходимо запросить адрес 194.85.162.82, адрес преобразуется к доменному имени «82.162.85.194.in-addr.arpa.» и будет запрошена запись типа PTR у системы DNS. В ответ на такой запрос будет получен ответ вида: «82.162.85.194.in-addr.arpa. PTR helios.cs.ifmo.ru.», что означает, что за этим IP закреплено имя «helios.cs.ifmo.ru.». Следует учитывать, что для администрирования такой зоны требуется иметь подсеть класса C (/24), а такими сетями владеют немногие администраторы доменов. Поэтому не исключены ошибки, связанные с неактуальностью данных. Получение записи PTR совсем не означает, что преобразование сработает в обратную сторону.

Для каждого узла в системе DNS может быть неограниченное число записей одного типа. Существует большое множество типов записей. Наиболее известные:

- SOA – Start of Authority (см. далее) ;
- NS – nameserver – показывает, где искать имя;
- A – показывает соответствие имени IP адресу;
- AAAA – соответствие имени IPv6 адресу;
- PTR – соответствие IP адреса имени;
- MX – mail exchanger – указывает на почтовый сервер (см. SMTP);
- TXT – несёт в себе произвольную текстовую информацию;
- CNAME – canonical name – или привязка дополнительных имён к основному;
- NULL – запись позволяет передавать любые бинарные данные и используется крайне редко оторванными от реальности индивидами для построения VPN поверх системы DNS.

Последовательность запроса IP адреса хоста helios.cs.ifmo.ru с клиентского компьютера можно продемонстрировать на схеме:



Первый запрос («helios.cs.ifmo.ru A?») в этой системе отправлен серверу, указанному в настройках резолвера клиентской системы. Этот запрос рекурсивный. Это означает, что получатель, если не знает ответа на этот запрос, должен попытаться его выяснить и отправить запрос далее. В нашем случае он отправлен на ns.cs.ifmo.ru. Пусть NS ничего не знает о запрашиваемом домене. Классическая схема определения имени будет работать как показано. У NS есть указание на корневые серверы, которым от отправит запрос от клиента [2]. Корневые серверы тоже не знаю о запрашиваемом домене (helios.cs.ifmo.ru), но отвечают NS-у, что домен .ru обслуживается таким-то сервером. NS повторяет эти шаги до тех пор, пока не будет получен ответ, что helios.cs.ifmo.ru соответствует такому-то IP адресу, либо ответ NXDOMAIN, означающий, что запись не найдена. Получив такой ответ, он будет отправлен клиенту [10]. В рассмотренном примере клиенту был отправлен ответ вида «helios.cs.ifmo.ru A 194.85.162.82».

Особенностью системы DNS является кеширование. Каждый, полученный системой NS, ответ кэшируется. Поэтому алгоритм не будет выполняться каждый раз при запросах доменных имён из, например, домена ifmo.ru, а запросы будут сразу отправляться серверу доменных имён, обслуживающих эту доменную зону. Кроме того, для уменьшения нагрузки на систему DNS существует механизм форвардинга запросов. Если NS не знает куда отправить запрос и у него указаны форвардеры (forwarders), он перешлёт запрос им. Такая схема часто используется для отправки неизвестных запросов на DNS серверы провайдера. Встречаются исключительно кэширующие доменные серверы, которые не предоставляют информации о каких-либо зонах, исключительно занимаясь форвардингом и кешированием запросов.

Для работы с системой DNS существует довольно много утилит. В Microsoft Windows – это программа nslookup, в UNIX – программы dig, nslookup, host, drill и другие.

Пример работы программы host, использованной для запроса записи типа NS для корневого домена:

```
$ host -t ns .
. name server i.root-servers.net.
. name server b.root-servers.net.
. name server m.root-servers.net.
. name server g.root-servers.net.
. name server e.root-servers.net.
```

```
. name server h.root-servers.net.
. name server f.root-servers.net.
. name server j.root-servers.net.
. name server d.root-servers.net.
. name server c.root-servers.net.
. name server a.root-servers.net.
. name server k.root-servers.net.
. name server l.root-servers.net.
```

Программа nslookup имеет интерактивный и не интерактивный варианты использования. Во втором результат показывается сразу, но возможно выполнение только одного запроса (аналогично утилите host), а в интерактивном – возможно выполнение нескольких запросов. Это можно проиллюстрировать следующим примером:

```
$ nslookup
> helios
Server:         192.168.10.1
Address:        192.168.10.1#53

Name:   helios.cs.ifmo.ru
Address: 192.168.10.10
> set type=soa
> gmail.com
Server:         192.168.10.1
Address:        192.168.10.1#53

Non-authoritative answer:
gmail.com
        origin = ns1.google.com
        mail addr = dns-admin.google.com
        serial = 2012061200
        refresh = 21600
        retry = 3600
        expire = 1209600
        minimum = 300

Authoritative answers can be found from:
gmail.com       nameserver = ns1.google.com.
gmail.com       nameserver = ns4.google.com.
```

```
gmail.com       nameserver = ns2.google.com.
gmail.com       nameserver = ns3.google.com.
ns1.google.com  internet address = 216.239.32.10
ns2.google.com  internet address = 216.239.34.10
ns3.google.com  internet address = 216.239.36.10
ns4.google.com  internet address = 216.239.38.10
```

