Networks

- OSI
- TCP/IP
- HTTP
- DNS

OSI

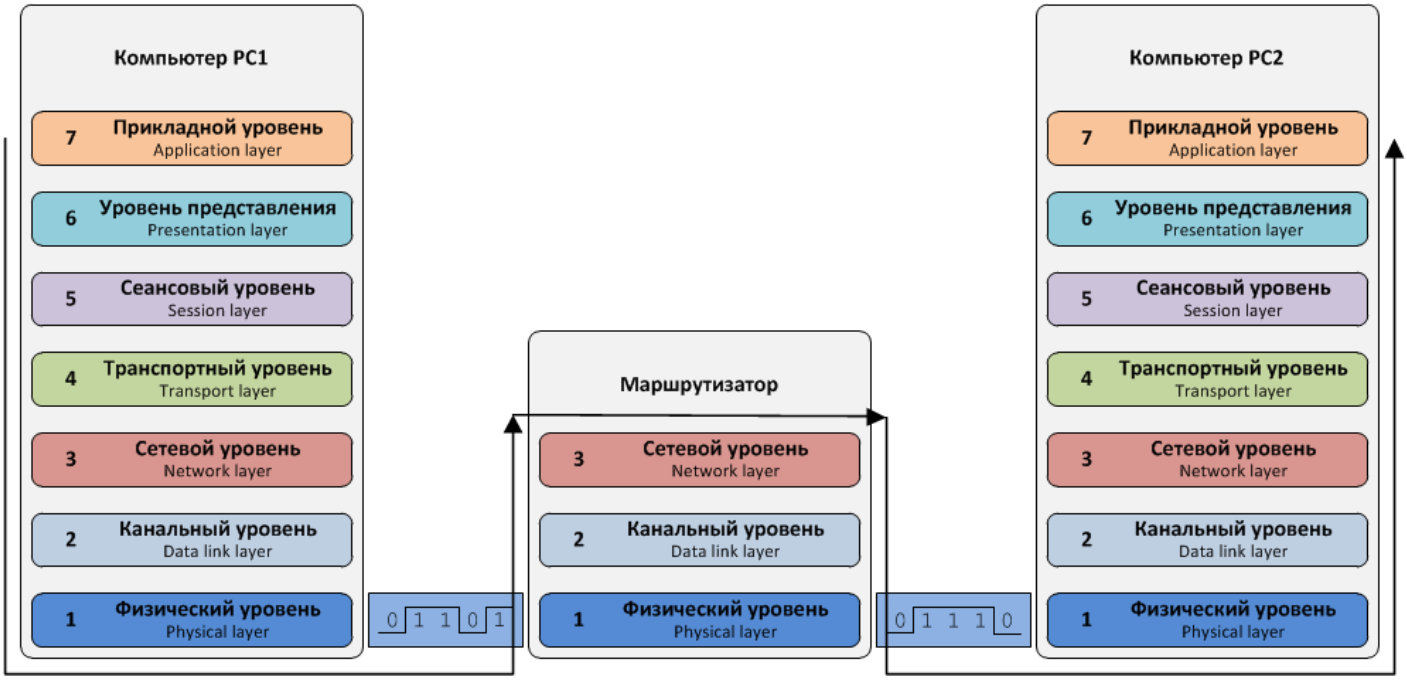
Модель OSI

Модель OSI (Open Systems Interconnection) - это стандартная модель взаимодействия открытых систем, которая определяет способ передачи данных между различными компьютерными устройствами. Эта модель была разработана Международной организацией по стандартизации (ISO) для обеспечения совместимости между различными производителями оборудования и программного обеспечения.

Модель OSI состоит из семи уровней, каждый из которых отвечает за определенный аспект передачи данных. Каждый уровень работает независимо друг от друга, что позволяет разработчикам создавать устройства и программное обеспечение, которые соответствуют только определенным уровням модели OSI.

