Internet Engineering Task Force (IETF)
Request for Comments: 9308
Category: Informational
ISSN: 2070-1721

M. Kühlewind
Ericsson
B. Trammell
Google Switzerland GmbH

September 2022

Applicability of the QUIC Transport Protocol

Применимость транспортного протокола QUIC

Аннотация

В этом документе обсуждается применимость транспортного протокола QUIC с упором на предостережения, влияющие на разработку и развёртывание прикладных протоколов на базе QUIC. Документ предназначен для разработчиков отображений прикладных протоколов на транспорт QUIC и самих прикладных протоколов.

Статус документа

Документ не относится к категории Internet Standards Track и публикуется лишь для информации.

Документ является результатом работы IETF[1] и представляет согласованный взгляд сообщества IETF. Документ прошёл открытое обсуждение и был одобрен для публикации IESG[2]. Не все одобренные IESG документы являются кандидатами в Internet Standard, см раздел 2 RFC 7841.

Информацию о текущем статусе документа, ошибках и способах обратной связи можно найти по ссылке https://www.rfc-editor.org/info/rfc9308.

Оглавление

[1]Internet Engineering Task Force - комиссия по решению инженерных задач Internet.
[2]Internet Engineering Steering Group - комиссия по инженерным разработкам Internet.

1. Введение

Новый транспортный протокол QUIC [QUIC] обеспечивает ряд расширенных функций. Хотя протокол разрабатывался для HTTP, он предоставляет возможности для более широкого круга приложений. QUIC инкапсулируется в UDP. QUIC версии 1 интегрирован с TLS 1.3 [TLS13] для шифрования всего содержимого и большей части данных управления. Версию HTTP, использующую QUIC, называют HTTP/3 [QUIC-HTTP].

В этом документе даны рекомендации для разработчиков приложений, которые хотят использовать протокол QUIC, не реализуя его самостоятельно. Это включает общее руководство для приложений на основе HTTP/3 или напрямую через QUIC.

В последующих разделах рассматриваются конкретные предостережения в части применимости QUIC и проблемы, которые нужно учитывать разработчикам приложений, использующих транспорт QUIC.

2. Необходимость отката

QUIC использует UDP в качестве основы. Это позволяет реализацию в пользовательском пространстве и обеспечивает прохождение промежуточных устройств (включая NAT) без обновления имеющейся инфраструктуры.

Измерения показывают, что от 3% [Trammell16] до 5% [Swett16] сетей блокируют весь трафик UDP, при этом мало свидетельств других форм систематических недостатков UDP по сравнению с TCP [Edeline16]. Блокировка подразумевает, что все приложения, работающие на основе QUIC, должны быть готовы к отказу в соединениях через такие сети или разработаны с возможностью отката к другому транспортному протоколу. В случае HTTP таким откатом является использование TLS на основе TCP.

Спецификация транспортных служб IETF (Transport Services или TAPS) [TAPS-ARCH] описывает систему с общим API для разных протоколов. Это особенно актуально для QUIC, поскольку решает проблему отката с использованием нескольких протоколов.

В частности, нужно избегать отката к незащищённым протоколам или слабым версиям протоколов с защитой. В общем случае приложениям, реализующим откат, необходимо учитывать влияние тката на безопасность. Откат к TCP и TLS раскрывает управляющую информацию для изменения и манипуляций в сети. Кроме того, откат к TLS версии ниже 1.3, которая применяется в QUIC версии 1, может вести к существенному ослаблению криптографической защиты. Например, результаты согласования протокола [RFC7301] имеют защиту конфиденциальности лишь при использовании TLS 1.3.

Эти приложения должны работать, возможно, с нарушением функциональности при отсутствии обеспечиваемых QUIC функций в резервном (fallback) протоколе. При откате к TLS через TCP наиболее очевидным отличием является отсутствие в TCP мультиплексирования потоков, требующее реализации такого мультиплексирования на уровне приложения. Кроме того, реализации TCP и сетевые пути зачастую не поддерживают опцию TCP Fast Open (TFO) [RFC7413], позволяющую передавать содержимое в первом пакете управления для новых соединений, тогда как возобновление сессий 0-RTT в QUIC позволяет это. Отметим, что имеются свидетельства блокировки данных в SYN даже при успешном согласовании TFO (см. [PaaschNanog]). И даже при сквозной работе Fast Open опция ограничена одним пакетом согласования TLS и данных приложения, в отличие от QUIC 0-RTT.

Хотя шифрование (в данном случае TLS) неразрывно связано с QUIC, согласование TLS через TCP может быть заблокировано. Если TLS через TCP не может поддерживаться, соединение следует прерывать, а затем приложение должно подобающим способом уведомить пользователя о недоступности защищённых коммуникаций.

Таким образом, любой механизм отката может приводить к снижению производительности и защищенности, однако такой откат не должен нарушать ожидания приложений в части конфиденциальности и целостности содержимого.

3. 0-RTT

QUIC поддерживает организацию соединений 0-RTT. Хотя такая же возможность обеспечивается в TLS 1.3 с TCP, с механизмом 0-RTT связаны свои возможности и проблемы для приложений, использующих QUIC.

Транспортный протокол, обеспечивающий организацию соединений 0-RTT, качественно отличается от протоколов без 0-RTT с точки зрения использующих протокол приложений. Стоимость закрытия соединения с последующим повторным открытием и попытки сохранить соединение открытым различаются (см. параграф 3.2. Восстановление сессий по сравнению с Keep-Alive).

Приложению нужно сознательно выбирать применение 0-RTT, поскольку с этим механизмом связан риск replay-атак. Для использующих 0-RTT прикладных протоколов требуется профиль, описывающий типы информации, которую можно передавать без опаски. Для HTTP этот профиль описан в [HTTP-REPLAY].

3.1. Replay-атаки

Повторная передача или злонамеренное воспроизведение данных из пакетов 0-RTT может приводить к получению серверной стороной множества копий одних и тех же данных.

Данные приложения, переданные в пакетах 0-RTT, могут обрабатываться неоднократно при их повторном использовании (replay). Приложения должны понимать, что можно безопасно передавать в 0-RTT. Прикладные протоколы, стремящиеся применять 0-RTT, должны тщательно анализировать и описывать возможное содержимое пакетов 0-RTT (см. параграф 5.6 в [QUIC-TLS]).

В некоторых случаях может быть достаточным ограничиться включением в 0-RTT данных, которые не вызывают на сервере действий с постоянным эффектом. Примерами безопасных действий являются инициирование извлечение данных и установка конфигурации. Идемпотентные операции, повторение которых не даёт дополнительного эффекта, также могут быть безопасными. Однако последовательность идемпотентных операций может быть неидемпотентной.

Когда сервер воспринял данные 0-RTT, уже нет способа выборочно отбросить полученные данные. Однако протоколы могут определять способы отклонения определённых действий, которые могут быть небезопасны при повторе.

