https://doi.org/10.26160/2474-5901-2025-49-67-70

РЕАЛИЗАЦИЯ QUIC-КЛИЕНТА И СЕРВЕРА С ПОДДЕРЖКОЙ ВОЗОБНОВЛЕНИЯ СЕССИЙ ЧЕРЕЗ SESSION TICKETS

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

Ключевые слова: QUIC, session tickets, RTT, возобновление сессий, aioquic, Python, early data, мультиплексирование потоков.
Аннотация. Протокол QUIC улучшает веб-соединения по сравнению с TCP, интегрируя TLS для шифрования и снижения задержек. В статье представлен пример реализации QUIC-клиента и сервера, основанный на библиотеке aioquic, с акцентом на использование session tickets для возобновления сессий. Это позволяет сократить время отклика (RTT) до 0-RTT, повышая эффективность. Рассматриваются алгоритм работы, диаграммы и примеры выполнения, показывающие снижение задержек в приложениях с высокими требованиями к скорости, таких как видео и онлайн-игры.

IMPLEMENTATION OF QUIC CLIENT AND SERVER WITH SESSION RESUMPTION SUPPORT VIA SESSION TICKETS

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

Keywords: QUIC, session tickets, RTT, session resumption, aioquic, Python, early data, stream multiplexing.
Abstract. The QUIC protocol enhances web connections compared to TCP by integrating TLS for encryption and delay reduction. The article presents an example implementation of a QUIC client and server based on the aioquic library, focusing on session tickets for session resumption. This reduces RTT to 0-RTT, improving efficiency. The algorithm, diagrams, and execution examples demonstrate lower delays in high-speed applications like video and online games.

Введение. Протокол QUIC (Quick UDP Internet Connections) – это современный транспортный протокол на базе UDP, разработанный для ускорения веб-соединений. Он сочетает шифрование TLS, мультиплексирование потоков (параллельную передачу нескольких данных без блокировок) и снижение задержек по сравнению с TCP [1, 2]. Ключевой особенностью является возобновление сессий через *session tickets* – зашифрованные токены, которые сервер выдает клиенту после первого соединения. Это позволяет повторно подключаться без полного рукопожатия, снижая время круглого пути (RTT) до 0-RTT [3]. RTT — это время, за которое пакет данных проходит до получателя и обратно; оно используется для контроля перегрузки сети и адаптации скорости передачи [4]. В QUIC начальная оценка RTT (RTT0) часто берется из предыдущих сессий, что ускоряет адаптацию, особенно при смене сетей (например, с Wi-Fi на мобильный интернет) [5]. Без возобновления каждое соединение требует полного рукопожатия, увеличивая задержку.

Реализация. В этой работе реализованы клиент и сервер на Python с библиотекой *aioquic*, демонстрируя *session tickets* для оптимизации RTT, позволяя быстро возобновлять соединения без полного обмена данными каждый раз. На рисунке 1 показана диаграмма первого (1-RTT) и повторного (0-RTT) соединений.