Каждый уровень модели OSI имеет свою собственную функциональность и задачи:

1. Физический уровень: определяет физические характеристики передачи данных по среде связи, такие как кабели, разъемы, протоколы передачи данных и т.д.
2. Канальный уровень: обеспечивает передачу данных между устройствами в локальной сети и обеспечивает контроль ошибок.
3. Сетевой уровень: определяет адресацию и маршрутизацию данных, управляет трафиком и обеспечивает качество обслуживания.
4. Транспортный уровень: обеспечивает надежную доставку данных между устройствами и управляет потоком данных.
5. Сеансовый уровень: устанавливает и управляет соединением между устройствами и обеспечивает управление сеансами.
6. Уровень представления: преобразует данные в формат, понятный для приложений.
7. Прикладной уровень: предоставляет доступ к приложениям и определяет способы обмена данными между приложениями.



Понимание модели OSI является важным аспектом для разработки и настройки сетевых устройств и приложений. Она позволяет разработчикам создавать устройства и приложения, которые могут взаимодействовать с другими устройствами и приложениями, работающими в различных сетях и на разных уровнях.

Модель OSI является концептуальной моделью, предназначенной для описания и понимания взаимодействия компьютерных систем в сетях. Она предоставляет общую рамку и стандартные протоколы для передачи данных между различными уровнями и устройствами. Однако, в реальности, реализация сетевых технологий может отличаться от модели OSI. Фактическое взаимодействие между компьютерами и сетевыми устройствами может основываться на других сетевых моделях, таких как модель TCP/IP, которая широко используется в Интернете.

Модель OSI представляет собой идеализированную архитектуру, которая помогает разработчикам и инженерам понять принципы работы сетей и различных протоколов. Она также служит основой для разработки стандартов и протоколов, которые обеспечивают совместимость и взаимодействие между различными устройствами и сетями.

Важно понимать, что реальная реализация сетей и протоколов может отличаться от модели OSI. Например, в модели TCP/IP уровни сетевого и канального доступа объединены в один уровень. Кроме того, в различных протоколах и технологиях могут быть добавлены дополнительные уровни или изменены способы взаимодействия между уровнями модели OSI.

Таким образом, модель OSI представляет собой концептуальный фреймворк, который помогает в понимании сетевых технологий, но конкретная реализация может отличаться в зависимости от используемых протоколов и технологий.

Уровни модели OSI

Модель					
Уровень (layer)		**Тип данных (PDU[1])**	**Функции**	**Примеры**	**Оборудование**
Host layers	7. Прикладной (application)	Данные	Доступ к сетевым службам	HTTP, FTP, POP3, SMTP, WebSocket	Хосты (клиенты сети), Межсетевой экран
	6. Представления (presentation)		Представление и шифрование данных	ASCII, EBCDIC, SSL, gzip	
	5. Сеансовый (session)		Управление сеансом связи	RPC, PAP, L2TP, gRPC	
	4. Транспортный (transport)	Сегменты (segment) / Датаграммы (datagram)	Прямая связь между конечными пунктами и надёжность	TCP, UDP, SCTP, Порты	
Media[2] layers	3. Сетевой (network)	Пакеты (packet)	Определение маршрута и логическая адресация	IPv4, IPv6, IPsec, AppleTalk, ICMP	Маршрутизатор, Сетевой шлюз, Межсетевой экран
	2. Канальный (data link)	Биты (bit)/ Кадры (frame)	Физическая адресация	PPP, IEEE 802.22, Ethernet, DSL, ARP, сетевая карта.	Сетевой мост, Коммутатор, точка доступа
	1. Физический (physical)	Биты (bit)	Работа со средой передачи, сигналами и двоичными данными	USB, RJ («витая пара», коаксиальный, оптоволоконный), радиоканал	Концентратор, Повторитель (сетевое оборудование)

TCP/IP

TCP/IP:

В мире современных сетей TCP/IP (Transmission Control Protocol/Internet Protocol) является наиболее широко распространенным набором протоколов для передачи данных. Он обеспечивает основу для связи и обмена информацией в Интернете и других компьютерных сетях. В этой статье мы рассмотрим основные аспекты TCP/IP, его составляющие и принципы работы.

