https://doi.org/10.26160/2572-4347-2025-23-25-32

АНАЛИЗ ПАРАЛЛЕЛЬНЫХ ПОТОКОВ В HTTP/3 С ИСПОЛЬЗОВАНИЕМ ИНСТРУМЕНТА QVIS

Хабаров С.П.
Санкт-Петербургский государственный лесотехнический университет им. С.М. Кирова, Санкт-Петербург, Россия

Ключевые слова: HTTP/3, QUIC, QVIS, параллелизм, мультиплексирование, HoLB, qlog, aioquic.
Аннотация. В статье рассмотрена возможность использования инструмента QVIS для анализа параллельных потоков в HTTP/3. На основе разработанного клиента на Python с использованием библиотеки aioquic проведено исследование мультиплексирования и подтверждено отсутствие блокировки в начале очереди. Показана эффективность QVIS для визуализации поведения QUIC и HTTP/3.

ANALYSIS OF PARALLEL STREAMS IN HTTP/3 USING THE QVIS TOOL

Khabarov S.P.
Saint-Petersburg State Forest Technical University, Saint-Petersburg, Russia

Keywords: HTTP/3, QUIC, QVIS, parallelism, multiplexing, HoLB, qlog, aioquic.
Abstract. This paper examines the use of the QVIS tool for analyzing parallel streams in HTTP/3. Based on a custom Python client developed with the aioquic library, an investigation of multiplexing is conducted, confirming the absence of head-of-line blocking The effectiveness of QVIS for visualizing QUIC and HTTP/3 behavior is demonstrated

Введение. Современные веб-приложения всё активнее используют протокол HTTP/3, построенный на основе QUIC [1], который призван устранить ключевые ограничения HTTP/2. В первую очередь, блокировку начала очереди (head-of-line blocking). Одно из основных преимуществ HTTP/3 – это поддержка истинного параллелизма за счёт независимости потоков на транспортном уровне. Однако для анализа такого поведения требуются специальные инструменты, способные визуализировать низкоуровневые сетевые события.

В данной работе исследуется возможность использования системы qvis [2] для анализа параллельных потоков в HTTP/3. С этой целью был разработан простейший, но функционально полный, клиент на Python с использованием библиотеки aioquic, который способен подключаться к HTTP/3-серверу, парсить HTML, извлекать ресурсы параллельно в рамках одного QUIC-соединения и генерировать qlog-файлы для последующей визуализации процесса сетевого взаимодействия.

В качестве тестового сайта был выбран www.cloudflare.com, так как он полностью поддерживает HTTP/3, содержит много <img> ресурсов, что

позволяет активировать большое число потоков, а также обеспечивать стабильность и высокую доступность.

Постановка задачи. Разработать методику анализа параллельных потоков в HTTP/3 с использованием qvis, основанную на визуализации поведения разработанного клиента при массовой параллельной загрузке ресурсов, и подтвердить наличие эффективного мультиплексирования при отсутствии блокировки начала очереди. Такой подход позволяет создать прозрачную, контролируемую и воспроизводимую среду для проведения исследований особенностей работы HTTP/3, чего сложно достичь при использовании браузеров или других “черных ящиков”.

Реализация HTTP/3-клиента. Общая структура клиента включает в себе три основных компонента, которые обеспечивают реализацию соединения по HTTP/3, параллельную загрузку ресурсов и управление процессом скачивания. Код клиента построен на базе библиотеки aioquic, обеспечивающей поддержку QUIC и HTTP/3 в асинхронной среде Python.

```
import asyncio, os
from urllib.parse import urljoin, urlparse
from bs4 import BeautifulSoup
from aioquic.asyncio import connect, QuicConnectionProtocol
from aioquic.quic.configuration import QuicConfiguration
from aioquic.h3.connection import H3Connection
from aioquic.quic.logger import QuicFileLogger

class H3Client(QuicConnectionProtocol):
    . . .
    def quic_event_received(self, event):
    . . .
    async def send_request(self, host: str, path: str):
    . . .
async def download_resource(client, host, path, file_path):
     . . .
async def main(host, output_dir):
     . . .
asyncio.run(main("www.cloudflare.com", "cloudflare_img"))
```

Основным компонентом является класс H3Client, наследуемый от QuicConnectionProtocol, который реализует логику HTTP/3-соединения, включая обработку QUIC-событий и отправку/приём HTTP/3 запросов. В этом классе описаны два метода.