Пример работы утилиты dig показан ниже. Ключ –t означает тип запрашиваемой записи, а AXFR – специальный тип, запрашивающий передачу (трансфер) всей зоны.

```
$ dig @127.2 zhmylove.ru -t AXFR


; <<>> DiG 9.9.2-P1 <<>> @127.2 zhmylove.ru -t AXFR
; (1 server found)
;; global options: +cmd
zhmylove.ru.              10800   IN      SOA     ns.zhmylove.ru. support.zhmylove.ru.
201412260 86400 7200 2419200 3600
zhmylove.ru.            10800   IN      MX      10 mail.zhmylove.ru.
zhmylove.ru.          10800   IN     TXT     "v=spf1 a ip4:188.65.66.173 mx:mail.zhmylove.ru
-all"
zhmylove.ru.            10800   IN      A       188.65.66.173
zhmylove.ru.            10800   IN      NS      ns.zhmylove.ru.
zhmylove.ru.            10800   IN      NS      ns2.zhmylove.ru.
zhmylove.ru.            10800   IN      NS      ns3.zhmylove.ru.
zhmylove.ru.            10800   IN      NS      ns4.zhmylove.ru.
_dmarc.zhmylove.ru.          10800      IN            TXT          "v=DMARC1\;  p=reject\;
rua=mailto:support@zhmylove.ru"
_domainkey.zhmylove.ru. 10800   IN      TXT     "o=-"
_adsp._domainkey.zhmylove.ru. 10800 IN  TXT     "dkim=all"
mail._domainkey.zhmylove.ru.    10800    IN             TXT                      "k=rsa\;
p=MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQDZgoUGCzm+3LTxCXKK9vbEOJ3LX2S/jHbCSUhdpjhcqt4cg2W3qogQUju
/ek73TPeUIjvWfJQniTaFzYlqTk2MDYt9JPZCkR/j39SCybA0QIxogIe6s4dEXzt0AQXdeAPGTQIGShAOyDPyfYJEXHuWsOkE
eBaWsxytibEAo/kA7wIDAQAB"
mail.zhmylove.ru.       10800   IN      A       188.65.66.173
ns.zhmylove.ru.         10800   IN      A       188.65.66.173
ns2.zhmylove.ru.        10800   IN      A       188.65.66.173
ns3.zhmylove.ru.        10800   IN      A       188.65.66.173
ns4.zhmylove.ru.        10800   IN      A       188.65.66.173
ns.tun.zhmylove.ru.     10800   IN      A       188.65.66.173
srv.zhmylove.ru.        10800   IN      A       188.65.66.173
tun.zhmylove.ru.        10800   IN      NS      ns.tun.zhmylove.ru.
www.zhmylove.ru.        10800   IN      CNAME   srv.zhmylove.ru.
zhmylove.ru.              10800   IN      SOA     ns.zhmylove.ru. support.zhmylove.ru.
201412260 86400 7200 2419200 3600
;; Query time: 0 msec
```

```
;; SERVER: 127.0.0.2#53(127.0.0.2)
;; WHEN: Tue Dec 30 14:20:03 2014
;; XFR size: 22 records (messages 1, bytes 854)
```

В данном выводе сочетаются одновременно много механизмов DNS и утилиты dig. Во-первых, для запроса был вручную указан сервер по IP адресу: 127.2. Для трансфера была запрошена зона zhmylove.ru. В зоне присутствуют наиболее часто используемые записи. Кроме того, в данной зоне есть записи для работы SPF, DMARC, DOMAINKEY и DKIM (см. SMTP). Также в зоне есть делегирование поддомена: tun.zhmylove.tk делегируется серверу ns.tun.zhmylove.ru. Причем, очевидно, IP адрес сервера ns.tun.zhmylove.ru знает только этот сервер, но его не получить классическим способом. Эту проблему можно формально описать так: клиент хочет запросить запись типа CNAME с именем s123.tun.zhmylove.ru. По описанному ранее алгоритму запрос приходит серверу DNS, отвечающему за домен zhmylove.ru. Сервер говорит, что за tun.zhmylove.ru отвечает ns.tun.zhmyove.ru, но чтобы определить его адрес, требуется обратиться к tun.zhmylove.ru – возникает зацикливание. Для решения этой проблемы существуют, так называемые, glue-записи. Это запись для поддоменов. В показанном примере – это ns.tun, указывающая на IP адрес DNS сервера поддомена. Обратим внимание, что иногда glue запись перестаёт совпадать с основной. В этом случае, требуется её вручную обновить.

Говоря проще, glue записи нужны чтобы заранее сообщить резолверу IP-адреса доменных серверов, отвечающих за делегированную зону.

Кроме типов записей, существуют также классы записи. Использовались только три класса: IN (INET), CH (CHAOSNET) и HS (HESIOD). Классы задумывались для того, чтобы система DNS могла работать не только в сети Интернет. В настоящее время классы CH и HS не используются. Однако, в некоторых случаях записи CHAOSNET можно использовать для получения версии ПО bind:

```
$ host -t txt -c ch version.bind sunrise.cs.ifmo.ru
Using domain server:
Name: sunrise.cs.ifmo.ru
Address: 194.85.162.83#53
Aliases:


version.bind descriptive text "name server"
```

Вернёмся к рассмотренному примеру. Зона состоит из записей. Каждая запись обязательно имеет следующий формат:

```
[ИМЯ] [TTL] [CLASS] TYPE VALUE
```

В квадратные скобки обозначают, что в конфигурационном файле сервера BIND эти поля при некоторых обстоятельствах можно не указывать. В этом случае имя будет повторять имя предыдущей записи, TTL – сформирован по определённым (описанным далее) правилам, а класс будет иметь значение IN. Поля TYPE и VALUE являются обязательными. Поле TTL означает время в секундах, в течении которого резолвер может считать кешированное значение валидным. Интерес представляет запись SOA. В поле VALUE для этой записи необходимо указывать пять значений. Целиком в файле зоны это выглядит так:

```
zhmylove.ru.   SOA ns.zhmylove.ru. support.zhmylove.ru. (
   201412260      ; serial number YYYYMMDDHH
   1d             ; slave refresh
```

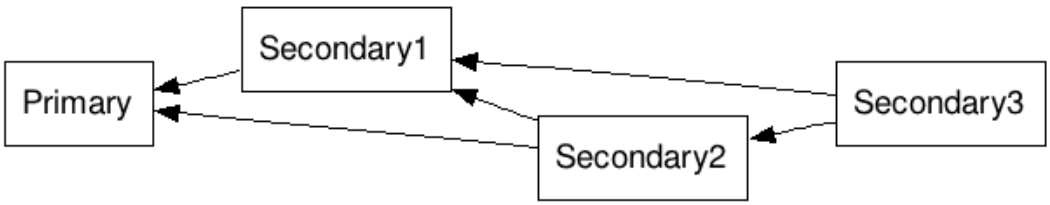
```
2h              ; slave retry time in case of problems
4w              ; slave expiration time
1h              ; max caching time in case of failed lookups
)
```

Первое значение – серийный номер зоны. Это показатель того, нужно ли вторичному серверу запрашивать актуальное содержимое зоны. Если серийный номер записи SOA из зоны на вторичном севере меньше, чем серийный номер на первичном, вторичный запрашивает всю зону. Кроме того, при изменении serial, BIND может отправлять уведомления указанным в конфигурации серверам, чтобы ускорить процесс обновления зоны. Второе поле – время запросов записи SOA вторичным сервером в обычном режиме. Третье поле аналогично второму, но в режиме, когда первичный сервер не отвечает (например, неисправна сеть). Обычно, это время меньше, чтобы быстрее среагировать на восстановление первичного сервера. Четвёртое поле – это время от последнего успешного трансфера зоны, через которое вторичный сервер перестаёт отвечать на запросы. Иными словами, сколько может жить зона после отключения мастера. Последнее поле служит сразу для нескольких целей. Во-первых, оно означает, сколько можно для данной зоны кешировать ответ NXDOMAIN (запись не найдена). Во-вторых, это TTL по умолчанию для остальных записей, если не указана директива $TTL или оно не задаётся для записи вручную. Максимальное значение поля по стандарту – 3 часа. Для BIND эти поля являются signed 32bit int, а поле serial – unsigned с максимальной разницей между значениями 2147483647.

Файл зоны формируется из набора записей и директив. Рассмотрим пример файла зоны cs.ifmo.ru:

ПРИМЕР ФАЙЛА

Фактически, мастер сервер – это сервер, на котором администрируется зона. А вторичные серверы – это серверы, которые эту зону получают и служат для уменьшения нагрузки на первичный и для отказоустойчивости. Первичный сервер не обязательно мастер зоны. Первичным называется тот сервер, с которым осуществляется синхронизация. Рассмотрим пример:



На рисунке первичным сервером для Secondary2 и Secondary3 является Secondary1, который при этом вторичен для Primary.

Сам сервер, с файлами зон, конфигурируется единым конфигурационным файлов, в котором необходимо описать каждую зону. Рассмотрим типовой конфигурационный файл для BIND:

```
// C-style comment
/* C multiline
 * comment
 */
# shell-style comment
options {
```

```
        directory "/etc/named";
        listen-on { any; };
        forwarders { 194.85.162.163; };
        recursion yes;
        allow-query { any; };
        allow-recursion { any; };
        notify yes;
        also-notify { 192.168.10.1; };
        allow-update { none; };
        allow-transfer { none; };
};
zone "." {
        type hint;
        file "named.root";
};
zone "localhost" {
        type master;
        file "localhost.zone";
};
zone "0.0.127.in-addr.arpa" {
        type master;
        file "0.0.127.in-addr.arpa";
};
zone "embedded.ifmo.ru" {
        type slave;
        masters { 194.85.162.163; };
        file "slave/embedded.ifmo.ru";
};
zone "xxx.com" {
        type forward;
        forwarders { 194.85.162.164; };
};
```

Используя такой конфигурационный файл, демон named будет отвечать на запросы к указанным зонам, а неизвестные запросы отправлять на адрес 194.165.162.163. Запросы к зоне xxx.com будут отправлены на 194.165.162.164, зона embedded.ifmo.ru будет скачиваться с указанного мастера и запросы к этой зоне будут обслуживаться без обращения к мастеру. Зоны

localhost и 0.0.127.in-addr.arpa требуются для некоторых клиентов, которые запрашивают эти адреса у системы DNS. Зона точка позволяет подсказать серверу (hint) IP адресы корневых серверов.

В секции options описаны настройки сервера. В частности, разрешены обычные и рекурсивные запросы от кого-угодно, включено уведомление вторичных серверов об изменениях зон (notify), настроены форвардеры, указан каталог, относительно которого будут рассматриваться все пути и указаны интерфейсы, на которых сервер вешает слушающий сокет. В нашем случае – на всех. После запуска сервера, он будет обслуживать запросы согласно конфигурации.

Стоит отметить, что некоторые серверы настраиваются в режиме только кеширования (forward-only). Такие серверы не являются мастерами каких-либо зон, только обслуживая рекурсивные запросы и кешируя ответы. Они используются для снижения нагрузки на основные серверы и уменьшения времени ответа в пределах корпоративной сети.

Протокол HTTP

В глобальной сети для передачи документов, чаще всего, используется протокол гипертекстовой передачи данных – HyperText Transmission Protocol. Наибольшее распространение получили версии 0.9, 1.0 и 1.1. Протокол описывает запросы и ответы для получения каких-либо ресурсов, идентифицируемых уникальными идентификаторами ресурсов. Версия 0.9 подразумевает однострочный запрос, состоящий из метода и указания ресурса. Поэтому для версии 0.9 справедлив, например, такой запрос:

```
GET /
```

В ответ на такой запрос, сервер предоставит клиенту содержимое документа, связанного с ресурсом /.

В отличие от версии 0.9, в версии 1.0 появились заголовки запроса/ответа, которые отделяются от тела пустой строкой (или последовательностью символов \n\n). Появились методы POST и HEAD. Первый позволяет передать аргументы запроса в теле, а второй позволяет получить от севера только код ответа и заголовки. Для версии 1.0 справедлива такая структура запроса/ответа:

```
МЕТОД
Заголовки

Тело
```

Среди заголовков выделяются, например, Content-type, который указывает тип содержимого. Содержимое может иметь любой тип из набора MIME. В заголовках сервер может указать клиенту кодировку документа (charset), установить cookie (Set-Cookie) (cookie – это переменная или массив переменных, хранимых клиентом HTTP и добавляемых ко всем запросам из домена, указанного в cookie; используется для идентификации клиентов), указать расположение ресурса (Location), например, при его отсутствии по запрошенному пути. Для ускорения работы протокола, могут использоваться заголовки Last-Modified и ETag, отражающие, требуется ли перезапрашивать документ целиком или можно ли воспользоваться версией из кеша. Для работы с этим же механизмом существует заголовок запроса If-Modified-Since, который указывает серверу время изменения кешированной на клиенте версии и, если сервер считает, что документ с этого времени не изменялся, возможен ответ Not Modified. Например, запросим HEAD запрос для проверки, изменялся ли документ:

```
HEAD /files/NIX/Freya2.txt HTTP/1.1
Host: zhmylove.ru
```

```
If-Modified-Since: Tue, 11 Nov 2014 15:29:43 GMT

HTTP/1.1 304 Not Modified
Server: nginx
Date: Sun, 4 Jan 2015 13:36:38 GMT
Last-Modified: Tue, 11 Nov 2014 15:29:43 GMT
Connection: keep-alive
ETag: "54622b67-ae9"
Expires: Thu, 31 Dec 2037 23:55:55 GMT
Cache-Control: max-age=315360000
Content-Type: application/octet-stream
```

Для идентификации хоста, к которому отправляются запросы в версии 1.0 есть заголовок Host. Этот заголовок позволяет реализовать обработку на одном сервере сразу нескольких доменов.

Версия протокола 1.1 является незначительным улучшением версии 1.0, но внесла ряд существенных изменений. В частности, заголовок Host стал обязательным, а keep-alive (удержание или не разрыв соединения после двустороннего обмена) соединения – стандартным поведением сервера. Рассмотрим пример запроса HTTP версии 1.1:

```
helios$ telnet 0 80
Trying 0.0.0.0...
Connected to 0.
Escape character is '^]'.
GET /~korg HTTP/1.1
Host: helios.cs.ifmo.ru

HTTP/1.1 301 Moved Permanently
Server: nginx/1.1.8
Date: Tue, 30 Dec 2014 13:25:02 GMT
Content-Type: text/html
Content-Length: 184
Location: http://helios.cs.ifmo.ru/~korg/
Connection: keep-alive

<html>
<head><title>301 Moved Permanently</title></head>
<body bgcolor="white">
```

```
<center><h1>301 Moved Permanently</h1></center>
<hr><center>nginx/1.1.8</center>
</body>
</html>
<html>
<head><title>400 Bad Request</title></head>
<body bgcolor="white">
<center><h1>400 Bad Request</h1></center>
<hr><center>nginx/1.1.8</center>
</body>
</html>
Connection to 0 closed by foreign host.
```

Как видно, сервер ответил на запрос кодом 301, добавил ряд заголовков, в частности, Location, указывающий, куда именно перенесён запрашиваемый документ (в примере, добавлен символ '/' в конец URI), указал время отправки документа (Date), серверное ПО (Server), принудительно указал Connection: keep-alive и не разорвал соединение. Также, для клиентов, не поддерживающих автоматический редирект, сервер сформировал html-страницу с описанием проблемы и ссылкой на запрашиваемый документ. Длину html-страницы сервер указал в заголовке Content-Lenght, а тип, соответственно, в Content-Type.

Как видно, взаимодействие по протоколу HTTP является plaintext. Протокол передаёт данные без изменения, что приводит к открытой передаче, например, данных форм, в т.ч. паролей. Протокол HTTP не предусматривает шифрование данных, поэтому был разработан механизм SSL (Secure Socket Layer), реализующий ассиметричное шифрование с подписью публичного ключа корневым сертификатом. Это позволяет клиенту, во-первых, удостовериться, что сервер, зашифровавший данные, валиден с точки зрения доверенных центров авторизации (ему доверяют), а во-вторых, позволяет шифровать передаваемый трафик. HTTP, работающий поверх SSL принято называть HTTPS (от HTTP secure). Традиционно, HTTP сервер работает на порту 80, а HTTPS – на 443. Рассмотрим пример HTTPS, для этого нам понадобится программа, способная установить шифрованное соединение и проверить валидность удалённой стороны. В нашем случае, это клиент openssl:

```
helios$ openssl s_client -connect 0:443
CONNECTED(00000003)
depth=3 /C=SE/O=AddTrust AB/OU=AddTrust External TTP Network/CN=AddTrust External CA Root
verify error:num=19:self signed certificate in certificate chain
verify return:0
---
Certificate chain
 0 s:/OU=Domain Control Validated/OU=Free SSL/CN=helios.cs.ifmo.ru
   i:/C=GB/ST=Greater Manchester/L=Salford/O=COMODO CA Limited/CN=COMODO RSA Domain Validation Secure Server CA
 1 s:/C=GB/ST=Greater Manchester/L=Salford/O=COMODO CA Limited/CN=COMODO RSA Domain Validation Secure Server CA
```

```
   i:/C=GB/ST=Greater Manchester/L=Salford/O=COMODO CA Limited/CN=COMODO RSA Certification
Authority
 2 s:/C=GB/ST=Greater Manchester/L=Salford/O=COMODO CA Limited/CN=COMODO RSA Certification
Authority
   i:/C=SE/O=AddTrust AB/OU=AddTrust External TTP Network/CN=AddTrust External CA Root
 3 s:/C=SE/O=AddTrust AB/OU=AddTrust External TTP Network/CN=AddTrust External CA Root
   i:/C=SE/O=AddTrust AB/OU=AddTrust External TTP Network/CN=AddTrust External CA Root
---
Server certificate
-----BEGIN CERTIFICATE-----
MIIFVTCCBD2gAwIBAgIRALDUojXzto1T/7IieT8mRZowDQYJKoZIhvcNAQELBQAw
gZAxCzAJBgNVBAYTAkdCMRswGQYDVQQIExJHcmVhdGVyIE1hbmNoZXN0ZXIxEDAO
BgNVBAcTB1NhbGZvcmQxGjAYBgNVBAoTEUNPTU9ETyBDQSBMaW1pdGVkMTYwNAYD
VQQDEy1DT01PRE8gUlNBIERvbWFpbiBWYWxpZGF0aW9uIFNlY3VyZSBTZXJ2ZXIg
Q0EwHhcNMTQxMTIxMDAwMDAwWhcNMTUwMjE5MjM1OTU5WjBSMSEwHwYDVQQLExhE
b21haW4gQ29udHJvbCBWYWxpZGF0ZWQxETAPBgNVBAsTCEZyZWUgU1NMMRowGAYD
VQQDExFoZWxpb3MuY3MuaWZtby5ydTCCASIwDQYJKoZIhvcNAQEBBQADggEPADCC
AQoCggEBAKSwg4suLal6KzeJBasZzMfolfpH70NadDEkCZQvhSUni1dL6hlOcHNS
C8+zFfl2pFRZIOmUvhJFWwBueX5kL/Aq4L1Sml4faz3AUh5whdby5+QBtNb/xtc7
AVaS5EfYuA5NiJB5Tek2wHxJu0mFmQkPh/+fEvcUjM8YxtCquacjIJiJSP5z05IM
gGslMIgrApui5bd/KGQk5yKk2dEUEDITgmlpymFe3auafCRb7YaO1cr/DxYhPhKD
gAFgmFir+5qCWdnrV6Mib+IP1pctZ3M3Jca4qKz/NmsSdTdbK7JrzyBz3ERHJZFe
8JiF2fSnp5dwGDBgQCHIcACvHKd0LNcCAwEAAaOCAeUwggHhMB8GA1UdIwQYMBaA
FJCvajqUWgvYkOoSVnPfQ7Q6KNrnMB0GA1UdDgQWBBTZdSiM70DcRb0cnYEn6KNy
Lti3MDAOBgNVHQ8BAf8EBAMCBaAwDAYDVR0TAQH/BAIwADAdBgNVHSUEFjAUBggr
BgEFBQcDAQYIKwYBBQUHAwIwTwYDVR0gBEgwRjA6BgsrBgEEAbIxAQICBzArMCkG
CCsGAQUFBwIBFh1odHRwczovL3NlY3VyZS5jb21vZG8uY29tL0NQUzAIBgZngQwB
AgEwVAYDVR0fBE0wSzBJoEegRYZDaHR0cDovL2NybC5jb21vZG9jYS5jb20vQ09N
T0RPUlNBRG9tYWluVmFsaWRhdGlvblNlY3VyZVNlcnZlckNBLmNybDCBhQYIKwYB
BQUHAQEEeTB3ME8GCCsGAQUFBzAChkNodHRwOi8vY3J0LmNvbW9kb2NhLmNvbS9D
T01PRE9SU0FEb21haW5WYWxpZGF0aW9uU2VjdXJlU2VydmVyQ0EuY3J0MCQGCCsG
AQUFBzABhhhodHRwOi8vb2NzcC5jb21vZG9jYS5jb20wMwYDVR0RBCwwKoIRaGVs
aW9zLmNzLmlmbW8ucnWCFXd3dy5oZWxpb3MuY3MuaWZtby5ydTANBgkqhkiG9w0B
AQsFAAOCAQEAWeIBPXETs6zN1mEYqsJYuzFVfqKyQO5+nA2yiIdoIuVe4BFtGeLz
h1N19frWGGtU2GebQvae2DQ4Q59gn17GMLd3krxTww8EX9TB1Ifg8LoHwAJdjDP7
powDUu/0QccOviCGu2B8V0bsiVgzT2R6d118NXn7y33XUhjPPxvYUfyygSw96fFd
Jl8lvC+097E/FZclNregfcINczVWQTOVZhxqdEJ6eMbNTGOfj5JlDs7GYeV60vTl
qW69G0CsSn1lMVvjjtICkasbsQMCUyXLDvsMmIzK+0Awv1WNDcqqK10BBYdYKCuJ
hBUgpNQJLDLrqGEJexIowTSPZmGS1r/dug==
-----END CERTIFICATE-----
subject=/OU=Domain Control Validated/OU=Free SSL/CN=helios.cs.ifmo.ru
```

```
       issuer=/C=GB/ST=Greater Manchester/L=Salford/O=COMODO CA Limited/CN=COMODO RSA Domain
Validation Secure Server CA

       ---

       No client certificate CA names sent

       ---

       SSL handshake has read 6107 bytes and written 343 bytes

       ---

       New, TLSv1/SSLv3, Cipher is DHE-RSA-AES256-SHA

       Server public key is 2048 bit

       Secure Renegotiation IS supported

       SSL-Session:

           Protocol  : TLSv1

           Cipher    : DHE-RSA-AES256-SHA

           Session-ID: 20F19B6DF18057806FAF374282EE176245FFE453ECFF25ED599DC9FD0015EB23

           Session-ID-ctx:

           Master-Key:
BE68063997D6C65374B93E26E4F2D815B907D9F16A067B75E9DC6B9C8AF723FD7DAD0EBAFA76BE37CE38235823F0A54F

           Key-Arg   : None

           Start Time: 1419946479

           Timeout   : 300 (sec)

           Verify return code: 19 (self signed certificate in certificate chain)

       ---

       GET /~korg/

       <!DOCTYPE html>

       <HTML><HEAD><TITLE></TITLE></HEAD><BODY>

             <IMG SRC="/~korg/favicon.ico" STYLE="position:fixed; right:10px; bottom:10px;"
ALT="KorG" />

             <H4>СПО</H4>

             <A HREF=unix/>Задания на лабораторные работы по UNIX (модуль 5)</A><BR>

             <A HREF=unix/SPO_PRE.pdf>Актуальная лекционная презентация по основам UNIX (модуль
5)</A><BR>

             <A HREF=unix/SPO_PRETEST.pdf>Актуальная лекционная презентация по элементам UNIX
(модуль 5)</A><BR>

             <A HREF=unix/SPO_PREEXAM.pdf>Актуальная лекционная презентация по тонким материям
UNIX (модуль 5)</A><BR>

             <A HREF=regexp/>Задания на лабораторные работы по regexp (модуль 6)</A><BR>

             <A HREF=regexp/regexp.pdf>Традиционная презентация по regexp (модуль 6)</A><BR>

             <A HREF=regexp/utils.pdf>Лекция по утилитам для работы с regexp (модуль 6)</A><BR>

             <A HREF=c/SPO_LEC.pdf>Актуальная лекционная презентация по C (модуль 8)</A><BR>

             <A HREF=exam/SPO_EXAM1_2014.pdf>Экзаменационные вопросы за первый семестр 2013/2014
учебного года</A><BR>

             <A HREF=exam/SPO_EXAM2_2014.pdf>Экзаменационные вопросы за второй семестр 2013/2014
учебного года</A><BR>


             <H4>Разное</H4>
```

```
        <A HREF=uirs.html>УИРС/Практика</A><BR>
</BODY></HTML>
closed
```

Видно, что openssl проверила сертификат, вывела его содержимое, вывела путь сертификации (цепочку центров авторизации, которые подтверждают его валидность). Вывела результат:

```
verify error:num=19:self signed certificate in certificate chain
```

И предоставила шифрованное (хоть и недоверенное) соединение нам. В соединение был отправлен запрос "GET /~korg/", который из-за неуказанной версии относится к версии 0.9 и был сразу обработан сервером. Протокол HTTP 0.9 не поддерживает keep-alive, поэтому после отправки ответа соединение было закрыто, о чем сообщает строка "closed".

Протокол HTTP является простым протоколом, который подвергается лёгкому разграничению прав доступа. Для разграничения прав доступа используются HTTP прокси-серверы. Прокси – это сервер-представитель, который взаимодействует с клиентом и с конечным сервером. Фактически, прокси – посредник при обмене данными. Прокси предназначены для решения различных задач, таких как:

- работа черными списками сайтов, клиентов и ресурсов;
- кеширование данных в локальной сети;
- аутентификация клиентов;
- аудит доступа к ресурсам.

Для работы через прокси, обычно, требуется его явно указать HTTP-клиенту (браузеру). Компания Microsoft предложила механизм автоматического определения параметров proxy, называемый WPAD (web proxy auto discovery). Суть механизма заключается в том, что клиентская система отправляет HTTP запрос на специальный узел wpad и запрашивает у него файл wpad.dat, который представляет собой скрипт на javascript в формате PAC (proxy auto configuration) и применяет настройки посредством выполнения этого скрипта. Пример скрипта можно получить загрузив http://wpad.cs.ifmo.ru/wpad.dat , который приведён ниже:

```
function FindProxyForURL(url, host)
{
        if (
                isInNet(host, "0.0.0.0", "255.0.0.0") ||
                isInNet(host, "10.0.0.0", "255.0.0.0") ||
                isInNet(host, "127.0.0.0", "255.0.0.0") ||
                isInNet(host, "172.16.0.0", "255.240.0.0") ||
                isInNet(host, "192.168.0.0", "255.255.0.0")
        ) return "DIRECT";

        if (dnsDomainIs(host,".scopus.com"))
                return "PROXY proxy.ifmo.ru:3128";

        return "PROXY 192.168.2.1:3128";
```

```
}
```

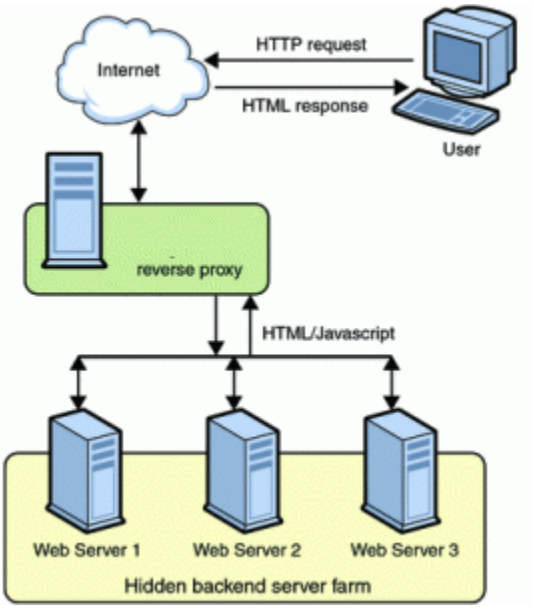
Кроме такого способа определения параметров прокси, возможна ситуация, так называемого, прозрачного проксирования (transparent proxy), при которой на стороне сетевого экрана исходящие TCP подключения перенаправляются на прокси-сервер, который их обрабатывает. Этот подход имеет ряд проблем. Во-первых, станет невозможной работа веб-серверов не на 80-м порту, а во-вторых, невозможна работа HTTPS. Это обусловлено тем, что клиент, при работе по HTTPS, ожидает от сервера получения шифрованного приватным ключом траффика. Прокси-сервер же, получает от сервера уже готовый ответ и, за неимением приватного ключа, не может предоставить клиенту требуемый, шифрованный ответ. Эта проблема решается добавлением в прокси сервер поддержки метода CONNECT, который фактически, предоставляет клиенту TCP-соединение, по которому передаются шифрованные данные. Фактически, прокси сервер теряет свой смысл, поэтому почти всегда используя метод CONNECT можно подключиться только на tcp/443 порт. Пример использования метода CONNECT:

```
$ telnet 0 3128

Trying 0.0.0.0...

Connected to 0.

Escape character is '^]'.

CONNECT http://helios.cs.ifmo.ru:443 HTTP/1.1

Host: helios.cs.ifmo.ru:443

SSH-2.0-OpenSSH_5.8p2_hpn13v11 FreeBSD-20110503
```

В примере умышленно на 443-й порт повешен SSH-сервер.

Теоретически, существуют два механизма организации прозрачного проксирования HTTPS. Первый – это SSL strip. В теле ответов клиенту все https:// схемы заменяются на http://. Прокси связывается с конечным сервером посредством HTTPS, а с клиентом общается по незащищённому каналу. Это приводит, например, к проблеме безопасности передаваемых данных. Второй механизм сложнее и называется SSL Interception. Он заключается в том, что прокси для взаимодействия с клиентом динамически формирует SSL wildcard сертификаты (например, сертификат, позволяющий доверять всем хостам в домене ifmo.ru: "*.ifmo.ru,ifmo.ru"). Сгенерировать wildcard сертификат для "*" и доменов второго уровня, например "*.ru" по понятным причинам нельзя. Механизм SSL Interception от клиента требует только доверия к центру авторизации, который подписывает генерируемые прокси-сервером сертификаты, что позволяет, например, не указывать настройки прокси при работе в корпоративной сети (с прокси) и в домашней (без прокси).

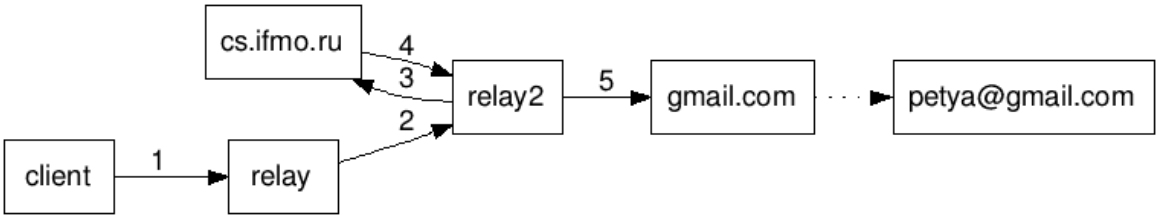
Кроме того, существует такое понятие, как Reverse proxy – обратный прокси-сервер – это прокси сервер, который выполняет функцию обработки и распределения запросов из глобальной сети в локальную. Например, осуществлять load balancing между несколькими локальными веб-серверами, или не нагружать «тяжёлые» веб-серверы запросами к статическому контенту. Например, reverse proxy можно проиллюстрировать так:



Протокол SMTP

В современных сетях для отправки почты используется протокол SMTP (simple mail transfer protocol). Этот протокол описывает принципы взаимодействия систем, участвующих в пересылке почты и позволяет отправлять письмо одному или нескольким адресатам (осуществляя групповую рассылку).

Рассмотрим один из возможных вариантов доставки письма конечному пользователю:



Пусть client и relay – это SMTP клиент и сервер соответственно. Чаще всего, такая ситуация встречается в рамках одной компании, когда клиенты пользуются корпоративным relay-ем для отправки почты. Клиент пишет письмо на vasya@cs.ifmo.ru и отправляет его на relay. Relay пользуется системой DNS (запрашивая записи типа MX) для получения адреса SMTP сервера, обслуживающего домен cs.ifmo.ru. В качестве этого адреса используется адрес с наименьшим приоритетом. Последующие адреса используются только в случае недоступности всех более приоритетных серверов. В рассматриваемом примере SMTP сервер, обслуживающий домен cs.ifmo.ru, является внутренним сервером, не доступным из глобальной сети, поэтому в ответ на MX запрос relay получает адрес relay2, которому и пересылается письмо. В свою очередь, relay2 при обработке письма доставляет его на сервер cs.ifmo.ru, который обрабатывая пришедшее письмо определяет, что существует привязка (алиас) вида vasya: vasya@gmail.com и пересылает письмо на vasya@gmail.com (по описанным ранее причинам, снова через relay2). Получив это письмо, relay2 пересылает его на SMTP сервер gmail.com, который уже доставляет письмо до адресата (не обязательно по протоколу SMTP, что отмечено прерывистой стрелкой на схеме).

