Протоколы транспортного уровня. Формат заголовков протоколов транспортного уровня. TCP, UDP, QUIC.

Транспортный уровень моделей OSI, TCP/IP.

Транспортный уровень моделей OSI и TCP/IP одинаков как по функциям, так и по названию. Термин TCP/IP – это комбинация двух протоколов. **Протокол IP** функционирует на сетевом Уровне 3 модели OSI, он является **протоколом дейтаграммного типа** без предварительного соединения (connectionless), который обеспечивает доставку сообщения через сеть по возможности, т.е. доставку с наибольшими возможными усилиями **(best-effort delivery)**, но без гарантий, т.е. доставка не надежная. **Протокол управления передачей TCP** работает на транспортном Уровне 4 модели OSI и является протоколом, ориентированным на предварительное соединение (connection-oriented), что обеспечивает контроль потока и надежность доставки. Когда эти протоколы объединены, они обеспечивают более широкий объем услуг: малую задержку и высокую надежность. Всемирная сеть Интернет строится на основе набора (стека) протоколов TCP/IP.

Основной функцией транспортного уровня является транспортировка сообщений и управление потоком информации от источника до устройства назначения, с обеспечением надежности доставки. Контроль доставки сообщения из одного конца соединения до другого и надежность обеспечены целым рядом параметров, передаваемых в заголовках сегментов:

- номерами последовательности передаваемых сегментов данных,
- размером, так называемого, скользящего окна,
- квитированием, т.е. подтверждением приема сообщения.

Транспортный уровень устанавливает логическое соединение между двумя конечными точками сети. Протоколы транспортного уровня сегментируют данные, посланные приложениями верхнего уровня на передающей стороне, и повторно собирают (реассемблируют) из полученных сегментов целое сообщение на приемной стороне.

Таким образом, протоколы транспортного уровня:

– реализуют сегментацию данных и повторную сборку целого сообщения из полученных сегментов. Большинство сетей имеет ограничение на объем передаваемых сообщений. Поэтому Транспортный уровень делит большое сообщение уровня приложений на сегменты данных, размер которых соответствует требованиям **протокола единиц данных** Protocol Data Unit – **PDU** более низких уровней сетевой модели. Кроме того, если в процессе контроля обнаружится, что принятое сообщение содержит ошибку, то возникает необходимость повторной передачи всего большого сообщения. При обнаружении ошибки в одном из принятых сегментов только данный сегмент будет передан повторно. Сегменты могут быть направлены одному или многим узлам назначения;

– обеспечивают многочисленные одновременно протекающие процессы обмена данными. На каждом конечном узле сети может быть запущено много разных приложений. Множество одновременно протекающих процессов обмена данными верхнего уровня может быть мультиплексировано поверх одного логического транспортного соединения. Чтобы передавать потоки данных соответствующим приложениям, протокол транспортного уровня должен идентифицировать каждое приложение. В протоколах TCP и UDP в качестве **идентификатора приложения используют номер порта**. Номер порта в заголовке сегмента транспортного уровня указывает, какое приложение создало передаваемое сообщение, и какое должно обрабатывать полученные данные на приемной стороне. При множестве одновременно протекающих обменах данными каждому из приложений или услуг назначается свой **адрес (номер порта)** так, чтобы транспортный уровень мог определить, с каким конкретно приложением или службой передаваемые данные должны взаимодействовать.

Наиболее известными протоколами транспортного уровня являются **протокол контроля передачи** (Transmission Control Protocol – **TCP**) и **протокол дейтаграмм пользователя** (User Datagram Protocol – **UDP**). Кроме того, на транспортном уровне функционирует **протокол надежной доставки**

(Reliable Transport Protocol – **RTP**), взаимодействие которого с протоколом EIGRP.

Протокол контроля передачи TCP является ориентированным на предварительное соединение (connection-oriented). Помимо деления сообщения на сегменты и идентификации приложений TCP обеспечивает контроль потока и надежность. Он взаимодействует с протоколами прикладного уровня: HTTP, SMTP, FTP, Telnet и другими. Протокол UDP является протоколом дейтаграммного типа connectionless, взаимодействует с такими протоколами прикладного уровня, как система доменных имен – DNS, передачи потока видеоданных – Video Steaming, голос поверх IP – Voice over IP и рядом других. Следует отметить, что система DNS взаимодействует как с TCP, так и с UDP.