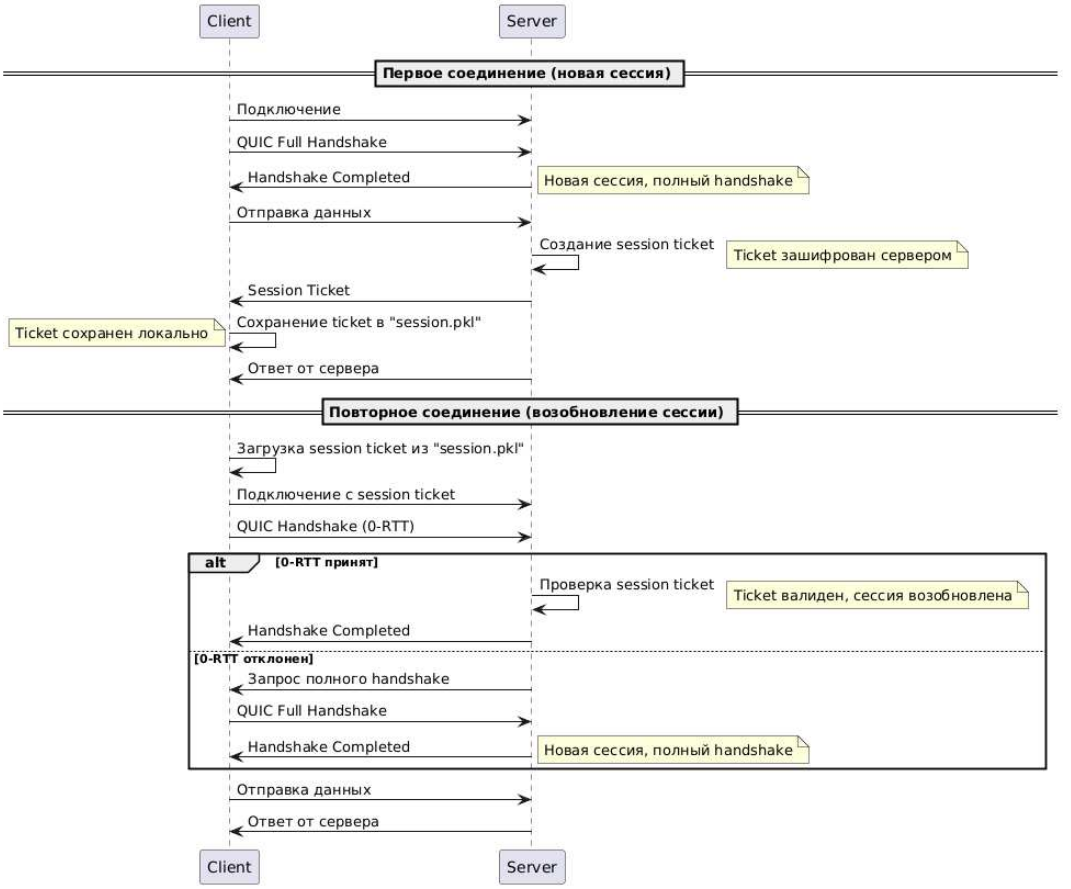

Рис. 1. Диаграмма последовательности работы взаимодействия сервера и клиента для первого (1-RTT) и повторного (0-RTT) соединений

При первом подключении клиент устанавливает соединение с сервером, и они проходят полный процесс рукопожатия (QUIC Full Handshake), который занимает один круг (1-RTT). После этого клиент отправляет тестовое сообщение, а сервер отвечает и создаёт *session ticket* – зашифрованный "пропуск" с параметрами сессии. Этот ticket отправляется клиенту, который сохраняет его в файл *session.pkl*. Без этого *ticket* каждое новое соединение требовало бы полного рукопожатия, что замедляет процесс.

При повторном подключении клиент загружает сохранённый *ticket* из файла *session.pkl* и отправляет его серверу вместе с запросом. Это позволяет сразу передать данные без ожидания полного обмена (0-RTT). Сервер проверяет *ticket*: если он валиден, сессия возобновляется мгновенно, и сервер подтверждает подключение. Если *ticket* не подходит, сервер запрашивает

полное рукопожатие, повторяя процесс как при первом соединении. После этого клиент отправляет данные, а сервер отвечает, используя сохранённые настройки для ускорения.

На рисунке 2 показан пример логов выполнения при повторном подключении (сессия возобновлена, данные отправлены без полного рукопожатия, *ticket* использован).

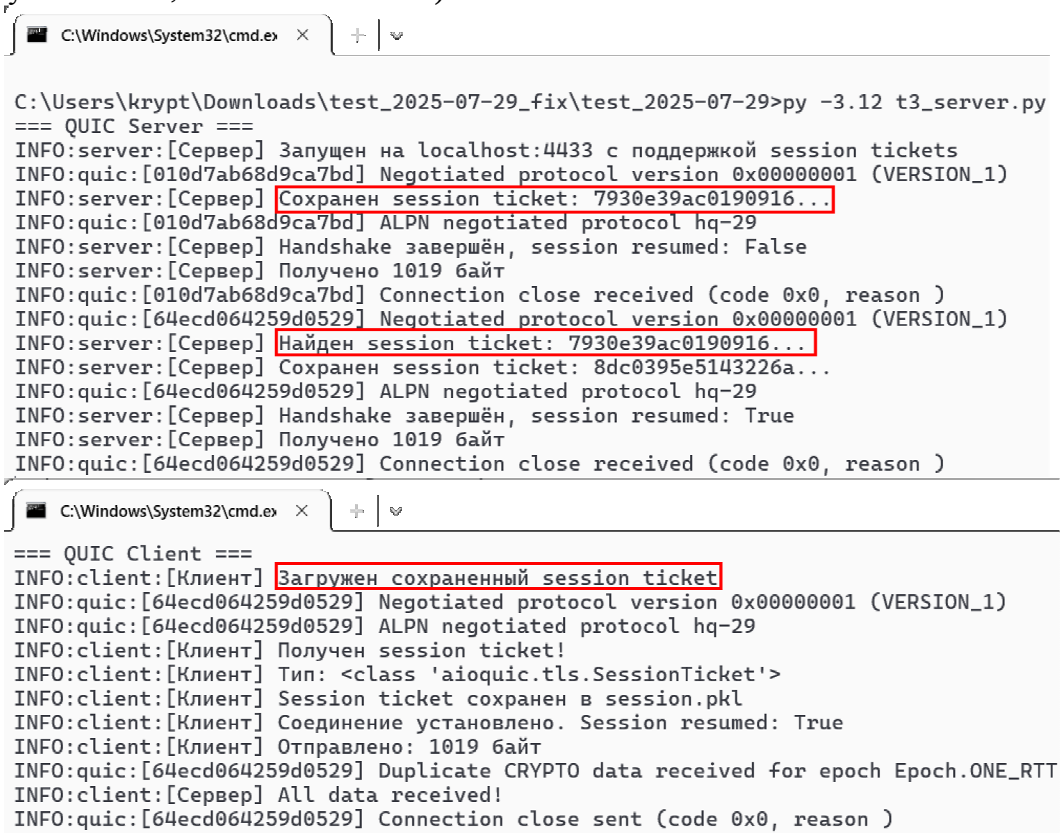

Рис. 2. Работа приложения при повторном подключении клиента, формирование session ticket

Заключение. Реализация показывает, как *session tickets* помогают ускорить работу QUIC, снижая время ожидания (RTT) до нуля (0-RTT) и позволяя отправлять данные сразу при подключении. Это делает работу быстрее для приложений, где важна скорость, таких как видео или онлайн-игры.

Список литературы / References

1. Thomson M., Turner S. Using TLS to Secure QUIC. – Internet Engineering Task Force (IETF), 2021. – 58 p. – doi.org/10.17487/RFC9001.
2. Iyengar J., Thomson M. QUIC: A UDP-Based Multiplexed and Secure Transport. – Internet Engineering Task Force (IETF), 2021. – 151 p. – doi.org/10.17487/RFC9000.
3. Rüth J., Bitsch A., Wehrle K. Session Resumption Protocols and Efficient Forward Security for TLS 1.3 // Journal of Cryptology. 2021, vol. 34, p. 20. – doi.org/10.1007/s00145-021-09385-0.

4. Chatzoglou E., Kouliaridis V., Karopoulos G., Kambourakis G. Revisiting QUIC attacks: a comprehensive review on QUIC security and a hands-on study // International Journal of Information Security. 2022, vol. 22, pp. 347-365. doi.org/10.1007/s10207-022-00630-6.
5. