1. Что такое сокет и какова его роль в сетевом программировании? +

2. Какие различия между протоколами TCP и UDP? +

3. Какие особенности работы сокетов без подключения (UDP) по сравнению с сокетами с подключением (TCP)? +

4. Что произойдёт, если в TCP-соединении не закрыть сокет корректно? +

5. В каких случаях целесообразно использовать TCP, а когда — UDP? +

6. Что такое IP-адрес и порт? Почему для создания сокетов необходимы оба параметра? +

7. Что такое "три рукопожатия" (three-way handshake) в TCP и зачем оно нужно? +

**Сокет** — это программный интерфейс, который обеспечивает двустороннюю связь между узлами в компьютерной сети. Это конечная точка для отправки или получения данных по сети. В сетевом программировании сокет играет роль в организации обмена информацией между приложениями, работающими на разных устройствах или на одном устройстве. Сокет состоит из комбинации IP-адреса и номера порта. IP-адрес идентифицирует конкретное устройство в сети, а порт указывает конкретное приложение или процесс на этом устройстве. Сокеты обеспечивают возможность отправки и получения данных между двумя устройствами в сети. Например, когда клиент хочет отправить запрос на сервер, он создает сокет, устанавливает соединение с сервером и передает данные. Сервер, в свою очередь, создает свой сокет для прослушивания входящих запросов и обработки их.

Если сокет в TCP-соединении не закрыть корректно, это может привести к различным проблемам:

**Утечка ресурсов:** открытые сокеты используют системные ресурсы (файловые дескрипторы, память и т.д.). Если сокет не закрыть корректно, эти ресурсы не будут освобождены. Со временем это может привести к исчерпанию доступных ресурсов системы, особенно если приложение работает долго или активно взаимодействует с сетью.

**Невозможность установить новое соединение:** если сокеты не закрываются корректно, то порты, которые использовались для соединений, могут оставаться занятыми (даже если физически соединение неактивно). Это может привести к исчерпанию доступных портов. В результате, при попытке установить новое соединение, могут возникнуть ошибки, так как все доступные порты будут заняты зависшими соединениями.

**Потеря данных:** если сокет закрывается некорректно (например, внезапное завершение без отправки финального сигнала FIN), то часть данных может быть потеряна. Это может привести к неполному или некорректному получению информации на стороне клиента или сервера.

**Неожиданное поведение приложений:** если одно из приложений не закрывает соединение правильно, другое приложение, участвующее в соединении, может остаться в "подвешенном" состоянии, ожидая завершения сеанса. Это может вызвать задержки, зависания или ошибки в работе программы. Клиент может ожидать данных от сервера, в то время как сервер уже завершил выполнение задачи и просто не закрыл соединение, что может вызвать таймауты или зависание на стороне клиента.

**IP-адрес и порт** — это ключевые компоненты, которые обеспечивают связь между устройствами в сети и помогают направлять данные к нужному получателю.

**IP (Internet Protocol) адрес** — это уникальный числовой идентификатор, который присваивается каждому устройству, подключённому к сети. Он необходим для того, чтобы определить местоположение устройства в сети, будь то локальная сеть или интернет.

**IP-адреса бывают двух версий:**

**IPv4:** представляет собой 32-битное число, обычно записываемое как четыре десятичных числа, разделённых точками (например, 192.168.1.1).

**IPv6:** представляет собой 128-битное число, записываемое в виде восьми групп шестнадцатеричных чисел, разделённых двоеточиями (например, 2001:0db8:85a3:0000:0000:8a2e:0370:7334).

**Функция IP-адреса** — указывать на конкретное устройство в сети, чтобы данные могли быть доставлены по нужному маршруту.

**Порт** — это числовой идентификатор (от 0 до 65535), который указывает на конкретное приложение или службу, работающие на устройстве. Каждый порт ассоциирован с определённым процессом или сервисом.

**Например:**

Порт 80 используется для HTTP (веб-сайты).

Порт 443 используется для HTTPS (безопасные веб-сайты).

Порт 22 используется для SSH (удалённое управление серверами).

**Функция порта** — позволяет различным процессам на одном устройстве обрабатывать различные типы данных. Если IP-адрес указывает на само устройство, то порт указывает на конкретное приложение или сервис внутри этого устройства.

**Почему для создания сокетов необходимы оба параметра?**

Для создания сокетов нужны как IP-адрес, так и порт, потому что:

IP-адрес определяет устройство, с которым нужно установить соединение. Например, в случае запроса веб-страницы IP-адрес указывает на сервер, где находится сайт.