1. История и значение TCP/IP TCP/IP был разработан в 1970-х годах в рамках исследовательского проекта, финансируемого Агентством передовых исследований обороны США (ARPA). Он был создан для обеспечения связи и обмена данными между компьютерами в сети ARPANET, предшественнице Интернета. Поскольку TCP/IP стал успешным стандартом, он был принят в качестве основы для Интернета и получил широкое применение во всем мире.

TCP/IP представляет собой набор протоколов, который обеспечивает точное, надежное и эффективное передачу данных через сети. Он определяет стандарты и правила для установления соединения, сегментации и сборки данных, маршрутизации, адресации и других аспектов сетевой связи.

Распределение протоколов по уровням модели TCP/IP	
Прикладной (Application Layer)	напр., HTTP, RTSP, FTP, DNS
Транспортный (Transport Layer)	напр., TCP, UDP, SCTP, DCCP *(RIP, протоколы маршрутизации, подобные OSPF, что работают поверх IP, являются частью сетевого уровня)*
Сетевой (Межсетевой) (Network Layer)	Для TCP/IP это IP *(вспомогательные протоколы, вроде ICMP и IGMP, работают поверх IP, но тоже относятся к сетевому уровню; протокол ARP является самостоятельным вспомогательным протоколом, работающим поверх канального уровня)*
Уровень сетевого доступа (Канальный) (Link Layer)	Ethernet, IEEE 802.11, WLAN, SLIP, Token Ring, ATM и MPLS, физическая среда и принципы кодирования информации, T1, E1

Основные протоколы

TCP/IP TCP/IP состоит из нескольких протоколов, каждый из которых выполняет определенную функцию в передаче данных. Вот некоторые из основных протоколов TCP/IP:

- IP (Internet Protocol): Он обеспечивает маршрутизацию и доставку пакетов данных между различными сетями. IP определяет уникальные IP-адреса, которые идентифицируют устройства в сети.
- TCP (Transmission Control Protocol): Он обеспечивает надежную и упорядоченную доставку данных между узлами в сети. TCP разбивает данные на сегменты, устанавливает соединение между отправителем и получателем, контролирует поток данных и обеспечивает контроль над ошибками.
- UDP (User Datagram Protocol): В отличие от TCP, UDP обеспечивает ненадежную и негарантированную доставку данных. Он используется для передачи данных, где небольшая задержка более важна, чем гарантированная доставка. UDP широко используется для стриминга мультимедийного контента, онлайн-игр и других приложений, где скорость передачи данных имеет большее значение, чем точность.
- ICMP (Internet Control Message Protocol): Он предоставляет механизм для отправки сообщений об ошибках и управления сетевым оборудованием, таким как маршрутизаторы. ICMP используется, например, для отправки сообщений о недоступности узлов или идентификации сетевых проблем.
- DNS (Domain Name System): DNS преобразует доменные имена, такие как example.com, в соответствующие IP-адреса, позволяя устройствам обращаться друг к другу по удобным для людей именам.

Принципы работы

TCP/IP TCP/IP основан на клиент-серверной архитектуре, где устройства в сети действуют либо в качестве клиентов, инициирующих соединение, либо в качестве серверов, ожидающих соединений и обрабатывающих запросы.

При передаче данных с использованием TCP/IP клиент и сервер взаимодействуют следующим образом:

1. Клиент инициирует соединение с сервером, отправляя запрос на определенный порт сервера.
2. Сервер принимает запрос и устанавливает TCP-соединение с клиентом.
3. Клиент и сервер обмениваются данными через установленное соединение.
4. По завершении передачи данных соединение закрывается.

Протоколы TCP и UDP обеспечивают надежность передачи данных в TCP/IP. TCP использует механизм подтверждений и переотправки пакетов для обеспечения доставки без ошибок и упорядочивания данных. UDP, с другой стороны, не предоставляет подтверждений и переотправки, что делает его быстрее, но менее надежным.

Применение

TCP/IP TCP/IP широко применяется в различных областях, включая:

- Интернет: TCP/IP является основным протоколом для связи и передачи данных в Интернете. Он обеспечивает глобальную связь и позволяет пользователям получать доступ к различным ресурсам и услугам в сети.
- Корпоративные сети: TCP/IP используется для связи компьютеров и устройств внутри организаций. Он позволяет сотрудникам обмениваться данными, обеспечивая эффективное взаимодействие и совместную работу.
- Облачные вычисления: TCP/IP является основой для облачных вычислений, позволяя передавать данные между облачными сервисами, управлять ресурсами и обеспечивать безопасное взаимодействие между облачными и клиентскими устройствами.
- Мобильные сети: TCP/IP используется в мобильных сетях для передачи данных между мобильными устройствами и сетевыми серверами. Он обеспечивает возможность доступа к интернету, обмена сообщениями и передачи мультимедийного контента через мобильные приложения.
- IoT (Internet of Things): TCP/IP играет важную роль в сетях IoT, где различные устройства могут обмениваться данными и взаимодействовать друг с другом. Он позволяет устройствам подключаться к сети, передавать данные и получать команды удаленного управления.

HTTP

HTTP база

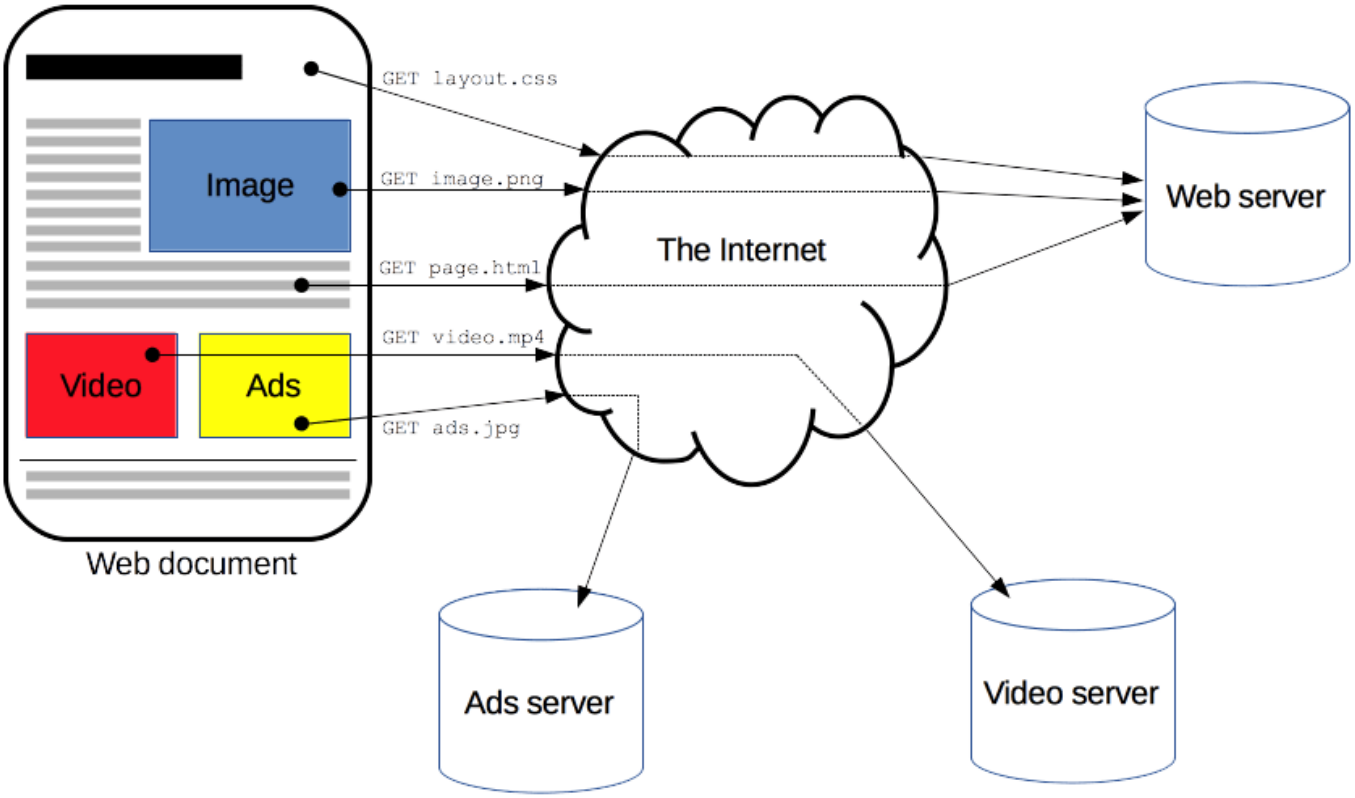
HTTP (Hypertext Transfer Protocol) - это протокол передачи гипертекста, который играет ключевую роль в коммуникации между клиентскими приложениями и веб-серверами. Является протоколом прикладного уровня, который определяет стандарты и правила для передачи данных в виде гипертекстовых документов. Гипертекстовые документы могут содержать ссылки на другие документы, образуя таким образом веб-сайты и страницы, которые мы привыкли видеть в браузерах.

Протокол основан на клиент-серверной архитектуре, где клиент отправляет запросы серверу, а сервер отвечает на эти запросы, предоставляя требуемую информацию или выполняя определенные действия. Запросы и ответы состоят из заголовков и, возможно, тела, которые содержат сами данные.

Принцип работы HTTP

Процесс обмена данными между клиентом и сервером в HTTP выглядит следующим образом:

1. Клиент инициирует соединение с сервером, отправляя HTTP-запрос.
2. Сервер принимает запрос и анализирует его.
3. Сервер генерирует HTTP-ответ, который содержит код состояния, заголовки и, возможно, тело с данными.
4. Ответ отправляется клиенту.
5. Клиент получает ответ и обрабатывает его в соответствии с кодом состояния и данными.



Клиенты и серверы взаимодействуют, обмениваясь одиночными сообщениями (а не потоком данных). Сообщения, отправленные клиентом, обычно веб-браузером, называются *запросами*, а сообщения, отправленные сервером, называются *ответами*.

Коды состояния

Коды состояния представляют собой трехзначные числа, которые указывают на результат обработки запроса сервером. Некоторые распространенные коды состояния включают:

- 200 OK: Запрос успешно выполнен, и сервер возвращает запрашиваемые данные.
- 404 Not Found: Запрашиваемый ресурс не найден на сервере.
- 500 Internal Server Error: Внутренняя ошибка сервера, которая мешает выполнить запрос.

Типы запросов

HTTP определяет несколько типов запросов, которые клиент может отправить серверу для выполнения определенных действий или получения данных. Некоторые из основных типов запросов включают:

- GET: Используется для получения данных с сервера. Клиент отправляет запрос на определенный ресурс и получает данные, связанные с этим ресурсом в ответе сервера.
- POST: Используется для отправки данных на сервер для обработки или создания нового ресурса. Клиент отправляет данные в теле запроса, и сервер обрабатывает эти данные соответствующим образом.

- PUT: Используется для обновления данных на сервере. Клиент отправляет данные, указывая идентификатор ресурса, который требуется обновить, и сервер обновляет соответствующий ресурс.
- DELETE: Используется для удаления ресурса на сервере. Клиент отправляет запрос на удаление указанного ресурса, и сервер удаляет его, если это возможно.
- PATCH: Используется для частичного обновления ресурса на сервере. Клиент отправляет только измененные или обновленные данные, и сервер применяет эти изменения к ресурсу.
- OPTIONS: Используется для получения информации о возможностях сервера и поддерживаемых методах запросов.

Каждый тип запроса имеет свою уникальную функцию и используется в различных сценариях. Они позволяют клиентам взаимодействовать с серверами и выполнять различные операции, такие как получение данных, отправка данных, обновление ресурсов и т. д.

Более показательные примеры:

GET запрос:

```
ET /search?query=example&page=1 HTTP/1.1
ost: example.com
```

римере, мы отправляем GET запрос на `example.com/search` с параметрами `query=example` и `e=1`. Весь запрос, включая параметры, передается через URL.