Итак, протокол транспортного уровня TCP помимо деления сообщения на сегменты и идентификации приложений обеспечивает:

1. Контроль потока.

2. Надежность доставки сообщения.

Для облегчения контроля и обеспечения надежности сообщения передаются частями (порциями), т.е. сегментами. При этом протокол транспортного уровня узла источника должен прослеживать каждый сегмент данных при передаче и повторно передавать любую часть сообщения, прием которой не был подтвержден устройством назначения. Транспортный уровень конечного узла на приемной стороне должен отследить получение данных и подтвердить это получение.

Контроль потока необходим, чтобы гарантировать, что источник, передавая данные с некоторой скоростью, не переполняет буферные устройства узла назначения. Если узел назначения не может обрабатывать данные в темпе их поступления, то может произойти переполнение буферов и потеря данных. Управление скоростью передачи данных обеспечивается изменением **размера окна (Window Size)**, который указывает, сколько байт данных должно быть передано за одну порцию. При переполнении буферных

устройств узел назначения посылает источнику требование уменьшения размера окна.

После получения каждой порции данных узел назначения посылает источнику **подтверждение принятых данных или подтверждение доставки (acknowledgment)**.

Подтверждение (квитирование) обеспечивает **надежность** сети передачи данных. Если подтверждение не получено, то неподтвержденная порция данных передается узлом источником повторно.

В дейтаграммных IP-сетях пакеты одного сообщения между двумя конечными устройствами могут проходить разными путями. Поэтому на узел назначения сегменты могут прийти не в том порядке, в котором были переданы. Надежный протокол транспортного уровня (TCP) должен восстановить правильный порядок сегментов и собрать переданное сообщение (реассемблировать его).

Адресация приложений, надежность, контроль потока, сегментация сообщений и их реассемблирование, реализуются путем задания ряда параметров в заголовке сегмента TCP (рис.1), размер которого 20 байт.

Рисунок 1. Формат заголовка сегмента TCP

Поля заголовка TCP сегмента определяют следующее:

– **Номер порта источника** (Source Port) – 16 бит номера порта, который посылает данные;

– **Номер порта назначения** (Destination Port) – 16 бит номера порта, который принимает данные;

– **Номер последовательности** (Sequence Number) – 32 бита номера первого байта в сегменте, используемого, чтобы гарантировать объединение частей (порций) данных в корректном порядке в устройстве назначения;

– **Номер подтверждения** (Acknowledgment Number) – 32 бита последовательного номера подтверждения принятых данных, (начальный номер байта следующей ожидаемой порции данных);

– **ДЗ** – длина заголовка (число 32-разрядных слов в заголовке);

– **Резерв** – разряды поля, установленные в ноль;

– **Код** – 6 разрядов, определяющих тип сегмента, например, сегмент установки соединения (SYN) и завершения сеанса (FIN), сегмент подтверждения принятых данных (ACK), срочного сообщения (URG);

– **Размер окна** (Window Size) – число байтов, передаваемых за одну порцию;

– **Контрольная сумма** (Checksum) – значение контрольной суммы заголовка и поля данных;

– **Индикатор** (Urgent pointer) – индицирует конец срочных данных;

– **Опции** (Option) – каждая текущая опция определяет максимальный размер TCP сегмента;

– **Данные** (Data) – сообщение протокола верхнего уровня.

Поскольку UDP является протоколом дейтаграммного типа, то в заголовке его сегмента (рис.2.) отсутствуют такие параметры, как Номер последовательности, Номер подтверждения, Размер окна.

Рисунок 2. Формат сегмента UDP

Поля UDP сегмента определяют следующее:

– **Номер порта источника** (Source Port) – 16 бит номера порта, который посылает данные;

– **Номер порта назначения** (Destination Port) – 16 бит номера порта, который принимает данные;

– **Длина** (Length) – число байтов в заголовке и в поле данных;

– **Контрольная сумма** (Checksum) – контрольная сумма заголовка и поля данных;

– **Данные** (Data) – сообщение протокола верхнего уровня.

Поскольку протокол UDP не обладает механизмами надежности, то она обеспечивается протоколами верхнего уровня приложений. Однако небольшой размер заголовка UDP и отсутствие дополнительной обработки номера последовательности, размера окна и пересылки подтверждения получения данных повышают скорость обработки и передачи сообщений по сравнению с протоколом TCP.