Метод quic_event_received обрабатывает входящие события от QUIC-стека, извлекает через H3Connection заголовки, данные, а также с помощью asyncio.Event сигнализирует о завершении потока. Метод send_request предназначен для отправки GET-запросов по указанному пути, ожидания ответа и возврата тела ресурса вместе с заголовками.

Второй компонент – это асинхронная функция download_resource, отвечающая за загрузку одного ресурса (например, изображения). Она вызывает send_request для получения данных, сохраняет их в локальный файл и выводит сообщение о завершении загрузки. Функция разработана как независимая асинхронная задача, что позволяет запускать множество таких загрузок параллельно в рамках одного соединения.

Третий компонент – функция main, которая управляет общим процессом. Она устанавливает соединение с сервером, загружает и парсит главную страницу (“/”) с помощью BeautifulSoup, после чего:

– извлекает все ссылки на изображения (<img src="...">);

– формирует абсолютные URL и определяет имена файлов;

– создаёт асинхронные задачи для загрузки каждого ресурса;

– запускает все задачи параллельно с помощью asyncio.gather.

Именно в этой функции реализован весь функционал, необходимый для исследования параллельных потоков при массовой загрузке статических ресурсов с внешнего сервера.

```
async def main(host, output_dir):
    try:
        cfg = QuicConfiguration(is_client=True, alpn_protocols=["h3"])
        cfg.quic_logger = QuicFileLogger("Logs")
        async with connect(host, 443, configuration=cfg,
                           create_protocol=H3Client) as client:
            response_data, _ = await client.send_request(host, path="/")
            html = response_data.decode("utf-8", errors="ignore")
            soup = BeautifulSoup(html, "html.parser")

            #  Формируем список задач для параллельной загрузки
            download_tasks = []
            for tag in soup.find_all('img', src=True):
                url = tag.get('src')
                full_url = urljoin(f"https://{host}", url)
                u = urlparse(full_url)
                filename = os.path.basename(u.path)
                if not filename: continue
                file_path = os.path.join(output_dir, filename)

                #  Создаём конкретную задачу (не await!)
                task = download_resource(client, u.netloc, u.path, file_path)
                download_tasks.append(task)

            # 4. Загружаем все задачи параллельно
            n = len(download_tasks)
            print(f"\n Запуск параллельной загрузки {n} ресурсов...")
            await asyncio.gather(*download_tasks, return_exceptions=True)

    except Exception as e:
        print(f" Ошибка: {e}")
```

Клиент настраивается на работу с сайтом cloudflare.com, который поддерживает HTTP/3 и содержит множество изображений. Это позволит активировать большое число потоков и зафиксировать их поведение. Логирование с помощью QuicFileLogger генерирует .qlog-файлы, которые необходимы для последующей визуализации в qvis.

Таким образом, код данного клиента представляет собой минимальную, но функционально полную платформу для исследования параллелизма в HTTP/3, которая сочетает простоту настройки, полный контроль над сетевыми операциями и совместимость с инструментами анализа протоколов.

Тестирование работы клиента. Тестирование выполнялось на компьютере с операционной системой Windows 10 с использованием Python 3.10.11 и aioquic 1.2.0. Целью тестирования являлось не только подтверждение работоспособности всех компонентов приложения, но и демонстрация ключевого преимущества HTTP/3 – мультиплексирования независимых потоков в рамках одного соединения.

На рисунке 1 слева представлен скриншот вывода программы во время выполнения параллельной загрузки, а справа содержимое локальной папки, куда сохранялись все 77 ресурсов, полученные с cloudflare.com.

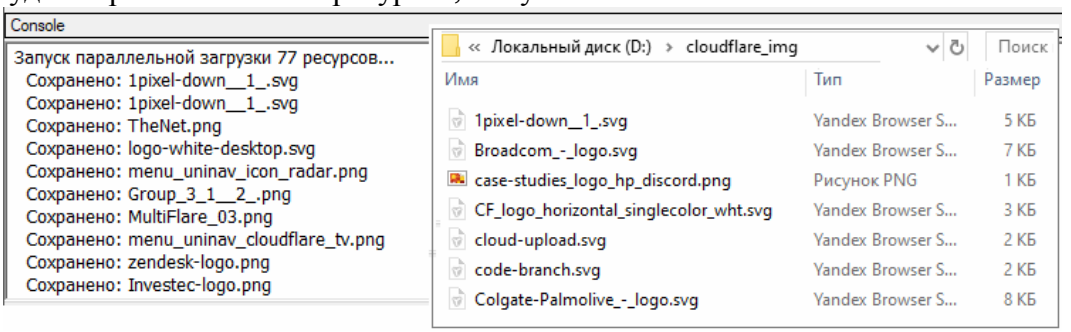

Рис. 1. Скриншоты процесса работы HTTP/3-клиента

Следует отметить, что в процессе всех запусков программы в папке Logs формировались .qlog-файлы, которые в JSON-формате содержат детальные логи всех QUIC- и HTTP/3-событий, таких как установление соединения, передача данных и управление потоками. Именно эти файлы и позволяют выполнять визуализацию в qvis.

Сочетание целенаправленно разработанного клиента, реального веб-ресурса с высокой нагрузкой по ресурсам и визуализации в qvis позволяет провести глубокий анализ поведения HTTP/3 и подтвердить все его преимущества в части параллелизма и эффективности передачи данных.

Визуализация параллельных потоков в qvis. Для проведения анализа поведения HTTP/3-соединения и оценки степени параллелизма была использована система визуализации qvis (QUIC and HTTP/3 Visualization Tool). Она представляет собой веб-интерфейс, который дает возможность исследовать трассировки, представленные в формате qlog, и наглядно представлять события на уровне QUIC и HTTP/3.

На рисунке 2 приведена вкладка Multiplexing, которая демонстрирует поведение клиента при параллельной загрузке множества ресурсов (изображений) с сайта www.cloudflare.com. Данные получены с помощью реализованного на базе aioquic HTTP/3-клиента, логирование которого осуществлялось с помощью QuicFileLogger.

Рис. 2. Диаграмма активности параллельных потоков

В верхней части рисунка представлена диаграмма, отображающая временные интервалы активности каждого потока (stream_id). Каждая горизонтальная полоса соответствует одному HTTP/3-потоку, а её длина – времени от первого до последнего события передачи данных.

Из анализа диаграммы следует, что почти все потоки начинаются практически одновременно после установки соединения. Это показывает, что клиент инициирует множество запросов параллельно, без ожидания завершения предыдущего. Такое поведение было бы невозможно при HTTP/2 с использованием одного TCP-соединения без дополнительных оптимизаций, поскольку там потоки также мультиплексируются, но подвержены блокировке в начале очереди (head-of-line blocking).

В нижней части рисунка 2 – диаграмма мультиплексированного потока данных (Multiplexed data flow), которая представляет собой временную последовательность полученных STREAM-фреймов. Каждый цветной прямоугольник соответствует одному фрейму из определённого потока. Положение по оси времени указывает момент получения STREAM-фрейма, а ширина — продолжительность его передачи.

В данном примере наблюдается плотное переплетение фреймов от различных потоков. Отсутствие длинных "чистых" блоков одного цвета говорит о том, что данные передаются интерливировано (перемешены), без чёткого разделения по потокам. Это свидетельствует о том, что QUIC-

транспорт эффективно мультиплексирует данные нескольких логических потоков в рамках одного UDP-соединения.

Надо отметить, что поскольку каждый поток является независимой сущностью в QUIC, то потеря пакета одного потока, не влияет на пакеты других потоков. Это принципиальное отличие от HTTP/2, где потеря TCP-сегмента может заблокировать весь поток данных, даже если другие логические потоки готовы к передаче.

Передача данных в HTTP/3 осуществляется на уровне логических потоков (stream), где каждый ресурс (например, изображение или CSS-файл) передаётся в одном или нескольких STREAM-фреймах. Процесс их отправки в сеть включает в себя несколько этапов, реализуемых на уровне QUIC-транспорта. Сначала HTTP/3-стек сервера формирует STREAM-фрейм, содержащий идентификатор потока (stream_id), смещение (offset) и часть полезной нагрузки (например, фрагмент файла).

Этот фрейм передаётся в QUIC-транспорт, который принимает решение об его упаковке в QUIC-пакет. QUIC-транспорт анализирует несколько факторов: текущий размер пакета; доступное пространство с учётом MTU и возможность слить нескольких фреймов в один пакет. При этом допускается упаковка в один QUIC-пакет одного или нескольких STREAM-фреймов из разных потоков, или даже части одного STREAM-фрейма вместе с другими типами фреймов (ACK, CRYPTO и др).

После формирования пакета к нему добавляется QUIC-заголовок, включающий номер пакета, идентификатор соединения и флаги. Затем весь пакет шифруется с использованием TLS 1.3, и только после этого передаётся через UDP-сокет как единая дейтаграмма. Таким образом, один QUIC-пакет может содержать данные из нескольких независимых потоков, обеспечивая мультиплексирование на уровне транспорта. В этом его отличие от HTTP/2, где все потоки мультиплексируются на уровне приложения и передаются в рамках одного TCP-соединения, которое подвержено блокировкам.

Исследовать и провести анализ описанного выше процесса можно с использованием того же инструмента qvis, используя другие его режимы работы. В частности, на вкладке Packetization (рис. 3) qvis позволяет получить структуру передачи данных с точки зрения клиента.

Верхняя часть скриншота отображает процесс отправки данных со стороны сервера. Каждый чёрный прямоугольник соответствует одному QUIC-пакету, а красные полосы – количеству фреймов внутри него. Плотное переплетение фреймов из разных потоков в одном QUIC-пакете демонстрирует эффективное мультиплексирование, которое характерно для протокола QUIC. Нижняя часть показывает небольшой объём данных, отправленных клиентом – преимущественно начальные запросы. Клиент инициирует соединение, сервер передаёт данные, клиент подтверждает их получение. Как видно из диаграммы, HTTP/3 позволяет серверу быстро и параллельно передавать ресурсы, без зависимости от ответов клиента.

Рис. 3. Диаграмма упаковки данных в qvis

Анализ этой диаграммы подтверждает, что HTTP/3 вместе с QUIC обеспечивают истинное мультиплексирование, эффективную упаковку данных и отсутствие head-of-line blocking, так как даже при большом объёме данных каждый поток работает независимо.

Заключение. Проведенное исследование показало, что кастомный HTTP/3-клиент может обеспечить массовую параллельную загрузку ресурсов по протоколу HTTP/3, а qvis пригоден для детального анализа поведения HTTP/3-приложений, включая исследование параллелизма, тестирование производительности и диагностику задержек.

Предложенный подход, как сочетание собственного клиента + qlog + qvis может быть использован как для тестирования беспроводных сетей [3,4], так и для исследования влияния сетевых условий (задержка, потери) на параллелизм,

Анализ трассировки в qvis позволяет сделать следующий вывод, что HTTP/3 обеспечивает истинную параллельность, когда клиент может открывать десятки потоков одновременно. При этом фреймы разных потоков передаются вперемешку, что оптимизирует использование канала и повышает скорость работы сетевых приложений.

Список литературы

1. IETF RFC 9000: QUIC: A UDP-Based Multiplexed and Secure Transport – Internet Engineering Task Force (IETF), 2021. – [Электронный ресурс]. – Режим доступа: https://datatracker.ietf.org/doc/rfc9000/
2. QUIC and HTTP/3 visualization toolsuite[Электронный ресурс]. – Режим доступа: https://qvis.quictools.info/#/files/
3. Бойцов А.К., Хабаров С.П. Современные беспроводные технологии в лесном хозяйстве // Актуальные вопросы в лесном хозяйстве: Материалы III международной научно-практической конференции. – СПб.: Полиграф-Экспресс, 2019. – С. 138-141.

4. Думов М.И., Хабаров С.П. Использование OMNET++ для моделирования беспроводных Wi-Fi сетей // Информационные системы и технологии: теория и практика: Сборник научных трудов. Выпуск 10. Часть 1. – СПб.: СПбГЛТУ им. С.М. Кирова, 2018. – С. 44-53.

References

1. IETF RFC 9000: QUIC: A UDP-Based Multiplexed and Secure Transport - Internet Engineering Task Force (IETF), 2021. [Electronic resource]. – Access mode: https://datatracker.ietf.org/doc/rfc9000/
2. QUIC and HTTP/3 visualization toolsuite [Electronic resource]. – Access mode: https://qvis.quictools.info/#/files/
3. Boytsov A.K., Khabarov S.P. Modern Wireless Technologies in Forestry // Actual Issues in Forestry: Proceedings of the III International Scientific and Practical Conference. – SPb.: Poligraf-Ekspress, 2019. – P. 138-141.
4. Dumov M.I., Khabarov S.P. Using OMNET++ for Modeling Wireless Wi-Fi Networks // Information Systems and Technologies: Theory and Practice: Collection of Scientific Papers. Issue 10. Part 1. – SPb.: SPbSFU n.a. S.M. Kirov, 2018. – P. 44-53.

Хабаров Сергей Петрович – кандидат технических наук, старший научный сотрудник, доцент, доцент кафедры информационных систем и технологий	**Khabarov Sergey Petrovich** – candidate of technical sciences, senior researcher, associate professor, associate professor of the Department of information systems and technologies
Serg.Habarov@mail.ru	

Received 25.08.2025