имер ответа сервера на GET запрос (HTTP 1.1 200 OK):

```
TTP/1.1 200 OK
ate: Mon, 20 May 2023 10:30:00 GMT
erver: Apache/2.4.7 (Ubuntu)
ontent-Type: text/html; charset=utf-8
ontent-Length: 342

 DOCTYPE html>
tml>
ead>
 <title>Пример страницы</title>
/head>
ody>
 <h1>Привет, мир!</h1>
/body>
/html>
```

POST запрос:

```
OST /submit HTTP/1.1
ost: example.com
ontent-Type: application/x-www-form-urlencoded
ontent-Length: 25

sername=example&password=123
```

римере, мы отправляем POST запрос на `example.com/submit`. Данные запроса, такие как `rname` и `password`, передаются в теле запроса. Заголовки `Content-Type` и `Content-Length` зывают на тип данных и длину тела запроса.

имер ответа сервера на POST запрос (HTTP 1.1 201 Created):

```
TTP/1.1 201 Created
ate: Mon, 20 May 2023 10:45:00 GMT
erver: Apache/2.4.7 (Ubuntu)
ontent-Type: application/json
ontent-Length: 54

'message": "Данные успешно созданы", "id": 12345}
```

боих примерах, первая строка представляет собой статус-линию ответа, которая ержит версию протокола (HTTP/1.1), код состояния (200 или 201) и текстовое описание тояния (OK или Created). Затем следуют заголовки, которые предоставляют ополнительную информацию о содержимом и свойствах ответа. Каждый заголовок едставлен в формате "Имя: Значение".

сле заголовков идет пустая строка, а затем следует тело ответа, которое содержит ктическое содержимое ответа сервера. Тело ответа может быть представлено в личных форматах, таких как HTML, JSON, XML и других, в зависимости от типа контента, занного в заголовке "Content-Type".

ды состояния HTTP указывают на успешность или ошибку выполнения запроса, а оловки и тело ответа предоставляют дополнительную информацию и данные, вращаемые сервером в ответ на запрос клиента.

новное отличие между GET и POST методами заключается в том, что GET используется я получения данных с сервера через URL, в то время как POST используется для отправки ных в теле запроса. GET запросы могут быть закладками или расшаренными ссылками, как параметры видны в URL, в то время как POST запросы обычно используются для правки конфиденциальной или чувствительной информации, такой как пароли или ные кредитных карт.

DNS

DNS (Domain Name System) - это распределенная система, которая переводит доменные имена (например, example.com) в соответствующие им IP-адреса компьютеров и устройств, а также выполняет обратное преобразование (перевод IP-адресов в доменные имена). DNS является фундаментальным протоколом в Интернете, используемым для разрешения имен в сети.



DNS использует два основных протокола и порты для своей работы:

1. Протокол DNS (Domain Name System): DNS использует собственный протокол DNS для обмена информацией между клиентскими устройствами и серверами DNS. Протокол DNS работает на прикладном уровне OSI (уровень 7).
2. Протоколы транспортного уровня:
   - TCP (Transmission Control Protocol): TCP используется для передачи данных DNS в случае больших объемов или сложных запросов, которые не могут быть переданы через одиночный пакет UDP.
   - UDP (User Datagram Protocol): UDP используется для основной передачи данных DNS. Он предоставляет ненадежный и безсоединительный транспорт данных, что делает его быстрее и более эффективным для большинства DNS-запросов.

Порты, используемые протоколами DNS:

- Порт 53: Это стандартный порт для DNS-серверов. Он используется как для TCP, так и для UDP протоколов. Клиентские устройства отправляют запросы DNS на порт 53 сервера DNS, а серверы DNS отправляют ответы на этот же порт клиентского устройства.

DNS находится на прикладном уровне OSI (уровень 7). Он предоставляет службу разрешения имен, которая облегчает использование доменных имен вместо IP-адресов. DNS работает

поверх протоколов транспортного уровня (TCP и UDP) и использует протокол DNS для обмена информацией. На прикладном уровне, DNS запросы и ответы передаются через приложения (браузеры, почтовые клиенты и т.д.), чтобы перевести доменные имена в соответствующие IP-адреса и наоборот.

DNS находится на прикладном уровне OSI, потому что его функциональность относится к обработке запросов и предоставлению служб разрешения имен, в то время как протоколы транспортного уровня обеспечивают надежную и эффективную передачу данных между клиентом и сервером DNS.

Это база:

- Распределенная структура: DNS работает на основе распределенной базы данных, называемой DNS-кэшами или серверами имен. Они разделены по всему миру и содержат информацию о доменных именах и соответствующих IP-адресах.
- Резолверы DNS: Клиентские устройства, такие как компьютеры или мобильные устройства, используют резолверы DNS для запроса соответствующего IP-адреса доменного имени. Резолверы могут быть настроены вручную или автоматически получать информацию от DNS-серверов, предоставляемых интернет-провайдерами.
- DNS-записи: DNS-записи содержат информацию о доменных именах и их ассоциированных IP-адресах. Некоторые распространенные типы DNS-записей включают записи типа A (преобразование доменного имени в IPv4-адрес), AAAA (преобразование доменного имени в IPv6-адрес), CNAME (псевдонимы для других доменных имен), MX (записи почтовых серверов) и NS (серверы имен для домена).
- Процесс разрешения DNS: Когда пользователь вводит доменное имя в веб-браузере или приложении, резолвер DNS инициирует процесс разрешения DNS. Резолвер отправляет запрос DNS на ближайший DNS-сервер, который затем перенаправляет запрос по иерархии DNS-серверов до тех пор, пока не будет найден соответствующий IP-адрес. Затем IP-адрес возвращается обратно резолверу, который затем использует его для установления соединения с запрашиваемым сервером.
- DNS-кэширование: Для улучшения производительности и снижения нагрузки на DNS-серверы, резолверы DNS могут временно сохранять полученные записи в своем кэше. При следующем запросе на тот же домен или на другой домен, резолвер DNS сначала проверяет свой локальный кэш на наличие соответствующей записи. Если запись найдена, резолвер использует сохраненный IP-адрес без необходимости отправлять новый запрос DNS. Это позволяет ускорить процесс разрешения и снизить нагрузку на DNS-серверы.
- Однако, если записи не найдены в локальном кэше резолвера или если записи в кэше устарели, резолвер отправляет запрос на DNS-серверы, начиная с корневых серверов. Запросы передаются по иерархии DNS-серверов, начиная с корневых серверов, затем серверов верхнего уровня, серверов авторитетных зон и наконец, до серверов, которые содержат требуемую запись. Когда найден соответствующий IP-адрес, он возвращается резолверу, который затем сохраняет запись в свой кэш и использует ее для будущих запросов.

- DNS также играет роль в других функциях, таких как обратное преобразование IP-адресов в доменные имена (резолюция обратных записей), маршрутизация почты с помощью записей MX, обнаружение служб с помощью записей SRV и другие.

  DNS является неотъемлемой частью функционирования Интернета, обеспечивая преобразование доменных имен в IP-адреса и позволяя пользователям получать доступ к веб-сайтам, отправлять электронную почту и выполнять другие сетевые операции с использованием удобных доменных имен вместо запоминания числовых IP-адресов

Запросы:

DNS-запросы могут быть классифицированы как рекурсивные и итеративные (нерекурсивные).