Комбинация номера порта и IP-адреса образует комплексный адрес, так называемый **сокет (socket address)**, который определяет не только уникальное устройство, но и программное обеспечение, используемое для создания и обработки сообщения, например, 192.168.10.17:1275; 10.1.10.6:53.

Номера портов делятся на несколько типов:

– **известные номера** (Well Known Ports), диапазон адресов которых находится в пределах от 0 до 1023;

– **зарегистрированные** порты с номерами от 1024 до 49151;

– **динамические** порты с номерами от 49151 до 65535, которые обычно динамически присваиваются пользователям.

Номера известных портов заданы организацией Internet Assigned Numbers Authority **(IANA)**, распределяющей адреса в Интернете. Номера известных портов назначаются протоколам и службам сервиса уровня приложений. Номера некоторых известных портов протокола TCP приведены на рис.3.

Номера известных портов

Протоколы	FTP	Telnet	SMTP	HTTP	HTTPS	POP3
Порты	20, 21	23	25	80	443	110

Рисунок 3. Номера известных портов

В приложении протокола передачи файлов FTP используются два известных (стандартных) номера порта 20 и 21. Порт 20 используется для передачи данных, а порт 21 – для управления соединением.

Среди номеров известных портов протокола UDP наиболее распространенными являются: протокол TFTP – 69, RIP – 520.

Служба DNS с номером порта 53 и простой протокол управления сетью (Simple Network Management Protocol – **SNMP**) с номером порта 161 взаимодействуют как с протоколом TCP, так и с UDP.

Зарегистрированные порты назначаются как пользователям, так и приложениям. Когда зарегистрированные порты не используются для ресурсов сервера, они могут быть использованы динамически клиентом как номер порта источника. Из зарегистрированных портов можно отметить альтернативные порты протокола HTTP – 8008 и 8080.

Заголовок TCP сегмента (рис.3.) содержит последовательный номер (Sequence Number), используемый, чтобы гарантировать объединение частей (сегментов) сообщения в том порядке, в котором они были переданы. Протокол UDP не имеет такого механизма, поэтому возможны ошибки при объединении сегментов данных при передаче по сложной сети. Однако скорость передачи данных с использованием протокола UDP выше, чем TCP.

Если необходимо узнать, какие TCP соединения активны на сетевом конечном узле, то можно использовать команду **netstat** в режиме командной строки. В распечатке команды (рис.4.) указаны: протокол (TCP), локальные адреса узлов с динамически назначенными номерами порта, внешние адреса (или имена) узлов назначения с номером порта, а также состояние связи.

