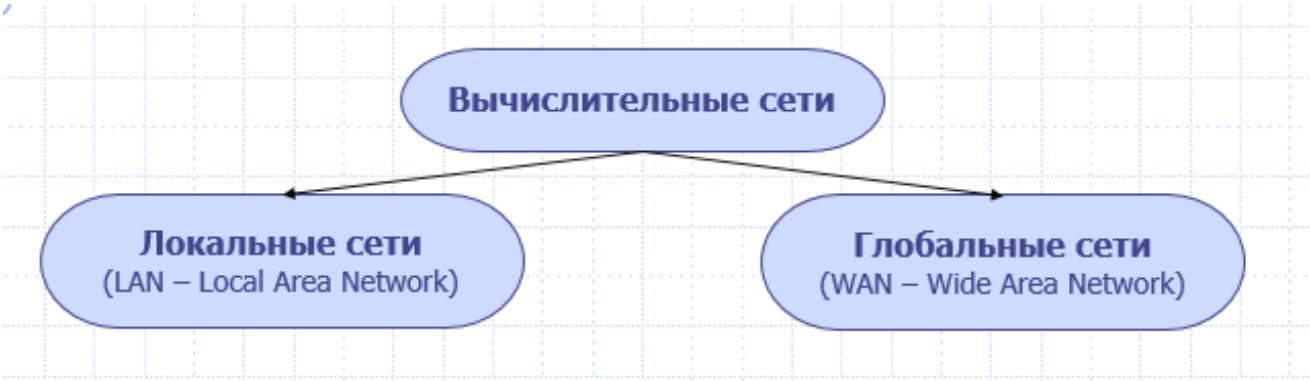
1. Понятие вычислительной сети. Классификация сетей ЭВМ. Локальные и глобальные вычислительные сети. Понятия трафика и пропускной способности. Понятие сетевого ресурса, клиента, сервера.

Вычислительная (компьютерная) сеть – совокупность вычислительных устройств, соединенных с помощью каналов связи и средств коммутации в единую систему для обмена сообщениями и совместного решения общей задачи (доступа пользователей к программным, техническим, информационным и организационным ресурсам сети).

Классификация сетей ЭВМ:

- Персональные сети (PAN – Personal Area Network) – Bluetooth
- Локальные вычислительные сети (LAN – Local Area Network) – Ethernet, Wi-Fi;
- Муниципальные (MAN – Metropolitan Area Network) – Wi-MAX;
- Глобальные (WAN – Wide Area Network) – Internet.



	Количество узлов	Принцип адресации	Адрес компьютера
Локальная	Ограничено (несколько сотен)	Физическая, адрес компьютера – адрес сетевого адаптера	Адрес компьютера уже имеется, компьютер может узнать своих соседей путем опроса
Глобальная	Не ограничено (миллионы), географически разнесены	Логическая адресация (IP-адрес)	Нельзя узнать, кто есть в сети

Сервер – узел сети, который выполняет запросы клиентов и обеспечивает им совместный доступ к некоторому ресурсу. Ресурсом может быть файловое хранилище, база данных, устройство печати, веб сайт, вычислительные мощности и др.

Клиент – узел сети, обращающийся к ресурсам удалённого узла-сервера с помощью сообщений-запросов. Узел может одновременно выступать в роли клиента и сервера.

Сетевой ресурс – устройство или часть информации, к которой может быть осуществлен удалённый доступ с другого компьютера (Принтер, модем, дисковая память, процессор).

Трафик – количество сообщений, передаваемых в сети (во всей сети) за определённый период времени.

Пропускная способность – предельное количество сообщений, которые могут передаваться в единицу времени между двумя узлами.

2. Логическая структура вычислительных сетей. Концепция и основные понятия эталонной модели взаимодействия открытых систем (ISO/OSI). Функции отдельных уровней OSI.

Логическая топология (структура) сети – определяет реальные пути движения сигналов при передаче данных по используемой физической топологии. То есть физическая структура — это как мы расположили устройства, а логическая — это через какие устройства будут проходить пакеты.

Эталонная модель взаимодействия открытых систем – обеспечивает взаимодействие различных сетевых устройств друг с другом. Модель определяет различные уровни взаимодействия систем. Каждый уровень выполняет определенные функции при таком взаимодействии.

Функции уровней:

1. **Физический уровень**: определяет метод передачи данных, какая среда используется (передача электрических сигналов, световых импульсов или радиоэфир), уровень напряжения, метод кодирования двоичных сигналов. (кабель)
2. **Канальный уровень**: определяет логическую топологию сети, правила получения доступа к среде передачи данных, решает вопросы, связанные с адресацией физических устройств в рамках логической сети и управлением передачей информации между сетевыми устройствами. (Сетевая плата)
3. **Сетевой уровень:** отвечает за доставку сообщений между узлами разных логических сетей, а также за логическую адресацию сетевых устройств и маршрутизацию сообщений. (IPv4, IPv6)
4. **Транспортный уровень:** отвечает за надежную передачу пакетов или потока данных между узлами сети. Он отвечает за разбиения блока (потока) данных на пакеты, передачу их по сети и сборку этих пакетов в правильном порядке. Также он отвечает за надежность доставки и должен обеспечивать повторную передачу при сбое. Появляется понятие портов – для идентификации программ. (TCP, UDP)
5. **Сеансовый уровень:** способствует взаимодействию между устройствами, запрашивающими и поставляющими услуги. Сеансы связи контролируются посредством механизмов, которые устанавливают, поддерживают, синхронизируют и управляют диалогом между поддерживающими связь объектами. Этот уровень также помогает верхним уровням идентифицировать доступный сетевой сервис и соединиться с ним. (Sockets)
6. **Уровень представления:** Основная задача уровня представления данных — преобразование данных во взаимно согласованные форматы (синтаксис обмена), понятные всем сетевым приложениям и компьютерам, на которых работают приложения. На этом уровне также решаются задачи компрессии и декомпрессии данных и их шифрование. (SSL, MIME)
7. **Прикладной уровень:** обеспечивает протоколы, необходимые для выполнения конкретных функций сетевой службы. (HTTP, FTP, SMTP, POP3)

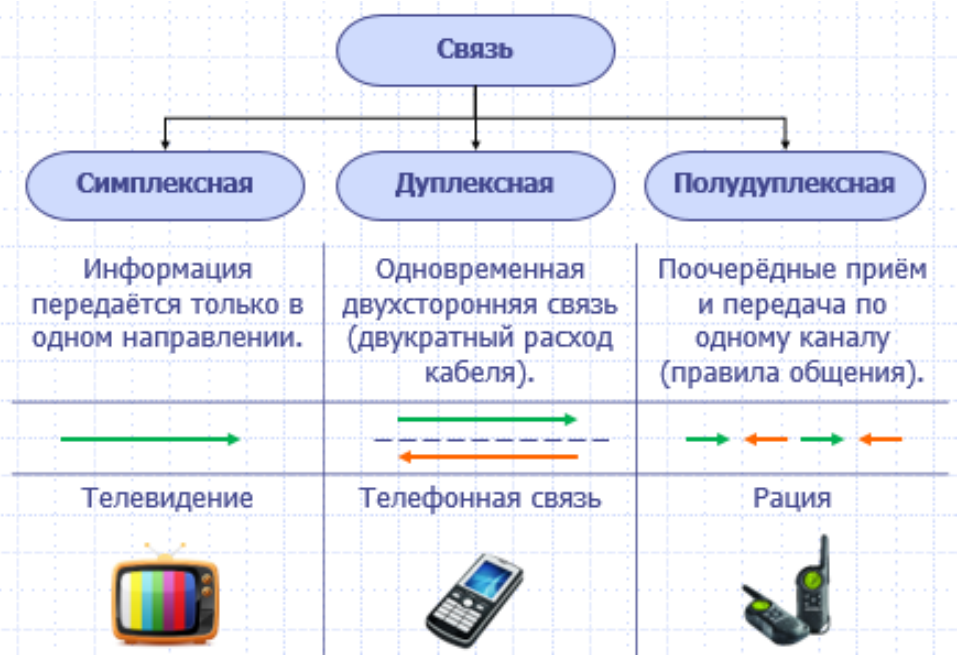
Протокол – правила и стандарты взаимодействия между узлами на одном уровне.

Интерфейс – правила и стандарты взаимодействия между уровнями на одном узле.

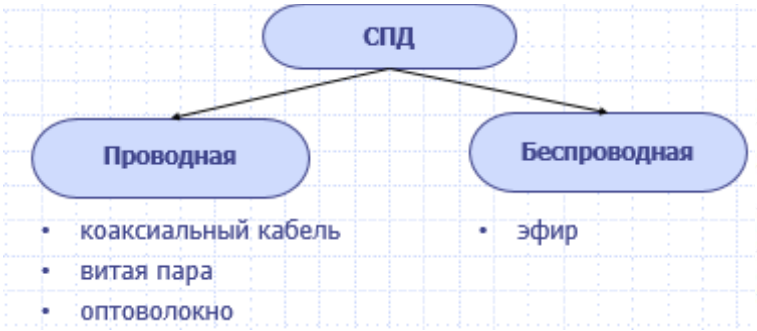
Весь путь должен проходить строго с верхнего на нижний и с нижнего на верхний. Такие процессы получили название **инкапсуляция** (с верхнего на нижний) и **деинкапсуляция** (с нижнего на верхний).

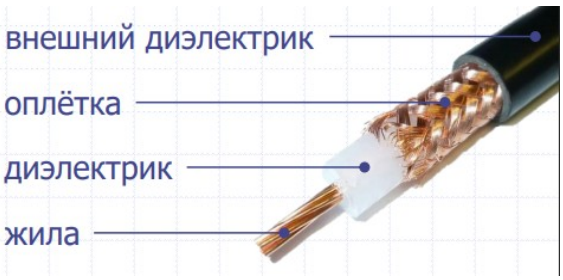
3. Физический уровень OSI. Задачи и функции физического уровня OSI. Среда передачи данных (СПД), ее виды и характеристики (витая пара, коаксиальный кабель, оптоволокно, эфир).

- Отвечает за создание физической связи между двумя узлами (кабели, разъёмы, сигналы);
- Стандартизируются параметры сигналов, способы кодирования сигналов, виды сигналов;
- Передача битов.



СПД – физическая среда, по которой распространяется сигнал, использующийся для передачи данных.

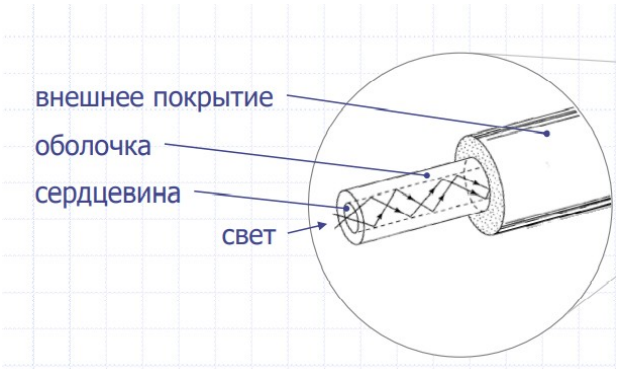




Коаксиальный кабель (RG-58)

- Сигнал передается по центральной жиле
- Оплётка – экран для защиты сигнала от наводок
- Эффективная длина: тонкий (до 25м), толстый (до 500м);
- Максимальная скорость – 10 Мбит/с

Витая пара – это вид кабеля связи, который представляет собой одну или несколько пар изолированных проводников, скрученных между собой. Имеет минимум 4 жилы (для дуплексной связи). Скручивание позволяет устранить перекрёстные помехи между парами.



Оптоволокно

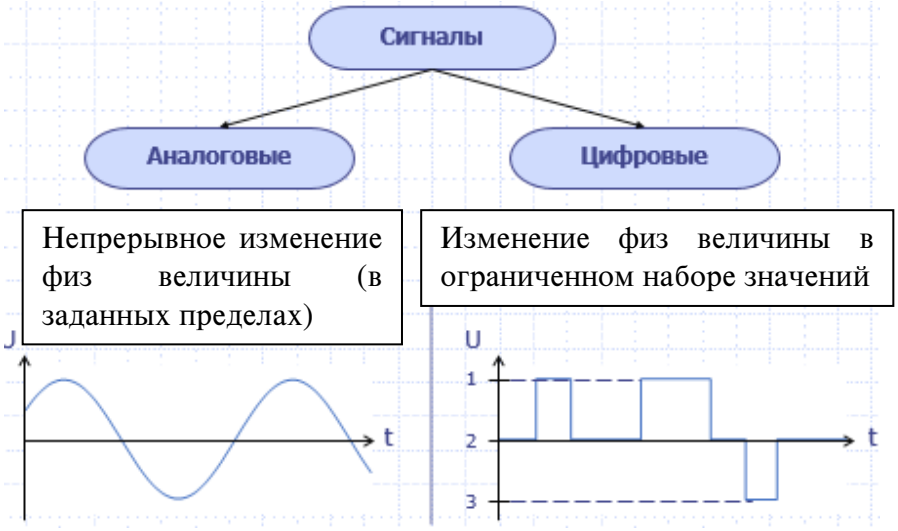
- Носитель сигнала – пучок света;
- Сердцевина с высоким коэффициентом преломления, оболочка с низким коэффициентом преломления;
- Высокая помехозащищенность и секретность;
- Высокая скорость передачи данных, до сотен Гбит/с.

Эфир

Носитель сигнала – радиоволны в заданном диапазоне частот. Чем выше частота, тем выше скорость передачи, но уменьшается дальность распространения сигнала и проникающая способность.

Bluetooth – 2,4 ГГц до 2Мбит/с; Wi-Fi – 2,4 ГГц до 300Мбит/с; 5 ГГц выше 1Гбит/с.

4. Передача данных с использованием различных видов кодирования сигналов. Аналоговые и цифровые сигналы. Методы кодирования аналоговых и цифровых сигналов.



Кодирование аналоговых сигналов

Модуляция – изменение какой-либо физической величины по определённому закону.

Демодуляция – восстановление исходного сигнала.

1) Амплитудная модуляция (АМ) – изменение амплитуды несущего сигнала.

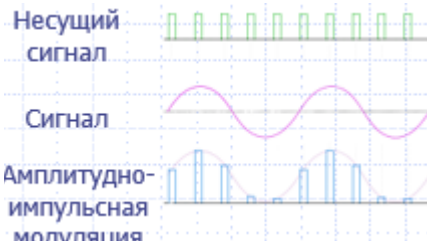
Достоинства: узкая ширина спектра; простота кодирования.

Недостатки: требуется высокая мощность передатчика; низкая помехоустойчивость.

2) Частотная модуляция (ЧМ) – изменение частоты несущего сигнала при той же амплитуде.

Достоинства: высокая помехозащищенность; эффективное использование мощности передатчика; сравнительная простота получения.

Недостатки: широкий спектр сигнала.



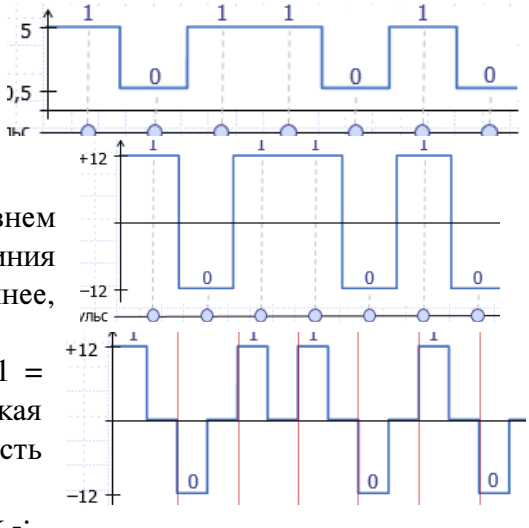
3) Импульсная модуляция (ИМ) – изменение амплитуды, частоты, фазы периодических импульсов.

Достоинства: многоканальность, высокая помехоустойчивость.

Недостатки: сложность аппаратуры.

Кодирование цифровых сигналов

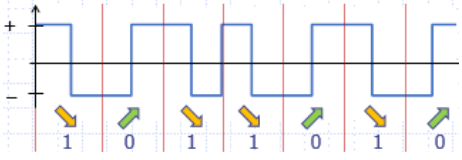
Transistor-Transistor Logic (TTL) – кодирование сигнала уровнем напряжения:1 = 4,5 – 5,0 В 0 = 0,2 – 0,5 В. Тактовый генератор + линия связи для синхроимпульса. Сигнал быстро затухает уже на расстоянии 5-10 м. Применяется в микросхемах.



Non-Return to Zero (NRZ) – кодирование уровнем напряжения:1 = +12 В 0 = -12 В. Тактовый генератор + линия связи для синхроимпульса. Сигнал затухает медленнее, устойчив к помехам.

Return to Zero (RZ) – кодирование уровнем напряжения:1 = +12 В 0 = -12 В. Самосинхронизирующийся. Высокая помехозащищенность. Выше частота – выше скорость передачи.

Манчестерский код – кодирование методом перехода от + к -: 1 = + к - 0 = - к +. Самосинхронизирующийся. Высокая помехозащищенность. Используется в локальных сетях (технология Ethernet).



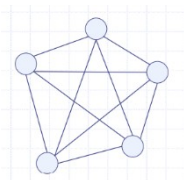
5. Разновидности физических сетевых топологий. Сравнительный анализ топологий "шина", "звезда", "кольцо".

Топология – граф связей между узлами сети.

1. Полно-связные;
2. Неполно-связные – шина, звезда, кольцо, гибридная.

Полно-связные

Каждый узел связан со всеми остальными узлами сети. На практике не используется.



Достоинства:

- Высокое быстродействие;
- Нет конфликтов при передаче данных;
- Надёжность.

Недостатки:

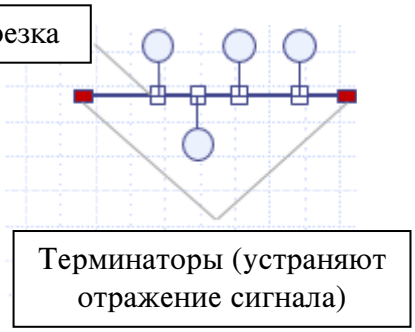
- Число узлов в сети ограничено количеством доступных портов на каждом узле;
- Высокий расход кабеля;
- Высокая стоимость.

Шина

Все узлы подключены к общему каналу передачи данных (коаксиальный кабель). Полудуплексная связь.

Достоинства: простота широковещательной передачи; простота подключения (при наличии врезки).

Недостатки: относительно низкая скорость, подключение без врезки = остановка сети; разрыв кабеля – X.

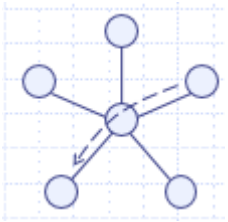


Звезда

Сигнал передается между узлами через один центральный узел. Полудуплексная или дуплексная связь. В качестве центрального узла может выступать сервер или специальное устройство – концентратор (Hub).

Достоинства: разрыв кабеля не выводит из строя всю сеть.

Недостатки: надежность зависит от центрального узла; высокий расход кабеля.