1. Рекурсивный DNS-запрос: В рекурсивном запросе резолвер полностью обрабатывает запрос от начала до конца. Резолвер отправляет запрос на DNS-сервер и ожидает полного ответа. Если DNS-сервер не имеет информации о запрашиваемом домене в своем кэше, он может перенаправить запрос на другой сервер и продолжать делать это, пока не будет найден и вернутся полные данные о домене. Затем резолвер возвращает полный ответ клиенту, который инициировал запрос. Рекурсивные запросы обычно используются клиентскими устройствами, чтобы получить полные данные о домене и упростить процесс для конечного пользователя.
2. Итеративный (нерекурсивный) DNS-запрос: В итеративном запросе резолвер отправляет запрос на DNS-сервер и ожидает ответа с максимально доступной информацией о запрашиваемом домене. Если DNS-сервер не имеет информации о домене, он возвращает резолверу наиболее близкий результат или информацию о сервере, который может предоставить дальнейшую информацию. Затем резолвер должен отправить дополнительные запросы на другие серверы для получения полного ответа. Итеративные запросы требуют более активного участия резолвера в процессе разрешения и могут быть использованы, например, серверами DNS для получения информации о других серверах, необходимых для обработки запросов.
3. Запросы типа Forward (Прямой запрос): Это обычные DNS-запросы, в которых клиентская машина или резолвер обращается к DNS-серверу для получения информации о домене или ресурсе.
4. Запросы типа Reverse (Обратный запрос): В обратных DNS-запросах клиентская машина или резолвер отправляет запрос на DNS-сервер, чтобы получить доменное имя, соответствующее определенному IP-адресу. Обратные запросы часто используются для идентификации домена, связанного с IP-адресом, например, при отображении имени хоста при сканировании сети или анализе журналов.

Путь DNS запроса:

Полная цепочка DNS-запроса может выглядеть следующим образом:

1. Браузер отправляет запрос на DNS-резолвер. Если запись о доменном имени не найдена в локальном DNS-кэше браузера, то он проверяет кэш операционной системы, файл hosts и кэш роутера, если они есть.
2. Если запись не найдена в локальных кэшах и файлах, то резолвер отправляет запрос на первый DNS-сервер в списке, настроенном в операционной системе или роутере.
3. Первый DNS-сервер, получив запрос, проверяет свой кэш. Если запись найдена, то сервер возвращает IP-адрес в DNS-резолвер, который передает его браузеру.
4. Если запись не найдена в кэше первого DNS-сервера, то он перенаправляет запрос на следующий DNS-сервер в цепочке. Обычно это будет DNS-сервер на более высоком уровне иерархии DNS.
5. Процесс перенаправления продолжается, пока DNS-сервер не найдет запись в своем кэше или не получит ее от другого DNS-сервера выше в иерархии.
6. Когда DNS-сервер находит запись о доменном имени, он возвращает ее IP-адрес в DNS-резолвер, который передает его браузеру.
7. Браузер использует полученный IP-адрес для установки соединения с веб-сервером, который хостит веб-страницу, соответствующую доменному имени.
8. Если в процессе запроса DNS-записи была найдена новая запись, она может быть кэширована в локальном DNS-кэше браузера, операционной системы, файле hosts, кэше роутера и на других DNS-серверах на пути запроса, для более быстрого доступа в будущем.

DNS записи:

В системе DNS существует несколько типов записей, которые используются для хранения информации о доменных именах и их связи с IP-адресами и другими данными. Вот некоторые из основных типов записей DNS:

1. Запись A (Address): Используется для связи доменного имени с IPv4-адресом.
2. Запись AAAA (IPv6 Address): Аналогично записи A, но используется для связи доменного имени с IPv6-адресом.
3. Запись CNAME (Canonical Name): Устанавливает псевдоним для указанного доменного имени, позволяя одному домену быть псевдонимом для другого.
4. Запись MX (Mail Exchange): Указывает почтовый сервер, отвечающий за доставку электронной почты для домена.
5. Запись NS (Name Server): Определяет DNS-серверы, которые являются авторитетными для домена и отвечают за предоставление информации о домене.
6. Запись PTR (Pointer): Используется для обратного преобразования IP-адреса в доменное имя.
7. Запись TXT (Text): Позволяет хранить произвольные текстовые данные для доменного имени, часто используется для добавления дополнительной информации, например, SPF-записей для авторизации отправителей электронной почты.
8. Запись SRV (Service): Используется для обнаружения служб, предоставляемых на указанном домене или порту, и содержит информацию о доменном имени, порте, протоколе и приоритете.

9. Запись SOA (Start of Authority): Содержит информацию об авторитетном сервере для домена, включая контактные данные, периоды обновления и другие настройки.
10. Запись SPF (Sender Policy Framework): Определяет список IP-адресов или доменных имен, которые авторизованы для отправки электронной почты от имени домена.