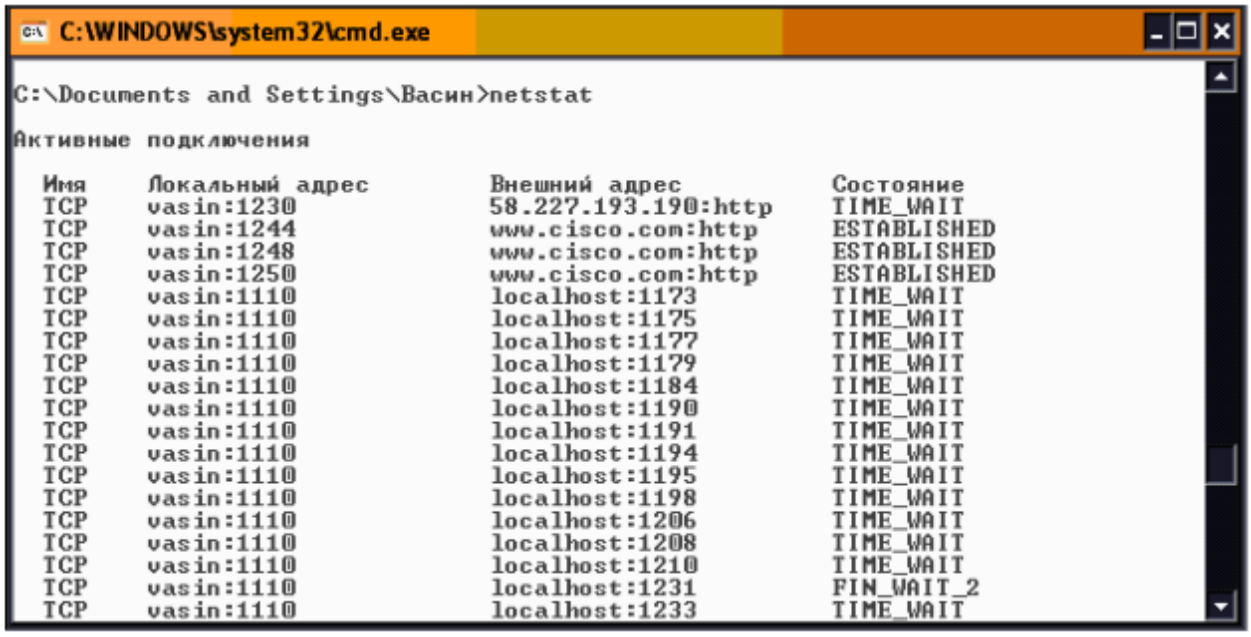

Рисунок 4. Результат выполнения команды netstat

В данном примере номер порта локального адреса является динамически назначаемым зарегистрированным портом источника с номером больше 1023. Для адреса www.cisco.com внешний порт задан символически: http. Состояние связи может быть с установленным соединением (ESTABLISHED) или с ожиданием окончания соединения (TIME_WAIT), когда был послан запрос окончания соединения (FIN).

Протокол QUIC. Формат заголовка QUIC.

Протокол QUIC – новый транспортный протокол, предназначенный для обеспечения соединения с низкой задержкой через Интернет. Новая технология построена на основе протокола UDP (что напрямую отражено в названии - Quick UDP Internet Connections), поэтому с её помощью можно передавать данных без необходимости в выделенном сквозном соединении.

QUIC был разработан компанией Google для решения проблем основного транспортного протокола TCP (Transmission Control Protocol), который широко используется в интернете, однако имеет недостатки, среди которых – высокий уровень задержек и проблемы с контролем перегрузок, который могут привести к проблемам с производительностью.

Для решения этих проблем QUIC предоставляет ряд ключевых функций, которые повышают производительность и безопасность, среди них – мультипоточность, приоритезация, управление перегрузками и сквозное шифрование.

Решение проблем TCP в QUIC

До недавнего времени TCP являлся основным транспортным протоколом в Интернете, который отвечает за обеспечение надёжного соединения между устройствами, однако TCP имеет ряд недостатков, которые так или иначе решены в QUIC

- **Миграция соединения**

В QUIC соединения идентифицируются с помощью 64-битного ID, который можно продолжать использовать даже после смены IP адреса пользователем. Из-за этого при смене IP адреса пользователя не возникает разрыва соединения, которое случается при смене на TCP. Также важно отметить, что на практике это довольно часто возникающая ситуация, особенно для смартфонов.

- **Оптимизированный ACK**

В QUIC каждый пакет имеет свой уникальный последовательный номер, поэтому не возникает проблемы различия пакетов при их ретрансмите. В QUIC поддерживается до 256 диапазонов NACK, помогая отправителю быть более устойчивым к перестановке пакетов и использовать меньше байтов в процессе. В TCP используется выборочный SACK, который решает проблему не во всех случаях.

- **Восполнение потерь**

В QUIC вызывается два TLP до того, как сработает Retransmission TimeOut (RTO), что отличается от реализации в TCP. Это позволяет быстрее обрабатывать восполнение хвостовых задержек, особенно для коротких и чувствительных к задержкам передач.

- **Управление перегрузкой**

QUIC находится в модели OSI на уровне приложений, позволяя проще обновлять главный алгоритм транспорта, который управляет отправкой, основываясь на параметрах сети. Большинство TCP-реализаций используют алгоритм CUBIC, который не оптимален для трафика, чувствительного к задержкам. Недавно разработанные алгоритмы вроде BBR, более точно моделируют сеть и оптимизируют задержки. QUIC позволяет использовать BBR и обновлять этот алгоритм по мере его совершенствования.

- **Многопоточность**

QUIC позволяет передавать несколько потоков данных через одно соединение, снижая расходы на создание и поддержание отдельных связей для каждого потока. В свою очередь TCP ничего подобного не имеет, поэтому для передачи данных несколькими потоками при работе с ним приходится открывать несколько различных соединений.

- **Приоритезация потоков**

QUIC позволяет устанавливать приоритеты потоков в зависимости от важности передаваемых данных, что позволяет ускорять процесс.