Порт указывает на конкретное приложение или сервис на этом устройстве, который должен обработать запрос. Если бы существовал только IP-адрес, то сервер не знал бы, какому процессу на устройстве направить полученные данные.

**TCP — протокол транспортного уровня**

**TCP** – протокол транспортного уровня, используется для передачи сообщений между устройствами в сети. Файлы не передаются целиком, а дробятся и передаются в виде небольших сообщений. Они передаются другому устройству – получателю, где повторно собираются в файл.

**Например**, человек хочет скачать картинку. Сервер обрабатывает запрос и высылает в ответ требуемое изображение. Ему, в свою очередь, необходим путь или канал, по которому он будет передавать информацию. Поэтому сервер обращается к сетевому сокету для установки требуемого соединения и отправки картинки. Сервер дробит данные, инкапсулирует их в блоки, которые передаются на уровень TCP получателя при помощи IP-протокола. Далее получатель подтверждает факт передачи.

**Особенности TCP**

**Система нумерации сегментов.** TCP отслеживает передаваемые и принимаемые сегменты, присваивая номера каждому из них. Байтам данных, которые должны быть переданы, присваивается определенный номер байта, в то время как сегментам присваиваются порядковые номера.

**Управление потоком.** Функция ограничивает скорость, с которой отправитель передает данные. Это делается для обеспечения надежности доставки, в том числе чтобы компьютер не генерировал пакетов больше, чем может принять другое устройство. Если говорить простым языком, то получатель постоянно сообщает отправителю о том, какой объем данных может быть получен.

**Контроль ошибок.** Функция реализуется для повышения надежности путем проверки байтов на целостность.

**Контроль перегрузки сети.** Протокол TCP учитывает уровень перегрузки в сети, определяемый объемом данных, отправленных узлом. Протокол TCP в таких ситуациях замедляет скорость передачи данных, чтобы предотвратить ухудшение ситуации и минимизировать потерю пакетов.

**Примеры применения сетевого протокола TCP**

Протокол TCP гарантирует доставку, а также обеспечивает целостность данных, передаваемых в сети. Поэтому он применяется для передачи данных, которые чувствительны к нарушению целостности, — например, текстов, файлов и т.п. Вот несколько протоколов, которые работают по TCP:

* SSH, FTP, Telnet: в данных протоколах TCP используется для обмена файлами.
* SMTP, POP, IMAP: протоколы, где TCP отвечает за передачу сообщений электронной почты.
* HTTP/HTTPS: протоколы, где TCP отвечает за загрузку страниц из интернета.

Эти примеры работают на уровне приложений стека TCP/IP и передают данные вниз к TCP, на транспортный уровень.

**UDP — протокол транспортного уровня**

Если нам очень важна скорость передачи, а вот потеря пакетов не так критична (как, например, в голосовом или видеотрафике), то лучше использовать UDP, или User Datagram Protocol. В отличие от TCP **он обеспечивает передачу данных без получения подтверждения от пользователя.** **Проще говоря, просто отправляет пакеты и не ждет ничего в ответ. Из-за этого достигается высокая скорость в ущерб надежности.**

Чаще всего UDP применяется в чувствительных ко времени службах, где потерять пакеты лучше, чем ждать. Звонки в Skype или Google Meet, **стриминг видео,** **онлайн-трансляции** используют этот протокол из-за того, что **они чувствительны ко времени и рассчитаны на определенный уровень потерь.** **Вся голосовая связь через интернет работает по протоколу UDP.** Также UDP очень часто **используется в онлайн-играх**. Аналогичная история с DNS-серверами, поскольку они должны быть быстрыми и эффективными.

**Примеры использования протокола UDP**

Примерами протоколов, использующих UDP-протокол, являются:

* DNS — протокол, преобразующий домены в IP-адреса, чтобы сделать возможной загрузку интернет-ресурса через браузер.
* SNMP — протокол, позволяющий системному администратору проводить мониторинг, контролировать производительность сети и изменять конфигурацию подключенных устройств.
* DHCP — протокол, отвечающий за автоматическое назначение IP-адреса клиенту.

**Разница между TCP и UDP**

Ключевым различием между TCP и UDP является скорость, поскольку TCP сравнительно медленнее UDP. В целом, UDP является быстрым, простым и эффективным протоколом, однако повторная передача потерянных пакетов данных возможна только в TCP.