Некоторые реализации и развёртывания TLS могут обеспечивать частичную и даже полную защиту от replay-атак, которая может служить для контроля рисков.

3.2. Восстановление сессий по сравнению с Keep-Alive

Поскольку QUIC инкапсулируется в UDP, использующие QUIC приложения должны иметь дело с короткими тайм-аутами бездействия сети. Развёрнутые промежуточные устройства с поддержкой состояния обычно создают состояние для потока UDP по первому переданному пакету и сохраняют его в течение периода бездействия значительно более короткого, чем для TCP. В [RFC5382] предложен период бездействия для TCP не менее 124 минуту, хотя в литературе не упоминается о широком применении этого правила. Однако короткий период ожидания для UDP чётко документирован. Согласно исследованию 2010 г. ([Hatonen10]), приложения UDP могут предполагать, что любая привязка NAT или иное состояние могут истекать после 30 секунд бездействия. В параграфе 3.5 [RFC8085] рассмотрены интервалы keep-alive для UDP и требуется минимальное значение 15 секунд с рекомендацией использовать более долгие интервалы или совсем отказаться от keep-alive.

За счёт применения идентификаторов соединений протокол QUIC более устойчив к сменам привязки NAT по тайм-ауту. Однако это помогает лишь в тех случаях, когда конечная точка сохраняет доступность по адресу, известному её партнёру и этот партнёр передаёт данные после тайм-аута.

Некоторые соединения QUIC могут быть неустойчивы к смене привязки NAT, поскольку инфраструктура маршрутизации (в частности, балансировщики) применяет квартет адресов и номеров портов для направления трафика. Кроме того, промежуточные устройства, функции которых отличаются от трансляции адресов, также могут влиять на путь. В частности, некоторые межсетевые экраны не пропускают серверный трафик, для которого у них нет свежего состояния для соответствующего пакета, переданного от клиента.

Приложения QUIC могут настраивать период бездействия для контроля риска тайм-аутов. Периоды бездействия и тайм-аут простоя в сети отличаются от тайм-аута бездействия соединения, который определяется меньшим значением тайм-аута бездействия конечных точек (см. параграф 10.1 в [QUIC]). Здесь возможны три варианта.

- Игнорирование проблемы, если протокол прикладного уровня включает взаимодействия с очень коротким интервалом бездействия или совсем без него или протокол достаточно устойчив к смене привязки NAT.
- Предотвращение продолжительных интервалов бездействия.
- Восстановление сессии после долгого бездействия с применением 0-RTT, когда это уместно.

Первый вариант проще всего, но подходит лишь для определённых приложений.

Сервер или клиент в приложении QUIC могут передавать кадры PING для сохранения активности (keep-alive) соединения и состояний на пути от возникновения тайм-аута. Рекомендации по использованию keep-alive зависят от приложения в основном из-за требований к задержке и частоте сообщений. В этом случае прикладное отображение должно указывать, кто отвечает за сохранение соединения - клиент или сервер. Хотя [Hatonen10] предполагается, что 30 секунд будет достаточно для общедоступной сети Internet при наличии в пути NAT, предпочтительней большие значения, если развёртывание может выдерживать смену привязки NAT или известно, что оно размещается в контролируемой среде (например, ЦОД), для снижения загрузки сети и расчётных издержек.

Передача кадров PING с интервалом менее 30 секунд в периоды бездействия может в некоторых случаях приводить к избыточному непродуктивному трафику и неприемлемому расходу батарей в (мобильных) устройствах. Кроме того, тайм-ауты меньше 30 секунд могут затруднять обработку временных перебоев в сети, таких как перенос виртуальных машин (Virtual Machine или VM) или потеря связи в результате перемещения (см. [RFC8085], особенно параграф 3.5).

Клиент (но не сервер) может также использовать восстановление сессий вместо передачи трафика keep-alive. В этом случае клиент, желающий передать серверу данные через соединение, которое бездействовало дольше тайм-аута сервера (доступен через транспортный параметр idle_timeout), может просто подключиться заново. По возможности для этого можно применять восстановление 0-RTT, снижающее задержку, связанную с повторной организацией. Этот подход применим лишь в тех случаях, когда можно безопасно использовать 0-RTT и перезапуск выполняет клиент.

Компромисс между возобновлением и keep-alive следует выбирать отдельно для каждого приложения. В общем случае приложениям следует применять keep-alive лишь в ситуациях, когда непрерывность коммуникаций очень важна, например, [QUIC-HTTP] рекомендует использовать keep-alive только при наличии находящихся в обработке запросов.

4. Использование потоков

Функция мультиплексирования потоков QUIC позволяет приложениям использовать несколько потоков в одном соединении без возникновения конфликтов (блокировка head-of-line) между ними. Данные потоков передаются в кадрах и один пакет QUIC в линии может включать один или несколько кадров потоков.

Потоки могут быть односторонними или двухсторонними и инициировать их может как клиент, так и сервер. В одностороннем потоке данные может передавать лишь его инициатор.

Потоки и соединения могут передать не более 2^{62}-1 байтов в каждом направлении из-за ограничений кодирования смещений в потоке и управления потоком данных в соединении. В маловероятном на сегодняшний день случае достижения предела потребуется организовать новое соединение.

Потоки можно открывать и закрывать независимо, аккуратно или жёстко. Приложение может аккуратно закрыть исходящее направление, указав QUIC установку бита FIN в кадре STREAM. Входящее направление нельзя аккуратно закрыть без созданного партнёром FIN (как в TCP). Однако конечная точка может жёстко закрыть исходящее направление или запросить у партнёра жёсткое закрытие входящего направления, эти действия не зависят одно от другого.

QUIC не предоставляет интерфейса для исключительной обработки какого-либо потока. Если закрывается критичный для приложения поток, приложение может генерировать сообщения об ошибках на прикладном уровне для информирования другой стороны и/или вышележащего уровня, что может приводить к разрыву соединения QUIC.

Отображение данных приложения в потоки зависит от приложения и для HTTP/3 описано в [QUIC-HTTP]. Имеется несколько базовых принципов разработки приложений, использующих потоки.

- В рамках потока обеспечивается упорядочение. Если приложению нужно передать данные в определённом порядке, их следует направлять в один поток. Порядок передачи, приёма и доставки в разных потоках не гарантируется.
- Несколько потоков обеспечивают распараллеливание. Данные могут обрабатываться независимо и может возникать блокировка head-of-line (пробка), если принудительно упорядочивать их приём данных, отправленных в разных потоках.
- Потоки могут обеспечивать ориентацию на сообщения и позволяют отклонять сообщения. Если сообщение сопоставлено с одним потоком, сброс потока для завершения срока действия неподтвержденного сообщения может служить для эмуляции неполной гарантии для этого сообщения.

Если получатель QUIC открыл максимально разрешённое число одновременных потоков, а отправитель говорит, что нужны ещё потоки, это не ведёт к автоматическому увеличению максимального числа потоков у получателя. Поэтому приложениям следует учитывать максимальное число разрешённых, открытых в данный момент и используемых в данный момент потоков при задании сопоставления данных с потоками.

QUIC выделяет числовой идентификатор (stream ID) для каждого потока. Хотя взаимосвязь между этими идентификаторами и типами потоков чётко задана в QUIC версии 1, будущие спецификации могут поменять её для других версий. Реализациям QUIC следует раскрывать свойства каждого потока (инициатор, число направлений, идентификатор), а приложениям следует запрашивать эти свойства, не пытаясь вывести их из идентификатора потока.

Метод назначения идентификаторов созданным приложением потокам может зависеть от реализации транспорта, поэтому приложениям не следует предполагать, что потоку, который ещё не создан, будет назначен определённый идентификатор. Например, HTTP/3 использует идентификаторы потоков для указания уже созданных потоков и не принимает допущений относительно будущих идентификаторов и способов их назначения (раздел 6 в [QUIC-HTTP]).

4.1. Мультиплексирование потоков

Потоки имеют смысл только для приложений, поскольку информация о потоке передаётся внутри границы шифрования QUIC и пакет не раскрывает сведений о передаваемых в нем потоках. Следовательно, мультиплексирование потоков не предназначено для их дифференцирования с точки зрения обработки в сети. Поэтому трафик приложений, требующий различной обработки в сети, следует передавать с разными квинтетами (5-tuple), т. е. через разные соединения QUIC. С учётом способности QUIC передавать данные приложения в первом RTT соединения (если предыдущее соединение с тем же хостом было организовано для предоставления требуемых свидетельств), стоимость организации нового соединения очень мала.

4.2. Приоритизация

Приоритизация потоков не раскрывается ни сети, ни получателю. Приоритизацией управляет отправитель и транспорту QUIC следует предоставлять приложениям интерфейс для приоритизации потоков [QUIC]. Приложения могут реализовать свою схему приоритизации - прикладной протокол, работающий на основе QUIC может определять явные сообщения для сигнализации приоритета, такие как заданы в [RFC9218] для HTTP. Прикладной протокол может задавать правила, позволяющие конечной точке задавать приоритет на основе контекста, или предоставлять высокоуровневый интерфейс и сохранить контроль приоритета за приложением.

Обработка приоритета при повторной передаче может быть реализована отправителем на транспортном уровне. В [QUIC] рекомендуется повторять передачу потерянных данных до отправки новых, если приложение явно не задаёт иное. Когда конечная точка QUIC использует для передачи потоки с полной гарантией, приоритизация повторной передачи будет давать преимущества в большинстве случаев, заполняя пропуски и освобождая окно управления потоком данных. Для потоков с частичной гарантией или без гарантии планирование более высокого приоритета повторов по сравнению с данными высокоприоритетных потоков может оказаться нежелательным. Для таких потоков QUIC может предоставлять явный интерфейс для контроля приоритизации или выводить решения о приоритете из уровня гарантий для потока.

4.3. Упорядоченная и надёжная доставка

Потоки QUIC обеспечивают упорядоченную и надёжную доставку. Хотя реализация может предоставлять опции использования потоков без полной гарантии или сохранения порядка, большинство реализаций предполагает надёжную доставку с сохранением порядка.

В соответствии с этим допущением конечная точка, получающая данные потока, может не продвигаться вперёд, пока не станут доступными данные, смежные с началом потока. В частности, получатель может приостановить поток данных, пока приложению не будут доставлены непрерывные данные (см. параграф 2.2 в [QUIC]). Для поддержки такой логики приёма конечная точка будет отправлять данные потока, пока они не будут подтверждены, чтобы данные из начала потока были переданы и подтверждены первыми.

Конечная точка с иным поведением при передаче, не согласующая его с партнёром, может столкнуться с проблемами производительности или блокировкой.

4.4. Блокировка управления потоком данных

Управление потоком данных QUIC (раздел 4 в [QUIC]) обеспечивает средства управления доступом к ограниченным буферам, которые есть у конечной точки для входящих данных. Этот механизм ограничивает объем данных, которые могут находиться в буферах конечных точек или в сети. Однако в некоторых случаях эти пределы могут создавать условия, которые могут приводить к неоптимальной работе и даже блокировке соединений.

Блокировки в управлении потоком данных возможны для любых протоколов, применяющих QUIC, хотя связанные с этим проблемы зависят от способа потребления данных реализацией и кредита управления потоком данных. Понимание причин блокировки позволит избежать проблем.

Размер и частота обновлений кредита управления потоком данных могут влиять на производительность. Использующие QUIC приложения часто имеют потребителя данных, читающего данные из транспортных буферов. У некоторых реализаций могут быть независимые буферы приёма на транспортном и прикладном уровне. Потребление данных не всегда подразумевает их незамедлительную обработку, однако общим методом реализации служит расширение кредита управления потоком данных для отправителя путём отправки кадров MAX_DATA и/или MAX_STREAM_DATA по мере потребления данных. На доставку этих кадров влияет задержка в канале передачи от получателя к отправителю данных. Если кредит своевременно не расширяется, передающее приложение может быть заблокировано, фактически останавливая передачу.

Крупные сообщения приложений могут вызывать блокировку, если получатель не считывает данные из транспорта поэтапно. Если сообщение больше доступного кредита управления потоком данных и получатель не освобождает дополнительный кредит, может возникнуть блокировка. Это возможно даже без достижения пределов управления потоком данных в потоке (stream), поскольку пределы управления потоком данных могут быть использованы другими потоками.

Формат сообщений с префиксом размера упрощает потребителю данных оставлять несчитанные данные в транспортном буфере и, таким образом, удерживать кредит управления потоком данных. Если пределы управления потоком данных не позволяют передать оставшуюся часть сообщения, возникает блокировка. Префикс размера может также позволять обнаружение таких блокировок. Когда прикладные протоколы имеют сообщения, которые можно обработать одним блоком, приёмный кредит управления потоком данных для всего сообщения делает такую блокировку менее вероятной.

Потребитель данных может охотно считывать данные, как только они станут доступны, чтобы получатель расширил кредит управления потоком данных и снизилась вероятность возникновения блокировки. Однако такому потребителю данных могут потребоваться другие меры удержания ответственности партнёра за дополнительное состояние, которое он сохраняет для частично обработанных сообщений.

Блокировка также может возникать при взаимозависимости данных из разных потоков. Предположим, что данные из одного потока прибывают раньше, чем данных из другого потока, от которого они зависят. Блокировка может возникнуть, если первый поток останется несчитанным, препятствуя расширению кредита управления потоком данных для второго потока. Для снижения вероятности такой блокировки взаимозависимых данных отправителю следует избегать отправки таких данных, пока данные, от которых они зависят, не будут учтены в кредите управления потоком данных на уровне потока и соединения.

Некоторые случаи блокировки можно преодолеть путём отмены затрагиваемых потоков с помощью STOP_SENDING или RESET_STREAM. Отмена некоторых потоков в некоторых протоколах может приводить к обрыву соединения.

4.5. Обязательства по ограничению числа потоков

Конечные точки QUIC отвечают за передачу совокупного предела числа потоков, которые они позволяют открыть партнёру. Исходные ограничения анонсируются в транспортных параметрах initial_max_streams_bidi и initial_max_streams_uni. По мере открытия и закрытия потоков они потребляются и совокупное значение инкрементируется. Пределы можно увеличить с помощью кадров MAX_STREAMS, но механизмов снижения нет. По достижении предельного числа потоков, дополнительные потоки создать невозможно и использующее QUIC приложение не сможет двигаться дальше. В этом случае соединения могут быть прерваны по тайм-ауту или явным закрытием, как описано в разделе 10 10.

Использующему QUIC приложению, сообщившему кумулятивный предел числа потоков, может потребоваться закрыть соединение раньше достижения предела, например, при остановке сервера для планового обслуживания. Незамедлительное закрытие соединения вызывает грубое завершение активно используемых потоков. В зависимости от использования приложением потоков QUIC это может быть нежелательно или пагубно в плане поведения или производительности приложения.

Более аккуратное закрытие выполняется путём прекращения отправки увеличений числа потоков и естественного завершения соединения по достижении предела. Однако требуемое для этого время зависит от партнёра и непредсказуемое время закрытия может не соответствовать целям приложения или операционным задачам. Применяющие QUIC приложения могут быть консервативными в части предельного числа открытых потоков для снижения неопределённости и обязательств. Однако излишняя консервативность по числу потоков влияет на их распараллеливание. Балансировка этих требований может зависеть от приложения и его развёртывания.

Вместо предотвращения грубого закрытия на основе ограничения числа потоков может применяться механизм аккуратного закрытия на прикладном уровне, чтобы сообщить о намерении закрыть соединение в некоторый будущий момент. В HTTP/3 для этого применяется кадр GOAWAY, при получении которого клиент прекращает создание новых потоков, даже до достижения порога. Вместо этого клиент создаёт новое соединение, в котором будут открываться последующие потоки. Как только все потоки старого соединения будут закрыты, его можно безопасно прервать сразу или по тайм-ауту бездействия (см. 10. Завершение соединений).

5. Пакетизация и задержка

QUIC предоставляет интерфейс, позволяющий приложению создавать множество потоков, однако приложение обычно не может контролировать отображение передаваемых через эти потоки данных в кадры и связывание кадров в пакеты.

По умолчанию многие реализации пытаются упаковать кадры STREAM одного или нескольких потоков в каждый пакет QUIC для минимизации расхода пропускной способности и вычислительных издержек (см. раздел 13 в [QUIC]). Если для заполнения пакета недостаточно данных, реализация может подождать в течение короткого времени для оптимизации пропускной способности за счёт задержки. Эта задержка может настраиваться или устанавливаться динамически на основе наблюдаемой картины передачи данных приложением.

Если приложению нужна малая задержка и передача лишь мелких блоков данных (chunk), может оказаться полезным задать протоколу QUIC незамедлительную передачу всех данных. Если предположение предполагает использовать определённую картину передачи, оно может задать для QUIC время ожидания при сборке кадров в пакет.

Точно так же, приложение обычно не может контролировать размер пакетов QUIC в линии. QUIC предоставляет возможность добавлять кадры PADDING для произвольного увеличения размера пакетов. Заполнение применяется в QUIC для того, чтобы передавать дейтаграммы не меньше согласованного размера (см. параграфы 8.1 и 14.1 в [QUIC]) и проверять путь после переноса соединения (см. параграф 8.2 в [QUIC]), а также для определения PMTU (Datagram Packetization Layer PMTU Discovery или DPLPMTUD, см. параграф 14.3 в [QUIC]).

Заполнение может применяться приложением для сокращения утечки сведений о передаваемых им данных. Реализация QUIC может предоставлять прикладному уровню интерфейс для контроля применением заполнения.

6. Обработка ошибок

QUIC рекомендует конечным точкам информировать партнёра обо всех обнаруженных ошибках. Ошибки могут возникать на транспортном и прикладном уровне. Транспортные ошибки, такие как нарушение протокола, влияют на соединение в целом. Использующее QUIC приложение может определять свои средства обнаружения и сигнализации для ошибок (см., например, раздел 8 в [QUIC-HTTP]). Ошибки приложения влияют на соединение или отдельный поток.

QUIC определяет пространство кодов, используемых для обработки ошибок на транспортном уровне. QUIC рекомендует конечным точкам использовать наиболее конкретный код, хотя разрешены наиболее подходящие коды, включая базовые ошибки.

Применяющие QUIC приложения определяют своё пространство кодов, независимое от QUIC и других приложений (см., например, параграф 8.1 в [QUIC-HTTP]). Значения кодов ошибок приложения могут совпадать с кодами ошибок на уровне соединения или потока.

Ошибки соединения ведут к его прерыванию. Они передаются с помощью кадров CONNECTION_CLOSE, содержащих поля кода и причины ошибки (может иметь размер 0). Для сигналов о транспортных и прикладных ошибках применяются разные типы кадров CONNECTION_CLOSE.

Ошибки потока ведут к прерыванию потока. Они передаются с помощью кадров STOP_SENDING или RESET_STREAM, содержащих лишь код ошибки.

7. Эффективность подтверждений

В QUIC версии 1 без расширений применяется стратегия подтверждений TCP (см. параграф 13.2 в [QUIC]), т. е. рекомендуется подтверждать каждый второй пакет. Однако для создания и обработки подтверждений QUIC расходуются ресурсы отправителя и получателя. С подтверждениями также связаны расходы на пересылку и расход пропускной способности каналов, что может влиять на производительность в некоторых типах сетей. Приложения могут повысить общую производительность, применяя иную стратегию с меньшей частотой подтверждений. В [QUIC-ACK-FREQUENCY] описано расширение для сигнализации желаемой задержки подтверждений и обсуждаются варианты его применения, а также влияние на контроль перегрузок и восстановление при ошибках.

8. Выбор порта и обнаружение конечных точек приложения

В общем случае номера портов служат для двух целей: «во-первых, они обеспечивают идентификаторы демультиплексирования для различения транспортных сессий между парой конечных точек, во-вторых, они могут указывать прикладной протокол и связанную службу, к которой подключены процессы» (раздел 3 в [RFC6335]). Допущение о возможности идентификации приложения в сети по номеру порта сегодня не так верно по причине инкапсуляции и механизмов динамического назначения портов, как отмечено в [RFC6335].

Поскольку QUIC является транспортным протоколом общего назначения, в нем не задаётся требований по использованию конкретного порта UDP. Для приложений с откатом к TCP, у которых ещё нет альтернативного сопоставления с UDP, обычно целесообразна регистрация (при необходимости) и использование порта UDP, соответствующего уже зарегистрированному для приложения порту TCP. Например, принятым по умолчанию для HTTP/3 [QUIC-HTTP] является порт UDP 443, аналогично HTTP/1.1 и HTTP/2 через TLS по протоколу TCP.

С учётом широкого распространения в практике управления сетями допущения об однозначном сопоставлении номера порта с приложением, использование портов, которые невозможно просто сопоставить с зарегистрированным именем службы, может вызывать блокировку или иные изменения в поведении пересылки на элементах сети, таких как межсетевые экраны, определяющие приложения по номерам портов.

Приложения могут определять иной механизм обнаружения конечных точек, позволяющий использовать номера портов, отличные от принятых по умолчанию. Например, HTTP/3 (параграфы 3.2 и 3.3 в [QUIC-HTTP]) задаёт применение HTTP Alternative Services [RFC7838] для источника HTTP, чтобы анонсировать доступность эквивалентной конечной точки HTTP/3 на неком порту UDP путём использования h3 в качестве маркера согласования протокола прикладного уровня (Application-Layer Protocol Negotiation или ALPN) [RFC7301].

ALPN позволяет клиенту и серверу согласовать, какой из нескольких возможных протоколов будет применяться в данном соединении. Следовательно, на одном порту UDP может поддерживаться несколько приложений, разделяемых по маркерам ALPN. Применяющие QUIC приложения должны регистрировать маркер ALPN, используемый в согласовании TLS.

Поскольку QUIC версии 1 не определяет полностью механизм согласования, HTTP/3 требует QUIC версии 1 и определяет маркер ALPN (h3) для применения лишь с этой версией. До сих пор не выбрано единого подхода для управления использованием разных версий QUIC ни в HTTP/3, ни в общем случае. Использующие QUIC прикладные протоколы должны рассматривать управление версиями протокола QUIC. Решения для этих протоколов могут основываться на выборе, сделанном другими протоколами, такими как HTTP/3.

8.1. Выбор порта источника

Некоторые протоколы UDP уязвимы для атак с отражением, где злоумышленник способен использовать сторонний трафик для атаки на службы (denial of service). Например, приведённые ниже номера портов связаны с приложениями, уязвимыми для атак с отражением (часто из-за неверной нестройки серверов).

- 53 - DNS [RFC1034].
- 123 - NTP [RFC5905].
- 1900 - SSDP [SSDP].
- 5353 - mDNS [RFC6762].
- 11211 - memcache.

Службы могут блокировать порты источника, связанные со протоколами, уязвимыми для атак с отражением, чтобы избежать издержек на обработку большого числа пакетов. Однако такая практика оказывает негативное влияние на клиентов, не только требуя организации нового соединения, но и приводя к отказу некоторых экземпляров от применения QUIC для службы на некоторое время и откату к отличному от UDP протоколу (2. Необходимость отката).

В результате реализациям клиентов рекомендуется избегать применения портов источника, связанных с протоколами, для которых известно об уязвимости для атак с отражением. Отметим, что в соответствии с общими рекомендациями для реализации клиентов в [RFC6335], приложения будут избегать использования эфемерных портов 49152-65535. Отметим, что другие номера портов также могут применяться для отражения.

9. Перенос соединений

QUIC поддерживает перенос соединений на стороне клиента. Если меняется IP-адрес клиента, конечная точка QUIC по-прежнему связывает пакеты с имеющимся транспортным соединением по полю Destination Connection ID (11. Раскрытие информации и идентификаторы соединений) в заголовке QUIC. Это подходит для случаев, когда адрес меняется, таких как смена привязки NAT, преднамеренная смена локального интерфейса, завершение срока действия временного адреса IPv6 [RFC8981], указание сервером предпочтительного адреса (параграф 9.6 в [QUIC]).

Использование идентификатора соединения ненулевого размера для сервера строго рекомендуется для всех клиентов, которые размещены или могут находиться за NAT. Такие идентификаторы настоятельно рекомендуются при поддержке активного переноса соединений. Если соединение преднамеренно переносится на другой путь, используется новый идентификатор соединения для минимизации возможности привязки сетевыми наблюдателями. Другая конечная точка QUIC применяет идентификатор соединения для связывания разных адресов с тем же соединением и элементом, если представлен идентификатор соединения ненулевого размера.

Базовая спецификация QUIC версии 1 поддерживает одномоментное использование лишь одного сетевого пути, что позволяет применять варианты аварийного переключения. Требуется проверка пути, чтобы конечные точки могли предотвратить атаки с подменой адресов при смене пути. Проверка пути занимает не менее 1 RTT и при смене пути может сбрасываться контроль перегрузки. Поэтому перенос обычно влияет на производительность.

Пробные пакеты QUIC, которые можно одновременно передавать по нескольким путям, служат для проверки адресов и измерения характеристик пути. Эти пакеты не могут передавать данные приложения, но, вероятно, будут включать кадры заполнения. Конечные точки могут использовать сведения о получении таких пакетов как входные данные для контроля перегрузок на данном пути. Приложения могут использовать сведения о пробных пакетах для выбора пути.

В QUIC версии 1 поддерживается активный перенос лишь для клиентов. Однако серверы могут указать в процессе согласования своё предпочтение переноса соединения на другой адрес после согласования. Это может служить, например, для перехода с адреса, используемого несколькими серверами, на уникальный адрес данного экземпляра. Сервер может представить адрес IPv4 и IPv6 в транспортном параметре при согласовании TLS, а клиент может выбрать один из этих двух адресов (см. параграф 9.6 в [QUIC]).

10. Завершение соединений

Соединения QUIC прерываются одним из 3 способов: неявный тайм-аут простоя, явное незамедлительное закрытие, явный сброс без учёта состояния.

QUIC не предоставляет механизма аккуратного завершения соединений, применяющее QUIC приложение может определить свой процесс аккуратного завершения (см., например, параграф 5.2 в [QUIC-HTTP]).

Тайм-аут бездействия QUIC включается через транспортные параметры. Клиент и сервер анонсируют время ожидания и в соединении действует меньшее из этих значений. По тайм-ауту соединения просто закрываются. Поэтому приложению следует поддерживать возможность настройки своего максимального значения, а также иметь доступ к рассчитанному меньшему значению для соединения. Приложение может настраивать максимальный тайм-аут бездействия для новых соединений на основе числа открытых или ожидаемых соединений, поскольку сокращение времени ожидания позволяет быстрее освобождать ресурсы.

Данные приложений, передаваемые в потоках или дейтаграммах откладывают тайм-аут бездействия QUIC. Приложения со своим механизмом keep-alive будут, таким образом, сохранять соединение QUIC. Приложения без keep-alive могут использовать механизмы транспортного уровня (см. параграф 10.1.2 в [QUIC] и 3.2. Восстановление сессий по сравнению с Keep-Alive). Однако интерфейсы QUIC для управления поведением транспорта могут различаться, что влияет на отказоустойчивость.

Незамедлительное закрытие указывается кадром CONNECTION_CLOSE (см. 6. Обработка ошибок) и приводит к немедленному закрытию всех потоков, что может влиять на приложения (см. 4.5. Обязательства по ограничению числа потоков).

Сброс без учёта состояния служит крайней мерой для конечных точек без доступа к состоянию соединения. Получение сигнала о таком сбросе служит признаком непоправимой ошибки, отличающейся от ошибок соединения тем, что не представляется сведений прикладного уровня.

11. Раскрытие информации и идентификаторы соединений

QUIC раскрывает в сеть некоторую информацию из нешифрованной части заголовка до организации контекста шифрования или потому, что сведения предназначены для использования в сети. Для получения дополнительной

информации об управляемости QUIC см. [QUIC-MANAGEABILITY]. Протокол QUIC использует длинный заголовок, раскрывающий некоторую дополнительную информацию (версия и идентификатор соединения у источника), хотя короткий заголовок раскрывает лишь идентификатор соединения у получателя. В QUIC версии 1 длинный заголовок применяется при организации соединения, а короткий - при передаче данных в имеющемся соединении.

Идентификатор соединения может иметь нулевой размер. Такой размер каждая из конечных точек соединения может выбирать индивидуально и он может применяться в каждом пакете, за исключением первого, который клиент передаёт при организации соединения.

Конечная точка, выбравшая идентификатор нулевого размера, будет получать пакеты с идентификатором соединения для получателя нулевого размера. Конечной точке потребуется использовать другие сведения, такие как IP-адрес и номер порта источника и получателя. Это может означать неспособность конечной точки сопоставить дейтаграммы с соединениями, если эти значения меняются, что делает невозможной для соединения смену привязки NAT или перенос на новый путь.

11.1. Создаваемый сервером идентификатор соединения

QUIC поддерживает созданные сервером идентификаторы соединений, передаваемые клиентам при организации соединения (см. параграф 7.2 в [QUIC]). Серверам за балансировщиком нагрузки может потребоваться смена идентификатора соединения в процессе согласования, кодируя отождествление сервера или сведения о пуле балансирования нагрузки, чтобы обеспечить балансировку без поддержки состояния.

Развёртывания серверов с балансировщиками нагрузки и другой инфраструктурой маршрутизации должны гарантировать, что эта инфраструктура согласованно маршрутизирует пакеты экземпляру сервера, который имеет состояние соединения даже при смене адресов, портов или идентификатора соединения. Это может потребовать координации между серверами и инфраструктурой. Одним из методов является кодирование маршрутной информации в идентификаторе соединения. Пример этого метода представлен в [QUIC-LB].

11.2. Снижение возможности привязки путём смены идентификаторов

Если конечные точки не вносят свежих идентификаторов соединения, клиенты не могут сократить связываемость при смене адресов. Выбор значений, которые сторонний наблюдатель не может привязать к клиенту, обеспечивает невозможность тривиально сопоставить активность на разных путях с использованием идентификатора соединения.

Хотя достаточно надёжные схемы генерации идентификаторов соединений смягчают проблему привязки, они не обеспечивают полной защиты. Анализ сроков действия секстетов (6-tuple - адреса отправителя и получателя, а также перенесённый Connection ID) позволяет выявить связи.

В случаях, когда перенос соединений в пуле серверов происходит редко, наблюдателю легко связать два идентификатора соединений. И наоборот, когда каждый сервер одновременно обрабатывает множество переносов, даже раскрытия сопоставлений с серверами может быть недостаточно.

Наиболее эффективное смягчение таких атак обеспечивается устройством сети и/или эксплуатационной практикой с помощью архитектуры распределения нагрузки, когда на один серверный адрес загружается множество потоков данных, а также координация времени переноса для повышения числа одновременных переносов и другие меры.

11.3. Использование серверных пакетов Retry для перенаправления

QUIC поддерживает пакеты Retry, которые сервер может передавать в ответ на пакет Initial от клиента. Сервер может указать в этом пакете новый идентификатор соединения и клиент будут передавать другой пакет Initial с выбранным сервером идентификатором. Этот механизм можно использовать для перенаправления соединения на дугой сервер, например, из соображений производительности или при поэтапном обновлении серверов пула, когда они могут применять разные версии QUIC.

В этом случае предполагается, что все серверы того или иного пула координируются с балансировщиками нагрузки, пересылающими трафик по идентификаторам соединений. Сервер может указать идентификатор соединения в пакете Retry так, что балансировщик перенаправит следующий пакет Initial другому серверу того же пула. Дополнительно балансировщик может напрямую предлагать выгрузку Retry, как описано в [QUIC-RETRY].

Подход, описанный в разделе 4 [RFC5077] для создания квитанций возобновления TLS, служит примером, который подходит также для маркером проверки. Однако настоятельно рекомендуется применять более современные криптографические алгоритмы.

12. QoS и DSCP

QUIC, как определено в [QUIC], имеет один контроллер перегрузок и обработчик восстановления. Это подразумевает, что все пакеты в соединении QUIC или, по меньшей мере, пакеты с одним квинтетом (5-tuple) {dest addr, source addr, protocol, dest port, source port}, которые имеют один код дифференцированного обслуживания (Diffserv Code Point или DSCP) [RFC2475] будут иметь близкую трактовку в сети, поскольку отклики о потерях или задержке пакетов служат вводом для контроллера перегрузок. Поэтому пакетам одного соединения следует использовать один код DSCP. В параграфе 5.1 [RFC7657] рассматривается взаимодействие Diffserv с протоколами доставки дейтаграмм [RFC7657] (в этом отношении взаимодействие с QUIC похоже на взаимодействие с SCTP).

При мультиплексировании нескольких потоков в одном соединении QUIC выбирать следует значение DSCP, связанное с наибольшим приоритетом среди мультиплексируемых потоков.

Если желательна разная трактовка в сети, например, путём указания разных DSCP, можно использовать несколько соединений QUIC с одним сервером. В общем случае рекомендуется минимизировать число соединений QUIC с одним сервером, чтобы избежать роста издержек и, более важно, конкуренции механизмов контроля перегрузки.

Как и в других применениях Diffserv вход пакета в сеть, не поддерживающую значени DSCP, может приводить к тому, что пакет не получит в сети желаемой трактовки. Значение DSCP в этом пакете может быть также меняться по мере прохождения пакета через сеть, изменяя запрошенную трактовку.

13. Согласование версии и криптографии

Версия QUIC может полностью менять поведение протокола за исключением назначения нескольких полей заголовка, объявленных инвариантными [QUIC-INVARIANTS]. Версия QUIC с большим номером не обязательно обеспечивает лучший сервис, но может просто предоставлять иной набор функций. Поэтому от приложений требуется способность выбирать используемую версию QUIC.

Новая версия может применять схему шифрования, отличную от TLS 1.3 и выше. [QUIC] задаёт требования к криптографическому согласованию, реализованному в данное время TLS 1.3 и описанному в отдельной спецификации [QUIC-TLS]. Это разделение предназначено для облегчения управления версиями с разным криптографическим согласованием.

Реестр QUIC Versions, организованный в [QUIC], позволяет выполнять предварительную регистрацию для экспериментов. Регистрация важна для предотвращения конфликтов, в том числе с экспериментальными версиями. Экспериментальные версии не следует использовать в течение длительного времени или регистрировать постоянно, чтобы минимизировать риск взятия отпечатков (fingerprinting) по номеру версии.

14. Развёртывание новых версий

QUIC версии 1 не задаёт механизма согласования версии в базовой спецификации, но [QUIC-VERSION-NEGOTIATION] предлагает расширение, обеспечивающее совместимое согласование версии.

Этот подход использует трехэтапный механизм развёртывания, обеспечивающий поэтапное развёртывание и эксперименты с разными версиями в большой серверной системе. В этом подходе все серверы в системе должны воспринимать соединения, использующие новую версию (этап 1) до того, как какой-либо сервер анонсирует её (этап 2), а аутентификация новой версии (этап 3) выполняется лишь после полного развёртывания анонсов новой версии.

Подробности приведены в разделе 5 [QUIC-VERSION-NEGOTIATION].

15. Доставка дейтаграмм по протоколу QUIC без гарантий

[RFC9221] задаёт расширение QUIC для передачи приёма дейтаграмм по протоколу QUIC без гарантий. В отличие от работы напрямую через UDP, приложениям, использующим службу дейтаграмм QUIC, не требуется реализовать свой механизм контроля перегрузок в соответствии с [RFC8085], поскольку для дейтаграмм QUIC обеспечивается контроль перегрузок.

Для дейтаграмм QUIC нет управления потоком данных, поэтому такие блоки данных могут отбрасываться при перегрузке получателя. Тогда как надёжный транспортный сервис QUIC обеспечивает на уровне потока интерфейс для упорядоченной передачи и приёма данных в нескольких потоках QUIC, служба дейтаграмм имеет интерфейс неупорядоченной доставки сообщений. При необходимости может применяться кадрирование прикладного уровня для мультиплексирования раздельных потоков дейтаграмм без гарантий в одном соединении QUIC.

16. Взаимодействие с IANA

Этот документ не требует действий IANA. Однако раздел 8 рекомендует приложениям, которые уже зарегистрировали порт TCP, но хотят задать QUIC в качестве транспорта, регистрировать порт UDP, идентичный заданному для TCP.

17. Вопросы безопасности

См. обсуждение вопросов безопасности в [QUIC] и [QUIC-TLS]. Соображения безопасности для базового транспортного протокола применимы и к использующим QUIC приложениям. Вопросы связывания, атак с воспроизведением (replay) и случайных значений, рассмотренные в [QUIC-TLS], следует цчитывать при развёртывании и применении QUIC.

Кроме того, переход на другой адрес раскрывает связь адреса клиента с сервером и может также раскрывать связь с путём, если идентификатор соединения не меняется или поток данных может быть сопоставлен иным способом. При поддержке переноса требуется учитывать это в плане конфиденциальности пользователей.

Разработчикам приложений следует учитывать, что при любом откате, применяемом при невозможности использовать QUIC по причине блокировки UDP в сети, следует обеспечивать такие же свойства защиты, как в QUIC. Если это невозможно, соединению не следует позволять приложению явно выполнять откат к менее безопасному варианту (см. 2. Необходимость отката).

В [QUIC-HTTP] даны рекомендации по вопросам безопасности для HTTP. Однако такие вопросы, как кросс-протокольные атаки, анализ и заполнение трафика, перенос могут быть актуальны и для других приложений, применяющих QUIC.

18. Литература

18.1. Нормативные документы

[QUIC] Iyengar, J., Ed. and M. Thomson, Ed., "QUIC: A UDP-Based Multiplexed and Secure Transport", RFC 9000, DOI 10.17487/RFC9000, May 2021, <https://www.rfc-editor.org/info/rfc9000>.

[QUIC-INVARIANTS] Thomson, M., "Version-Independent Properties of QUIC", RFC 8999, DOI 10.17487/RFC8999, May 2021, <https://www.rfc-editor.org/info/rfc8999>.

[QUIC-TLS] Thomson, M., Ed. and S. Turner, Ed., "Using TLS to Secure QUIC", RFC 9001, DOI 10.17487/RFC9001, May 2021, <https://www.rfc-editor.org/info/rfc9001>.

18.2. Дополнительная литература

[Edeline16] Edeline, K., Kühlewind, M., Trammell, B., Aben, E., and B. Donnet, "Using UDP for Internet Transport Evolution", DOI 10.48550/arXiv.1612.07816, 22 December 2016, <https://arxiv.org/abs/1612.07816>.

[Hatonen10] Hätönen, S., Nyrhinen, A., Eggert, L., Strowes, S., Sarolahti, P., and M. Kojo, "An Experimental Study of Home Gateway Characteristics", Proc. ACM IMC 2010, November 2010, <https://conferences.sigcomm.org/imc/2010/papers/p260.pdf>.

[HTTP-REPLAY] Thomson, M., Nottingham, M., and W. Tarreau, "Using Early Data in HTTP", RFC 8470, DOI 10.17487/RFC8470, September 2018, <https://www.rfc-editor.org/info/rfc8470>.

[PaaschNanog] Paasch, C., "Network support for TCP Fast Open", NANOG 67 Presentation, 13 June 2016, <https://www.nanog.org/sites/default/files/Paasch_Network_Support.pdf>.

[QUIC-ACK-FREQUENCY] Iyengar, J. and I. Swett, "QUIC Acknowledgement Frequency", Work in Progress, Internet-Draft, draft-ietf-quic-ack-frequency-02, 11 July 2022, <https://datatracker.ietf.org/doc/html/draft-ietf-quic-ack-frequency-02>.

[QUIC-HTTP] Bishop, M., Ed., "HTTP/3", RFC 9114, DOI 10.17487/RFC9114, June 2022, <https://www.rfc-editor.org/info/rfc9114>.

[QUIC-LB] Duke, M., Banks, N., and C. Huitema, "QUIC-LB: Generating Routable QUIC Connection IDs", Work in Progress, Internet-Draft, draft-ietf-quic-load-balancers-14, 11 July 2022, <https://datatracker.ietf.org/doc/html/draft-ietf-quic-load-balancers-14>.

[QUIC-MANAGEABILITY] Kühlewind, M. and B. Trammell, "Manageability of the QUIC Transport Protocol", RFC 9312, DOI 10.17487/RFC9312, September 2022, <https://www.rfc-editor.org/info/rfc9312>.