- **Шифрование**

QUIC по умолчанию обеспечивает сквозное шифрование, помогая защитить передачу данных. Для шифрования в QUIC используется TLS 1.3, который устанавливает ключи сессии, а после этого шифрует каждый пакет информации. В TCP также используется TLS, но из-за этого появляется необходимость в дополнительном рукопожатии между клиентом и сервером. В QUIC же этого не происходит, поскольку он соединяет собственное рукопожатие с RTT от TLS. Также в QUIC заменён записывающий слой TLS на собственный формат, но при этом сохраняется полная совместимость, и при этом ещё передаются дополнительный метаданные с целью защиты от манипуляций соединением через firewall и proxy.

Дополнительно в QUIC

- **QPACK**

В HTTP второй версии были представлены сжатые заголовки - HPACK, которые позволяет конечным хостам сокращать количество передаваемой информации. В частности, HPACK имеют динамические таблицы, заполненные предыдущими HTTP запросами и ответами. Это позволяет хостами обращаться к прошлой информации без необходимости запрашивать её снова. Используя TCP, HTTP запросы и ответы отправляются по порядку их пришествия, последовательно. А QUIC при помощи мультипоточности может отправлять их во множественном количестве одновременно, но тогда не гарантируется последовательность отправки пакетов.

- **QUIC заголовки**

Заголовки в QUIC бывают двух типов: длинные и короткие. Различают их по объёму передаваемой информации, в длинных её хранится больше. Длинные заголовки хранят в себе информацию о версии, ID адресата и отправителя и различные флаги, как например, форма заголовка. В самом частом сценарии длинные заголовки используются для первичного рукопожатия, но как только соединение установлено, отправитель начинает отправлять короткие пакеты, которые являются наиболее часто используемой опцией в трафике QUIC. В коротких пакетах хранится информации об ID адресата, номере самого пакета и зашифрованная информация пакета.

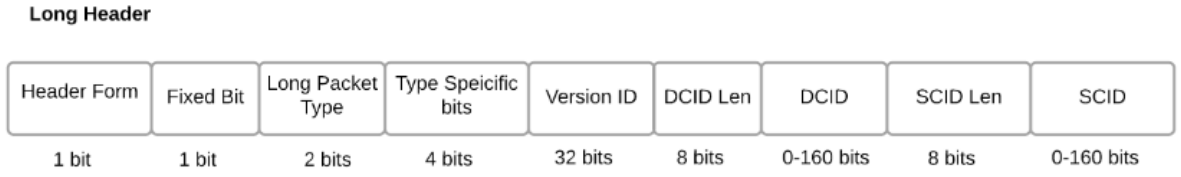

Рисунок 5. Формат длинного заголовка

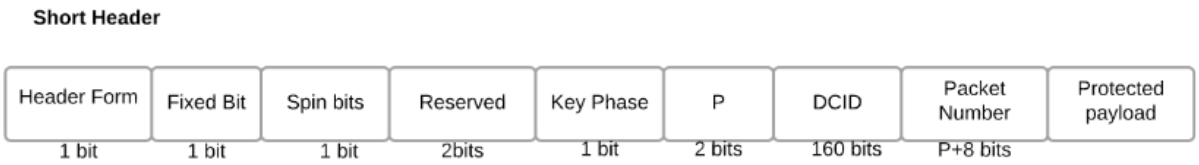

Рисунок 6. Формат короткого заголовка

Длинный заголовок содержит следующие поля в своем заголовке, как показано на рисунке 5.

– Форма заголовка (HF) - определяет тип заголовка

– Фиксированный бит (FB) - указывает, является ли пакет действительным или нет. Если он установлен в 0, пакет недействителен.

– Тип длинного пакета (T) - указывает тип пакета с длинным заголовком

– Биты, зависящие от типа - биты, специфичные для типов пакетов с длинными заголовками

– Идентификатор версии (VID) - 32 бита для идентификации версии QUIC

– Длина идентификатора целевого соединения (DCID Len)

– Идентификатор целевого соединения (DCID)

– Длина идентификатора подключения к источнику (SCID Len)

– Идентификатор подключения к источнику (SCID)

– Короткий заголовок содержит следующие поля в своем заголовке, как показано на рисунке 6.