Протокол SMTP, аналогично протоколам FTP и HTTP является plaintext протоколом. Ниже приведён пример взаимодействия клиента и SMTP сервера.

```
$ telnet mail.cs.ifmo.ru smtp

Trying 192.168.4.8...
```

```
Connected to mail.cs.ifmo.ru.
Escape character is '^]'.
220 mail.cs.ifmo.ru ESMTP Exim 4.84 Mon, 29 Dec 2014 20:53:46 +0300
HELO admin
250 mail.cs.ifmo.ru Hello helios.cs.ifmo.ru [192.168.10.10]
MAIL FROM: president@whitehouse.gov
250 OK
RCPT TO: admin@cs.ifmo.ru
250 Accepted
RCPT TO: vasya@cs.ifmo.ru
250 Accepted
DATA
354 Enter message, ending with "." on a line by itself
From: xxx@xxx.com
To: vasya@cs.ifmo.ru
Subject: New videos available

Hi, vasya!
Our archieves had updated last saturday.
.
250 OK id=1Y5eWd-000FuT-Ls
HELP
214-Commands supported:
214 AUTH STARTTLS HELO EHLO MAIL RCPT DATA NOOP QUIT RSET HELP
QUIT
221 mail.cs.ifmo.ru closing connection
Connection to mail.cs.ifmo.ru closed by foreign host.
```

Письмо разделяется на конверт, заголовки и тело. Конверт включает в себя информацию, кому и от кого доставлять письмо (MAIL FROM и RCPT TO). Конверт не входит в состав письма и требуется только серверу для обработки этого письма. На каждом промежуточном сервере формируется свой конверт. Заголовки следуют сразу после DATA и заканчиваются пустой строкой, а телом письма называется всё от этой пустой строки до заключительной точки. После ввода заключительной точки, соединение не сбрасывается, что позволяет отправлять сразу несколько разных писем в рамках одного подключения. В примере такой keep-alive был продемонстрирован вызовом HELP после постановки письма в очередь. Формально, сервер не может изменять заголовки, но может добавлять собственные в верхнюю часть, поэтому при анализе корреспонденции порядок чтения заголовков не меняется. Однако, это правило иногда нарушается. В частности, для изменения значения поля Date.

Протокол SMTP не избавлен от проблемы оригинальности отправителя, что приводит к большому числу спама, mail-ботов и, как следствие, троянов и спуфинга (подмены source адреса) у писем, рассылаемых по электронной почте. Валидность отправителя проверить вообще не представляется возможным, но существует ряд решений, позволяющих сильно снизить поток нежелательной почты. Во-первых, часто реализуется механизм проверки, что после HELO указан IP или FQDN (см. DNS). Для проверки заявленного MAIL FROM используется так называемый Callout. Когда серверу требуется проверить MAIL FROM, он подключается к серверу отправителя, заявляет специальный MAIL FROM: <> и в качестве RCPT TO указывает полученный от отправителя MAIL FROM (фактически, имитируя отправку письма отправителю). Если сервер отправителя корректно отвечает на такой RCPT TO, письмо считается валидным.

Кроме этих механизмов есть такие широко используемые механизмы, как SPF и DKIM (переработанная версия DOMAINKEY и DMARC). Они основываются на проверке домена отправителя. SPF – на проверке валидности IP адреса, с которого может прийти письмо, а DKIM – проверки ключа, которым отправитель шифрует часть заголовков. При этом, оба механизма основываются на системе DNS, рассмотренной ранее. У DNS сервера запрашиваются записи типа TXT, в которых хранятся в первом случае форматированная строка, а во втором – публичный ключ от шифра. Если получателю удаётся пройти проверку SPF и DKIM, письмо считается валидным. Различные системы (например, gmail и yandex) доверяют таким подписанным письмам. Однако, не смотря на эти механизмы, защитить весь поток почты не получится из-за списков рассылки, придя на которые, письмо начинает пересылаться сервером, обслуживающим список. Соответственно, тот сервер не может быть учтён в SPF (может иметь любой IP адрес) и DKIM (не имеет приватного ключа), поэтому не весь поток нашей исходящей почты будет защищён.

Помимо рассмотренных механизмов пользуются популярностью такие способы определения валидности письма, как проверка содержимого письма, разбор всех заголовков и расчёт коэффициента «человечности» - прогнозирование, человеком ли было написано письмо. Широкое распространение получила программа SpamAssassin, которая учитывает множество критериев и на основе них доставляет (или рекомендует SMTP серверу) письмо получателю.

Оглавление

Введение 1

Физические аспекты 1

Канальные протоколы 6

- Основы Ethernet 6
- Технология VLAN 7
- Агрегирование каналов 9
- Сетевые мосты 9
- Максимально передаваемый блок данных (MTU) 9

Протоколы сетевого уровня 10

- Протокол IPv4 10
- Протокол ARP 13
- Протокол IPv6 14
- Маршрутизация 15
- Туннелирование 18

Транспортные протоколы 19

- Протокол ICMP 19
- Протокол TCP 20
- Протокол UDP 22
- Псевдозаголовок TCP/UDP 23
- Сетевые экраны и раздел пакетные фильтры 23

Прикладные программы и протоколы 24

- Протокол TELNET 25
- Протокол FTP 26
- Программы удалённого управления rlogin, rsh и ssh 28
- Распределённая система доменных имён (DNS) 30
- Протокол HTTP 39
- Протокол SMTP 46