Кольцо

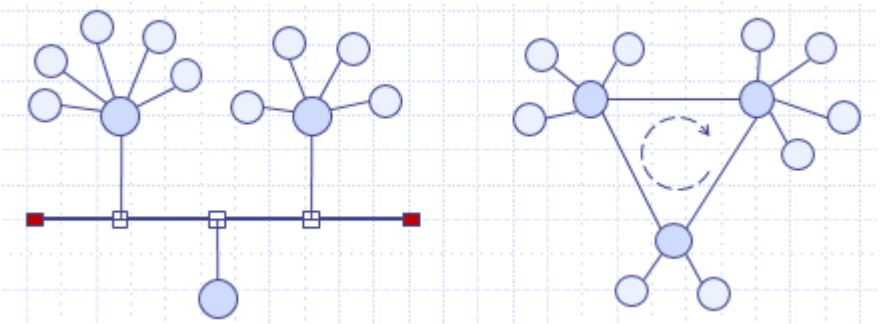
Сигнал передается в одном направлении между узлами покругу. Симплексная связь между узлами и дуплексная связь по всей сети.

Достоинства: низкий расход кабеля; возможность передачи данных в разных сегментах сети.

Недостатки: низкая надежность; подключение узла = остановка сети; разрыв кабеля – X.

Гибридная топология

Комбинация шины, звезды и кольца. Применяется для связывания сетей разных топологий. Часто используется кольцо из звезд.



6. Канальный уровень OSI. Методы доступа к среде передачи данных CSMA/CD и CSMA/CA. Диаграмма перехода между состояниями.

- Физическая адресация устройств (MAC адреса)
- Логическая топология сети (граф вычислительной сети, в котором ребра соответствуют путям передачи данных между узлами (без учёта физических связей)).
- Правила доступа и совместного использования СПД несколькими парами абонентов
- Передача пакетов (кадров) данных (англ. frame)
- Проверка ошибок физического уровня

MAC-адрес – физический адрес устройства; уникальный в пределах локальной сети, зашитый производителем или заданный администратором сети. Структура кадра данных:

Преамбула	MAC-адрес получателя	MAC-адрес отправителя	Тип / Размер	Содержимое	Контрольная сумма
8 октетов	6 октетов	6 октетов	2 октета	46 – 1500 октетов	4 октета

Сетевая плата читает кадры с соотв. MAC-адресом получателя. После прочтения пересчитывает контрольную сумму и если значение совпадает то пакет принимается, если нет – отбрасывается.

Коллизия – наложение нескольких сигналов в общей СПД при попытке нескольких узлов осуществить передачу одновременно.

Домен коллизий – сегмент сети с общей СПД, в котором одновременная передача данных несколькими узлами приводит к коллизии.

CSMA/CD

Узел прослушивает СПД на предмет занятости. При отсутствии сигнала данных (СПД свободна) узел выполняет передачу данных.

Во время передачи кадра узел продолжает слушать СПД и если обнаруживает коллизию (слышит не то, что пишет), то останавливает передачу, посылает всем пакет затора (jam signal) и ждёт в течение случайного промежутка времени (интервал отсрочки), находимого с помощью специального алгоритма, после чего повторяет передачу кадра.

После 16 неудавшихся попыток отправки кадра он отбрасывается – ошибка передачи.

Используется проводной технологией Ethernet (IEEE 802.3).

CSMA/CA

Узел прослушивает СПД. Когда среда освобождается, узел отсчитывает случайный интервал времени, и если среда всё ещё свободна – передаёт широковещательный служебный кадр RTS (Request to Send).

Узел назначения, получив RTS-кадр, вырабатывает кадр занятости среды CTS (Clear To Send) с указанием, как долго среда будет занята.

Передающий узел, получив CTS, начинает передачу данных. Все остальные узлы, получив CTS, ждут указанное время.

После приёма данных узел назначения отвечает квитанцией о доставке – кадром ACK. Если передающий узел не получил ACK в течение определенного времени, он фиксирует факт коллизии, ждёт в течение случайного промежутка времени (интервал отсрочки) и процесс передачи повторяется.

Используется беспроводными технологиями Wi-Fi (IEEE 802.11).

7. Канальный уровень OSI. Методы доступа с передачей маркера: шина с передачей маркера (показать диаграмму перехода между состояниями), логическое кольцо на физической звезде, физическое кольцо с передачей маркера.

Метод с передачей маркера в топологии шина.

Логическое «кольцо», физическое «шина». Право передачи переходит от узла к узлу по кольцу – передача маркера. Каждый узел кроме собственного адреса (MyID) хранит адрес след. узла (NextID), которому передает маркер. Если у узла с маркером имеется пакет для передачи, то он передает пакет, а затем и маркер, целевому узлу. Если пакета для передачи нет, то маркер передается узлу с адресом NextID.

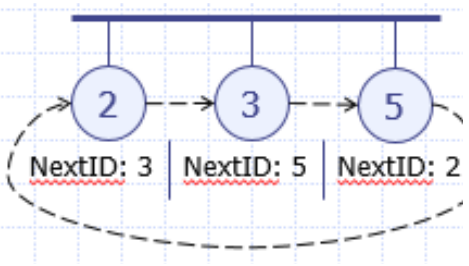
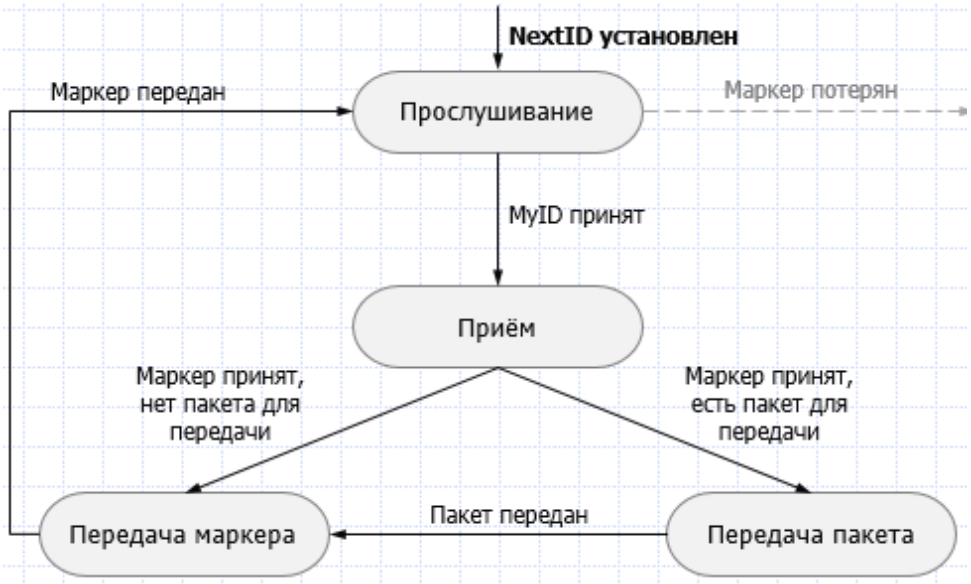
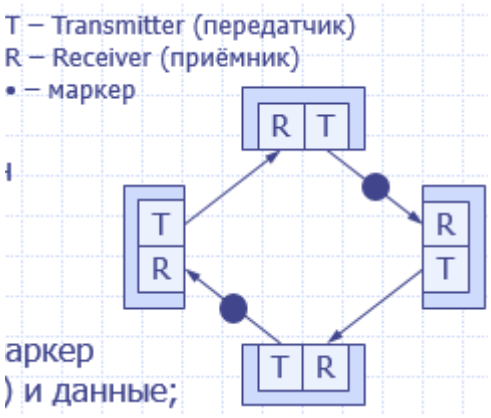


Диаграмма перехода между состояниями



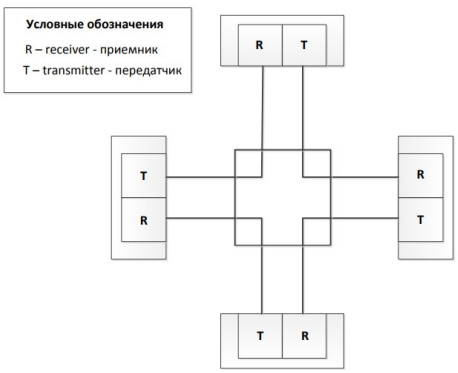
Метод с передачей маркера в топологии кольцо.

Логическое «кольцо», физическое «кольцо». В кольце циркулирует один или несколько маркеров. Если необходимо передать данные, то узел дожидается прихода маркера и передает маркер с ID получателя в заголовке и данные. Когда маркер и данные доходят до узла-получателя, тот их принимает, а маркер и данные передает дальше по кольцу. Когда маркер с данными возвращается к узлу-отправителю, проверяется, совпадает ли пришедший пакет с переданным (корректно ли переданы данные).



Логическое кольцо на физической звезде.

При работе с такой топологией в сети может существовать несколько маркеров одновременно. По сути, маркер постоянно циркулирует по сети, и пропускная способность мало зависит от трафика, следовательно, она максимальна. Скорость передачи маркера очень высокая.



8. Сетевой уровень OSI. Соединение N сетей с помощью (N–1) мостов. Основы маршрутизации.

- Доставка сообщений в глобальной сети (между узлами разных локальных сетей);
- Логическая адресация узлов сети;
- Маршрутизация сообщений;
- Передача пакетов данных с заданными параметрами качества обслуживания;
- Обеспечивается протоколом интернета IP.

Мост – устройство, объединяющие две подсети в единую сеть (мост сетевого уровня пересылает IP-пакеты.

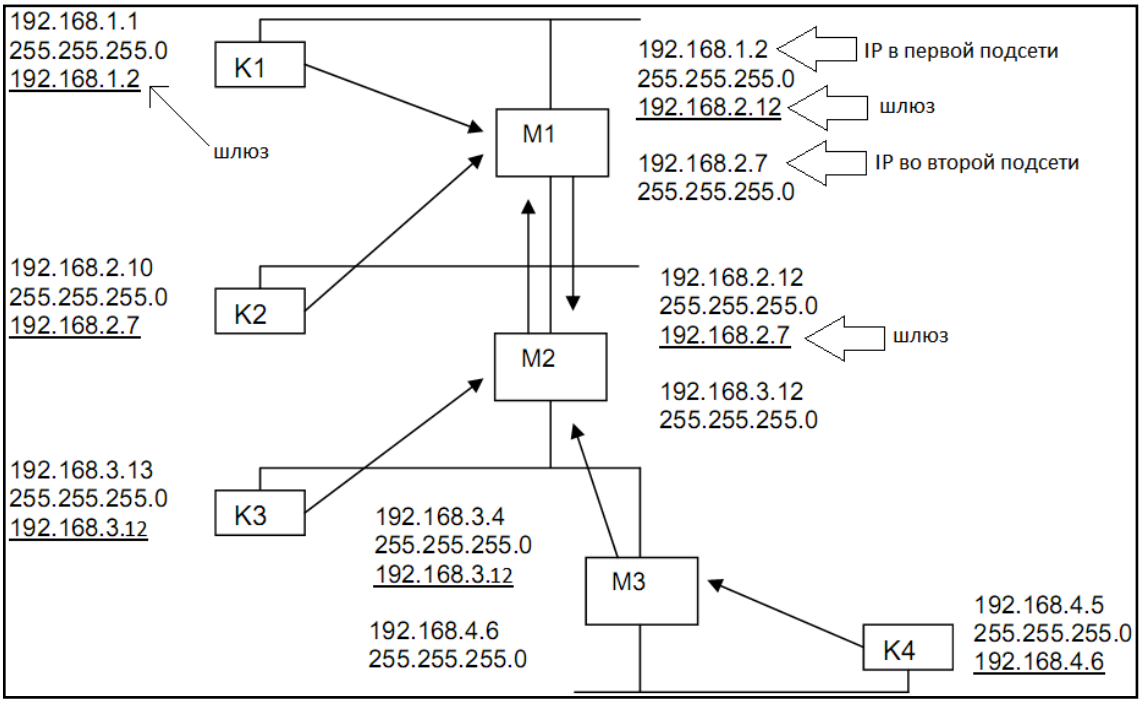
From: 192.168.1.1

To: 192.168.3.13

Отправителем будет наложена маска по операции AND. Получим:

192.168.1.1 AND 255.255.255.0 = 192.168.1.0 | 192.168.3.13 AND 255.255.255.0 = 192.168.3.0

Результаты не равны, поэтому пакеты направляется стандартному шлюзу, т.е. будет получен MAC-адрес шлюза по его IP (по средства ARP) и пакет передан по этому MAC-адресу (хотя в заголовке останутся старые IP). Мост (являющийся шлюзом) сравнит IP получателя со своим IP: 192.168.3.13 ! = 192.168.1.2



Мост видит, что пакет не ему, тогда он наложит маску на IP получателя и на оба своих IP:

192.168.3.13 AND 255.255.255.0 = 192.168.3.0

192.168.1.2 AND 255.255.255.0 = 192.168.1.0

192.168.2.7 AND 255.255.255.0 = 192.168.2.0

IP целевой подсети не равен IP ни одной из сетей, к которой подключен мост, поэтому пакет передается на стандартный шлюз первого моста (так же вычисляется MAC по IP с помощью ARP)

Второй мост точно так же проверяет не предназначен ли пакет лично ему, затем накладывает маску по AND на целевой адрес пакета и на два своих IP. Видит, что

192.168.3.0 == 192.168.3.0

Значит, пакет идет в одну из сетей, к которым подключен мост. Тогда мост получает MAC адрес целевого узла (MAC(192.168.3.13)) и отправляет по нему пакет.

Однако, как видно из схемы, доставка пакета в четвертую подсеть (например, узлу K4) извне невозможна в силу отсутствия перехода от M2 к M3 (пакет будет крутится между M1 и M2 до тех пор, пока его время жизни не станет равно нулю и он не будет уничтожен). То есть максимум 2 сети можно соединить мостами. Таким образом, очевидно, что построение крупной сети типа Internet на мостах невозможно. Нужно использовать маршрутизаторы.

Маршрутизаторы отличаются от мостов тем, что у них есть таблицы, указывающие куда надо передавать пакет. Для маршрутизатора можно указать, что пакет с адресом, попадающим в заданный диапазон, надо передавать туда-то. То есть вместо основного шлюза целевая таблица маршрутизации. Так же, в отличии от мостов, маршрутизатор имеет множество входов.

Для управления маршрутизатором есть утилита route (нужно указывать для какого протокола IPv4 или IPv6). Она позволяет конфигурировать таблицу маршрутизации данного узла.

9. Транспортный уровень OSI. Задачи и функции уровня. Классы транспортных протоколов. Передача данных с установкой и без установки соединения.

На пути от отправителя к получателю пакеты могут быть искажены или утеряны. Хотя некоторые приложения имеют собственные средства обработки ошибок, существуют и такие, которые предпочитают сразу иметь дело с надежным соединением.

Транспортный уровень обеспечивает приложениям или верхним уровням стека — прикладному, представления и сеансовому — передачу данных с той степенью надежности, которая им требуется.

Функции транспортного уровня:

- Идентификация программ-абонентов на узлах сети;
- Передача сообщений с установкой и без установки соединения;
- Разбиение сообщений на пакеты и их сборка в правильном порядке;
- Уведомление о доставке с повторной передачей утерянных пакетов.

10. Принципы IP-адресации. Протоколы ARP и RARP, принципы их работы. Уязвимость протокола ARP. Понятие маски и стандартного шлюза. Протокол DHCP и принцип его работы.

IP-адрес – это уникальная 32-разрядная последовательность двоичных цифр, с помощью которой компьютер однозначно идентифицируется в IP-сети.
Для удобства работы с IP-адресами 32-разрядную последовательность обычно разделяют на 4 части по 8 битов (на октеты), каждый октет переводят в десятичное число и при записи разделяют эти числа точками. В адресе кодируется номер **подсети** и номер **узла**.
Классы: A – 0|7-разрядный номер сети|24-разрядный номер узла, B – 1|0|14-р сети|16-р узла, C – 1|1|0|21-р сети|8-р узла, D (для маршрутизаторов) – 1|1|1|0|28-разрядный номер группы.
Маска подсети – битовая маска, старшие разряды которой установлены в единицу, а младшие – в ноль; определяет, какая часть IP-адреса относится к номеру подсети, а какая – к номеру узла.
Вычисление номера подсети: IP & Маска (побитовая операция «И»). После / кол-во 1 в маске.
Зарезервированные IP: 0.0.0.0 – адрес несущ. узла, имеет особое значение, **127.0.0.1** – обращение к самому себе (не задействует канальный уровень), **255.255.255.255** – широковещательный адрес любой сети, передача сообщений сразу всем узлам, обеспечивается канальным уровнем.
ARP(Address Resolution Protocol) – позволяет по IP узнать MAC, RARP(Revers...) – наоборот.
Для передачи IP-дейтаграммы по локальной сети необходимо знать MAC-адрес.

- Посылается широковещательный кадр в сеть с указанием искомого IP-адреса.
- Узел с искомым IP-адресом посылает ответ, в который вкладывает свой MAC-адрес.

Тип канального протокола		Тип сетевого протокола(IP)
Длина физ адреса	Длина лог адреса	Операция
Физический адрес отправителя (MAC)		
Логический адрес отправителя (IP)		
Физический адрес получателя (MAC)		
Логический адрес получателя (IP)		

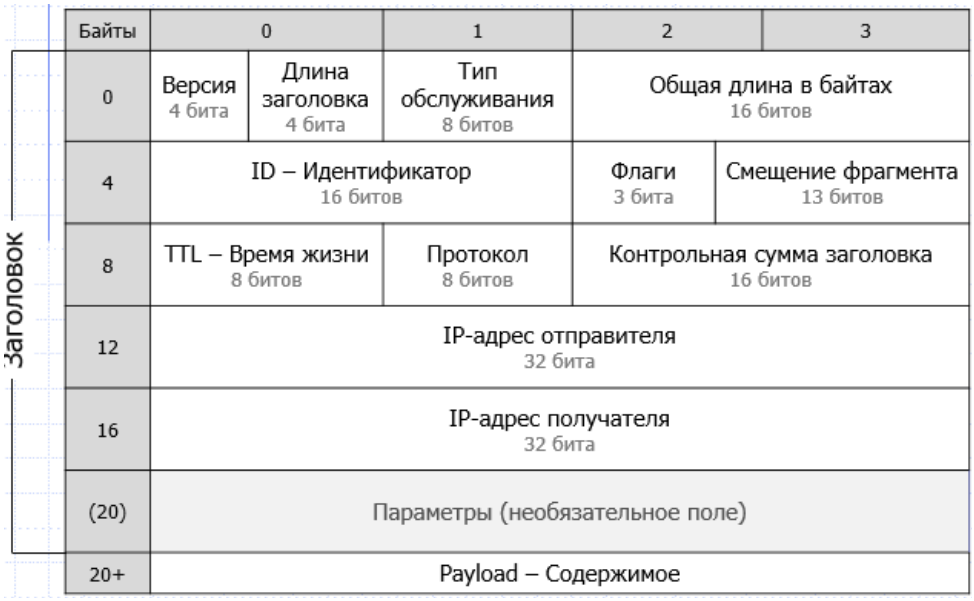
Кэширование ARP-ответов:
Если перед посылкой каждого сообщения делать ARP-запрос, то в сети увеличится трафик и снизится скорость. Чтобы этого избежать, результаты кэшируются узлом отправителем. ARP-кэш – это таблица, которая содержит записи соответствий: IP-адрес ⇔ MAC-адрес (dynamic(автоматически, со врем. устар.)/static(вручную)).
Отсутствует проверка подлинности ARP-ответа.

Атака посредника – злоумышленник тайно ретранслирует (и может изменять) сообщения между двумя сторонами, которые считают, что они непосредственно общаются друг с другом.