– Форма заголовка (HF)

– Исправленный бит (FB)

– Бит вращения - бит задержки вращения

– Зарезервированные биты

– Ключевой этап - идентификация ключа защиты пакетов

– Длина номера пакета (P)

– Идентификатор целевого соединения (DCID)

– Номер пакета

– Полезная нагрузка пакета

- **ID связи**

В QUIC существует Connection ID или CID. У каждой связи есть свой набор CID, каждый из которых может быть использован для выделения связи. CID независимо выбирается конечными приложениями, между которыми происходит отправка. Целью CID является доставка информации до получателя независимо от смены его UDP или IP адреса в сети

- **Stream Frame**

Поскольку QUIC подключается клиент и сервер одним соединением, появляется необходимость создания нескольких потоков внутри этого соединения для передачи данных в режиме мультиплекс. Этим и занимается Stream Frame, он создает несколько внутренних потоков, доставляющих информацию. В каждом таком потоке есть механизм контроля данных и отслеживание потерь, если потеря происходит, то это не влияет на другие потоки. В каждом потоке содержится информация о его порядковом номере, смещение, которое позволяет определить откуда потоку начинать передавать пакет, длину передаваемой информации и само содержимое.

- **Нумерация пакетов**

Нумерация представлена числом в диапазоне от 0 до $2^{.61}$ - 1. Это используется для определение криптографического одноразового номера защиты. Каждая конечная точка поддерживает отдельный номер пакета для отправки и получения. Номера пакетов ограничены, потому что им необходимо быть представленными целиком в поле ACK. Номера пакетов могут быть сокращены и кодироваться в от 1 до 4 байт.

Установление соединения TCP

Поскольку TCP является протоколом, ориентированным на предварительное соединение (connection-oriented), то сначала необходимо установить сессию между приложениями конечных устройств. Узел отправитель инициализирует соединение, которое должно быть

подтверждено узлом получателем. Программное обеспечение протокола TCP обмениваются сообщениями через сеть, чтобы проверить, что передача разрешена и что обе стороны готовы к ней.

Соединение между двумя устройствами производится в три этапа (рис.7).

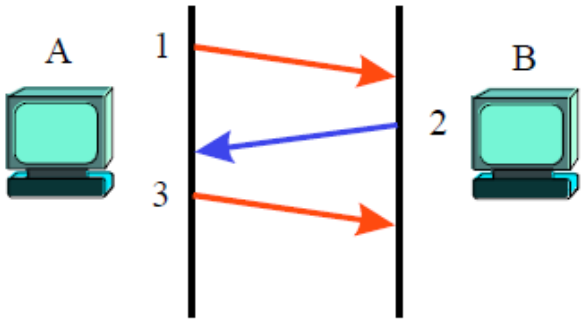

Рисунок 7. Установление соединения TCP

Во-первых, узел отправитель инициализирует установление связи путем посылки узлу получателю запроса синхронизации SYN (1).

Во-вторых, узел получатель подтверждает запрос синхронизации и задает свои параметры синхронизации ACK (2).

В-третьих, узлу получателю посылается подтверждение, что обе стороны готовы, чтобы соединение было установлено (3).

Такой механизм получил название трехэтапного установления связи (Three-way handshake). Оба узла должны согласовать начальные номера последовательности передаваемых частей информации, что происходит через обмен сегментами синхронизации (SYN) и подтверждения (ACK).

Установление соединения QUIC

В протоколе QUIC соединение начинается с рукопожатия, в котором сразу определяются параметры коммуникации и криптографическая составляющая QUIC-TLS. Трехстороннее рукопожатие TCP и трехстороннее рукопожатие TLS 1.3 объединяются в один трехпакетный обмен (рис.8). В результате экономится время, затрачиваемое на полный круг (Round Trip

Time, RTT). Для коротких сессий это позволяет значительно повысить производительность.

