**1. Физическая и логическая инфраструктура сети**

Под *инфраструктурой сети* понимают множество взаимосвязанных технологий и систем, которые администраторы должны досконально знать, чтобы успешно поддерживать работу сети и устранять неполадки.

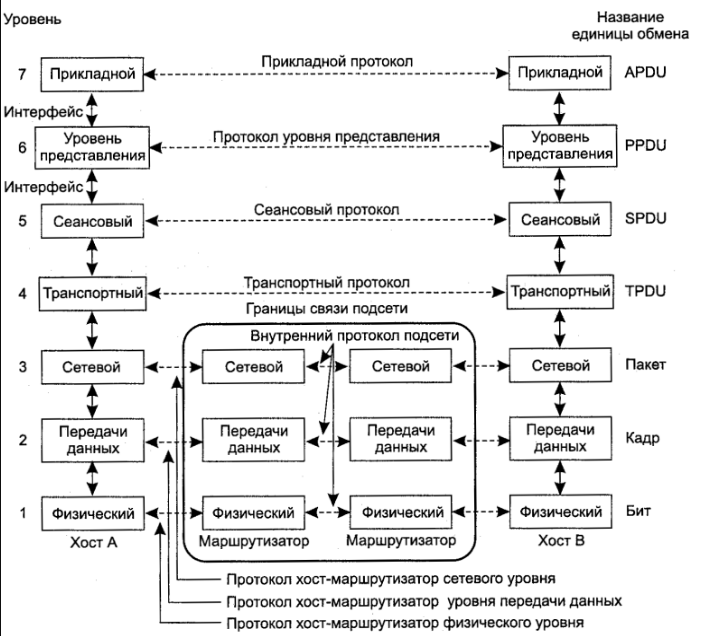
Под *физической инфраструктурой сети* подразумевают ее топологию, то есть физическое строение сети со всем ее оборудованием: кабелями, маршрутизаторами, коммутаторами, мостами, концентраторами, серверами и узлами. К физической инфраструктуре также относятся транспортные технологии: Ethernet. 802.11b, коммутируемая телефонная сеть общего пользования (PSTN), ATM — в совокупности они определяют, как осуществляется связь на уровне физических подключений.

*Логическая инфраструктура сети* состоит из всего множества программных элементов, служащих для связи, управления и безопасности узлов сети, и обеспечивает связь между компьютерами с использованием коммуникационных каналов, определенных в физической топологии. Примеры элементов логической инфраструктуры сети: система доменных имен (Domain Name System. DNS), сетевые протоколы, например TCP/IP.

**2. Эталонная модель OSI**

Сетевая модель OSI (англ.​ open systems interconnection basic reference model — базовая эталонная модель взаимодействия открытых систем, сокр. ​ ЭМВОС​ ; 1978 год) — сетевая модель стека сетевых протоколов OSI/ISO.

Эталонная модель OSI:



**Физический уровень** занимается реальной передачей необработанных битов по каналу связи.

**Уровень передачи данных (канальный)**

Основная задача уровня передачи данных — быть способным передавать «сырые» данные физического уровня по надежной линии связи, свободной от необнаруженных ошибок с точки зрения вышестоящего сетевого уровня.

**Транспортный уровень**

Основная функция транспортного уровня — принять данные от сеансового уровня, разбить их при необходимости на небольшие части, передать их сетевому уровню и гарантировать, что эти части в правильном виде прибудут по назначению.

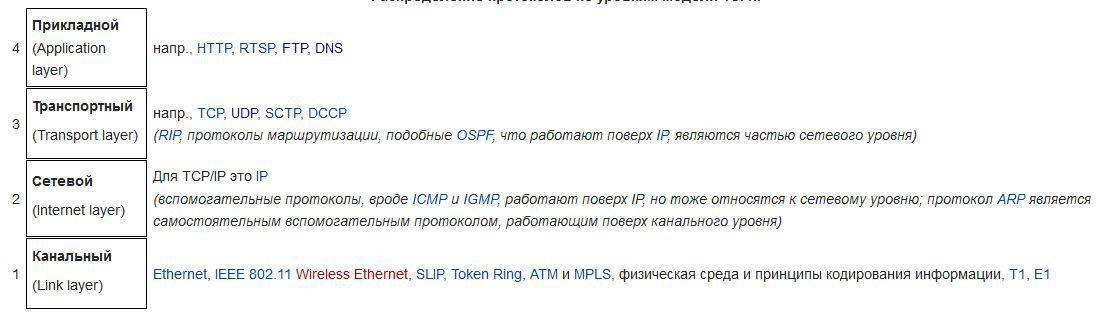
**Сеансовый уровень** позволяет пользователям различных компьютеров устанавливать сеансы связи друг с другом.

**Уровень представления** занимается по большей части синтаксисом и семантикой передаваемой информации.

**Прикладной уровень** содержит набор популярных протоколов, необходимых пользователям.

**3. Эталонная модель TCP/IP**

TCP/IP является основным стеком протоколов Интернет, реализующего модель с коммутацией пакетов. Она содержит четыре уровня: прикладной, транспортный, межсетевой и физический + плюс передачи данных.



**Прикладной уровень**

На прикладном уровне (Application layer) работает большинство сетевых приложений.

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

* массе своей эти протоколы работают поверх TCP или UDP и привязаны к определённому порту, например:
  + HTTP на TCP­порт 80 или 8080,
  + FTP на TCP­порт 20 (для передачи данных) и 21 (для управляющих команд),
  + SSH на TCP­порт 22,
  + запросы DNS на порт UDP (реже TCP) 53,
  + обновление маршрутов по протоколу RIP на UDP­порт 520.

**Транспортный уровень**

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

TCP — «гарантированный» транспортный механизм с предварительным установлением соединения, предоставляющий приложению надёжный поток данных, дающий уверенность в безошибочности получаемых данных, перезапрашивающий данные в случае потери и устраняющий дублирование данных.

UDP протокол передачи датаграмм без установления соединения.

**Сетевой уровень**

Сетевой уровень (Internet layer) изначально разработан для передачи данных из одной (под)сети в другую. Примерами такого протокола является X.25 и IPC в сети ARPANET.

**Канальный уровень**

Канальный уровень (Link layer) описывает, каким образом передаются пакеты данных через физический уровень, включая ​***кодирование*** (то есть специальные последовательности бит, определяющих начало и конец пакета данных). Ethernet, например, в полях заголовка пакета содержит указание того, какой машине или машинам в сети предназначен этот пакет.

**4. Анализ и критика эталонных моделей OSI и TCP/IP. Гибридная модель стека протоколов.**

**Критика модели и протоколов OSI**

Основных причин неудачи модели OSI было четыре:

­ несвоевременность

­ неудачная технология

­ неудачная реализация

­ неудачная политика

**Критика эталонной модели TCP/IP**

Во­первых, в этой модели нет четкого разграничения концепций служб, интерфейса и протокола.

Во­вторых, модель TCP/IP отнюдь не является общей и довольно плохо описывает любой стек протоколов, кроме TCP/IP. Так, например, описать технологию Bluetooth с помощью модели TCP/IP совершенно невозможно.

В­третьих, хост­сетевой уровень в действительности не является уровнем в том смысле, который обычно используется в контексте уровневых протоколов. Это скорее интерфейс между сетью и уровнями передачи данных.

В­четвертых, в модели TCP/IP не различаются физический уровень и уровень передачи данных.

**Гибридная модель**

В этой модели I-й уровень стека TCP/IP заменен первыми тремя уровнями стека OSI. Функционально, 6-й, 5-й и 4-й уровни гибридной модели полностью соответствуют уровням стека TCP/IP (уровни 2-3), а 3-й, 2-й и 1-й уровни определены немного по-другому, чем в стеке OSI, а именно:

·        3-й уровень (сетевой) определяет однородные сети с маршрутизацией типа DVB, G3, GPRS, FR, ATM, Ethernet, 802.11, ArcNet, X25 и др.;

·        2-й уровень (канальный) включает в себя подуровень контроллеров пакетирования однородных сетей и подуровень компрессоров цифровых потоков видео (MPEG), речи, данных (MNP, V42bis, LBZ);

·        1-й уровень (физический) включает в себя подуровень линейных кодеков борьбы с групповыми и одиночными ошибками цифрового потока канала связи.

**5. Физический уровень. Физические носители.**

Физический уровень (Physical layer) имеет дело с передачей битов по физическим каналам связи, таким, как коаксиальный кабель, витая пара, оптоволоконный кабель или цифровой территориальный канал. К этому уровню имеют отношение характеристики физических сред передачи данных, такие как полоса пропускания, помехозащищенность, волновое сопротивление и другие. На этом же уровне определяются характеристики электрических сигналов, передающих дискретную информацию, такую как крутизна фронтов импульсов, уровни напряжения или тока передаваемого сигнала, тип кодирования, скорость передачи сигналов. Кроме того, здесь стандартизируются типы разъемов и назначение каждого контакта.

Физический уровень :

* передача битов по физическим каналам;
* формирование электрических сигналов;
* кодирование информации;
* синхронизация;
* модуляция.

Реализуется аппаратно.

Функции физического уровня реализуются во всех устройствах, подключенных к сети. Со стороны компьютера функции физического уровня выполняются сетевым адаптером или последовательным портом.

**6. Канальный уровень. Службы канального уровня. Сетевые адаптеры.**

Одной из задач *канального уровня* является проверка *доступности среды передачи*. Другая задача *канального уровня* — реализация механизмов *обнаружения и коррекции ошибок*. Для этого на *канальном уровне*  *биты* группируются в наборы, называемые ***кадрами*** (***frames***). *Канальный уровень* обеспечивает корректность передачи каждого *кадра* помещая специальную последовательность *бит* в начало и конец каждого *кадра*, для его выделения, а также вычисляет *контрольную сумму*, обрабатывая все байты *кадра* определенным способом, и добавляет *контрольную сумму* к *кадру*.

Функции канального уровня

* Надежная доставка *пакета*:

1. Между двумя соседними станциями в сети с произвольной топологией.
2. Между любыми станциями в сети с типовой топологией:

* проверка доступности разделяемой среды;
* выделение *кадров* из потока данных, поступающих по сети; формирование *кадров* при отправке данных;
* подсчет и проверка *контрольной суммы*.

Реализуются программно-аппаратно.

Сетевые адаптеры – это сетевое оборудование, обеспечивающее функционирование сети на физическом и канальном уровнях.

Это устройство решает задачи надежного обмена двоичными данными, представленными соответствующими электромагнитными сигналами, по внешним линиям связи. Как и любой контроллер компьютера, сетевой адаптер работает под управлением драйвера операционной системы, и распределение функций между сетевым адаптером и драйвером может изменяться от реализации к реализации.

Компьютер, будь то сервер или рабочая станция, подключается к сети с помощью внутренней платы – сетевого адаптера.

**7. Обнаружение и исправление ошибок.**

Методы обнаружения ошибок основаны на передаче в составе блока данных избыточной служебной информации, по которой можно судить с некоторой степенью вероятности о достоверности принятых данных. В сетях с коммутацией пакетов такой единицей им формации может быть PDU любого уровня, для определенности будем считать, что мы контролируем кадры.

Избыточную служебную информацию принято называть контрольной суммой, или контрольной последовательностью кадра (Frame Check Sequence, FCS). Контрольная сумма вычисляется как функция от основной информации, причем не обязательно путем сумми​рования. Принимающая сторона повторно вычисляет контрольную сумму кадра по известному алгоритму и в случае ее совпадения с контрольной суммой, вычисленной передающей стороной, делает вывод о том, что данные были переданы через сеть корректно. Рассмотрим несколько распространенных алгоритмов вычисления контрольной суммы, отличающихся вычислительной сложностью и способностью обнаруживать ошибки в данных.

В процессе хранения данных и передачи информации по сетям связи неизбежно возникают ошибки. Контроль целостности данных и исправление ошибок — важные задачи на многих уровнях работы с информацией (в частности,**​**физическом,**​**канальном,**​**транспортном​уровнях**​**сетевой модели OSI​).

В системах связи возможны несколько стратегий борьбы с ошибками:

* обнаружение ошибок в блоках данных и ​***автоматический запрос повторной передачи*** повреждённых блоков — этот подход применяется, в основном, на канальном и транспортном уровнях;
* обнаружение ошибок в блоках данных и отбрасывание повреждённых блоков — такой подход иногда применяется в системах потокового мультимедиа, где важна задержка передачи и нет времени на повторную передачу;
* ***исправление ошибок****​*(​​***forward error correction****​*)применяется на физическом уровне.

**8. Протоколы распределения канала. TDM- и FDM-мультиплексирование.**

Частотное мультиплексирование (FDM). Мультиплексирование с разделением частоты (Рис 2) используется в телефонных сетях для организации передачи голосового сигнала, а также может применяться в кабельном телевидении.

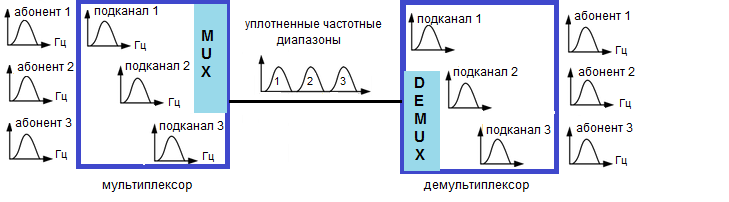


Рис 2.  Частотное мультиплексирование и демультиплексирование

Основная идея : На первом этапе   идет процесс разделения общего широкополосного канала связи на отдельные полосы частот (подканалы), на которые накладываются абонентские частотные диапазоны. На втором этапе,  чтобы избежать взаимного влияния уплотненных пользовательских диапазонов, в каждый подканал добавляется страховая частотная неинформативная полоса, так называемая полоса расфильтровки.

Временное мультиплексирование (TDM).  
Мультиплексирование с разделением по времени (Рис 3) широко применяются в сетевых технологиях PDH, SDH/SONET, АТМ, Ethernet, PON. 

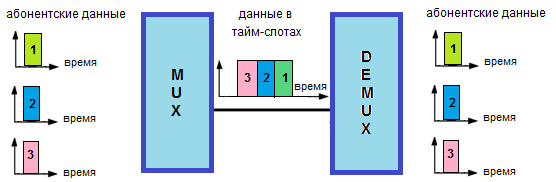


Рис 3. Временное мультиплексирование и демультиплексирование

Суть этого метода мультиплексирования с разделением времени заключается в следующем:  с помощью TDM-мультиплексора входные абонентские каналы последовательно подключаются  к общему каналу связи на определенный интервал времени, так называемый тайм-слот, а на приемной стороне  демультиплексор разуплотняет общий  поток на отдельные выборки и распределяет их по соответствующим приемным абонентским каналам.

**9. Протоколы произвольного доступа.**

В протоколе произвольного доступа передающий узел всегда передает данные в канал с максимальной скоростью, то есть R бит/с. Когда возникает коллизия, каждый вовлеченный в нее узел передает свой кадр повторно до тех пор, пока ему не удастся пройти по каналу без коллизий.

Однако, испытав коллизию, узел, как правило, не повторяет передачу тут же, а выжидает в течение случайного интервала времени. Благодаря разной длительности случайных интервалов времени существует ненулевая вероятность того, что интервал, выбранный одним из узлов, окажется меньше, чем у других вовлеченных в коллизию узлов, и он успеет «пропихнуть» свой кадр в канал без коллизии.

**10. Протоколы последовательного доступа.**

Двумя желательными свойствами протокола коллективного доступа являются, во-первых, возможность единственного активного узла передавать свои данные с максимальной пропускной способностью канала R бит/с, во-вторых, возможность для каждого из М активных узлов передавать свои данные со скоростью R/M бит/с. Протоколы ALOHA и CSMА удовлетворяют первому требованию, но не удовлетворяют второму. Это подвигло исследователей на создание нового класса протоколов — протоколов последовательного доступа. Как и в случае с протоколами произвольного доступа, существуют десятки протоколов последовательного доступа, и у каждого есть множество вариантов. Здесь мы рассмотрим два наиболее важных протокола последовательного доступа. Первый из них — протокол опроса. При использовании протокола опроса один из узлов должен быть назначен главным (управляющим) узлом. Главный узел поочередно опрашивает все узлы. Например, сначала главный узел посылает сообщение узлу 1, сообщая ему, что он может передать некоторое максимальное количество кадров. После того как узел 1 передает несколько кадров, главный узел разрешает передать некоторое количество кадров узлу 2. (Главный узел может определить момент завершения передачи очередным узлом по отсутствию сигнала в канале.) Данная процедура продолжается бесконечно, при этом главный узел в цикле опрашивает все узлы.

Протокол передачи маркера. Здесь не существует главного узла, но есть специальный кадр – маркер, который пере- дается от одного узла к другому и наличие которого разрешает передавать им данные. Проблема – требуется наличие специальных алгоритмов для экстраординарных случаев, наподобие «потери» маркера.

**11. Концентраторы, коммутаторы, мосты.**

**Концентратор** работает на физическом уровне сетевой модели OSI, повторяет приходящий на один порт сигнал на все активные порты. В случае поступления сигнала на два и более порта одновременно возникает коллизия, и передаваемые кадры данных теряются. Таким образом, все подключённые к концентратору устройства находятся в одномдомене коллизий. Концентраторы всегда работают в режимеполудуплекса, все подключённые устройства Ethernet разделяют между собой предоставляемую полосу доступа.

Работает по следующему принципу: копирует все полученные пакеты во все порты. При этом может возникнуть проблема, при которой по двум и более портам приходят пакеты в одно и то же время. Другая проблема — безопасность — все пакеты доходят до всех компьютеров сети, поэтому существует возможность несанкционированного доступа к информации. И, наконец, ещё одной проблемой является то, что копирование пакетов повышает нагрузку на сеть, причём весьма существенно — весь трафик сегмента сети поступает к каждому из компьютеров и тем самым загружает сеть.

**Коммутатор** хранит в памяти таблицу коммутации (хранящуюся в ассоциативной памяти), в которой указывается соответствие MAC-адреса узла порту коммутатора. При включении коммутатора эта таблица пуста, и он работает в режиме обучения. В этом режиме поступающие на какой-либо порт данные передаются на все остальные порты коммутатора. При этом коммутатор анализирует кадры (фреймы) и, определив MAC-адрес хоста-отправителя, заносит его в таблицу на некоторое время. Впоследствии, если на один из портов коммутатора поступит кадр, предназначенный для хоста, MAC-адрес которого уже есть в таблице, то этот кадр будет передан только через порт, указанный в таблице. Если MAC-адрес хоста-получателя не ассоциирован с каким-либо портом коммутатора, то кадр будет отправлен на все порты, за исключением того порта, с которого он был получен.

Существует три способа коммутации.

1. С промежуточным хранением (Store and Forward). Коммутатор читает всю информацию в кадре, проверяет его на отсутствие ошибок, выбирает порт коммутации и после этого посылает в него кадр.
2. Сквозной (cut-through). Коммутатор считывает в кадре только адрес назначения и после выполняет коммутацию. Этот режим уменьшает задержки при передаче, но в нём нет метода обнаружения ошибок.
3. Бесфрагментный (fragment-free) или *гибридный*. Этот режим является модификацией сквозного режима. Передача осуществляется после фильтрации фрагментов коллизий

**Сетевой мост** делит разделяемую среду передачи сети на части(часто называемые логическимисегментами), передавая информацию из одного сегмента в другой только в том случае, если такая передача действительно необходима, то есть если адрес компьютера назначения принадлежит другой подсети . Тем самым мост изолирует трафик одной подсети от трафика другой, повышая общую производительность передачи данных в сети. Локализация трафика не только экономит пропускную способность, но и уменьшает возможность несанкционированного доступа к данным, так как кадры не выходят за пределы своего сегмента, и злоумышленнику сложнее перехватить их.

**12. Сетевой уровень. Модели сетевого обслуживания. Дейтаграммная служба и служба виртуальных каналов.**

Сетевой уровень занимается управлением операциями подсети. Важнейшим моментом здесь является определение маршрутов пересылки пакетов от источника к пункту назначения. Маршруты могут быть жестко заданы в виде таблиц и редко меняться. Кроме того, они могут задаваться в начале каждого соединения, например, терминальной сессии. Наконец, они могут быть в высокой степени динамическими, то есть вычисляемыми заново для каждого пакета с учетом текущей загруженности сети. Если в подсети одновременно присутствует слишком большое количество пакетов, то они могут закрыть дорогу друг другу, образуя заторы в узких местах. Недопущение подобной закупорки также является задачей сетевого уровня.

Модель обслуживания на основе виртуальных каналов включает три фазы.

**Установка виртуального канала.​**Во время этой фазы отправитель связывается ссетевым уровнем, указывает адрес получателя и ждет, пока сеть установит виртуальный канал. Сетевой уровень определяет путь от отправителя до получателя, то есть последовательность линий связи и пакетных коммутаторов, через которые будут проходить все пакеты данного виртуального канала.

**Передача данных.​**Как только виртуальный канал установлен,данные могут начатьперемещение по виртуальному каналу.

**Разрыв виртуального канала.​**Эта процедура начинается,когда отправитель(илиполучатель) информирует сетевой уровень о своем желании разорвать виртуальный канал. Затем, как правило, сетевой уровень информирует оконечную систему на другой стороне сети о разрыве соединения и обновляет таблицы в каждом пакетном коммутаторе пути, показывая, что виртуального канала более не существует.

В сети с виртуальными каналами перед тем, как пакеты начинают идти, создается определенный маршрут следования. Это маршрут служит для поддержки логического соединения между удаленными станциями. Если маршрут установлен, то все пакеты между взаимодействующими станциями будут идти строго по этому маршруту.Поскольку на время логического соединения маршрут строго фиксирован, то такое логическое соединение в некоторой степени аналогично образованию канала в сетях с коммутацией каналов и называется виртуальным каналом. Каждый пакет теперь содержит идентификатор виртуального канала наряду с полем данных. Все узлы по маршруту знают, направлять такие пакеты – никакого решения по маршрутизации теперь эти узлы не принимают. В любое время каждая станция может установить один или несколько виртуальных каналов с другой станцией или станциями.

Главное различие с дейтаграммным подходом и классической маршрутизацией состоит в том, что в сетях с виртуальными каналами узел не принимает решение о отборе маршрута для каждого входящего пакета, а делает это только один раз – на этапе формирования виртуального канала.

Пакеты, которые посылаются протоколом, не требующим соединения, называются **дейтаграммами**. Дейтаграммной сети каждый пакет передается без ссылки на пакеты, которые идут до или после него. Каждый узел на основании контрольной информации заголовка пакета и собственных данных об окружающих узлах сети выбирает следующий узел, на который перенаправляется пакет. Пакеты с одним и тем же адресом назначения могут следовать от станции отправителя к станции назначения разными маршрутами. Конечный узел маршрута восстанавливает правильную последовательность пакетов и уже в этой последовательности передает их станции назначения.

**13. Основы маршрутизации. Классификация алгоритмов маршрутизации.**

Протоколы маршрутизации могут быть реализованы с помощью алгоритмов маршрутизации, которые классифицируются по следующим классификационным признакам:

## 1. По способу выбора наилучшего маршрута

Все алгоритмы маршрутизации делятся на 2 группы:

* Одношаговые алгоритмы маршрутизации

Каждый маршрутизатор при выборе маршрута определяет только одно звено этого маршрута. Ответственность только за один шаг этого маршрута.

Одношаговые алгоритмы реальней на этапе инициализации

* Многошаговые алгоритмы маршрутизации (алгоритм маршрутизации от источника)

Весь маршрут задается в уже отправленном пакете узлом источника. Многошаговые алгоритмы считаются перспективней

## 2. По способу построения таблиц маршрутизации

* Алгоритмы простой маршрутизации
  + - Таблиц маршрутизации как правило нет, или таблицы являются очень примитивными (они не передают никакой маршрутной информации)
    - Алгоритмы случайной маршрутизации – пакет посылается в случайном направлении
    - Лавинная маршрутизация (алгоритмы заполнения) – пакеты посылаются во все выходные направления, во все порты
    - Алгоритмы скорейшей передачи (алгоритм горячей картошки) – как только маршрутизатор поучает пакет – он старается скорее его отослать
    - Алгоритмы кратчайшей очереди (наименьшей загрузки) – информация идет на порт, который наименее загружен
    - Алгоритм по предыдущему опыту – таблица маршрутизации очень примитивна, есть запись, дублирующая предыдущую передачу
    - Сегодня используют лавинные алгоритмы. Они самые быстрые по доставке информации, а так же могут информировать об экстренной информации возникающей в сети передачи данных.
* Алгоритмы фиксированной и статической маршрутизации

Это алгоритмы, которые заносятся администратором сети, поэтому все записи являются статическими или меняется время функционирования сети. Используются для сетей с простой топологией и чаще всего применяются в теоретических приложениях.

1)    Однопутевые (одномаршрутные, безальтернативные)

2)    Многопутевые (многомаршрутные, допускающие альтернативу)

* **Адаптивные алгоритмы**, = алгоритмы динамической маршрутизации

Они самые реальные и самые распространенные. Автоматическое построение таблиц маршрутизации, эти алгоритмы адаптированы к изменениям в сети.

К Адаптивным алгоритмам предъявляются следующие требования:

a)     Адитивность – скорость адаптивности алгоритма к изменениям в сети. Для достижения скорости алгоритмы должны быть простыми

b)     Адаптивные алгоритмы маршрутизации должны обеспечивать если не оптимальные, то хотя бы наилучшие маршрутные решения.

c)     Сходимость алгоритма – это когда алгоритм после некоторого времени приводит к однозначному результату.

## 3. По месту выбора маршрутов (маршрутного решения)

По месту выбора маршрутного решения все алгоритмы маршрутизации делятся на три класса:

* Изолированные алгоритмы (локальные) – нет никакого обмена маршрутной информацией и каждый маршрутизатор принимает решение на основании той информации, которую он сам собрал.
* Централизованные – вся маршрутная информация со всех маршрутизаторов стекается в сетевой маршрутный центр – он ответственный за определение оптимальных маршрутов и сбор маршрутной информации. Возможны 2 подхода:
* Подход виртуального канала – маршрут определяется на основе оптимальной информации, посылаемой во все промежуточные маршрутизаторы. Недостатком является уязвимость маршрутного центра
* Подход формирования по таблице для каждого маршрутизатора.
* Распределенные – это самые распространенные алгоритмы, где все маршрутизаторы участвуют в сборе и распространении маршрутной информации и работа по выбору наилучшего маршрута распределена между всеми маршрутизаторами.

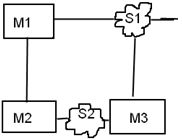
# 4. По виду информации которой обмениваются маршрутизаторы

Все алгоритмы делятся на 2 класса:

a)     Дистанционно-векторные алгоритмы – RIP протокол (протокол маршрутной информации). В дистанционно-векторном: каждый маршрутизатор периодически всем своим соседям передаёт вектор сообщения, где указывает адреса всех известных ему подсетей и расстояние до них, в качестве (расстояния используется промежуточные узлы). Недостатки плохая адаптация к отказам маршрутизаторов, возможность возникновения маршрутных петель.

Каждый маршрутизатор периодически всем своим соседям передает вектор сообщений, где указывается адреса известных ему подсетей и расстояния до них от данного маршрутизатора.

**Принцип действия:**



Каждый маршрутизатор распространяет соседям о всех своих близких соседях:

1.     Адрес соседней подсети

2.     Тип интерфейса  (М-М (маршрутизатор – марш) и М-S (марш – сеть) )

3.     Метрика интерфейса (пропускные способности каждого из путей, время задержки и метрика надежности)

 Соседний маршрутизатор получает информацию без коррекции, и через некоторое время все маршрутизаторы будут иметь полную информацию о всех подсетях и маршрутизаторах. Вся информация записывается в Базу Данных, после чего каждый маршрутизатор знает топологию, определяет кратчайшие маршруты для всех подсетей.

Для этого используется **алгоритм Дейкстра** – алгоритм определения кратчайшего маршрута, где каждое звено этого маршрута записывается в таблицу маршрутизации. Вычисления происходят по всей метрике.

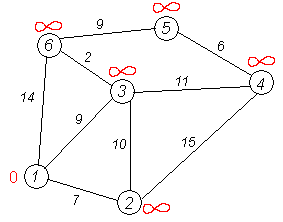
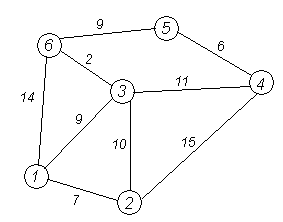
**Маршрутизатор не имеет сетевого адреса!!**

Для проверки в каждый маршрутизатор пересылается сообщение каждые 10 сек и если ответа нет, то таблица корректируется без учета вышедшего из строя маршрутизатора. Топология сети в течении длительной работы сети не меняется, информационные потоки тоже не меняются после их распределения.

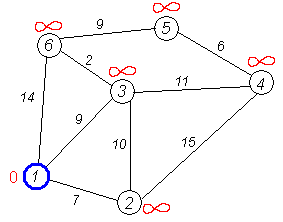
**14. Алгоритм маршрутизации, основанный на состоянии линий (канала).**

Рассмотрим выполнение алгоритма на примере графа, показанного на рисунке. Пусть требуется найти кратчайшие расстояния от 1-й вершины до всех остальных.

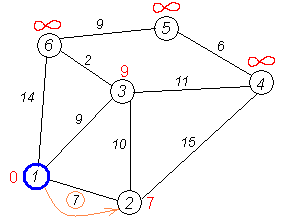
Кружками обозначены вершины, линиями — пути между ними (ребра графа). В кружках обозначены номера вершин, над ребрами обозначена их «цена» — длина пути. Рядом с каждой вершиной красным обозначена метка — длина кратчайшего пути в эту вершину из вершины 1.



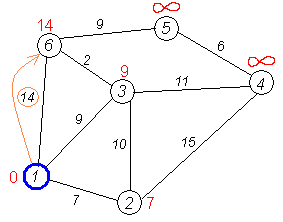
**Первый шаг**. Рассмотрим шаг алгоритма Дейкстры для нашего примера. Минимальную метку имеет вершина 1. Её соседями являются вершины 2, 3 и 6.



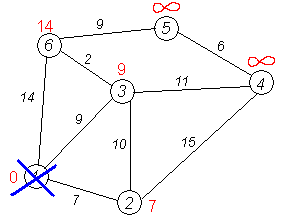
Первый по очереди сосед вершины 1 — вершина 2, потому что длина пути до неё минимальна. Длина пути в неё через вершину 1 равна сумме кратчайшего расстояния до вершины 1, значению её метки, и длины ребра, идущего из 1-й в 2-ю, то есть 0 + 7 = 7. Это меньше текущей метки вершины 2, бесконечности, поэтому новая метка 2-й вершины равна 7.



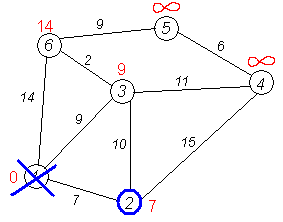
Аналогичную операцию проделываем с двумя другими соседями 1-й вершины — 3-й и 6-й.



Все соседи вершины 1 проверены. Текущее минимальное расстояние до вершины 1 считается окончательным и пересмотру не подлежит (то, что это действительно так, впервые доказал Э. Дейкстра). Вычеркнем её из графа, чтобы отметить, что эта вершина посещена.



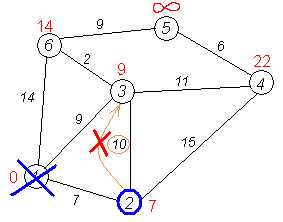
**Второй шаг**. Шаг алгоритма повторяется. Снова находим «ближайшую» из непосещенных вершин. Это вершина 2 с меткой 7.



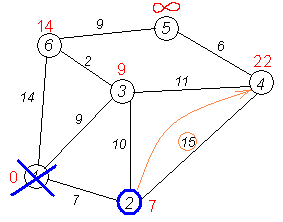
Снова пытаемся уменьшить метки соседей выбранной вершины, пытаясь пройти в них через 2-ю вершину. Соседями вершины 2 являются вершины 1, 3 и 4.

Первый (по порядку) сосед вершины 2 — вершина 1. Но она уже посещена, поэтому с 1-й вершиной ничего не делаем.

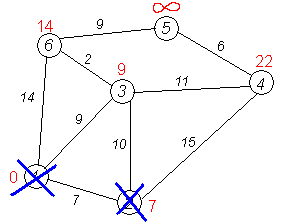
Следующий сосед вершины 2 — вершина 3, так как имеет минимальную метку из вершин, отмеченных как не посещённые. Если идти в неё через 2, то длина такого пути будет равна 17 (7 + 10 = 17). Но текущая метка третьей вершины равна 9<17, поэтому метка не меняется.



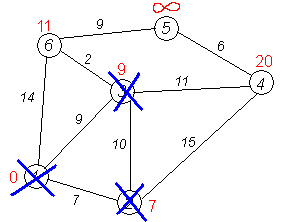
Ещё один сосед вершины 2 — вершина 4. Если идти в неё через 2-ю, то длина такого пути будет равна сумме кратчайшего расстояния до 2-й вершины и расстояния между вершинами 2 и 4, то есть 22 (7 + 15 = 22). Поскольку 22<https://studfiles.net/html/2706/123/html_FRiVjN7Sd8.x2cx/img-qY9RuO.png, устанавливаем метку вершины 4 равной 22.



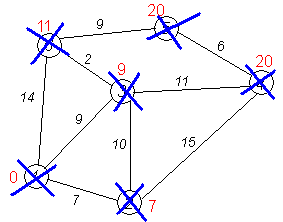
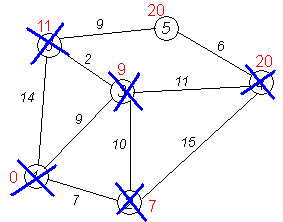
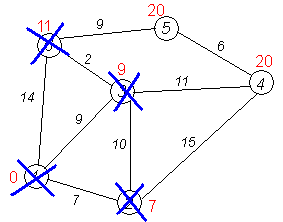
Все соседи вершины 2 просмотрены, замораживаем расстояние до неё и помечаем её как посещенную.



**Третий шаг**. Повторяем шаг алгоритма, выбрав вершину 3. После её «обработки» получим такие результаты:



**Дальнейшие шаги**. Повторяем шаг алгоритма для оставшихся вершин. Это будут вершины 6, 4 и 5, соответственно порядку.



**Завершение выполнения алгоритма**. Алгоритм заканчивает работу, когда нельзя больше обработать ни одной вершины. В данном примере все вершины зачеркнуты, однако ошибочно полагать, что так будет в любом примере - некоторые вершины могут остаться незачеркнутыми, если до них нельзя добраться. Результат работы алгоритма виден на последнем рисунке: кратчайший путь от вершины 1 до 2-й составляет 7, до 3-й — 9, до 4-й — 20, до 5-й — 20, до 6-й — 11.

1. **Алгоритм дистанционно-векторной маршрутизации.**

В **дистанционно-векторных алгоритмах (DVA)** каждый маршрутизатор периодически и широковещательно рассылает по сети вектор, компонентами которого являются расстояния от данного маршрутизатора до всех известных ему сетей. Пакеты протоколов маршрутизации обычно называют объявлениями о расстояниях, так как с их помощью маршрутизатор объявляет остальным маршрутизатора известные ему сведения о конфигурации сети.

Получив от некоторого соседа вектор расстояний (дистанций) до известных тому сетей, маршрутизатор наращивает компоненты вектора на величину расстояния от себя до данного соседа, он дополняет вектор информацией об известных ему самому других сетях, о которых он узнал непосредственно или из аналогичных объявлений других маршрутизаторов. Обновленное значение вектора маршрутизатор рассылает своим соседям. В итоге, каждый маршрутизатор узнает через соседние маршрутизаторы информацию обо всех имеющихся в составной сети сетях и о расстояниях до них. Затем он выбирает из нескольких альтернативных маршрутов к каждой сети тот маршрут, который обладает наименьшим значением метрики. Маршрутизатор, передавший информацию о данном маршруте, отмечается в таблице маршрутизации как следующий (next hop).

1. **Интернет-протокол. Адресация в протоколе IPv4. Классы сетей. CIDR. Маска подсети.**

Отдельные компьютерные сети были объединены в всемирную сеть Интернет благодаря маршрутизируемому протоколу сетевого уровня стека TCP/IP – интернет-протоколу IP. IP- протокол обеспечивает доставку пакетов данных между любыми узлами сети через произволь- ное число промежуточных узлов (маршрутизаторов). Он не гарантирует надежной доставки пакета до адресата – в частности, пакеты могут прийти не в том порядке, в котором были от- правлены, продублироваться, оказаться поврежденными (обычно поврежденные пакеты уни- чтожаются) или не прийти вовсе. Гарантию безошибочной доставки пакетов дают некоторые протоколы более высокого уровня – транспортного, например, TCP

**IP­адреса (Internet Protocol version 4, интернет протокол версии 4) – представляют собой основной тип адресов, используемый на сетевом уровне ​**моделиOSI​,для осуществления передачи пакетов междусетями.

В современной сети интернет используется IPv4. В протоколе IP этой версии каждому узлу сети ставится в соответствие IP-адрес длиной к октета (4 байта). При этом, компьютеры в подсетях объединяются общими начальными битами адреса. IPv4 использует 32-битные адреса, ограничивающие адресное пространство 2 в 32 степени возможными уникальными адресами. Традиционной формой записи IPv4 адреса является запись в виде 4-х десятичных чисел (от 0 до 255), разделенных точками с указываемой через знак косой черты длины маски подсети. Некоторые адреса IPv4 зарезервированы для специальных целей и не предназначены для глобальной маршрутизации.



Сети класса А — это огромные сети. Маска сети класса А: 255.0.0.0. В каждой сети такого класса может находиться 16777216 адресов.

Сети класса В — это средние сети. Маска такой сети — 255.255.0.0. Эта сеть содержит 65536 адресов.

Сеть класса С — маленькие сети. Содержат 256 адресов (на самом деле всего 254 хоста, так как номера 0 и 255 зарезервированы). Маска сети класса С — 255.255.255.0.

Если адрес начинается с последовательности битов 10, то данная сеть относится к классу В, а если с последовательности 110, то — к классу С. Если адрес начинается с последовательности 1110, то сеть является сетью класса D, а сам адрес является особым — групповым (multicast). Если в пакете указан адрес сети класса D, то этот пакет должны получить все хосты, которым присвоен данный адрес. Адреса класса Е зарезервированы для будущего применения.

Группа IETF приняла стандарт CIDR (Classless Inter-Domain Routing — бесклассовая внутридоменная маршрутизация). Данный стандарт позволяет использовать сетевую часть адреса любой длины, не обязательно кратной 8 бит. Сетевой адрес стандарта CIDR записывается четырьмя десятичными числами, разделенными точками, например a.b.c.d/x, где х указывает количество разрядов в сетевой части адреса.

В терминологии сетей TCP/IP **маской подсети** или **маской сети** называется битовая маска, определяющая, какая часть IP-адреса узла сети относится к адресу сети, а какая — к адресу самого узла в этой сети. Например, узел с IP-адресом 12.34.56.78 и маской подсети 255.255.255.0 находится в сети 12. Разбиение одной большой сети на несколько маленьких подсетей позволяет упростить маршрутизацию.34.56.0/24 с длиной префикса 24 бита. Маски подсети являются основой метода бесклассовой маршрутизации (CIDR). При этом подходе маску подсети записывают вместе с IP-адресом в формате «*IP-адрес/количество единичных бит в маске*». Число после слэша означает количество единичных разрядов в маске подсети.

1. **Протокол IP. Формат кадра. Фрагментация IP-дейтаграмм.**

***Internet Protocol*** (**IP**, досл. «межсетевой протокол») — [маршрутизируемый](https://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D1%80%D1%88%D1%80%D1%83%D1%82%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F) [протокол](https://ru.wikipedia.org/wiki/%D0%A1%D0%B5%D1%82%D0%B5%D0%B2%D0%BE%D0%B9_%D0%BF%D1%80%D0%BE%D1%82%D0%BE%D0%BA%D0%BE%D0%BB) [сетевого уровня](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D1%82%D0%BE%D0%BA%D0%BE%D0%BB%D1%8B_%D1%81%D0%B5%D1%82%D0%B5%D0%B2%D0%BE%D0%B3%D0%BE_%D1%83%D1%80%D0%BE%D0%B2%D0%BD%D1%8F) [стека](https://ru.wikipedia.org/wiki/%D0%A1%D1%82%D0%B5%D0%BA) [TCP/IP](https://ru.wikipedia.org/wiki/TCP/IP). Именно IP стал тем протоколом, который объединил отдельные [компьютерные сети](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%D0%BD%D0%B0%D1%8F_%D1%81%D0%B5%D1%82%D1%8C) во всемирную сеть [Интернет](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D1%80%D0%BD%D0%B5%D1%82). Неотъемлемой частью протокола является *адресация* сети.

IP объединяет сегменты сети в единую сеть, обеспечивая доставку пакетов данных между любыми узлами сети через произвольное число промежуточных узлов ([маршрутизаторов](https://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D1%80%D1%88%D1%80%D1%83%D1%82%D0%B8%D0%B7%D0%B0%D1%82%D0%BE%D1%80" \o "Маршрутизатор)). Он классифицируется как протокол [сетевого](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D1%82%D0%BE%D0%BA%D0%BE%D0%BB%D1%8B_%D1%81%D0%B5%D1%82%D0%B5%D0%B2%D0%BE%D0%B3%D0%BE_%D1%83%D1%80%D0%BE%D0%B2%D0%BD%D1%8F) уровня по [сетевой модели OSI](https://ru.wikipedia.org/wiki/%D0%A1%D0%B5%D1%82%D0%B5%D0%B2%D0%B0%D1%8F_%D0%BC%D0%BE%D0%B4%D0%B5%D0%BB%D1%8C_OSI). IP не гарантирует надёжной доставки пакета до адресата — в частности, пакеты могут прийти не в том порядке, в котором были отправлены, продублироваться, оказаться повреждёнными или не прийти вовсе.

Формат кадра



Версия – для IPv4 значение поля должно быть равно 4;

* IHL (Internet Header Length) – длина заголовка IP-пакета в 32-битных словах (dword). Именно это поле указывает на начало блока данных в пакете. Минимальное корректное значение для этого поля равно 5; 22 ∙ Длина пакета – длина пакета в октетах, включая заголовок и данные. Минимальное кор- ректное значение для этого поля равно 20, максимальное – 65 535;
* Идентификатор – значение, назначаемое отправителем пакета и предназначенное для определения корректной последовательности фрагментов при сборке пакета. Для фраг- ментированного пакета все фрагменты имеют одинаковый идентификатор;
* 3 бита флагов. Первый бит должен быть всегда равен нулю, второй бит DF (don’t fragment) определяет возможность фрагментации пакета и третий бит MF (more fragments) показы- вает, не является ли этот пакет последним в цепочке пакетов;
* Смещение фрагмента – значение, определяющее позицию фрагмента в потоке данных. Смещение задается количеством 8-ми байтовых блоков, поэтому это значение требует умножения на 8 для перевода в байты;
* Время жизни (TTL) – число маршрутизаторов, которые может пройти этот пакет. При прохождении маршрутизатора это число уменьшается на единицу. Если значение этого поля равно нулю, то пакет должен быть отброшен, и отправителю пакета может быть послано сообщение Time Exceeded (ICMP тип 11 код 0);
* Протокол – идентификатор интернет-протокола следующего уровня указывает, данные какого протокола содержит пакет, например, TCP, UDP, или ICMP;
* Контрольная сумма заголовка;
* Differentiated Services Code Point (DSCP) – используется для разделения трафика на клас- сы обслуживания, например для установки чувствительному к задержкам трафику;
* Указатель перегрузки (Explicit Congestion Notification, ECN) – Предупреждение о пере- грузке сети без потери пакетов. Является необязательной функцией и используется только если оба хоста ее поддерживают;
* Адреса источника-назначения;
* Опции.

**IP-фрагментация**это разбиение датаграммы на множество частей, которые могут быть повторно собраны позже. Для IP-фрагментации и повторной сборки используются такие поля из IP заголовка как источник, адресат, идентификация, полная длина, и смещение фрагмента, наряду с флажками "больше фрагментов" (MF) и "не фрагментировать" (DF).

1. **Протокол маршрутизации RIP, OSPF.**

Протокол маршрутной информации (Routing Information Protocol, RIP) — внутренний протокол маршрутизации дистанционно-векторного типа. Будучи одним из наиболее ранних протоколов обмена маршрутной информацией, он до сих пор чрезвычайно распространен в вычислительных сетях ввиду простоты реализации.

В настоящее время протокол RIP для IP представлен двумя версиями. RIP v.1 не поддерживает маски, т. е. он распространяет между маршрутизаторами информацию только о номерах сетей и расстояниях до них, но не о масках этих сетей, считая, что все адреса принадлежат к стандартным классам A, B или С. RIP v.2 передает данные о масках сетей, поэтому он в большей степени соответствует современным требованиям. Так как при построении таблиц маршрутизации работа второй версии принципиально не отличается от первой, то в дальнейшем для упрощения записей будет описываться работа первой версии.

Протокол маршрутизации OSFP (Open Shortest Path First) представляет собой протокол состояния связей, использующий алгоритм SPF поиска кратчайшего пути в графе(Алгоритм Дейкстры).

OSPF является протоколом маршрутизации с объявлением состояния о канале (link-state). Это значит, что он требует отправки объявлений о состоянии канала (link-state advertisement - LSA) во все роутеры, которые находятся в пределах одной и той же иерархической области. В объявления LSA протокола OSPF включается информация о подключенных интерфейсах, об использованных показателях и о других переменных. По мере накопления роутерами OSPF информации о состоянии канала, они используют алгоритм SPF для расчета наикратчайшего пути к каждому узлу.

1. **Устройство маршрутизатора.**

На сетевом уровне фактическая работа (то есть то, ради чего существует сетевой уровень) заключается в продвижении дейтаграмм, а ее исполнителями являются маршрутизаторы. Ключевой составляющей продвижения дейтаграмм является их передача с входной линии маршрутизатора на его выходную линию

Маршрутизатор состоит из четырех компонентов.

* Входные порты. Он выполняет функции физического уровня, завершая входную физическую линию маршрутизатора. Он также осуществляет функции канального уровня, необходимые для взаимодействия с функциями канального уровня на другой стороне линии связи. Еще он выполняет функции поиска и продвижения данных, так что пакет, переправленный в коммутационный блок маршрутизатора, на выходе из него появляется из того порта, из которого следует. Управляющие пакеты продвигаются из входного порта в маршрутный процессор.
* Коммутационный блок. Коммутационный блок соединяет входные порты маршрутизатора с его выходными портами. Коммутационный блок целиком располагается внутри маршрутизатора — сеть внутри сетевого маршрутизатора!
* Выходные порты. Выходной порт хранит пакеты, переправленные ему через коммутационный блок, а затем передает пакеты по выходной линии. Таким образом, выходной порт осуществляет функции физического и канального уровней, обратные функциям входного порта.
* Маршрутный процессор. Маршрутный процессор выполняет функции протоколов маршрутизации, обрабатывает информацию о маршрутах, а также выполняет функции управления сетью в маршрутизаторе.

1. **Протокол IPv6. Формат дейтаграммы.**

**IPv6** (англ.​***Internet Protocol version 6****​*)—новая версия протоколаIP,призванная решить проблемы, с которыми столкнулась предыдущая версия (IPv4) при её использовании в

Интернете, за счёт использования длины адреса 128 бит вместо 32.

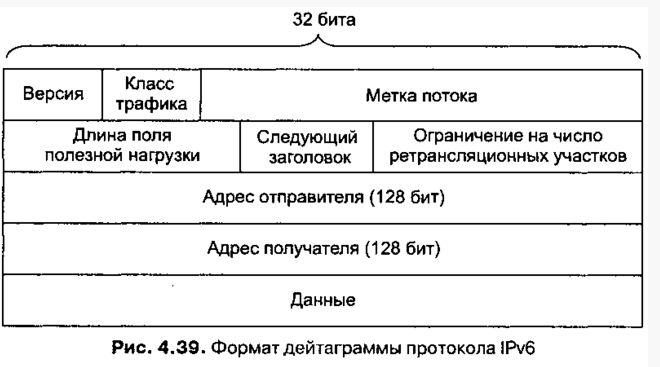
По​сле того, как адресное пространство в IPv4 закончится, два стека протоколов — IPv6 и IPv4 — будут использоваться параллельно (англ. dual stack), с постепенным увеличением доли трафика IPv6, по сравнению с IPv4. Такая ситуация станет возможной из­за наличия огромного количества устройств, в том числе устаревших, не поддерживающих IPv6 и требующих специального преобразования для работы с устройствами, использующими только IPv6.

Несмотря на огромный размер адреса IPv6, благодаря этим улучшениям заголовок пакета удлинился всего лишь вдвое: с 20 до 40 байт.

Из IPv6 убраны функции, усложняющие работу маршрутизаторов:

* Маршрутизаторы больше не должны фрагментировать пакет, вместо этого пакет отбрасы- вается с ICMP-уведомлением о превышении MTU (максимальный размер полезного блока данных одного пакета, который может быть передан протоколом без фрагментации);
* Из IP-заголовка исключена контрольная сумма. С учётом того, что канальные (Ethernet) и транспортные (TCP и UDP) протоколы имеют свои контрольные суммы, ещё одна кон- трольная сумма на уровне IP воспринимается как излишняя.

Существуют различные типа адресов IPv6: unicast, multicast, anycast.



* Версия. Это 4-разрядное поле идентифицирует номер версии протокола IP и для протокола IPv6 содержит значение 6.
* Класс трафика. Это 8-разрядное поле.
* Метка потока. Это 20-разрядное поле используется для идентификации «потока» дейтаграмм.
* Длина полезной нагрузки. Это 16-разрядное поле обрабатывается как целое число без знака и содержит количество байтов в 1Ру6-дейтаграмме, следующих за 40-разрядным заголовком дейтаграммы.
* Следующий заголовок. Это поле идентифицирует протокол, которому доставляется содержимое (поле данных) дейтаграммы.
* Ограничение на число ретрансляционных участков. Содержимое этого поля уменьшается на единицу на каждом маршрутизаторе, через который проходит дейтаграмма. Когда содержимое этого поля достигает нуля, дейтаграмма уничтожается.
* Адреса отправителя и получателя. Различные форматы 128-разрядных IPv6-адресов описаны в RFC 2373.
* Данные. Это полезная нагрузка IPv6-дейтаграммы. Когда дейтаграмма достигает пункта назначения, полезная нагрузка извлекается из нее и передается протоколу, указанному в поле следующего заголовка.

1. **Транспортный уровень. Службы транспортного уровня.**

Основная функция транспортного уровня — принять данные от сеансового уровня, разбить их при необходимости на небольшие части, передать их сетевому уровню и гарантировать, что эти части в правильном виде прибудут по назначению.

Протоколы транспортного уровня поддерживаются оконечными системами, однако не поддерживаются маршрутизаторами. Маршрутизаторы обрабатывают сообщения сетевого уровня и не оказывают влияния на сообщения транспортного уровня. На передающей стороне транспортный протокол преобразует сообщения приложения в сообщения транспортного уровня. Это преобразование иногда подразумевает разбиение сообщения, полученного от приложения, на несколько фрагментов с добавлением к каждому фрагменту заголовка транспортного уровня. Затем транспортный уровень передает свои сообщения сетевому уровню. На приемной стороне транспортный уровень получает сообщения сетевого уровня, преобразует их в сообщения транспортного уровня, отбрасывает заголовки, производит необходимую обработку (например, сборку) и передает результат приложению.

Существует несколько протоколов транспортного уровня. Например, в Интернете протоколами транспортного уровня являются TCP и UDP. Каждый из протоколов обладает собственным набором служб, предоставляющих услуги приложениям.

1. **Мультиплексирование и демультиплексирование на транспортном уровне (Механизмы идентификации двух процессов в сетевом взаимодействии???).**

Мультиплексирование – образование из нескольких отдельных потоков общего агрегированного потока, который передается по одному физическому каналу (способ разделения одного имеющегося физического канала между несколькими узлами сети). UDP принимает сообщения от различных процессов, различая их по заданным номерам портов. После добавления заголовка UDP передает пользовательскую дейтаграмму IP. На стороне приемника есть только один UDP. Однако мы можем иметь много процессов, которые могут получать пользовательские дейтаграммы от IP и здесь уже требуется демультиплексирование – операция, обратная мультиплексированию. После проверки ошибок отбрасывания заголовка UDP доставляет каждое сообщение к соответствующему процессу, основываясь на номере порта.

1. **Протокол UDP. Службы протокола UDP.**

Функции UDP сводятся к операциям мультиплексирования и демультиплексирования, а также несложной проверке наличия ошибок в данных. Таким образом, при использовании U DP приложение почти напрямую взаимодействует с протоколом сетевого уровня IP.

UDP получает сообщения от прикладного уровня, добавляет к ним поля номеров портов отправителя и получателя для демультиплексирования приемной стороной, а также два других специальных поля и передает полученный сегмент сетевому уровню. Сетевой уровень заключает сегмент в дейтаграмму и «по возможности» передает ее хосту назначения. Если последний успешно получает сегмент, протокол UDP с помощью поля номера порта получателя направляет данные сегмента нужному процессу. Обратите внимание на то, что протокол UDP не предусматривает процедуры рукопожатия перед началом передачи сегментов. Поэтому говорят, что UDP осуществляет передачу данных без установления соединения.

**UDP**— это протокол без установления соединения и с негарантированной доставкой пакетов. (Типа: крикнул что-нибудь, а услышат тебя или нет — неважно).

1. **Протокол TCP. Службы протокола TCP.**

**TCP**— это протокол с установлением соединения и с гарантированной доставкой пакетов. Сначала производится обмен специальными пакетами для установления соединения, происходит что-то вроде рукопожатия (-Привет. -Привет. -Поболтаем? -Давай.). Далее по этому соединению туда и обратно посылаются пакеты (идет беседа), причем с проверкой, дошел ли пакет до получателя. Если пакет не дошел, то он посылается повторно («повтори, не расслышал»).

1. **Управление TCP-соединением. Контроль перегрузок.**

В протоколе TCP соединения устанавливаются с помощью "тройного рукопожатия".Чтобы установить соединение, одна сторона (например, сервер) пассивно ожидает входящего соединения, выполняя примитивы LISTEN и ACCEPT, либо указав конкретный источник, либо не указав никого конкретно.

Другая сторона (например, клиент) выполняет примитив CONNECT, указывая IP-адрес и порт, с которым он хочет установить соединение, максимальный размер ТСР-сегмента и, по желанию, некоторые данные пользователя (например, пароль).

Примитив CONNECT посылает ТСР-сегмент с установленным битом SYN и сброшенным битом АСК и ждет ответа.

Когда этот сегмент прибывает в пункт назначения, TCP-сущность проверяет, выполнил ли какой-нибудь процесс примитив LISTEN, указав в качестве параметра тот же порт, который содержится в поле Порт получателя. Если такого процесса нет, она отвечает отправкой сегмента с установленным битом RST для отказа от соединения.

Если какой-либо процесс ожидает соединения на данном порту, то входящий ТСР-сегмент вручается этому процессу. Затем процесс может принять соединение или отказаться от него. Если процесс принимает соединение, он отсылает в ответ подтверждение.

Если два хоста одновременно попытаются установить соединение друг с другом, используя одну и ту же пару сокетов, то будет установлено лишь одно соединение, так как пара конечных точек однозначно определяет соединение.

Подход, применяемый в TCP, заключается в ограничении скорости передачи источников в зависимости от наблюдаемой перегрузки. Если перегрузки в сети не наблюдаются, источник может увеличивать скорость передачи; в противном случае скорость передачи принудительно снижается. При подобном подходе возникают три вопроса. Во-первых, каким образом ограничить скорость передачи данных источником? Во-вторых, как источник может определить наличие перегрузки на пути между ним и приемником? В-третьих, каков алгоритм изменения скорости передачи в зависимости от перегрузки? Мы рассмотрим эти вопросы на примере алгоритма Reno, использующегося в современных операционных системах для контроля перегрузки ТСР-соединений. Для определенности будем считать, что источник осуществляет передачу большого файла данных.

Сначала рассмотрим, каким образом в TCP удается ограничивать скорость передачи данных. Механизм контроля перегрузки использует дополнительную переменную CongWin, называемую окном перегрузки. Окно перегрузки влияет на скорость передачи данных в сеть следующим образом: объем неподтвержденных данных, которые может передать источник, не превышает минимального из значений CongWin и RcvWindow

Для того чтобы наглядно представить разницу между контролем перегрузок и контролем потока, предположим, что объем приемного буфера TCP настолько велик, что ограничение, накладываемое окном приема, можно считать несущественным. Таким образом, количество неподтвержденных данных определяется только переменной CongWin.

Ограничение объема неподтвержденных данных косвенно влияет на скорость передачи источника. Представьте себе соединение, в котором отсутствуют потери и задержки пакетов. Если в начале времени оборота источнику разрешается передача CongWin байтов, а по истечении времени оборота он получает квитанции для переданных данных, то скорость его передачи равна отношению CongWin ко времени оборота. Таким образом, меняя значение CongWin, протокол TCP может регулировать скорость передачи данных через соединение.

Теперь рассмотрим, каким образом TCP определяет наличие перегрузок на пути соединения. Введем событие «потеря пакета», наступающее по истечении интервала ожидания или при получении тройного подтверждения от принимающей стороны. При значительных перегрузках в сети буферы одного или нескольких маршрутизаторов переполняются, что приводит к потере дейтаграммы. Потеря дейтаграммы обнаруживается источником при наступлении события «потеря пакета», то есть по истечении интервала ожидания либо при получении трех дублирующих квитанций. Потеря пакета является признаком перегрузок в сети.

1. **Прикладной уровень. Протоколы прикладного уровня. Сетевые службы прикладного уровня.**

Прикладной уровень - седьмой уровень модели OSI, предназначенный для обеспечения взаимодействия пользователя с приложениями на удаленном компьютере.

Прикладной уровень обеспечивает:

- описание форм и методов взаимодействия прикладных процессов;

- управление заданиями, передачу файлов, управление системой и т.д.;

- идентификацию пользователей по их паролям, адресам и электронным подписям;

Прикладной уровень модели OSI является интерфейсом между приложением и средой передачи, т.е. уровнями с 1 по 6 эталонной модели взаимодействия открытых систем. Примечательно, что уровень 7 в действительности не содержит приложение, а лишь предоставляет интерфейс к среде передачи для приложения, которое не относится к модели OSI.

Протокол прикладного уровня — протокол верхнего (7-ого) уровня сетевой модели OSI, обеспечивает взаимодействие сети и пользователя. Уровень разрешает приложениям пользователя иметь доступ к сетевым службам, таким как обработчик запросов к базам данных, доступ к файлам, пересылке электронной почты. Также отвечает за передачу служебной информации, предоставляет приложениям информацию об ошибках и формирует запросы к уровню представления.

1. **Электронная почта. Протоколы SMTP, POP, IMAP.**

POP3 используется почтовым клиентом для получения сообщений электронной почты с сервера. Обычно используется в паре с протоколом SMTP.

Альтернативным протоколом для сбора сообщений с почтового сервера является IMAP.

В протоколе POP3 предусмотрено 3 состояния сеанса:

* авторизация: клиент проходит процедуру аутентификации;
* транзакция: клиент получает информацию о состоянии почтового ящика, принимает и удаляет почту;
* обновление: сервер удаляет выбранные письма и закрывает соединение.

IMAP – протокол прикладного уровня для доступа к электронной почте. Почтовая программа, использующая этот протокол, получает доступ к хранилищу корреспонденции на сервере так, как будто эта корреспонденция расположена на компьютере получателя. Электронными письмами можно манипулировать с компьютера пользователя (клиента) без необходимости постоянной пересылки с сервера и обратно файлов с полным содержанием писем. Для отправки писем используется протокол SMTP.

Преимущества по сравнению с POP3

IMAP был разработан для замены более простого протокола POP3 и имеет следующие преимущества по сравнению с последним:

* Письма хранятся на сервере, а не на клиенте.
* Поддержка нескольких почтовых ящиков (или папок).
* Возможно создание общих папок, к которым могут иметь доступ несколько пользователей.
* Информация о состоянии писем хранится на сервере и доступна всем клиентам. Письма могут быть помечены как прочитанные, важные и т. п.
* Поддержка поиска на сервере.
* Поддержка онлайн-работы.
* Предусмотрен механизм расширения возможностей протокола.

SMTP– это сетевой протокол, предназначенный для передачи электронной почты в сетях TCP/IP.

SMTP используется для отправки почты от пользователей к серверам и между серверами для дальнейшей пересылки к получателю. Для приёма почты почтовый клиент должен использовать протоколы POP3 или IMAP.

Чтобы доставить сообщение до адресата, необходимо переслать его почтовому серверу домена, в котором находится адресат.

1. **Формат сообщений электронной почты. MIME.**

Формат почтового сообщения Internet определен в документе RFC-822 (Standard for ARPA Internet Text Message). Почтовое сообщение состоит из трех частей: конверта, заголовка и тела сообщения. Пользователь видит только заголовок и тело сообщения. Конверт используется только программами доставки. Заголовок всегда находится перед телом сообщения и отделен от него пустой строкой. RFC-822 регламентирует содержание заголовка сообщения. Заголовок состоит из полей. Поля состоят из имени поля и содержания поля. Имя поля отделено от содержания символом ":". Минимально необходимыми являются поля Date, From, cc или То

**Multipurpose Internet Mail Extensions** (**MIME**) — стандарт, описывающий передачу различных типов данных по электронной почте, а также, шире, спецификация для кодирования информации и форматирования сообщений таким образом, чтобы их можно было пересылать по Интернету.

1. **Отличия HTTP 1.1 от HTTP 1.0.**

HTTP — протокол прикладного уровня, реализованный поверх протокола TCP/IP. HTTP определяет, как взаимодействуют между собой клиент и сервер, как запрашивается и передаётся контент по интернету.

**Отличия HTTP 1.1 от HTTP 1.0**

* **Новые HTTP-методы** — PUT, PATCH, HEAD, OPTIONS, DELETE.
* **Идентификация хостов.** В HTTP/1.0 заголовок Host не был обязательным, а HTTP/1.1 сделал его таковым.
* **Постоянные соединения.** Как говорилось выше, в HTTP/1.0 одно соединение обрабатывало лишь один запрос и после этого сразу закрывалось, что вызывало серьёзные проблемы с производительностью и проблемы с задержками. В HTTP/1.1 появились постоянные соединения, т.е. соединения, которые по умолчанию не закрывались, оставаясь открытыми для нескольких последовательных запросов. Чтобы закрыть соединение, нужно было при запросе добавить заголовок Connection: close. Клиенты обычно посылали этот заголовок в последнем запросе к серверу, чтобы безопасно закрыть соединение.
* **Потоковая передача данных,** при которой клиент может в рамках соединения посылать множественные запросы к серверу, не ожидая ответов, а сервер посылает ответы в той же последовательности, в которой получены запросы. Но, вы можете спросить, как же клиент узнает, когда закончится один ответ и начнётся другой? Для разрешения этой задачи устанавливается заголовок Content-Length, с помощью которого клиент определяет, где заканчивается один ответ и можно ожидать следующий.

HTTP/1.1 ввёл [сhunked encoding](https://ru.wikipedia.org/wiki/Chunked_transfer_encoding) — механизм разбиения информации на небольшие части (chunks) и их передачу.

* **Chunked Transfers** если контент строится динамически и сервер в начале передачи не может определить Content-Length, он начинает отсылать контент частями, друг за другом, и добавлять Content-Length к каждой передаваемой части. Когда все части отправлены, посылается пустой пакет с заголовком Content-Length, установленным в 0, сигнализируя клиенту, что передача завершена. Чтобы сказать клиенту, что передача будет вестись по частям, сервер добавляет заголовок Transfer-Encoding: chunked.
* В отличие от [базовой аутентификации](https://en.wikipedia.org/wiki/Basic_access_authentication) в HTTP/1.0, в HTTP/1.1 добавились [дайджест-аутентификация](https://ru.wikipedia.org/wiki/%D0%94%D0%B0%D0%B9%D0%B4%D0%B6%D0%B5%D1%81%D1%82-%D0%B0%D1%83%D1%82%D0%B5%D0%BD%D1%82%D0%B8%D1%84%D0%B8%D0%BA%D0%B0%D1%86%D0%B8%D1%8F) и [прокси-аутентификация](http://wiki.squid-cache.org/Features/Authentication).
* **Кэширование**.
* **Диапазоны байт** (byte ranges).
* **Кодировки**
* **Согласование содержимого** ([content negotiation](https://en.wikipedia.org/wiki/Content_negotiation)).
* **Клиентские куки**.
* **Улучшенная поддержка сжатия**.

1. **Отличия HTTP 2.0 от HTTP 1.1.**

HTTP/2 разрабатывался для транспортировки контента с низким временем задержки. Главные отличия от HTTP/1.1:

* бинарный вместо текстового
* мультиплексирование — передача нескольких асинхронных HTTP-запросов по одному TCP-соединению
* сжатие заголовков методом HPACK
* Server Push — несколько ответов на один запрос
* приоритизация запросов
* безопасность