Еще одно заметное различие между TCP и UDP заключается в том, что первый обеспечивает упорядоченную доставку данных от пользователя к серверу (и наоборот). UDP, в свою очередь, не проверяет готовность получателя и может доставлять пакеты вразнобой.

**TCP vs UDP**

|  |  |  |
| --- | --- | --- |
|  | **TCP** | **UDP** |
| **Состояние соединения** | Требуется установленное соединение для передачи данных (соединение должно быть закрыто после завершения передачи) | Протокол без соединения, без требований к открытию, поддержанию или прерыванию соединения |
| **Гарантия доставки** | Может гарантировать доставку данных получателю | Не гарантирует доставку данных получателю |
| **Повторная передача данных** | Повторная передача нескольких кадров в случае потери одного из них | Отсутствие повторной передачи потерянных пакетов |
| **Проверка ошибок** | Полная проверка ошибок | Базовый механизм проверки ошибок. Использует вышестоящие протоколы для проверки целостности |
| **Метод передачи** | Данные считываются как поток байтов; сообщения передаются по границам сегментов | UDP-пакеты с определенными границами; отправляются по отдельности и проверяются на целостность по прибытии |
| **Сферы применения** | Используется для передачи сообщений электронной почты, HTML-страниц браузеров | Видеоконференции, потоковое вещание, DNS, VoIP, IPTV |

Каждый протокол хорош под свои задачи, недаром они являются одними из самых распространенных в интернете. В завершение сравнения TCP и UDP можно выделить, что TCP применяется там, где важно доставить все данные в определенном порядке. Зона применения UDP, в свою очередь, — это голосовой и видеотрафик, где доставка всех пакетов не является обязательной.

Также серьезным отличием TCP от UDP является размер заголовков. У TCP он составляет 20-60 байт, а у UDP — всего 8 байт. Это показывает, насколько сложнее устроен протокол TCP, ведь он приоритизирует трафик и проверяет блоки данных на наличие ошибок.

**Тройное рукопожатие**

**"Три рукопожатия" (three-way handshake)** — это процесс установления соединения в протоколе TCP (Transmission Control Protocol), который позволяет двум устройствам в сети согласовать параметры и гарантировать надежное и упорядоченное соединение. Этот процесс необходим для установления корректного соединения между клиентом и сервером, чтобы обе стороны могли передавать данные безопасно и без потерь.

**Этапы "трёх рукопожатий":**

**SYN (synchronize):**

Клиент инициирует соединение, отправляя серверу сегмент с флагом SYN (synchronize), указывая своё начальное порядковое число (Порядковое число помогает TCP отслеживать каждый кусочек данных (байт), который передаётся. Например, первый кусочек данных имеет порядковое число "100", второй — "101" и тд) для потока данных. Это сообщение сообщает серверу о готовности клиента начать обмен данными.

Порядковое число используется для отслеживания последовательности байтов, чтобы убедиться, что все данные передаются в правильном порядке.

**SYN-ACK (synchronize-acknowledge):**

Сервер, получив запрос клиента, отвечает, отправляя клиенту сегмент с двумя флагами: SYN и ACK (acknowledge). Сервер подтверждает получение первого сегмента, увеличивая номер последовательности на единицу (Ack = x + 1) и отправляя своё собственное начальное порядковое число (например, Seq = y). Это означает, что сервер согласен установить соединение и готов начать передачу данных.

**ACK (acknowledge):**

Клиент, получив ответ от сервера, отправляет завершающий сегмент с флагом ACK, подтверждая получение номера последовательности от сервера, увеличенного на единицу (Ack = y + 1). Этот сегмент завершает установление соединения, и с этого момента данные могут передаваться.

**Зачем нужно "три рукопожатия"?**

**Согласование параметров соединения:**

TCP — это надёжный протокол, который обеспечивает передачу данных без потерь и дублирования. Три рукопожатия позволяют сторонам согласовать начальные порядковые номера данных, что критически важно для контроля целостности и последовательности данных при передаче.

**Подтверждение готовности обеих сторон:**

Этот процесс обеспечивает, что и клиент, и сервер готовы к передаче данных. Если соединение не будет установлено корректно, данные могут быть потеряны или переданы в неправильной последовательности.

**Предотвращение ошибок и дублирования данных:**

Порядковые номера, согласованные во время рукопожатия, позволяют избежать ситуации, когда повторяющиеся или потерянные пакеты от предыдущих соединений могут быть ошибочно приняты текущим соединением. Это помогает TCP обеспечивать корректность данных.