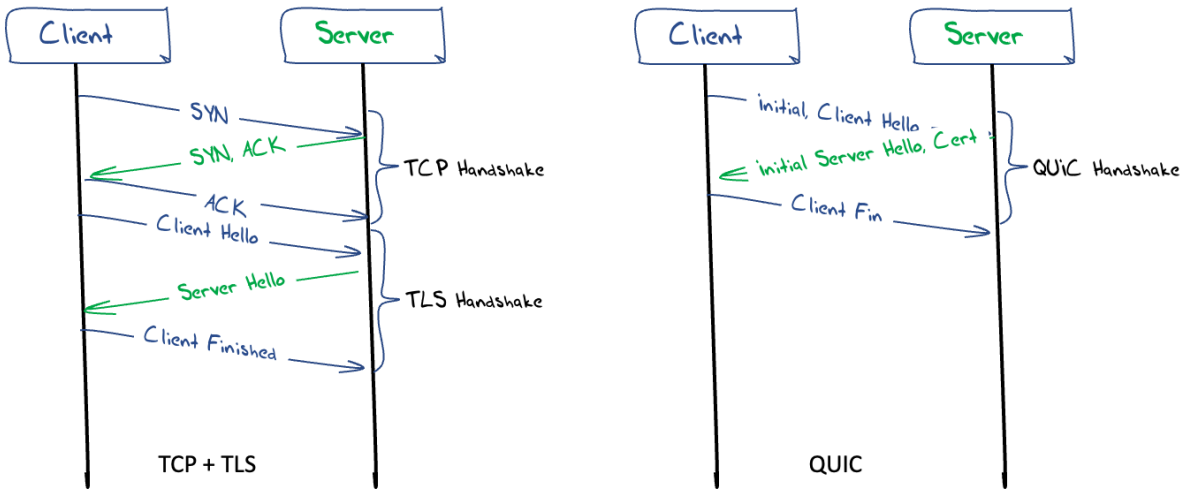

Рисунок 8. Сравнение рукопожатий TCP/TLS и QUIC

Нулевое рукопожатие (0-RTT) в QUIC также позволяет клиенту отправлять на сервер зашифрованные полезные данные уже в первом пакете, используя согласованные параметры предыдущего соединения и идентификатор TLS 1.3 с ранее выданным сервером ключом (PSK), хотя подобный 0-RTT-обмен потенциально уязвим для атак повторного воспроизведения.

Сравнение TCP и QIUC

QUIC основан на UDP и был разработан для решения проблем TCP, который является доминирующим транспортным протоколом, используемым в Интернете.

Одним из ключевых различий между QUIC и TCP является способ, которым они обрабатывают передачу данных. TCP — это протокол, ориентированный на связь, который устанавливает выделенное сквозное соединение между устройствами для передачи данных. Оно поддерживается в течение всего времени передачи, и данные передаются надежным, упорядоченным образом. QUIC, с другой стороны, является протоколом без связи, который позволяет передавать данные без необходимости в

выделенном сквозном соединении. Это обеспечивает большую гибкость и может привести к снижению задержки, поскольку уменьшаются накладные расходы на установление и поддержание соединения.

Чтобы сравнить скорость работы протоколов хорошо подойдёт следующая иллюстрация:

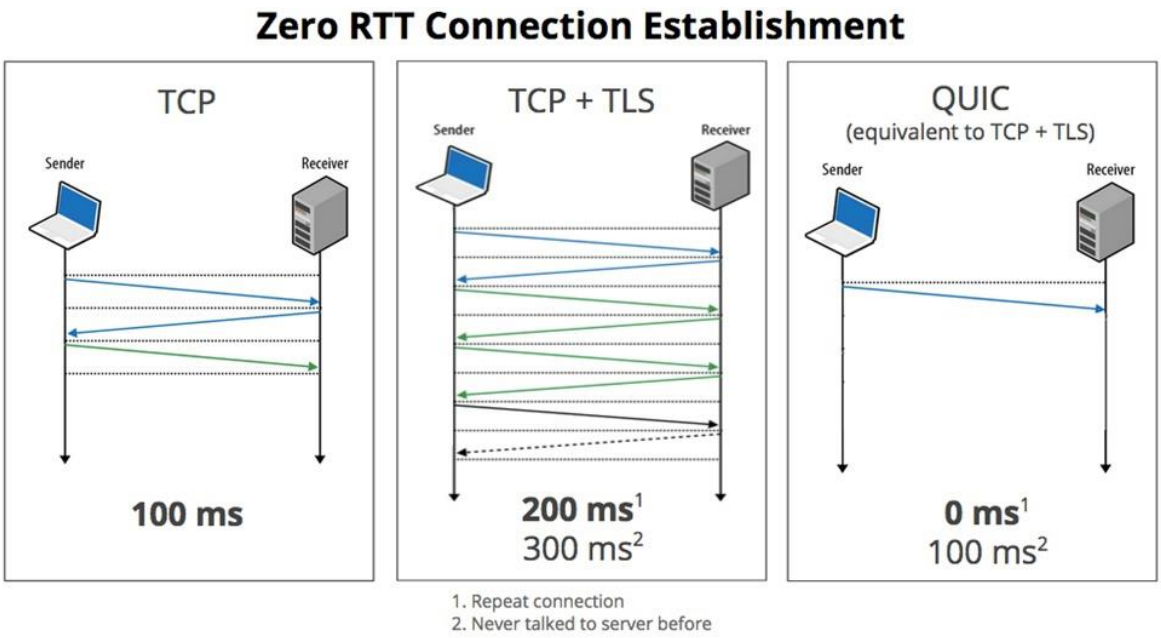

Рисунок 9. Установление соединения TCP, TCP/TLC, QUIC

На ней мы видим большой прирост скорости QUIC даже в сравнении с TCP без шифрования. Поскольку QUIC основан на UDP, для передачи информации нет необходимости в том, чтобы отправлять RTT в таком же количестве, как это делает TCP. Если связь между клиентом и сервером уже была установлена, QUIC не будет отправлять никаких RTT, но, если связь первичная, один RTT всё же будет отправлен. Это положительно сказывается на скорости передачи данных.

Применение протоколов TCP, UDP, QUIC.

TCP

Первым из рассматриваемых протоколов будет TCP, или Transmission Control Protocol, который используется для транспортировки сообщений между устройствами в сети.

В сети файлы не передаются целиком, а дробятся и передаются в виде относительно небольших сообщений. Далее 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.

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

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.

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.

QUIC

По мере того, как HTTP / 3 и QUIC набирают популярность и получают все большее распространение, ожидается появление разнообразных вариантов использования. Эти варианты использования включают прямую трансляцию и видеопотоки, видео по запросу, загрузки и веб-ускорение. К числу наиболее обнадеживающих сценариев применения этих технологий относятся:

1. **Веб- и мобильные приложения реального времени.** Эти приложения, такие как веб- и мобильные приложения с голосовой и видеосвязью, требуют низкой задержки и надежной передачи данных. Использование QUIC независимых потоков и механизмов контроля перегрузки делают его хорошим выбором для этих приложений, поскольку он позволяет быстро и эффективно отправлять, и получать данные. В многопоточном режиме протокола QUIC передача данных между разными потоками в пределах одного соединения не затрагивается.

2. **Связь с устройствами интернета вещей.** Устройства интернета вещей часто используют для связи такие протоколы, как TCP и MQTT. Однако эти протоколы могут быть подвержены таким проблемам, как высокая задержка и потеря пакетов, особенно в ограниченных средах. QUIC может стать более надежной и эффективной альтернативой, поскольку он разработан для хорошей работы в сетях с высокой задержкой и потерями. Его близкое к нулю время прохождения в оба конца (RTT) важно для повышения производительности сети и обеспечения положительного взаимодействия с пользователем.

3. **Интернет транспортных средств и подключенные автомобили.** QUIC может принести большую пользу экосистеме интернета транспортных средств (IoV). Эти системы полагаются на обмен данными в режиме реального времени для предоставления таких услуг, как управление дорожным движением, отслеживание транспортных средств и функции безопасности. Низкая задержка QUIC, возможности мультиплексирования и устойчивость к потере пакетов и переупорядочиванию пакетов могут обеспечить надежную и эффективную связь между транспортными средствами и компонентами инфраструктуры. Кроме того, использование QUIC шифрования TLS обеспечивает повышенную безопасность конфиденциальных данных об автомобиле.

4. **Облачные вычисления** включают в себя доставку вычислительных ресурсов через Интернет. Благодаря QUIC облачные приложения могут извлечь выгоду из низкой задержки и сквозного шифрования, что может улучшить взаимодействие с пользователем и безопасность.

5. **Приложения для платежей и электронной коммерции.** Эти приложения требуют безопасной и надежной передачи данных. Использование QUIC шифрования безопасности транспортного уровня (TLS) и надежных потоков HTTP3 делают его хорошим выбором для этих приложений, поскольку он помогает гарантировать безопасную передачу данных без перехвата. С точки зрения конечного пользователя, протокол

QUIC также улучшает взаимодействие с пользователем, обеспечивая более быстрые и бесперебойные транзакции.

Контрольные вопросы.

1) В чем различие между протоколами TCP и UDP?

2) По какой команде можно узнать, какие TCP соединения активны на сетевом конечном узле?

3) Для какого соединения предназначен протокол QUIC.

4) В каких случаях используется длинный и короткий заголовки протокола QUIC.

5) За сколько этапов выполняется предварительное установление соединения у протокола TCP?

6) Какая главная особенность протокола QUIC по сравнению с TCP при передаче данных.

7) Может ли видеотрафик передаваться с помощью протокола TCP и почему?