Шлюз – на этот адрес передаются все пакеты, адресованные не в данную сеть.
DHCP (Dynamic Host Control Protocol)
DHCP-сервер содержит у себя таблицу MAC-адресов и соответствующих им IP-адресов. Клиенты, которые вошли в сеть и не имеют параметров протокола IP для работы, обращаются с широковещательным запросом и обнаруживают DHCP-сервер, который им отвечает, указывает свой IP адрес и MAC-адрес, на который узел посылает запрос с просьбой выдать ему параметры конфигурации для работы в локальной сети.
DHCP-сервер ведет у себя таблицу IP-адресов и соответствующих им MAC-адресов. Когда к серверу приходит запрос, то в таблице появляется новая запись. Из пула свободных адресов

выдается IP, в таблицу вписывается MAC адрес того узла, от которого пришел запрос, а также время, до которого это соответствие является действительным («время аренды»).
В DHCP можно указать, что для определенных MAC адресов необходимо выдавать определенные IP-адреса.

11. Протокол IPv4. Формат пакетов и принципы работы. Функционирование протокола IP в локальной сети и в глобальной сети. Использование маски, стандартного шлюза и таблицы маршрутизации.



Версия – номер версии протокола – 4.

Длина заголовка – размер заголовка в 4х-байтовых словах; зависит от размера Параметров;

Тип обслуживания – параметры, управляющие качеством обслуживания; по битам:
[0-2] – приоритетность,
[3] – важность минимизации задержек при передаче,
[4] – важность скорости передачи (передавать по сети с низким трафиком), [5] – важность надёжности передачи, [6-7] – зарезервированы.

Общая длина в байтах – размер дейтаграммы, включая заголовок;

ID – номер дейтаграммы для правильной сборки фрагментов.

Флаги – управление фрагментацией:
[0] – зарезервирован, [1] – запрет фрагментации (DF), [2] – есть ли ещё фрагменты(MF).

Смещение фрагмента – позиция фрагмента в исходной дейтаграмме; для удобного приёма фрагментов не по порядку.

TTL – максимальное время, которое пакет может существовать в сети. Уменьшается на 1 при каждой пересылке пакета. Пакеты с TTL=0 отбрасываются.

Протокол – указывает, какой протокол следующего выше уровня (транспортного) инкапсулируется в дейтаграмме.

Контрольная сумма заголовка – пересчитывается и проверяется на каждой пересылке пакета, т.к. фрагментация и уменьшение TTL приводят к изменению заголовка.

Параметры – необязательное поле; содержит дополнительные опции различной длины, кратной байту. В общем случае длина не кратна 4 байтам – требуется дополнение для выравнивания по границе заголовка (управление маршрутизацией, служебная информация).

Функционирование протокола IP в локальной и глобальной сети

Для того чтобы работать в локальной сети достаточно установить IP-адрес, но для работы в глобальной сети необходимо задать:
1. Маску подсети
2. IP-адрес стандартного шлюза
3. Уникальный IP адрес

Например:
IP: 192.168.234.100 Маска: 255.255.255.0 Gateway: 192.168.234.19 (192.168.234.0 – номер подсети)

1) Если посылается пакет по протоколу IP, то берется адрес, на который отправляется пакет, и на него накладывается маска. Т.е., выясняют номер узла и номер сети, в которую направляется пакет. Если номер подсети совпадает с номером в своей подсети, то используется алгоритм передачи 1, а если нет, то алгоритм 2.

2) Пакет отправляется своему стандартному шлюзу. Шлем пакет по адресу 192.168.235.20. Номер подсети – 192.168.235.0. Пакет будет направлен компьютеру 192.168.234.19, но адресат остается прежним.

12. Протокол IPv6. Формат пакетов и основные отличия от протокола IPv4.

Новая версия призвана решить проблемы IPv4:

- недостаточная разрядность IP-адреса;
- отсутствие средств безопасности;
- большие накладные расходы на маршрутизаторах.

Адреса длиной 128 битов (16 байтов) вместо 32 битов (4 байтов). **Формат адреса** – 8 шестнадцатеричных числа в диапазоне от 0000 до FFFF, разделённых двоеточием

Удалены функции, которые усложняли работу маршрутизаторов, например, маршрутизаторы больше не разбивают пакет на фрагменты.

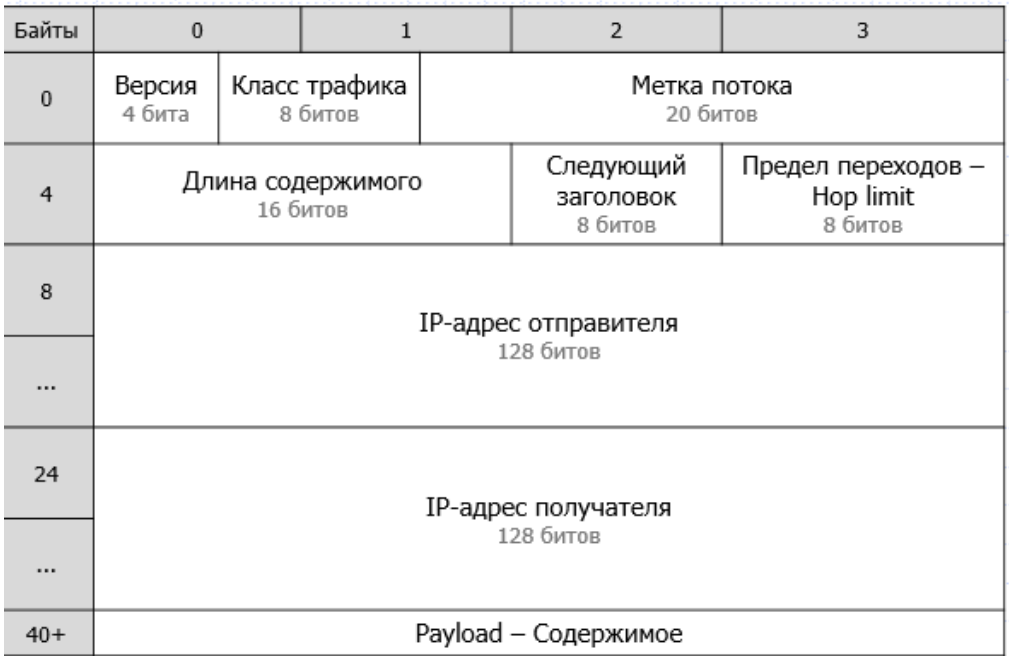
Появились заголовки расширений, при этом минимальный заголовок пакета удлинился с 20 байтов до всего 40 байтов.

Добавлены средства шифрования и безопасности.

Появились метки потоков.

Добавилось многоадресное вещание.

Поддержка сверхбольших дейтаграмм (англ. jambogram) с теоретическим пределом 4 ГБ.



Версия – номер версии протокола;

Промежуточные узлы могут реализовывать возможность управления трафиком, например, блокировки определённых соединений или установки приоритетов;

Метка потока позволяет идентифицировать соединение без необходимости анализа содержимого пакета 4-го транспортного уровня, что упрощает обработку трафика на промежуточных узлах. Для идентификации соединения используется кортеж из трёх элементов: адрес отправителя, адрес получателя, метка потока;

Класс трафика – управление качеством обслуживания, по битам: [0-2] – приоритетность; [3] – минимизация задержек; [4] – передавать по сети с низким трафиком; [5] – надёжность; [6] – Минимизация стоимости; [7] – зарезервирован.

Длина содержимого – размер содержимого в байтах. Не включает размер заголовков;

Предел переходов – Переименованное поле TTL из IPv4;

Следующий заголовок – тип следующего заголовка. Указывает тип заголовка транспортного уровня или одного из следующих доп. Заголовков IPv6:

- Заголовок фрагментации в IPv6 служит той же цели, что и смещение фрагмента, и флаги фрагментации в IPv4. Разделение IP-дейтаграмм на фрагменты и их сборку обеспечивает оконечные узлы, которые должны учитывать параметр Maximum Transmission Unit канального уровня. Маршрутизаторы избавлены от этой функциональности.
- Заголовок маршрутизации позволяет накопить информации о маршруте при передаче пакета в одну сторону, а потом использовать её для быстрой маршрутизации в обратную сторону.
- Заголовки аутентификации и шифрования обеспечивают безопасность;

- Заголовок опций позволяет передавать сверхбольшие дейтаграммы, превышающие максимальный размер 65,535 байтов.

13. Сравнительная характеристика протоколов с установкой и без установки соединения. Протокол UDP. Формат пакетов и принципы работы. Понятие порта.

Порт – двухбайтовый числовой идентификатор, который обозначает одного логического абонента на узле.

UDP – протокол транспортного уровня, обеспечивающий передачу дейтаграмм фиксированного размера:

- без установки соединения,
- без гарантий доставки,
- без подтверждения о доставке,
- без упорядочивания принимаемых пакетов в порядке отправления.

Обеспечивает более высокую скорость передачи данных, чем протокол TCP, но не предоставляет гарантий доставки. Используется для передачи мультимедиа-данных (аудио и видео), для которых реальный масштаб времени передачи важнее надёжности (допускается потеря пакетов).

Байты	0	1	2	3
0	Порт отправителя 16 битов		Порт получателя 16 битов	
4	Длина дейтаграммы 16 битов		Контрольная сумма* 16 битов	
8+	Данные			

Номер порта отправителя – задаёт порт, на который может быть отправлен ответ, если таковой ожидается; иначе – 0.

Номер порта получателя – обязательное поле.

Длина дейтаграммы – размер заголовка и данных в байтах.

Контрольная сумма* – для предотвращения ошибочной маршрутизации дейтаграмм; рассчитывается на основании части заголовка IP (IP-адрес отправителя, IP-адрес получателя, протокол), заголовка UDP и данных. Если контроль обеспечивается протоколом вышестоящего уровня, заполняется нулями. Не является обязательным для IPv4.

TCP – протокол транспортного уровня, обеспечивающий передачу потока данных:

- с установкой соединения,
- с гарантированной (надёжной) доставкой и повторной передачей утерянных пакетов,
- с подтверждением о доставке (квитирование),
- с разделением потока данных на сегменты и их сборкой в правильном порядке на принимающей стороне.

TCP позволяет управлять параметрами передачи данных и динамически подстраивать их под пропускную способность канала.

TCP обеспечивает дуплексное соединение с двумя потоками данных: «туда» (на передачу) и «обратно» (на приём).

14. Протокол TCP. Формат пакетов и общие принципы работы.

TCP – протокол транспортного уровня, обеспечивающий передачу потока данных:

- с установкой соединения,
- с гарантированной (надёжной) доставкой и повторной передачей утерянных пакетов,
- с подтверждением о доставке (квитирование),
- с разделением потока данных на сегменты и их сборкой в правильном порядке на принимающей стороне.

TCP позволяет управлять параметрами передачи данных и динамически подстраивать их под пропускную способность канала. TCP обеспечивает дуплексное соединение с двумя потоками данных: «туда» (на передачу) и «обратно» (на приём).

Sequence number (SN) Порядковый номер – смещение отправляемого сегмента в общем потоке данных в байтах. Начальное значение смещения для первого сегмента (логический ноль) выбирается при установке соединения.

Порядковый номер для следующего сегмента = номер текущего + размер текущего сегмента. Допустимо переполнение 32-разрядного значения смещения, т.е. диапазон «зацикливается» и отсчёт байтов продолжается с 0+, что позволяет передавать поток данных бесконечной длины. **Номер подтверждения (ACK-SN)** – квитанция; смещение в общем потоке данных в байтах, до которого все сегменты успешно приняты. Указывает, какой следующий порядковый номер ожидает принимающая сторона. **Длина заголовка** – размер заголовка TCP в 4х-байтовых словах; **Флаги** – биты, управляющие процессом передачи данных: URG (Urgent) – признак срочных данных; задействует поле Указатель срочных данных. ACK (Acknowledgement) – признак подтверждения; задействует поле Номер подтверждения (ACK-SN). PSH (Push) – обработать данные, не дожидаясь заполнения буфера: для отправляющей стороны – отправить по сети; для принимающей стороны – отдать данные на вышестоящий уровень. RST (Reset) – сброс соединения; применяется в ситуациях, когда произошла рассинхронизация соединения либо необходимо разорвать старое соединение после сбоя системы. SYN (Synchronize) – запрос синхронизации Порядкового номера (SN). FIN (Finalize) – готовность закрыть соединение: больше нет данных для отправки, но т.к. соединение дуплексное, то необходимо продолжать читать до получения FIN от другой стороны.

Указатель срочных данных – граница данных, которые помечаются как срочные (устарело).

Контрольная сумма* – для предотвращения ошибочной маршрутизации пакетов; рассчитывается на основании части заголовка IP (IP-адрес отправителя, IP-адрес получателя, протокол), заголовка TCP и данных. Заполняется отправителем и проверяется получателем.

Размер окна – общий объём данных в байтах, которые принимающая сторона способна принять на данный момент. Позволяет «приёмнику» управлять «передатчиком». (Передаётся вместе с квитанцией о доставке. Для стороны, которая получает квитанцию, означает, сколько байтов могут быть отправлены и ожидать подтверждение о доставке. Динамически подстраивается под пропускную способность канала в процессе передачи данных.)

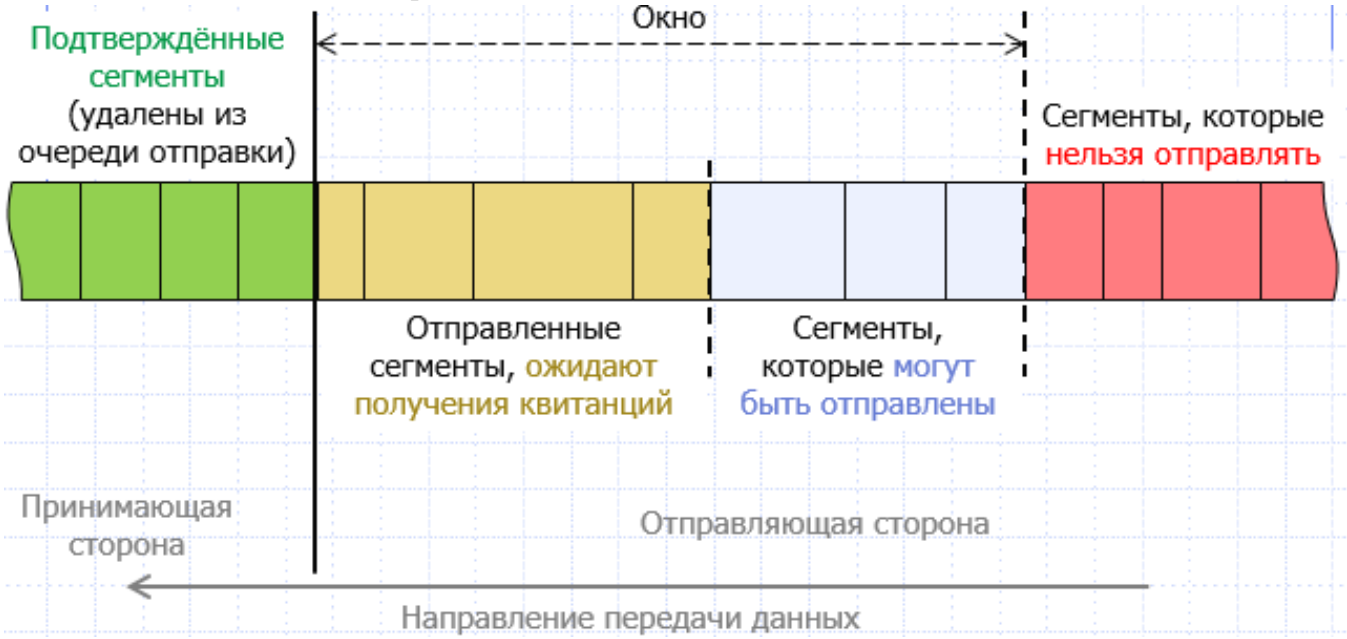
15. Принцип скользящего окна в протоколе TCP. Проблемы TCP.

Окно TCP – это общий объем данных, которые могут быть отправлены и не подтверждены.

Размер окна зависит от пропускной способности канала и подстраивается на основе адаптивного алгоритма.

Размер окна используется для согласования скоростей «приёмника» и «передатчика»:

- если принимающая сторона не готова принять новые сегменты (буфер заполнен, некуда писать), то она устанавливает размер окна равный нулю;
- отправляющая сторона перестаёт посылать сегменты, пока размер окна равен нулю;
- принимающая сторона периодически «зондирует» отправляющую сторону, посылая квитанцию на последний принятый байт.



После соединения размер окна устанавливается максимальным. Он влияет на количество данных, которые могут быть переданы без посылки подтверждения.

Принцип «скользящего окна»: окно по мере работы TCP смещается в правую сторону: как только приходит квитанция, левая граница смещается вправо.

Размер окна рассчитывается динамически следующим образом: при передаче каждого пакета вычисляется время его оборота. Размер окна определяется как средневзвешенное значение для десяти последних пакетов. Пакетам назначаются веса – от одного до десяти. Наибольший вес у самого последнего пакета, наименьший – у первого. Если используется такая формула, то с течением времени в значении размера окна будет учитываться история и сглаживаться пики, а при изменении пропускной способности сети произойдет адаптация. При этом резкое уменьшение пропускной способности не сильно сбросит окно.

Зондирование нулевым окном (получатель зондирует отправителя) – установка размера окна в 0. Показывает, что получатель "жив", соединение не разорвано, но пакеты в данный момент не могут быть приняты.

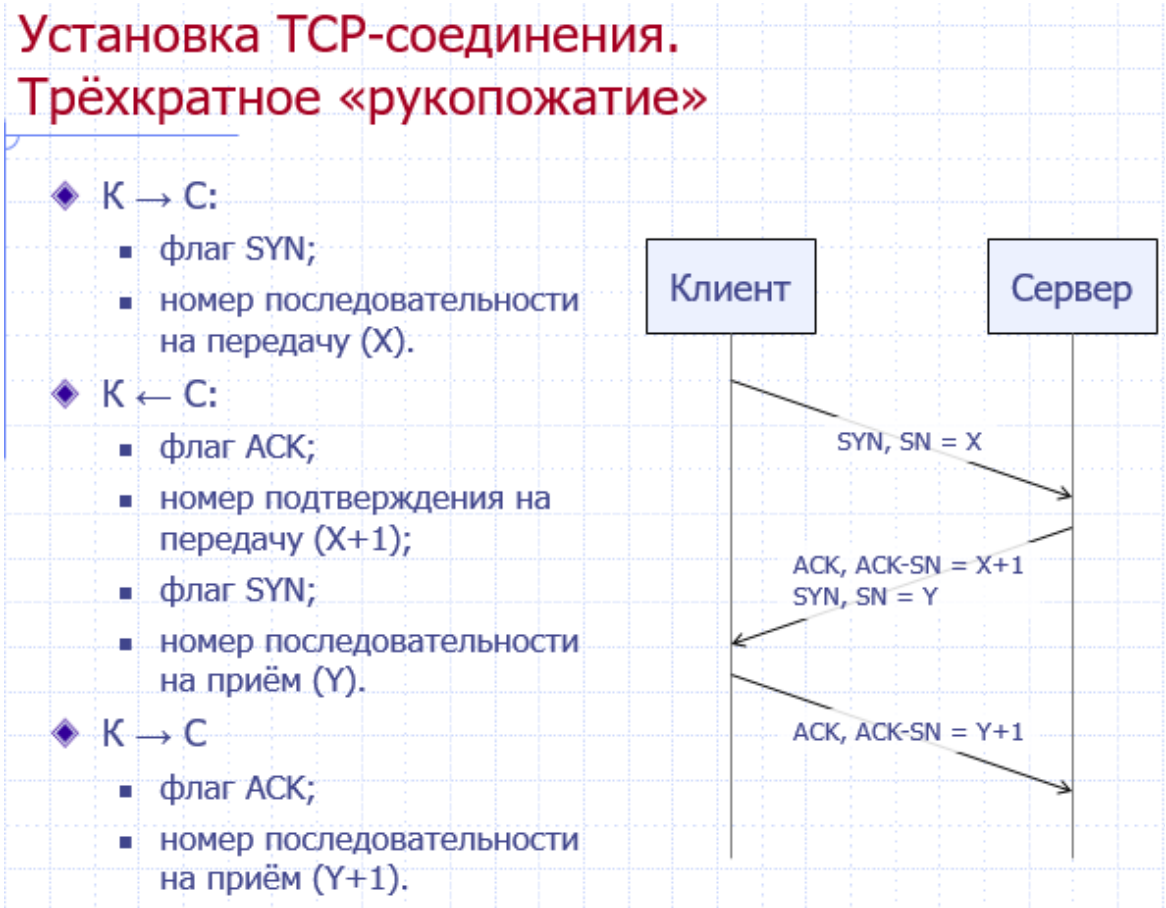
Проблемы TCP:

- Men in the middle;
- Парадокс дней рождений
- DoS атака;
- Атака запросами множества машин с разными IP.

16. Механизм установки TCP-соединения. Уязвимость TCP-протокола вида «парадокс дней рождения».

Сервер принимает запрос на установку соединения (пассивная сторона). Клиент посылает запрос (активная сторона).

1. Когда выполняется установка соединения, клиент посылает на заданный порт пакет с флагом установки соединения (SYN - синхронизировать порядковый номер (SN)). При этом, когда посылается пакет SYN, порядковый номер (SN) клиентом выбирается случайным образом и вписывается в пакет;
2. Сервер, получив запрос, отправляет на него ответ. В этом ответе устанавливаются флаги SYN, ACK (признак подтверждения (ACK-SN)). Тот IP адрес и порт, которые будут использоваться клиентом, берутся из ответного пакета, а для того чтобы сопоставить первый пакет и ответный пакет и нужен Random SN. Диапазон SN - Int32.
3. Получив пакет, клиент отправляет пакет с флагом ACK, после чего сервер переходит в состояние ESTABLISHED.



Парадо́кс дней рожде́ния. В группе, состоящей из 23 или более человек, вероятность совпадения дней рождения (число и месяц) хотя бы у двух людей превышает 50 %. Например, если в классе 23 ученика или более, то более вероятно то, что у какой-то пары одноклассников дни рождения придутся на один день, чем то, что у каждого будет свой неповторимый день рождения. Впервые эта задача была рассмотрена Рихардом Мизесом в 1939 году.

Во время установки TCP-соединения значения SN (номер последовательности) является по сути идентификатором абонента.

Злоумышленник может попытаться угадать значение ACK-SN в ответе от сервера и послать подтверждение на установку соединения вместо сервера, предварительно заставив его «молчать», перегрузив большим количеством запросов на установку соединения.

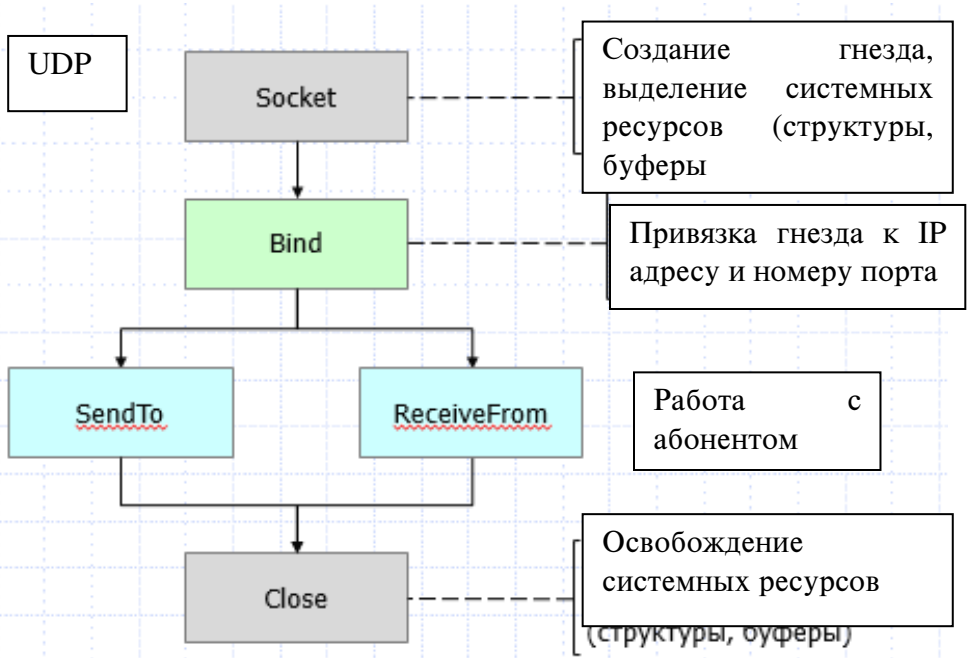
В некоторых Unix-серверах очень легко предсказать значение SN, выдаваемое сервером, из-за возникающего «парадокса дней рождений».

17. Интерфейс прикладного программирования Sockets («сетевые гнезда»). Взаимодействие клиента с сервером по протоколам UDP и TCP с помощью прикладного интерфейса Sockets.

Сокеты являются интерфейсом для стека протоколов TCP/IP, предоставляют возможность работы по протоколу TCP, UDP, с использованием «сырых» дейтаграмм IP.

Сокет (гнездо) – это точка подключения программы к сетевому интерфейсу.

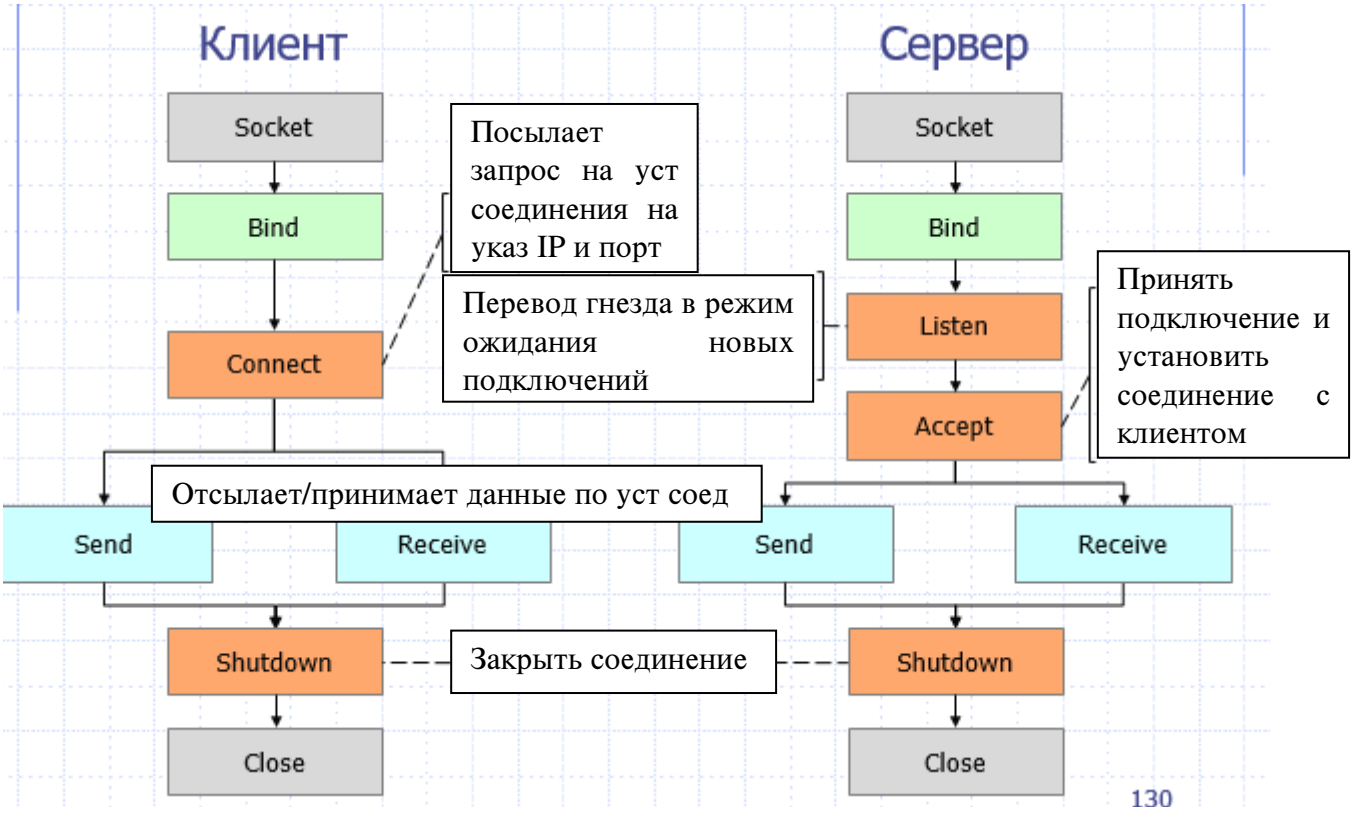
- У каждой программы, которая хочет использовать интерфейс, должно быть гнездо;
- гнезд может быть много, т.к. они идентифицируют канал связи для программы;
- гнездо имеет идентификатор – целочисленный номер, который по смыслу идентичен дескриптору файла в файловой системе.



В процессе обмена, как правило, используется два сокета — сокет отправителя и сокет получателя. Каждый процесс может создать «слушающий» сокет (серверный сокет) и привязать его к какому-нибудь порту операционной системы. Слушающий процесс обычно находится в цикле ожидания, то есть просыпается при появлении нового соединения.

Listen – создаёт очередь заданного размера для приёма входящих запросов на установку соединения и переводит гнездо в состояние прослушивания. Никакого ожидания не выполняет.

Accept – ждёт появления запроса на установку соединения, затем выбирает его из очереди, подтверждает его, отсылая пакет с флагами SYN и ACK, и дожидается ответного подтверждения. Создаёт и возвращает вызывающей стороне новое гнездо, которое служит для обмена данными с клиентом.



18. Виды узлов сети. Усилитель, повторитель, коммутатор, маршрутизатор, мост, шлюз.

Усилитель - используется для удлинения аналоговых линий связи. Увеличивает амплитуду сигнала.

Повторитель - используется для удлинения линий связи, в которых передается цифровой сигнал с одним из импульсных способов кодирования. Он принимает сигнал и потом просто повторяет его. Восстанавливает правильную форму.

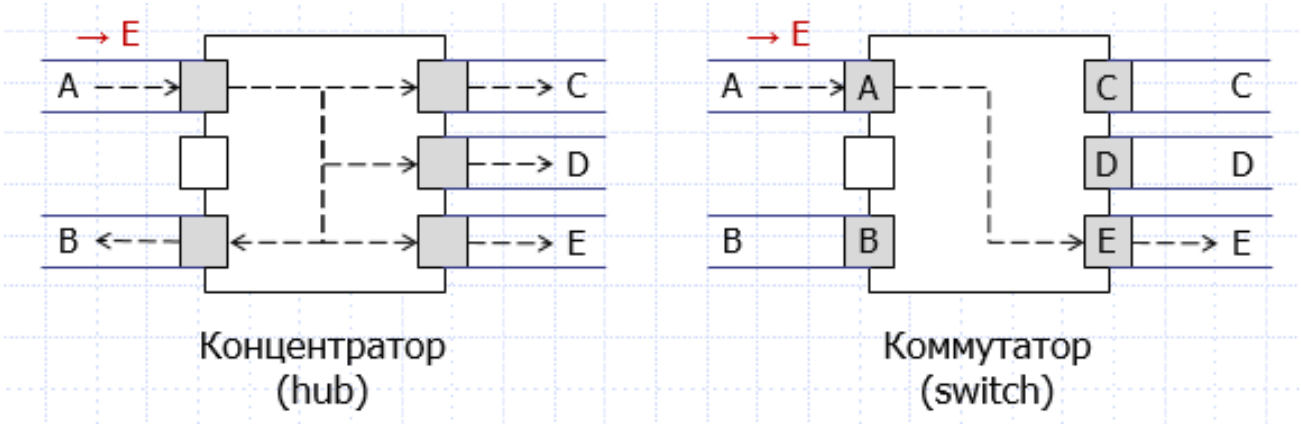
Мост – устройство, объединяющее две подсети в единую сеть. Мост сетевого уровня пересылает IP-пакеты;

Шлюз - аппаратный маршрутизатор или программное обеспечение для сопряжения компьютерных сетей, использующих разные протоколы (например, локальной и глобальной). На этот адрес передаются все пакеты, адресованные не в данную сеть;

Маршрутизатор — это устройство третьего сетевого уровня, он отличается от коммутатора тем, что работает не с кадрами Ethernet, а с IP-дейтаграммами – пакетами более высокого уровня, интерпретирует их. Маршрутизатор содержит таблицу маршрутизации, которая показывает, что делать с пакетами, которые имеют некоторые значения исходных IP-адресов и IP-адресов получателя (показывает, куда дальше должны быть отправлены пакеты).

Коммутатор – ретранслирует сигнал с одного входа на определённый выход. Распознаёт MAC-адреса и интерпретирует пакеты канального уровня. Применяется в топологии звезда.

- Без буферизации (перенаправляет пакет в нужный порт по MAC-адресу в заголовке;
- Буферизирующий (принимает пакет целиком в оперативную память, анализирует его заголовок и передаёт в нужный порт по MAC-адресу в заголовке;
- Маршрутизирующий (буферизирующий с дополнительной логикой маршрутизации пакетов канального уровня. Сокращает конфликты за счёт внутренней изоляции маршрутов.



19. Динамические системы именования. Принципы организации DNS. Рекурсивные и итеративные запросы. Выполнение DNS-запросов через интерфейс прикладного программирования Sockets.

DNS (Domain Name System) – Иерархическая распределенная база данных, которая хранит отображение имени в IP адрес. Стандартизирует форматы имен и запросы на получение каких-либо данных, ассоциирующихся с этими именами.

Принципы организации DNS

· В корне системы – пустое имя, не содержащее ни одного символа. От корня отходят домены первого уровня (COM, NET, ORG, RU, BY и др.), которые логически подчинены пустому имени. Далее могут следовать поддомены второго и т.д. уровней.

· Имена доменов не являются чувствительными к регистру;

· Имена доменов пишутся через точку в обратном порядке (третий.второй.первый);

· Имена доменов первого уровня оговорены заранее;

· Количество изменений на первом уровне должно быть минимально;

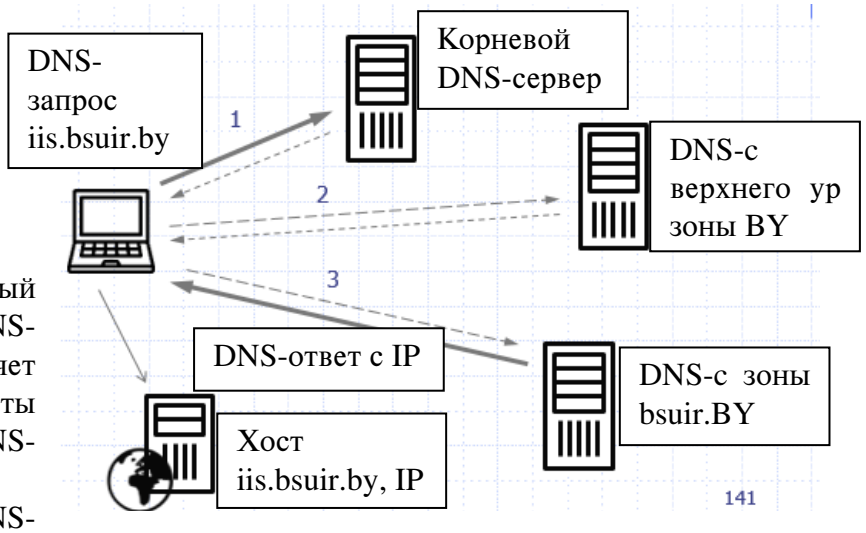
Когда клиент обращается к браузеру и открывает страницы, то постоянно идет преобразование имени в IP-адрес. Чтобы не забивать сеть повторяющимися запросами, используется кэширование: DNS-сервер запоминает имена и кэширует их на некоторый промежуток времени.

Итеративный запрос:
DNS-клиент опрашивает DNS-сервера по очереди.

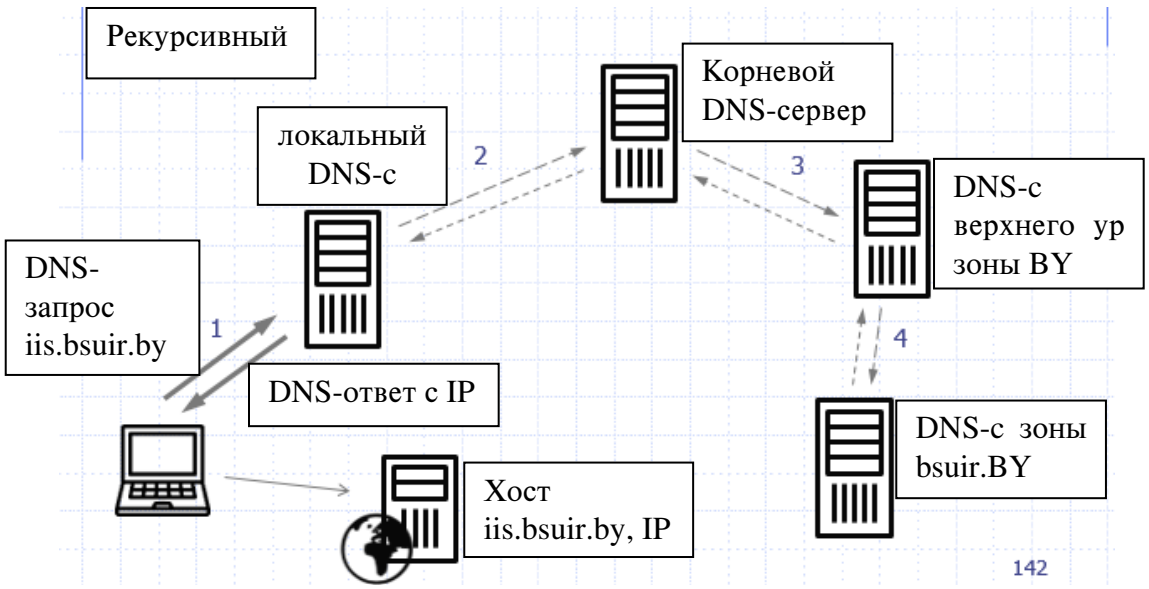
Рекурсивный запрос:
DNS-клиент перепоручает работу DNS-серверам.

Смешанный запрос:
DNS-клиент делает рекурсивный DNS-запрос к локальному DNS-серверу, который выполняет итеративный DNS-запрос. Ответы DNS **кэшируются** локальным DNS-сервером.

Снижает нагрузку на корневые DNS-сервера.



Протокол WHOIS – получение информации о домене: дата регистрации, срок действия, информация о владельце (название организации, адрес, электронная почта, телефон) и др.



20. Понятие сетевого экрана. Основные принципы его функционирования.

Сетевой экран (брандмауэр, firewall) – программа или аппаратно-программное устройство для фильтрации сетевого трафика с целью предотвращения угроз, исходящих из сети. Является обязательным компонентом операционной системы.

Работает на основе настраиваемых правил **на разных уровнях**:

· Канальный – фильтрует кадры (Ethernet, Wi-Fi);

· Сетевой – фильтрует IP-дейтаграммы;

· Транспортный – фильтрует UDP- и TCP-пакеты;

· Прикладной – распознаёт протоколы прикладного уровня (HTTP, FTP…) и фильтрует их данные; использует эвристические (не являющиеся гарантированно точными) подходы для фильтрации трафика.

Виды сетевых экранов по способу фильтрации

1) Сетевой экран без запоминания состояния

Использует статические правила, учитывающие только текущие параметры заголовков пакетов.

2) Сетевой экран с запоминанием состояния

Принимает решение динамически на основе текущего состояния сеанса и его предыстории.

Правила работы сетевых экранов

Работа сетевых экранов построена на использовании правил фильтрации. Правила бывают двух типов: **разрешающие и запрещающие**.

Правила применяются в определённом порядке, который по сути задаёт приоритет правил.

Стандартное поведение позволяет:

- Запретить всё, что не разрешено явно;
- Разрешить всё, что не запрещено явно;
- Запросить у пользователя создание нового правила с заданным поведением.

Параметры правил фильтрации: направление трафика(исходящий/входящий); IP-адреса; Тип протокола; Порты; Программы.

Назначения сетевых экранов: для защиты периметра, для защиты оконечного узла.

21. Технология NAT. Принципы организации и функционирования. Статическая и динамическая трансляция адресов и портов (PAT).

NAT (англ. Network Address Translation) – сетевая технология, которая позволяет узлам локальной сети с неуникальными локальными IP-адресами выходить в глобальную сеть Интернет.

Отличие обработки протоколов TCP и UDP в технологии динамического NAT (PAT)

Для протокола **TCP** новая запись в таблице NAT создаётся при передаче TCP-пакета с флагом SYN (установка соединения) из локальной сети в глобальную.

Пакеты на установку соединения, переданные из глобальной сети на адреса и порты NAT, отбрасываются.

Для протокола **UDP** (в котором отсутствует соединение) новая запись в таблице NAT создаётся при передаче первого UDP-пакета из локальной сети в глобальную.

Первый исходящий UDP-пакет «открывает» NAT для приёма ответных UDP-пакетов на выделенный глобальный порт от любого IP-адреса глобальной сети.

Статический NAT

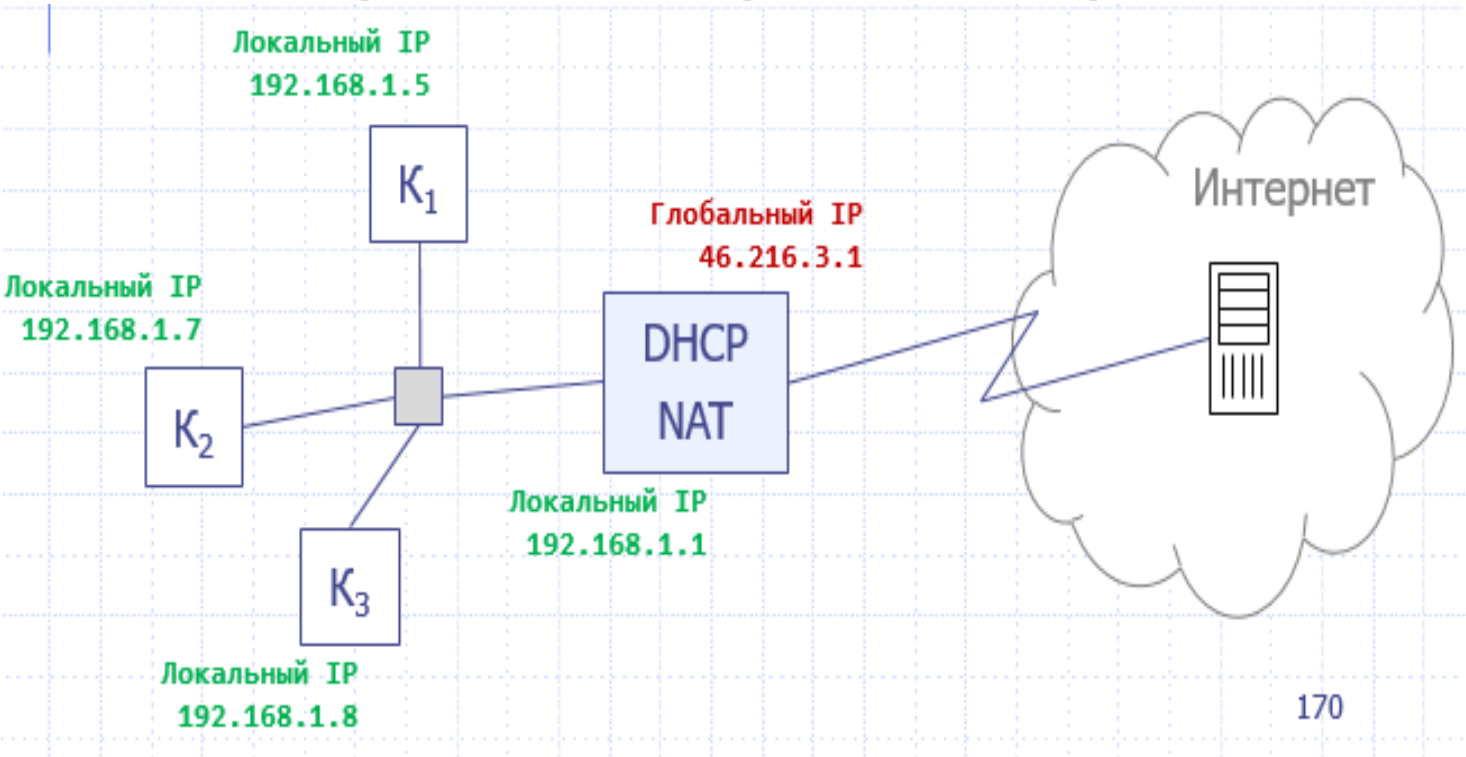
- Выделяет в пользование локальному узлу один из глобальных адресов.
- Подменяет локальный адрес глобальным адресом.
- Количество локальных узлов с одновременным доступом в глобальную сеть невелико.
- Обеспечивает установку соединения из глобальной сети с узлом в локальной сети.

Динамический NAT (PAT)

- Выделяет в пользование локальному узлу один из своих портов.
- Подменяет локальный адрес и порт глобальным адресом и портом из заданного диапазона.
- Количество локальных узлов с одновременным доступом в глобальную сеть велико.
- Запрещает установку соединение из глобальной сети с узлом в локальной сети.

DHCP – обеспечивает автоматическую выдачу локальных IP-адресов.

NAT – обеспечивает трансляцию локальных IP-адресов в глобальный IP-адрес.



22. Преодоление сетевых экранов при взаимодействии точка-точка по протоколу UDP (UDP Hole Punching).

Условие: два устройства хотят равноправно общаться по протоколу UDP, однако они оба находятся за брендмауерами, соответственно отправить сообщение извне невозможно (без статической настройки брендмауера, но это для лохов). вопрос - что делать?

Решение: третья сторона

Равноправное взаимодействие по протоколу UDP

Позволяет двум узлам, защищенным сетевыми экранами, установить равноправный прямой обмен данными по протоколу UDP.

Решает задачу обмена данными даже для узлов, находящихся в разных локальных сетях за устройствами с динамическим NAT (PAT).

Требует наличия третьей стороны для обмена информацией о точках доступа друг к другу (IP-адреса, порты). Англ. UDP hole punching («проделывание дырок» для UDP).

Пример:

1) Узлы A и B находятся за сетевыми экранами. Узлы A и B знают имена друг друга, но не знают IP-адресов и портов друг друга. Узлы A и B знают IP-адрес и порт третьей стороны – узла C.

2) Узлы A и B соединяются с C по протоколу UDP и аутентифицируются. (Сетевые экраны узлов A и B создают правила, разрешающие входящий трафик на IP-адреса и порты, с которых они обратились к C.) Узел C запоминает IP-адреса и порты узлов A и B.

3) Узел C сообщает узлам A и B IP-адреса и порты собеседников. (Сетевые экраны узлов A и B пропускают ответы от узла C на IP-адреса и порты, с которых они обращались к C ранее.)

4) Узел A сообщает узлу C, что он начинает «звонить» узлу B. Узел C сообщает узлу B, что ему «звонит» узел A.

5) Узел A «звонит» узлу B.

Сетевой экран узла A создаёт правило, разрешающее входящий трафик на IP-адрес и порт, с которых он «звонит» узлу B.

Сетевой экран узла B блокирует входящий трафик от узла A.

6) Узел B «звонит» узлу A навстречу.

Сетевой экран узла B создаёт правило, разрешающее входящий трафик на IP-адрес и порт, с которых он «звонит» узлу A.

Сетевые экраны узлов A и B пропускают встречные «звонки».

23. Преодоление сетевых экранов при взаимодействии точка-точка по протоколу UDP (UDP Hole Punching) в условиях, когда взаимодействующие узлы используют технологию NAT для выхода в Интернет.

ОТВЕТ В 22. (ВСЕ КОМПЫ ИСПОЛЬЗУЮТ NAT, В ПРИМЕРЕ ДОБАВЛЯЕТСЯ NAT, КОТОРЫЙ НАСТРАИВАЕТ ПОРТЫ ДЛЯ ЗАПРОСОВ)

Для протокола UDP (в котором отсутствует соединение) новая запись в таблице NAT создаётся при передаче первого UDP-пакета из локальной сети в глобальную. Первый исходящий UDP-пакет «открывает» NAT для приёма ответных UDP-пакетов на выделенный глобальный порт от любого IP-адреса глобальной сети. (это как раз позволяет “проделывать дырки” даже с NATом)

24. Технология виртуальных сетей VPN. Основные задачи. Способы реализации. Псевдо-каналы. Туннелирование.

VPN (Virtual Private Network) – это технология, которая решает задачу виртуализации и объединения физически различных сетей в одну логическую сеть.

Реализуется за счёт псевдо-каналов и туннелирования пакетов.

- Псевдо-канал – эмуляция телекоммуникационного канала поверх сети с коммутацией пакетов.
- Туннелирование – передача пакетов некоторого протокола (например, IP) в пакетах другого протокола.

Решает задачу объединения нескольких физически обособленных сетей в одну виртуальную частную сеть.

- Например, объединить несколько офисов одного предприятия.

Позволяет удалённому клиенту подключиться к другой локальной сети и стать частью этой сети.

- Например, сотруднику подключиться удалённо к локальной сети предприятия.

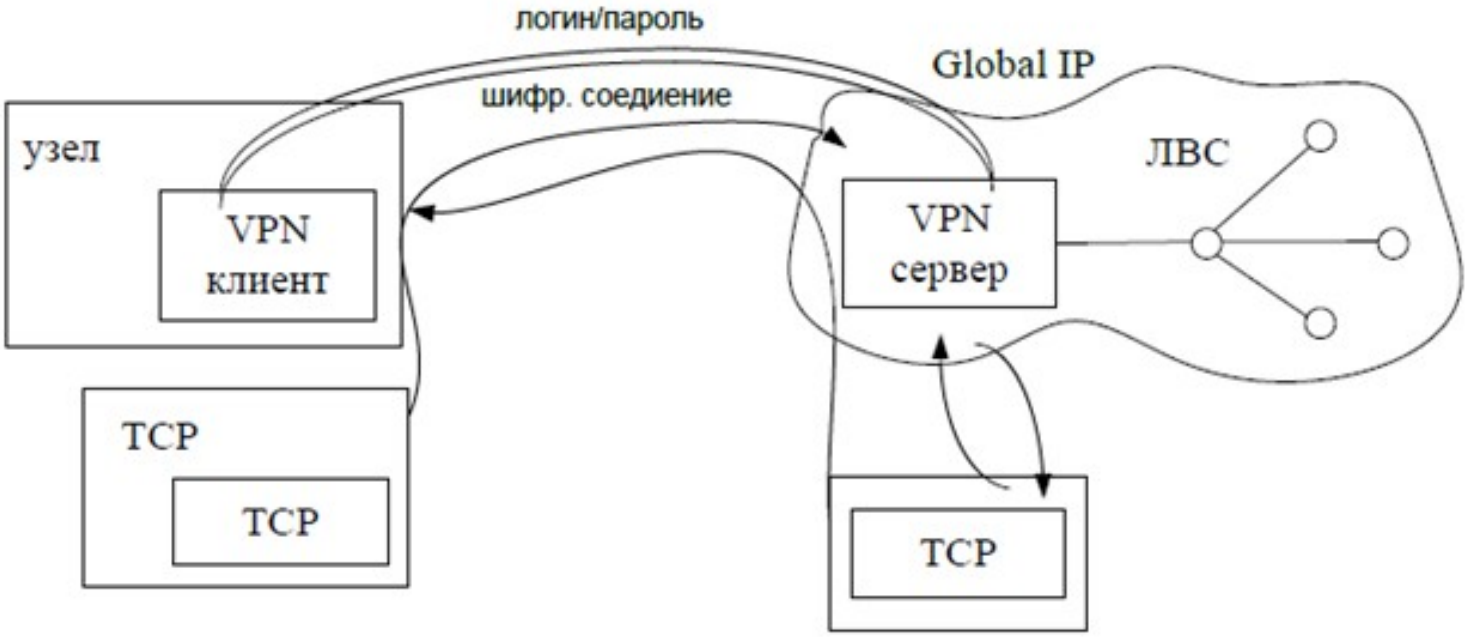
Позволяет виртуально изменить своё местоположение с помощью поставщика услуг VPN.

- Например, выйти в глобальную сеть от имени узла, расположенного в другой стране.

VPN-клиент устанавливает TCP соединение с VPN-сервером через глобальный IP-адрес, указывая также соответствующий логин и пароль.

Когда клиент хочет отправить данные в некоторый VPN, он вкладывает в TCP-пакет данные, которые передаются через сформированное соединение (VPN-клиент перехватывает все пакеты стека протоколов TCP-IP). Другими словами, передается пакет TCP, а сервер, в свою очередь, этот пакет «преобразует» и отправляет данные уже в рамках своей сети. Обратная процедура является аналогичной прямой процедуре.

Когда TCP-пакет приходит, он извлекается и передается дальше, т.е. создается впечатление того, что сообщение пришло из сети, хотя на самом деле это всего лишь имитация.



25. Электронная почта. Принципы организации и функционирования. Протоколы SMTP, POP3 и IMAP.

Электро́нная по́чта — технология и служба по пересылке и получению электронных сообщений между пользователями компьютерной сети.

Протокол передачи почты SMTP (англ. Simple Mail Transfer Protocol):

Порт сервера 25.

Обеспечивает доставку электронных писем, не рассчитан на работу с входящей почтой. Почтовые клиенты используют SMTP для отправки писем (современный порт сервера для отправки – 587).

Промежуточные сервера используют SMTP для пересылки отправленных писем адресату, на нужный почтовый сервер.

Данные в SMTP представлены в 7-разрядных символах ASCII. Каждый символ передаётся как 8 битов с установленным в 0 старшим разрядом. Расширения стандарта SMTP могут разрешить передачу полноценных 8-разрядных байтов данных.

Протокол доступа к почтовому ящику POP3 (англ. Post Office Protocol v3):

Порт сервера 110.

Служит для изъятия части или всей почты из почтового ящика.

Ориентирован на малый размер хранилища почтовых ящиков на сервере. Ориентирован на работу с почтой в автономном режиме.

Состояния сеанса: AUTHORIZATION (требуется вход) – состояние после подключения клиента к серверу; необходимо авторизоваться для получения доступа к ящику. TRANSACTION (накопление изменений) – состоояние после успешной авторизации; сервер заблокировал ящик, и клиент работает с почтой; все изменения накапливаются, но не применяются сразу. UPDATE (применение изменений) – состояние после выхода клиента: сервер применяет накопленные изменения.

После авторизации клиент может: просматривать состояние ящика, просматривать весь список и содержимое отдельных писем, помечать их к удалению, отменять все текущие изменения.

Положительный ответ сервера: +OK …, отрицательный: -ERROR …

Недостатки протокола POP3:

Отсутствие возможности управления письмами на сервере – перемещение по папкам, назначение ярлыков.

Необходимость «забирать» письма с почтового сервера для полноценной работы с почтой.

Отсутствие возможности работы несколькими клиентами с одним почтовым ящиком. Клиент А «забрал» письма, и клиент В их «не увидит».

Протокол доступа и управления сообщениями IMAP (англ. Internet Message Access Protocol):

Порт сервера 143.

Ориентирован на интерактивный режим работы с почтовым ящиком через интернет. Возможен одновременный доступ нескольких клиентов к одному ящику.

Возможность работы с несколькими ящиками (папками: «Входящие», «Исходящие», «Помеченные» и др.).

Гибкое управление почтой на сервере: ярлыки, пометки «важное», перемещение писем между ящиками.

Возможность отслеживать состояние письма (например, «прочитано», «отправлен ответ», «отложено», «удалено»).

Состояния сеанса: Not Authenticated (клиент не аутентифицировался) Authenticated (клиент аутентифицировался) Selected (клиент выбрал ящик) Logout (клиент выходит)

26. Прикладной протокол FTP. Принципы организации и функционирования. Команды и их формат.

FTP (File Transfer Protocol) – это протокол передачи файлов.

Использует раздельные соединения для управления и передачи файлов.

Соединение для управления устанавливается на порт сервера 21. Через это соединение передаются текстовые команды и ответы на них.

Соединение для передачи файлов открываются по мере необходимости. Инициатором соединения может выступать как сервер, так и клиент.

Режимы работы:

- Активный – инициатором соединения является сервер. Не пригоден, если клиент находится за устройством PAT или сетевым экраном.
- Пассивный – инициатором соединения является клиент. Позволяет клиенту использовать технологию PAT и сетевой экран.

После установления TCP-соединения клиент выполняет аутентификацию и авторизацию. Имя пользователя и пароль передаются в открытом виде.

После авторизации клиент с помощью соответствующих команд может: просматривать содержимое каталогов, менять текущий каталог, управлять файлами и каталогами на сервере, отправлять файлы на сервер и получать файлы с сервера.

Команда	Описание
Команды управления доступом	
USER name	Войти на сервер с использованием указанного пользователя (подключение уже должно быть установлено). Далее FTP-сервер запросит пароль. Возможен анонимный вход.
PASS pass	Передать серверу пароль. Выполняется сразу после команды USER.
ACCT info	Войти на сервер с использованием указанной учётной записи. Учётная запись не обязательно связана с пользователем, а команда с командой USER.
CWD path	Сменить текущий каталог на сервере.
CDUP	Перейти в родительский каталог на сервере.
SMNT path	Монтирование другой файловой системы по указанному пути на сервере.
REIN	Выход пользователя без разрыва соединения с сервером.
QUIT	Выход пользователя и разрыв соединения с сервером.
Команды параметров передачи	
PORT value	Войти в активный режим с указанными в команде IP-адресом и портом. Сервер сам подключается к клиенту для передачи данных.
PASV	Войти в пассивный режим. Сервер вернёт IP-адрес и порт, которые он «прослушивает» и к которым нужно подключиться, чтобы забрать данные.
TYPE code	Задать тип представления и хранения данных, в частности: ASCII (текстовый), Image (бинарный), интерпретацию «CRLF», а также логический размер байта в битах.

Команды передачи файлов	
RETR path	Начать передачу файла от сервера клиенту.
STOR path	Начать передачу файла от клиента серверу.
STOU	Аналогична STOR, но для файла генерируется уникальное имя в каталоге.
APPE path	Аналогична STOR, но данные добавляются в конец файла, если он уже существует.
ALLO args	Выделяет указанное количество байтов для файла в файловой системе на сервере.
REST marker	Возобновить разорванную передачу файла с заданной контрольной точки.
RNFR, RNTO	Переименовать файл. RNFR — что переименовывать, RNTO — во что переименовать.
ABOR	Прервать выполнение предыдущей команды (передачи). Соединение с сервером остаётся.
DELE path	Удалить указанный файл на сервере.

27. Протокол HTTP. Принципы организации и функционирования. Формат URL-адресов, структура запроса и ответа.

HTTP(Hyper-Text Transfer Protocol) — широко распространённый протокол передачи данных, изначально предназначенный для передачи гипертекстовых документов.

В соответствии со спецификацией OSI, HTTP является протоколом прикладного уровня.

Протокол HTTP предполагает использование клиент-серверной структуры передачи данных. Клиентское приложение формирует запрос и отправляет его на сервер, после чего серверное программное обеспечение обрабатывает данный запрос, формирует ответ и передаёт его обратно клиенту.

Для того, чтобы сформировать HTTP-запрос, необходимо составить стартовую строку, а также задать по крайней мере один заголовок — это заголовок Host, который является обязательным, и должен присутствовать в каждом запросе.

Коды ответов HTTP

100-199 – Информационные

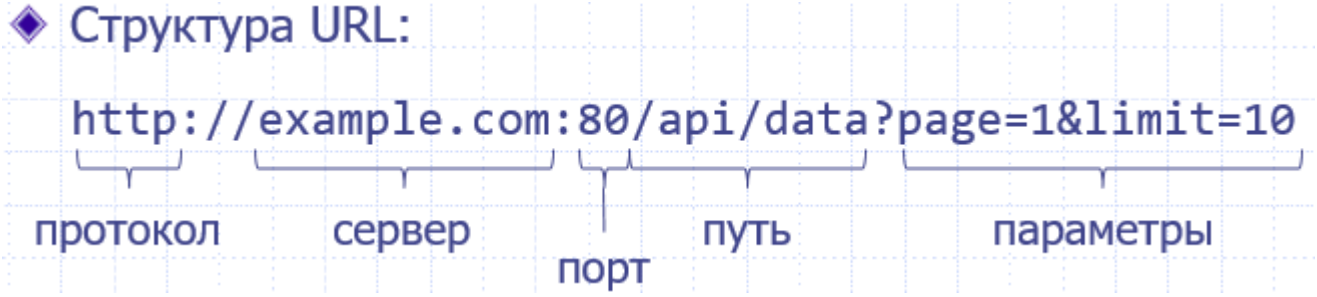
200-299 – Успех

300-399 – Переадресация

400-499 – Ошибка со стороны клиента

500-599 – Ошибка со стороны сервера

Uniform Resource Identifier



Структура запросов и ответов в 1.1

Запросы и ответы – это текст, состоящий из заголовков и содержимого.

Заголовок – это строка, оканчивающаяся символами «CRLF».

Блок заголовков отделяется от содержимого пустой строкой (двумя парами символов «CRLF»).

Первый заголовок запроса содержит тип, путь и версию протокола.

Первый заголовок ответа содержит версию протокола и код ответа.

Все заголовки делятся на: заголовки запроса; заголовки ответа; заголовки запросов и ответов.

```
GET / HTTP/1.1
```

```
HTTP/1.1 200 OK
```

28. Типы HTTP-запросов и их классификация. Основные HTTP-заголовки. Понятие "куки".

GET – получить текущее представление указанного ресурса.
HEAD – получить информацию о текущем представлении ресурса (аналогичен GET, но в ответе только заголовки, без содержимого).
PUT – заменить текущее представление указанного ресурса или создать новый ресурс.
POST – отправить данные на сервер, добавив к указанному ресурсу.
DELETE – удалить указанный ресурс.
PATCH – обновить, частично изменить указанный ресурс
(добавлен в 2010 году).
OPTIONS – получить параметры соединения с указанным ресурсом: доступные методы, форматы.
TRACE – проверить прохождение запроса к указанному ресурсу.
+ CONNECT – установить соединение (при использовании прокси-сервера).
Классификация:

- Безопасные (не изменяют ресурсы на сервере). Например, запросы на чтение (GET, HEAD, OPTIONS, TRACE)
- Небезопасные (могут изменять ресурсы). Например, запросы на изменение (PUT, POST, DELETE, PATCH)
- Идемпотентные (результаты повторных вызовов идентичны результату одиночного вызова). Например, GET, HEAD, OPTIONS, TRACE, PUT, DELETE.
- Не идемпотентные (повторный вызов приводит к новому изменению). Например, POST, PATCH.

Основные HTTP-заголовки

Заголовки управления соединением: Connection, Keep-Alive
Заголовки согласования типа содержимого: Accept, Accept-Encoding
Заголовки свойств содержимого: Content-Type, Content-Encoding, Content-Length
Заголовки условий: Last-Modified, ETag , If-Match
Заголовки «куки»: Set-Cookie, Cookie
Заголовок переадресации: Location
Заголовки кэширования: Cache-Control, Expires
Заголовки контекста запроса: Host, User-Agent
Заголовки контекста ответа: Allow, Server

HTTP cookie – это небольшой фрагмент данных, хранящий состояние для протокола HTTP. Отправляется сервером клиенту, который может сохранить эти данные и отсылать обратно с новыми запросами к данному серверу.
Применение:

- Управление сеансом – логины, корзины для виртуальных покупок;
- Персонализация – пользовательские предпочтения, оформление;
- Мониторинг – отслеживания поведения пользователя, таргетированная реклама.

«Куки» бывают:

- временные (сессионные) или постоянные (указан атрибут Expires или Max-Age);
- «безопасные», требующие протокол HTTPS (указан атрибут Secure);
- недоступные из JavaScript (указан атрибут HttpOnly).

29. Виды HTTP-соединений и их особенности. Организация конвейера запросов-ответов. Составной HTTP-ответ, размер которого не известен в момент отправки.

HTTP-соединение:

- Единовременное (close) – по одному соединению передается только один запрос ответ.
- Постоянное(keep-alive) – по одному соединению может передаваться несколько запросов и ответов последовательно. В ответе обязательно указываются Content-Length и Content-Type.

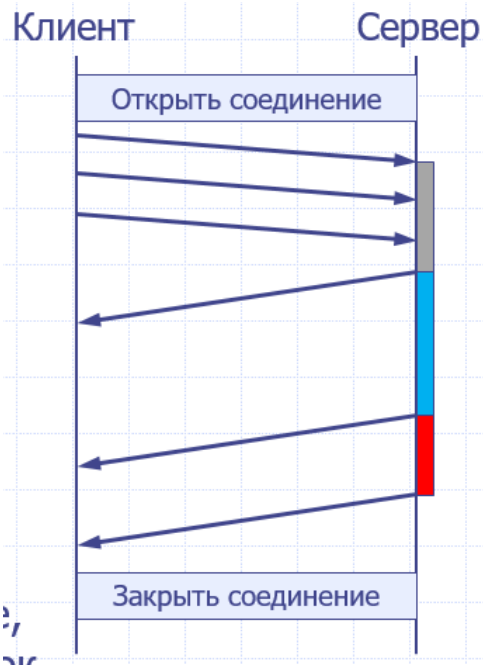
Конвейера запросов-ответов

Отсылка новых запросов по одному соединению без ожидания ответов на предыдущие запросы (HTTP/1.1).

Поддерживает только идемпотентные запросы (GET, HEAD, PUT, DELETE), повторное выполнение которых безопасно.

Можно повысить производительность, упаковав несколько HTTP запросов в один TCP-пакет.

В HTTP/1.1 отсутствуют средства сопоставления запросов и ответов, поэтому ответы обязаны приходить в том же порядке, что и отправленные запросы. Этот недостаток не позволяет полноценно использовать конвейерный режим запросов-ответов.



Составной HTTP-ответ, размер которого не известен в момент отправки

Указывается служебный заголовок Transfer-Encoding: chunked.

Ответ отправляется по частям (англ. chunks) с указанием размера каждой части. Конец ответа обозначается пустым блоком нулевой длины.

```
HTTP/1.1 200 OK
Content-Type: text/plain
Transfer-Encoding: chunked

3\r\n
Per\r\n
9\r\n
 aspera\r\n\r\n
3\r\n
ad \r\n
5\r\n
astra\r\n
0\r\n
\r\n
```

30. Использование заголовков условий HTTP при кэшировании содержимого ответов и совместного редактирования данных.

Современные веб-браузеры поддерживают кэширование редко меняющихся ресурсов.

Может быть кэширование не только страниц но и ресурсов страниц, например, изображений, таблиц стилей, библиотек JS.

Кэширование используется для ускорения загрузки страниц, но при этом расходует память на локальном диске компьютера для кэша.

Протокол HTTP поддерживает работу хэша

Чтобы браузер понимал, когда нужно запрашивать новую страницу, а когда брать из кэша используется заголовок Expires, который указывает до какого вр емени можно хранить ресурс в кэше. Но веб-сервера не всегда устанавливают этот заголовок. (Пример: Expires: Sun, 12 Jun 2016 10:35:18 GMT).

Если Expires не установлен то браузер может использовать поле Last-Modified, в котором указано время последнего изменения и если ресурс давно не менялся то может использовать ресурс из кэша. Но в данном случае возрастает шанс ошибки.

Запрос Get с условием(Conditional get):

Это запрос веб-серверу передать ресурс в том случае, если он был изменен.

При первом запросе веб-серверу:

- Ответ сервера содержит заголовок Last-Modified.
- Ресурс записывается в кэш.

При повторном обращении

- Запрос будет содержать заголовок If-Modified-Since: {здесь будет Last-Modified который мы получили при первом обращении}

В ответ на If-Modified-Since может прийти два ответа:

- 304 Not Modified. В этом случае могут быть переданы дополнительные заголовки: Cache-Control, Expires, Last-Modified
- 200 OK. Полная передача обновленной в ерсии ресурса.

ETag(entity tag) начиная с HTTP 1.1

Код который генерируется на основе содержимого ресурса.

Этот заголовок прикрепляется к ответу.(ETag: “123-8df”)

Запрос GET с условием:

If-None-Match: 123-8df

Заголовок Cache-Control начиная с HTTP 1.1

С помощью этого заголовка можно более гибко управлять кэшированием.

Cache-Control: private, max-age = 10

Возможные значения: no-store - не сохранять в кэш, no-cache - можно сохранять, но при использовании выполнить conditional get, public - информация доступна всем и ее можно кэшировать, private - может быть сохранена только в частном кэше браузера, max-age - время хранения в кэше в секундах.

ПРЕЗЫ. При кэшировании:

Клиент запрашивает данные.

Сервер возвращает данные с посчитанным хэш-кодом и датой последнего изменения.

Клиент кеширует данные.

При повторном запросе данных заголовки условий позволяют серверу узнать версию данных у клиента и избежать повторной передачи содержимого.

Совместное редактирование:

Клиент А получает данные, изменяет их локально и отправляет серверу.

Клиент В получает данные одновременно с клиентом А, изменяет их локально и отправляет серверу свои изменения позже, чем клиент А.

Изменения клиента А применяются, а изменения клиента В отбрасываются с ошибкой.

31. Аутентификации и авторизация в протоколе HTTP. Типы аутентификации Basic и Digest.

Аутентификация – установление личности клиента.

Авторизация – проверка прав доступа клиента.

Аутентификация в протоколе HTTP:

Осуществляется для каждого запроса отдельно. Сервер просит аутентификацию для доступа к ресурсу с помощью заголовка ответа WWW-Authenticate. Клиент выполняет аутентификацию и авторизацию с помощью заголовка Authorization.

Основные типы аутентификации в HTTP: Basic – имя пользователя + пароль, Digest – на основе дайджеста (хэша), Bearer – на основе маркера безопасности.

Basic

Имя типа: Basic

Имя пользователя и пароль объединяются в одну строку через двоеточие и кодируются в Base64.

Пример:

- Имя: «username»
- Пароль: «password»
- Строка для кодирования: «username:password»
- Запрос:

```
GET / HTTP1.1
Authorization: Basic dXNlcm5hbWU6cGFzc3dvcmQ=
```

Digest

Имя типа: Digest

Позволяет избежать передачи имени пользователя и пароля в открытом виде. Передаётся хэш-код пароля, имени пользователя, текущего времени и других параметров, полученных от сервера.

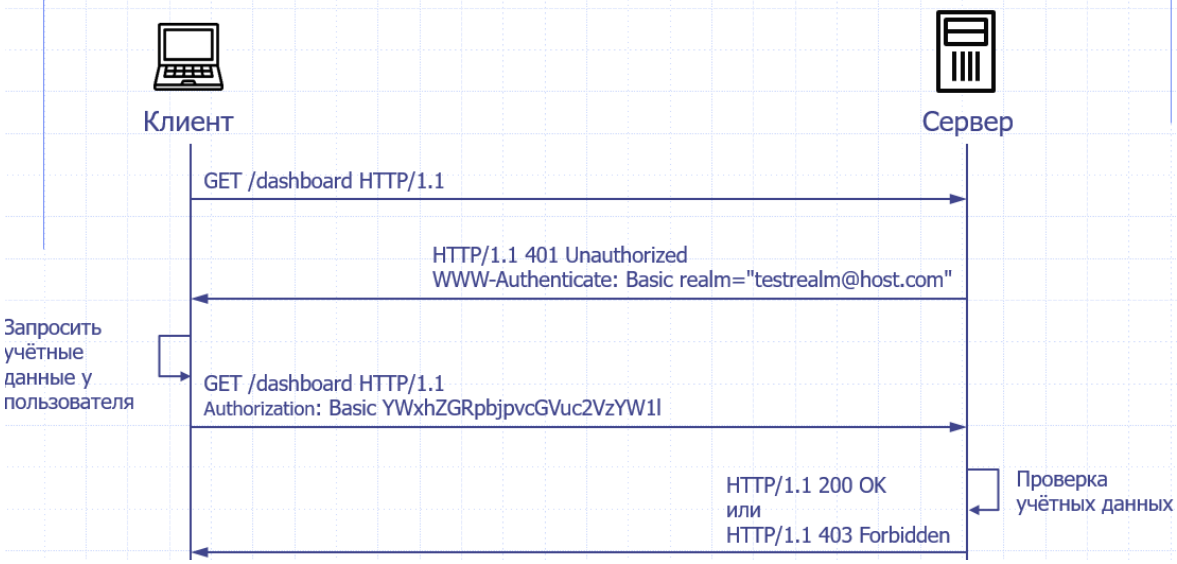
Заголовок WWW-Authenticate содержит:

- realm – защищённая область сайта;
- qop – степень защиты: обычная (auth) или с проверкой целостности (auth-int);
- algorithm – алгоритм хэширования (MD5, SHA-256);
- nonce – однократно используемая строка в формате Base64 nonce="<точное-время> MD5(<точное-время>:ETag:<секрет>)"

Заголовок Authorization содержит:

- username – имя или хэш-код имени пользователя;
- nc – счётчик повторного использования nonce;
- cnonce – однократно используемая строка от клиента;
- response – хэш-код, включающий: имя пользователя, пароль, значение nonce, полученное от сервера, а также значения realm и URI response="MD5(HA1:nonce:nc:cnonce:qop:HA2)", где HA1 = MD5(username:realm:password), HA2 = MD5(HTTP-method:URI)

Общая схема аутентификации в HTTP



32. Аутентификация в HTTP на основе маркера безопасности. Bearer

Имя типа: Bearer (носитель, предъявитель)

Используется для избегания частых аутентификаций.

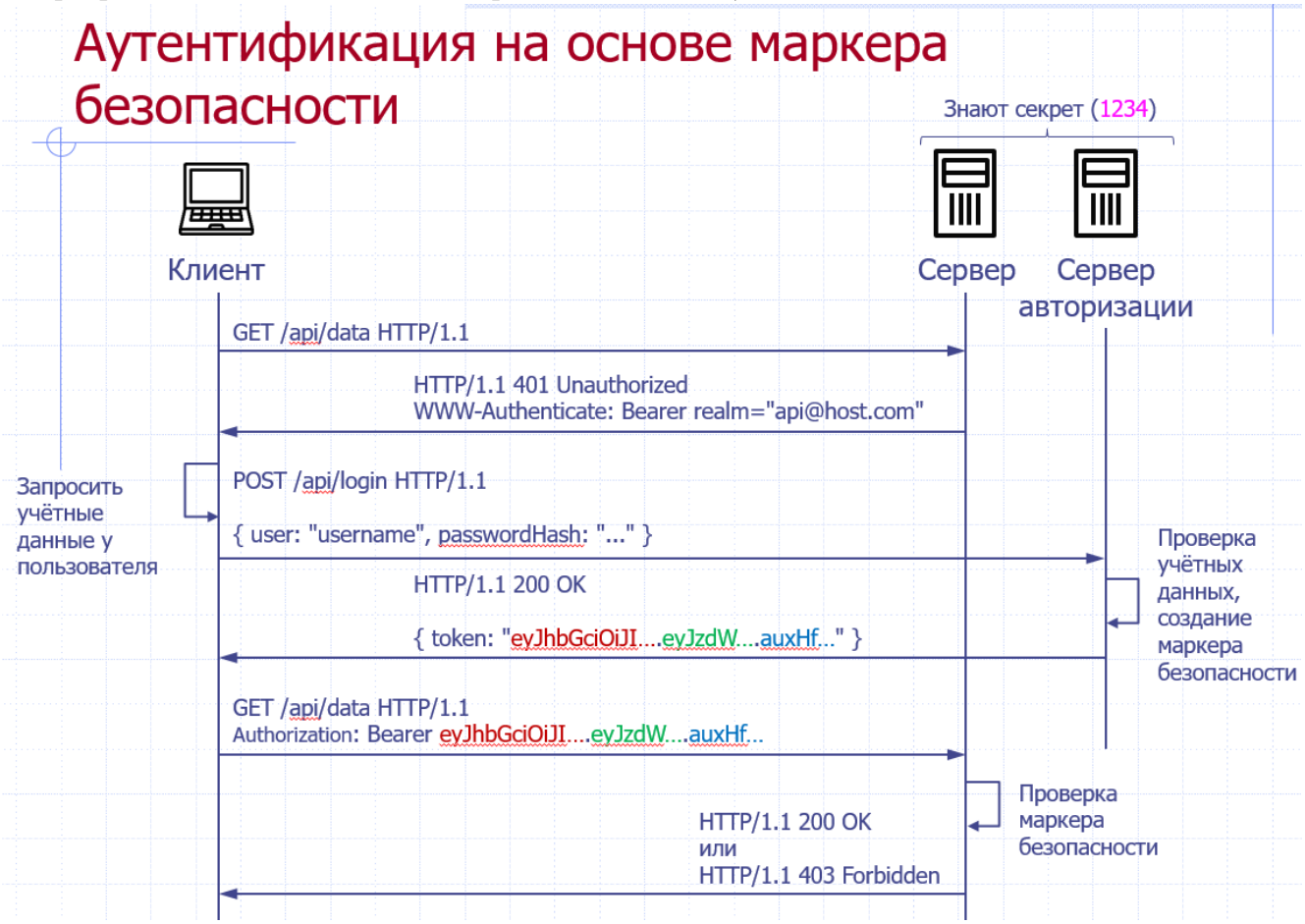
Ориентирован на использование шифрованных соединений.

Ориентирован на авторизацию через стороннюю службу аутентификации.

Обеспечивает аутентификацию и авторизацию без раскрытия учётных данных пользователя (имени, пароля).

Маркер безопасности (англ. security token):

- Маркер часто представляет собой зашифрованный и/или подписанный идентификатор пользователя со списком прав доступа.
- Полученный маркер безопасности может быть использован для осуществления доступа к ресурсам без необходимости аутентификации в течение определённого периода времени.
- Время действия маркера ограничено. После его истечения необходимо провести повторную аутентификацию.
- Существуют механизмы обновления маркера без повторной аутентификации (на основе другого маркера безопасности с бóльшим временем действия).



33. Технология TLS/SSL и протокол HTTPS.

Протокол HTTP не предоставляет средств обеспечения безопасности.

Для установки и поддержания защищённого соединения используется протокол защиты транспортного уровня TLS (англ. Transport Layer Security), который основан на ныне устаревшем протоколе SSL (англ. Secure Socket Layer).

TLS предоставляет три основных функции, которые обеспечивают безопасный и защищённый обмен данными:

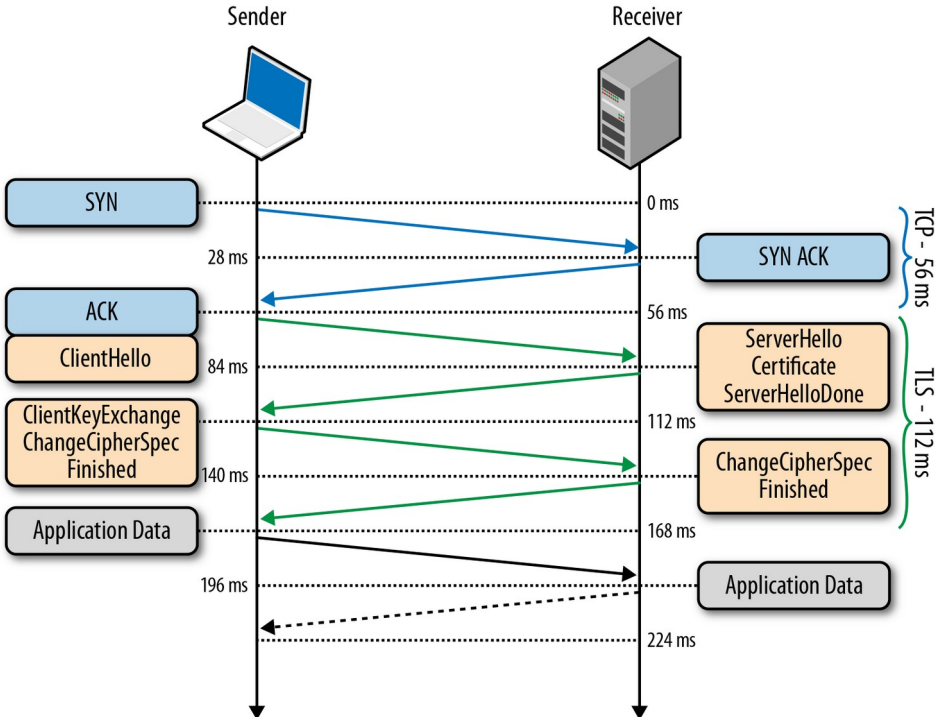
- Аутентификацию – клиент проверяет, что установил соединение с истинным сервером.
- Шифрование – защита от просмотра передаваемых данных третьей стороной.
- Целостность – защита передаваемых данных от искажения и подмены.

Сертификаты – основа работы TLS

- Сертификат – подписанный цифровой подписью файл, который содержит наборы ключей для симметричного и асимметричного шифрования.
- Сертификаты может выдавать и подписывать только ограниченный круг доверенных организаций (Setigo, GlobalSign, Comodo, ISRG) ■ Let's Encr t от ISRG: https://letsencrypt.org/ru/about/ ■ Self-signed сертификаты (подписанные самим собой) можно сгенерировать автоматически; не используются публично.
- Сертификат состоит из двух частей: ■ Публичной – передаётся клиенту при подключении к серверу; проверяется у доверенной компании, выдавшей и подписавшей сертификат (центр сертификации, англ. CA – Certificate Authority). ■ Приватной – секретный ключ, который «знает» только сервер; служит для идентификации сервера.
- Публичный и приватный ключи сертификата используются для реализации асимметричного шифрования.
- При установке TLS-соединения используется асимметричное шифрование, а при передаче данных – симметричное шифрование.

Соединение TLS

- Клиент устанавливает TCP-соединение с сервером.
- Клиент посылает на сервер спецификацию в виде обычного текста (версию протокола, которую он хочет использовать, поддерживаемые методы шифрования).
- Сервер утверждает версию используемого протокола, выбирает способ шифрования, и отправляет ответ клиенту вместе со своим публичным сертификатом.
- Клиент проверяет присланный сертификат и инициирует безопасный обмен ключами, используя асимметричное шифрование (открытый ключ сервера). В результате обмена клиент и сервер получают общий ключ для симметричного шифрования.
- Сервер обрабатывает присланное клиентом сообщение, расшифровывает его своим закрытым ключом, проверяет подпись, и отправляет клиенту заключительное сообщение, зашифрованное симметричным алгоритмом.
- Клиент расшифровывает полученное сообщение, проверяет подпись, и



если всё хорошо, то соединение считается установленным и начинается обмен данными приложений.

34. Сравнительный анализ классических и "одностраничных" веб-приложений.

Классическое веб-приложение:

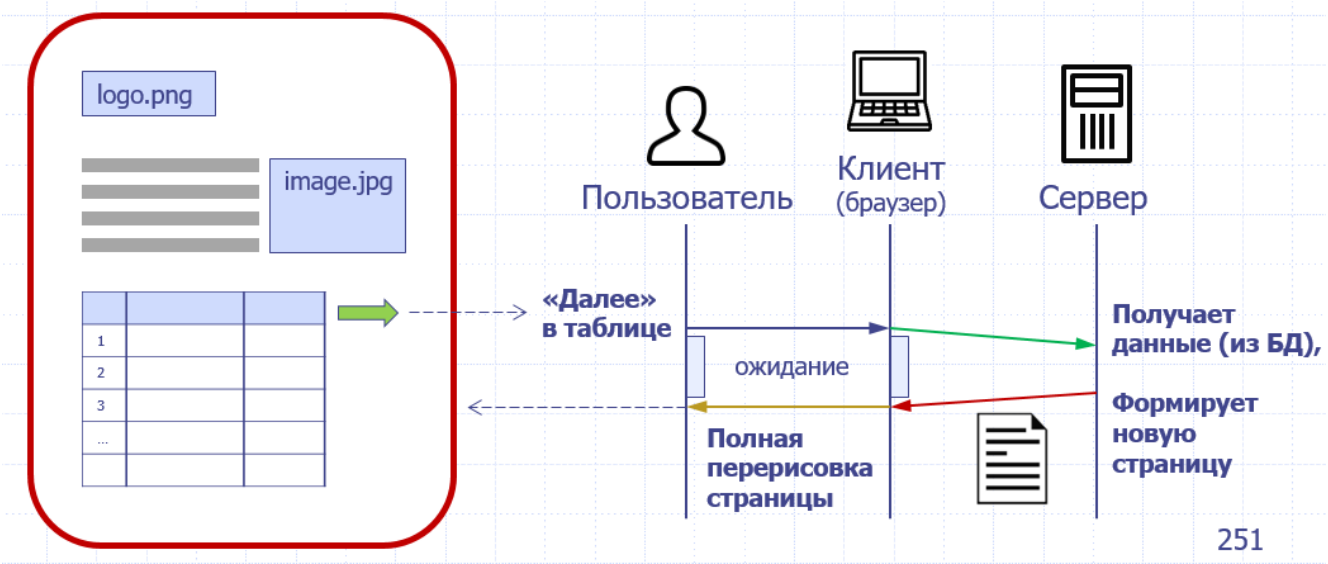
- Программная логика целиком реализуется на сервере с помощью любого языка программирования.
- Переход по ссылке внутри сайта приводит к HTTP-запросу и загрузке новой HTML-страницы с сервера.
- Данные приходят вместе с HTML-страницей.

Одностраничное веб-приложение:

- Программная логика целиком реализуется на клиенте с помощью языка JavaScript.
- Переход по ссылке внутри сайта приводит к видоизменению структуры HTML-страницы без необходимости выполнять HTTP-запрос к серверу.
- Данные приходят отдельно (обычно в формате JSON)

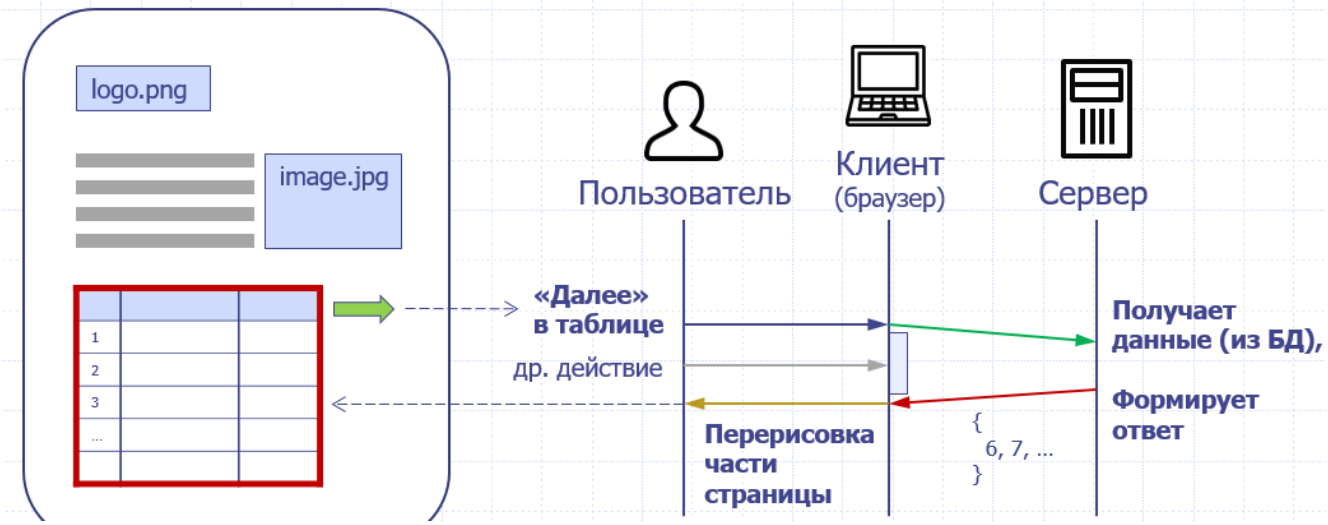
Классическое:

- Действие пользователя приводит к отправке запроса на сервер.
- Сервер формирует и отправляет новую страницу для отображения.
- Клиент (браузер) перерисовывает страницу целиком.
- Интерфейс пользователя блокируется до загрузки новой страницы.



Одностраничное:

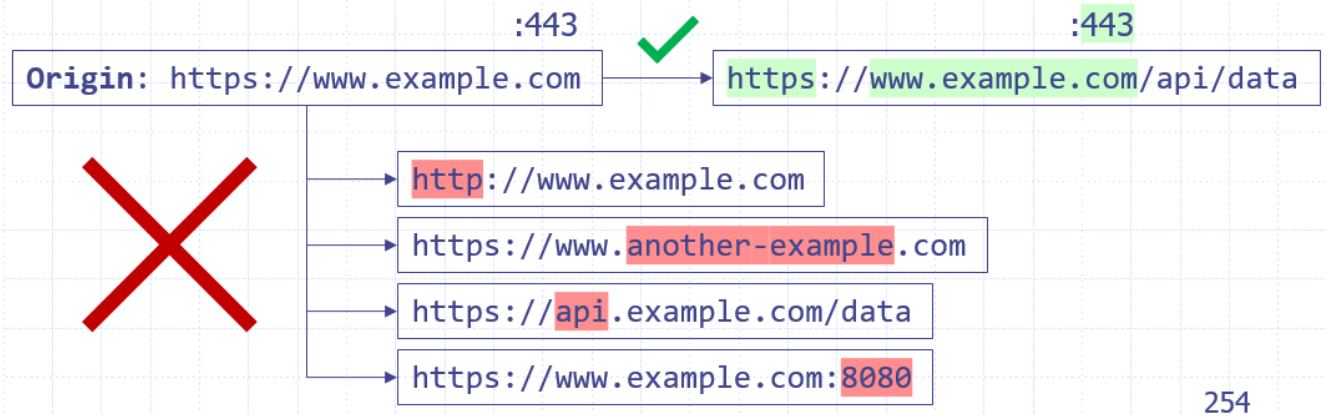
- Действие пользователя приводит к отправке запроса на сервер.
- Сервер передаёт в ответе только данные.
- Код на клиенте обновляет содержимое страницы, клиент (браузер) перерисовывает только часть страницы.
- Нет блокировки интерфейса пользователя.



35. Правило ограничения домена (принцип одного источника), совместное использование ресурсов между разными источниками (Cross-origin resource sharing - CORS).

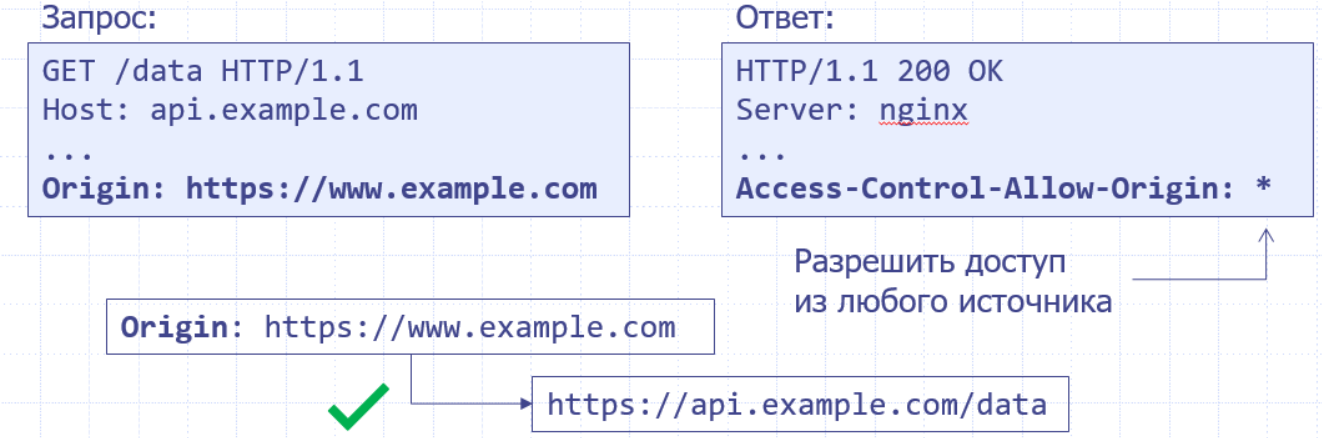
Правило ограничения домена

- Политика безопасности, по которой код на странице может получить доступ только к ресурсам из того же источника.
- Заголовок **Origin** устанавливается браузером (недоступен из кода).
- Источники считается одинаковыми, если у них совпадают протокол, сервер (домен и поддомены) и порт.
- Защищает веб-приложения от внедрения вредоносного кода и кражи cookies.



Совместное использование ресурсов между разными источниками

- Позволяет смягчить политику одинакового источника.
- Сервер устанавливает, из каких источников можно получить доступ к ресурсу, о чём сообщает в заголовке ответа Access-Control-Allow-Origin.



Предварительный запрос в CORS

Не нужен для «простых» запросов (методы GET, HEAD и POST без дополнительных заголовков).

Клиент посылает предварительный (англ. preflight) запрос OPTIONS, чтобы узнать, доступен ли ресурс для текущего источника.

Заголовки предварительного запроса, помимо Origin:

- Access-Control-Request-Method,
- Access-Control-Request-Headers.

Заголовки ответа:

- Access-Control-Allow-Origin – источник,
- Access-Control-Allow-Methods – методы,
- Access-Control-Allow-Headers – заголовки,
- Access-Control-Max-Age – сколько секунд актуальны результаты предварительного запроса (Methods, Headers).

36. Передача данных по протоколу HTTP в режиме приближенном к режиму реального времени. Опрос данных (Ajax Polling), долгий опрос данных (Comet Long Polling), передача потока данных (Comet Streaming). Веб-гнёзда (Web Sockets).

Постановка задачи:

Пользователь хочет отслеживать изменения на сервере. Пример: клиент, взаимодействующий со службой погоды и отображающий показания температуры, влажности, давления и пр. для заданного населённого пункта. **Подходы к реализации:**

- Pull-модели (клиент «вытаскивает» данные с сервера):
  - Ручное обновление
  - Опрос данных (AJAX polling)
- Push-модели (сервер «проталкивает» данные клиенту):
  - Долгий опрос данных (Comet long polling)
  - Передача потока данных (Comet streaming)
- Двусторонний обмен сообщениями
  - Веб-гнёзда (Web-Sockets)

1) Ручное обновление. Недостатки:

- Низкая частота обновлений, реальный масштаб времени отсутствует.
- Высокий трафик (полная перезагрузка страницы).
- Высокая нагрузка на сервер и сеть.

2) Опрос данных. Недостатки:

- Низкая частота обновлений, реальный масштаб времени отсутствует.
- Высокий трафик
  - частые запросы;
  - передача заголовков запроса и ответа.
- Высокая нагрузка на сервер и сеть.

Достоинства

- Отсутствует блокировка пользовательского интерфейса.

3) Долгий опрос данных. Недостатки:

- Средняя частота обновлений, близко к реальному масштабу времени.
- Средний трафик (передача заголовков запроса и ответа).
- Средняя нагрузка на сеть.

Достоинства

- Отсутствует блокировка пользовательского интерфейса.
- Низкая задержка для нечастых событий.

4) Потоковая передача данных. Недостатки:

- Может блокироваться на промежуточных узлах.

Достоинства:

- Высокая частота обновлений, низкая задержка, реальный масштаб времени.
- Низкий трафик (нет передачи заголовков запроса и ответа.)
- Низкая нагрузка на сеть.

Цель: полнодуплексная связь между клиентом и сервером, чтобы не только клиент мог отслеживать изменения на сервере, но и сервер мог отслеживать изменения на клиенте.

- TCP не подходит: проблемы безопасности, отсутствует CORS, порты блокируются сетевыми экранами.
- HTTP не подходит: диалоговый режим, не гарантирована поддержка соединения, не гарантирован порядок сообщений.

Создаётся HTTP-запросом со специальными заголовками, который переводит HTTP-соединение в режим Веб-гнезда.

Клиент и сервер могут посылать данные друг другу асинхронно.

Ориентированы на передачу отдельных сообщений (а не потока данных, как протокол TCP).

Обеспечивает фрагментацию с помощью кадров для передачи сообщений с неизвестной заранее длинной и мультиплексирования канала связи.

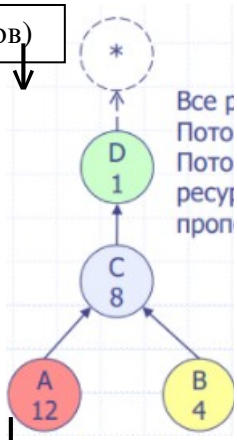
Сообщения от клиента серверу маскируются операцией XOR со случайным ключом для исключения «отравления кэша» на прокси-серверах. Сообщения от сервера клиенту не маскируются.

37. Проблемы протокола HTTP/1.1. Протокол HTTP/2.

Проблемы протокола HTTP/1.1

- Загрузка типичной веб-страницы в среднем требует совершить ~100 HTTP-запросов и передать ~2 МБ данных.
- Рост пропускной способности сетей не сопровождается уменьшением задержки при передаче.
- Производительность падает из-за высокой задержки при выполнении HTTP-запросов. Чем больше запросов, тем больше общая задержка и время загрузки страницы.
- В HTTP/1.1 ограничено число параллельных соединений одного клиента с одним сервером, что ограничивает скорость загрузки страницы.
- HTTP/1.1 работает поверх TCP, но не использует его возможности в полной мере. В частности, заголовок Content-Length делает невозможным прерывание передачи данных по HTTP-соединению без разрыва и повторной установки TCP-соединения.
- Конвейеризация запросов работает плохо: длительный запрос в начале очереди блокирует последующие короткие запросы.

D->C->A:B(3:1)(разделение ресурсов)



Протокол HTTP/2

Совместим с HTTP/1.1 по методам, кодам ответа, формату идентификатора ресурса (URI) и большинству заголовков.

HTTP/2 – бинарный протокол, разбор запросов и ответов ускоряется.

Бинарный формат, ориентированный на передачу сообщений, состоящих из кадров, в рамках потока данных.

Мультиплексирование потоков данных в рамках одного соединения – вместо конвейеризации запросов.

- Передача нескольких потоков в рамках одного TCP-соединения.
- Позволяет чередовать несколько запросов/ответов параллельно.
- Решает проблему блокировки очереди.

Приоритеты и зависимости между потоками данных. (Порядок чередования и доставки кадров клиентом/сервером влияет на производительность. Каждый поток может иметь вес от 1 до 256 и может зависеть от другого потока. Вес описывает важность потоков с точки зрения клиента и может служить указанием серверу, в какой пропорции выделять ресурсы на обработкую)

Сжатие заголовков методом HPACK – ускоряет серию запросов к одному серверу.

Возможность прервать передачу данных без разрыва соединения.

Посылка от сервера (server push) – сервер может самостоятельно отправить клиенту некоторые данные без запроса со стороны клиента.

Шифрование TLS 1.2+ не обязательно по стандарту, но требуется всеми реализациями.

Потоки, сообщения и кадры в HTTP/2

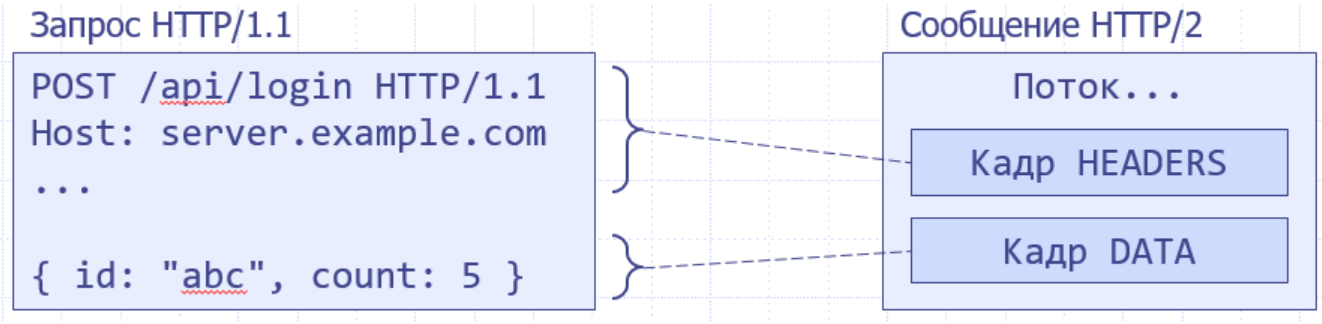
Поток – двунаправленный поток байтов в установленном соединении, который может нести одно или несколько сообщений.

Сообщение – полная последовательность кадров, которая соответствует логическому запросу или ответу.

Кадр – наименьшая единица связи в HTTP/2, которая содержит тип, длину, флаги, идентификатор потока и полезную нагрузку.

Типы кадров:

- HEADERS – соответствует заголовкам запроса HTTP/1.1,
- DATA – соответствует телу запроса HTTP/1.1,
- PRIORITY – задаёт приоритет потока,
- RST_STREAM – позволяет прервать поток, и др.


Запрос HTTP/1.1
POST /api/login HTTP/1.1
Host: server.example.com
...
{ id: "abc", count: 5 }
Сообщение HTTP/2
Поток...
Кадр HEADERS
Кадр DATA

38. Сравнительный анализ технологий распределённых вычислений: RPC, CORBA, веб-службы.

RPC (вызов удалённых процедур, англ. Remote Procedure Call)

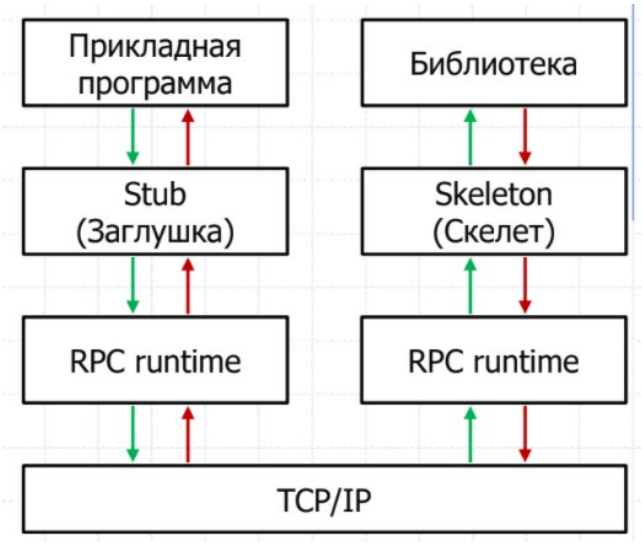
- зависимость от языка программирования;
- бинарный протокол, зависит от архитектуры и ОС;
- не учитывает распределённую среду и связанные с ней задержки и ошибки.

CORBA (запросы к удалённым объектам, англ. Common Object Request Broker Architecture)

- ссылки на удалённые объекты => чрезмерная сложность;
- недостатки RPC.

Веб-службы (Web Services)

- возможность взаимодействия в разнородной среде (архитектуры, ОС, ЯП);
- учитывает распределённую среду и связанные с ней задержки и ошибки;
- текстовый протокол SOAP.



RPC. Шаги:

- На языке IDL описывается интерфейс библиотеки.
- Компилятором IDL генерируются заглушка (stub) библиотеки для клиента и скелет (skeleton) сервера.
- Заглушка заменяет собой библиотеку и отвечает за установку соединения со скелетом сервера, отправку информации о вызываемой функции, сериализацию и отправку входных данных, а также десериализацию результирующих данных.
- Скелет сервера отвечает за прослушивание входящих запросов от заглушек клиентов, приём информации о вызываемой функции, десериализацию входных данных, вызов библиотеки, сериализацию и отправку результирующих данных.

Язык описания интерфейсов IDL (англ. Interface Definition Language)

COBRA

Ссылки на удалённые объекты, которые можно передавать между клиентами и другими службами. IDL с поддержкой ООП: создание удалённых объектов, вызов методов, изменение значений полей.

Проблемы:

Отдельный вызов по сети на каждом обращении к объекту. Для эффективной реализации требуется изменить интерфейс (в примере: установка всех
полей адреса одной функцией).

Все накладные расходы вызова по сети неявные.

Возможны ошибки временной недоступности сервера.

Ошибочным является сам подход с сокрытием факта взаимодействия по сети при выполнении вызовов функций и обращениям к удалённым объектам.

Веб-служба

Веб-служба (сетевой сервис) – один или несколько взаимодействующих логических серверов, обеспечивающих доступ к конкретному типу ресурса с помощью веб-протоколов (HTTP и базирующихся на нём).

Архитектура, ориентированная на службы (SOA – Service Oriented Architecture).

SOAP (Simple Object Access Protocol) – открытый стандарт на основе XML для передачи данных между клиентами и серверами веб-служб. (В протоколе HTTP используется только метод POST, который не кэшируется и позволяет передать данные в запросе и в ответе.)

WSDL (Web Services Description Language) – открытый формат на основе XML для описания интерфейсов (контрактов) веб-служб.

HTTP REST – архитектурный стиль построения веб-служб, использующий всю инфраструктуру HTTP (все методы и заголовки). В качестве формата передачи данных обычно используется JSON или XML.

39. Принципы архитектуры SOA, ориентированной на веб-службы. Протокол SOAP и формат WSDL.

Совместимость (англ. Interoperability)

- Стандартный протокол (HTTP), стандартный формат передачи данных (XML, JSON).
- Контракт вместо программного интерфейса. Контракт – правила взаимодействия клиента со службой, описывающие доступные операции, типы данных, входные и выходные параметры операций. Контракт имеет стандартный формат (WSDL).
- Возможность взаимодействия клиента и сервера, написанных на разных языках и разных платформах.

Автономность

- Устойчивость к (временной) недоступности зависимостей.
- Библиотека не способна работать в отсутствие других библиотек, от которых она зависит. Служба должна быть способна работать автономно – в отсутствие других служб, от которых она зависит.

Мобильность

- Отделение функциональности от хостинга, а хостинга от конфигурации (протоколов, форматов и точек доступа к службе).
- Позволяет переносить службу с одного сервера/адреса на другой без перекомпиляции.

Оговоренный жизненный цикл и многоверсионность

- В библиотеках версия кодируется четырьмя числами: «major.minor.patch.build». Версия службы определяется датой выпуска (год, месяц, день).
- Для службы оговаривается срок жизни каждой версии.
- Переход на новую версию предполагает одновременную работу старой и новой версии службы с заданным сроком миграции зависимых от неё служб и клиентов.

Самодокументируемость

- Служба должна предоставлять всю необходимую информацию для её использования. Для этого создаются специальные точки доступа к службе.
- Обычно службы предоставляют информацию о контракте (WSDL-схему).

Обнаруживаемость

- Наличие реестра и стандартных способов обнаружения службы для её использования.

Готовность служб к объединению в общую среду взаимодействия (шину)

- Маршрутизация сообщений между службами, очереди сообщений.
- Мониторинг, аудит, протоколирование.
- Управление развёртыванием служб.

40. Технология WCF, пример SOAP-службы и клиента этой службы. Управление поведением WCF-службы при создании её экземпляров и обработке параллельных запросов.

Windows Communication Foundation (WCF) — это платформа для создания приложений, ориентированных на службы. С помощью WCF можно передавать данные в виде асинхронных сообщений из одной конечной точки службы в другую.

Режимы создания экземпляра (InstanceContextMode):

1. Single Создается один экземпляр для всех клиентов, при этом он создается только когда приходит первый вызов службы. Если вызовов нет, то и объект экземпляра не создается.
2. PerCall Экземпляр создается на вызов каждого метода. При любом вызове не просто объекта, а даже любого его метода, создается отдельный экземпляр объекта. Позволяет не беспокоиться о синхронизации при режиме параллельного вызова Multiple.
3. PerSession Экземпляр объекта создается на одну сессию клиента. Когда клиент создает объект WeatherServiceClient и делает первый вызов, создается экземпляр объекта, т.е. когда клиент только создал объект, экземпляр класса с функционалом службы на серверной стороне еще не создается. WeatherService на стороне службы будет уничтожен, когда клиент вызовет метод Close объекта WeatherServiceClient.

Режимы параллельного вызова(ConcurrencyMode):

1. Single Все запросы обслуживаются одним потоком последовательно. Не возникает проблемы синхронизации нескольких клиентов, вызывающих службу: они образуют очередь, их запросы последовательно обрабатываются службой.
2. Multiple Режим параллельной обработки входящих запросов от всех клиентов с помощью пула потоков. Все эти потоки для клиентов берутся из пула. Чтобы не беспокоиться о синхронизации, можно использовать этот режим совместно с режимом создания экземпляра PerCall.
3. Reentrant Предполагает вызов в одном потоке (аналог Single), но в этом режиме допустимы рекурсивные вызовы. Пришедший рекурсивный запрос обрабатывается тем же потоком и возвращается назад. Этот режим более сложно устроен, т.к. требует сохранения контекста, чтобы понимать, когда произошел рекурсивный вызов, и не ставить его в очередь. Поэтому этот режим не используется по умолчанию.

41. Архитектурный стиль HTTP REST и службы на его основе.

HTTP REST – архитектурный стиль построения веб-служб, использующий всю инфраструктуру HTTP (все методы и заголовки). В качестве формата передачи данных обычно используется JSON или XML.

REST (англ. Representational State Transfer – «передача состояния представления»)

Основные принципы:

- Строгое разделение клиента и сервера.
- Сервер не хранит клиентское состояние представления, все необходимые данные передаются в запросе.
- Возможность кэширования на стороне клиента.
- Единообразие интерфейса.
- Возможность добавления промежуточных слоёв между клиентом и сервером (балансировка нагрузки, шифрование, кэширование и др.).

Позволяет строить производительные, масштабируемые, переносимые приложения (службы) с унифицированным интерфейсом.

Пример интерфейса службы HTTP REST

- RESTful веб-служба для работы с личным блогом
  - Авторизация – HTTP-заголовок `Authorization: Bearer <jwt>`

Действие	Метод	URI	Тело
Добавить новую запись	POST	/my/posts	{ title, text }
Получить список своих записей	GET	/my/posts?page={n}	-
Получить представление записи #id	GET	/my/posts/{id}	-
Обновить содержимое записи #id	PUT	/my/posts/{id}	{ title, text }
Удалить запись #id	DELETE	/my/posts/{id}	-
Получить список записей другого пользователя	GET	/{user-id}/posts	-
Отметить запись другого пользователя	POST	/{user-id}/posts/{post-id}/like	-

Отличия веб-служб на основе REST и SOAP

1) в REST есть некоторый ресурс, к которому можно применять стандартные операции. Путь в строке HTTP – путь к самому ресурсу и его идентификация. В веб-службах SOAP путь дает доступ к самой веб-службе, ее контракту;

2) В REST используется не SOAP (не XML), а HTTP;

3) SOAP полностью базируется на методе POST, который является небезопасным и не идемпотентным, HTTP REST основан на использовании всей инфраструктуры HTTP с возможностью кэширования и выполнения идемпотентных кэшируемых запросов, поэтому более безопасных GET. REST позволяет использовать заголовки управления HTTP – if-match, if-none-match, управление кэшированием

Для REST-служб нужно использовать не ServiceHost, а WebServiceHost. WebServiceHost отличается тем, что обеспечивает возможность создания точки доступа для прослушивания конкретного HTTP адреса.

42. Рассеянные (облачные) вычисления (cloud computing). Основные понятия, принципы организации и функционирования. Понятия IaaS, PaaS, SaaS.

Облачные технологии – это технологии распределённой обработки и хранения данных, которые абстрагируют место и способ хранения данных, платформу и особенности выполнения распределённых вычислений.

Центр (хранения и) обработки данных (сокр. ЦОД/ЦХОД).

Виртуализация как основа облачных вычислений.

Суть этой технологии состоит в том, что на компьютере выполняется операционная система, называемая хост-операционной системой (supervisor), под управлением которой выполняются другие операционные системы. То есть на одном компьютере можно запустить несколько операционных систем одновременно. При этом и операционные системы, и программы будут полноценно работать.

1) Можно создать датацентр, который будет состоять из дорогих и мощных компьютеров (многопроцессорных). На них установить операционную систему Supervisor и запускать под ее управлением клиентские операционные системы. Это позволяет масштабировать операционные системы, то есть поддерживать на одном многопроцессорном компьютере одновременную работу нескольких систем;

2) Виртуальный узел легко перемещать с одного физического узла на другой. Если выполняются backup-копии (snapshot'ы, т.е. снимок состояния работы в виде файла), то можно запустить эту копию на другом физическом узле. Это дает то, что не дает отсутствие виртуализации в случае сбоя или отказа оборудования – устойчивость работы;

3) Распределение мощностей.

Виды облачных моделей:

- Infrastructure as a Service (IaaS) – инфраструктура как услуга;

Пример –служба Amazon Web Services. Такого рода компании предоставляют возможность создать некоторое количество виртуальных машин, объединенных в локальную сеть. То есть инфраструктура предприятия развертывается не на самом предприятии, а у службы Amazon или кого-либо иного поставщика.

- Platform as a Service (PaaS) – платформа как услуга;

Когда появились службы, возникла необходимость устанавливать на компьютеры стандартные библиотеки, окружения, фреймворки (например, .Net), т.е. некоторый набор библиотек, позволяющих выполнять какие-то программы, а также разрабатывать свои приложения и запускать их на платформе.

Платформа предоставляет среду выполнения программ. Классический пример – Windows Azure, force.com.

PaaS также предоставляет средства коммуникаций между программами, включая средства для работы с базами данных, файловой системой, очередями, системами массового обслуживания. Например, в рамках Windows Azure предоставляется работа с MSSQL.

- Software as a Service (SaaS) – приложение как услуга.

Смысл состоит в том, что сама служба «виртуализует». Например, виртуализация службы электронной почты: организация может зарегистрироваться, получить сервер для хранения корреспонденции, почтовые ящики для своих сотрудников и работать с этой почтовой службой. Аналогичную возможность эта служба предоставляет многим организациям, которые логически обращаются к одному поставщику услуг, физически это работает на одних и тех же серверах, на одной и той же платформе, но внутри масштабируется. С точки зрения каждого заказчика, служба виртуализована, она работает только в его личных интересах. Получается, что на этом уровне не только платформа общая, здесь общая инфраструктура создания на уровне программы, на более высоком уровне и разделение идет на уровне заказчика. Это приводит к появлению понятия multi-tenant (множественная аренда).

Пример: Microsoft Outlook

43. Понятие масштабируемой распределённой системы. Вертикальная и горизонтальная масштабируемость.

Масштабируемость – способность системы справляться с увеличением рабочей нагрузки путём добавления ресурсов.

Виды масштабирования:

- Вертикальное масштабирование – наращивание ресурсов одного узла (ядер процессоров, оперативной памяти, дискового пространства).
- Горизонтальное масштабирование – наращивание количества узлов в распределённой системе.

Горизонтальное масштабирование – это великолепный способ масштабирования, т.к. имеется возможность выбрать мощность одного узла и найти его оптимальную стоимость.

Главная проблема горизонтального масштабирования – это необходимость использования ПО, которое должно иметь архитектуру, поддерживающую горизонтальное масштабирование.

До определенного уровня вертикальное масштабирование лучше горизонтального, т.к.

1. Горизонтальное масштабирование требует специального ПО;
2. Увеличить количество компонентов существующего компьютера дешевле, чем купить новый дополнительный компьютер;
3. Энергопотребление системы ниже, т.к. у нее меньше энергопотребляющих компонентов;
4. Занимаемый объем помещения меньше.

Однако у вертикальной масштабируемости есть предел по количеству памяти, количеству процессоров. При использовании нескольких процессоров необходима модель общей памяти.

Варианты:

1. Обеспечить синхронизацию (когерентность) кэшей процессоров;
2. Применять подходы при программировании, обеспечивающие синхронизацию