[QUIC-RETRY] Duke, M. and N. Banks, "QUIC Retry Offload", Work in Progress, Internet-Draft, draft-ietf-quic-retry-offload-00, 25 May 2022, <https://datatracker.ietf.org/doc/html/draft-ietf-quic-retry-offload-00>.

[QUIC-VERSION-NEGOTIATION] Schinazi, D. and E. Rescorla, "Compatible Version Negotiation for QUIC", Work in Progress, Internet-Draft, draft-ietf-quic-version-negotiation-10, 27 September 2022, <https://datatracker.ietf.org/doc/html/draft-ietf-quic-version-negotiation-10>.

[RFC1034] Mockapetris, P., "Domain names - concepts and facilities", STD 13, RFC 1034, DOI 10.17487/RFC1034, November 1987, <https://www.rfc-editor.org/info/rfc1034>.

[RFC2475] Blake, S., Black, D., Carlson, M., Davies, E., Wang, Z., and W. Weiss, "An Architecture for Differentiated Services", RFC 2475, DOI 10.17487/RFC2475, December 1998, <https://www.rfc-editor.org/info/rfc2475>.

[RFC5077] Salowey, J., Zhou, H., Eronen, P., and H. Tschofenig, "Transport Layer Security (TLS) Session Resumption without Server-Side State", RFC 5077, DOI 10.17487/RFC5077, January 2008, <https://www.rfc-editor.org/info/rfc5077>.

[RFC5382] Guha, S., Ed., Biswas, K., Ford, B., Sivakumar, S., and P. Srisuresh, "NAT Behavioral Requirements for TCP", BCP 142, RFC 5382, DOI 10.17487/RFC5382, October 2008, <https://www.rfc-editor.org/info/rfc5382>.

[RFC5905] Mills, D., Martin, J., Ed., Burbank, J., and W. Kasch, "Network Time Protocol Version 4: Protocol and Algorithms Specification", RFC 5905, DOI 10.17487/RFC5905, June 2010, <https://www.rfc-editor.org/info/rfc5905>.

[RFC6335] Cotton, M., Eggert, L., Touch, J., Westerlund, M., and S. Cheshire, "Internet Assigned Numbers Authority (IANA) Procedures for the Management of the Service Name and Transport Protocol Port Number Registry", BCP 165, RFC 6335, DOI 10.17487/RFC6335, August 2011, <https://www.rfc-editor.org/info/rfc6335>.

[RFC6762] Cheshire, S. and M. Krochmal, "Multicast DNS", RFC 6762, DOI 10.17487/RFC6762, February 2013, <https://www.rfc-editor.org/info/rfc6762>.

[RFC7301] Friedl, S., Popov, A., Langley, A., and E. Stephan, "Transport Layer Security (TLS) Application-Layer Protocol Negotiation Extension", RFC 7301, DOI 10.17487/RFC7301, July 2014, <https://www.rfc-editor.org/info/rfc7301>.

[RFC7413] Cheng, Y., Chu, J., Radhakrishnan, S., and A. Jain, "TCP Fast Open", RFC 7413, DOI 10.17487/RFC7413, December 2014, <https://www.rfc-editor.org/info/rfc7413>.

[RFC7657] Black, D., Ed. and P. Jones, "Differentiated Services (Diffserv) and Real-Time Communication", RFC 7657, DOI 10.17487/RFC7657, November 2015, <https://www.rfc-editor.org/info/rfc7657>.

[RFC7838] Nottingham, M., McManus, P., and J. Reschke, "HTTP Alternative Services", RFC 7838, DOI 10.17487/RFC7838, April 2016, <https://www.rfc-editor.org/info/rfc7838>.

[RFC8085] Eggert, L., Fairhurst, G., and G. Shepherd, "UDP Usage Guidelines", BCP 145, RFC 8085, DOI 10.17487/RFC8085, March 2017, <https://www.rfc-editor.org/info/rfc8085>.

[RFC8981] Gont, F., Krishnan, S., Narten, T., and R. Draves, "Temporary Address Extensions for Stateless Address Autoconfiguration in IPv6", RFC 8981, DOI 10.17487/RFC8981, February 2021, <https://www.rfc-editor.org/info/rfc8981>.

[RFC9218] Oku, K. and L. Pardue, "Extensible Prioritization Scheme for HTTP", RFC 9218, DOI 10.17487/RFC9218, June 2022, <https://www.rfc-editor.org/info/rfc9218>.

[RFC9221] Pauly, T., Kinnear, E., and D. Schinazi, "An Unreliable Datagram Extension to QUIC", RFC 9221, DOI 10.17487/RFC9221, March 2022, <https://www.rfc-editor.org/info/rfc9221>.

[SSDP] Donoho, A., Roe, B., Bodlaender, M., Gildred, J., Messer, A., Kim, Y., Fairman, B., and J. Tourzan, "UPnP Device Architecture 2.0", 17 April 2020, <https://openconnectivity.org/upnp-specs/UPnP-arch-DeviceArchitecture-v2.0-20200417.pdf>.

[Swett16] Swett, I., "QUIC Deployment Experience @Google", IETF96 QUIC BoF Presentation, 20 July 2016, <https://www.ietf.org/proceedings/96/slides/slides-96-quic-3.pdf>.

[TAPS-ARCH] Pauly, T., Trammell, B., Brunstrom, A., Fairhurst, G., and C. Perkins, "An Architecture for Transport Services", Work in Progress, Internet-Draft, draft-ietf-taps-arch-14, 27 September 2022, <https://datatracker.ietf.org/doc/html/draft-ietf-taps-arch-14>.

[TLS13] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <https://www.rfc-editor.org/info/rfc8446>.

[Trammell16] Trammell, B. and M. Kühlewind, "Internet Path Transparency Measurements using RIPE Atlas", RIPE 72 MAT Presentation, 25 May 2016, <https://ripe72.ripe.net/wp-content/uploads/presentations/86-atlas-udpdiff.pdf>.

Благодарности

Большое спасибо рецензентам Last Call - Chris Lonvick и Ines Robles.

Эта работа частично поддерживалась в рамках гранта Еврокомиссии Horizon 2020 № 688421 Measurement and Architecture for a Middleboxed Internet (MAMI) и контракта № 15.0268 с Государственным секретариатом Швейцарии по образованию, исследованиям и инновациям. Эта поддержка не предполагает одобрения (endorsement).

Участники работы

Ниже перечислены те, кто внёс существенный вклад в текст документа или предоставил отклики.

Gorry Fairhurst
Ian Swett
Igor Lubashev
Lucas Pardue
Mike Bishop
Mark Nottingham
Martin Duke
Martin Thomson
Sean Turner
Tommy Pauly

Адреса авторов

Mirja Kühlewind
Ericsson
Email: mirja.kuehlewind@ericsson.com

Brian Trammell
Google Switzerland GmbH
Gustav-Gull-Platz 1
CH-8004 Zurich
Switzerland
Email: ietf@trammell.ch

Перевод на русский язык

Николай Малых

nmalykh@protokols.ru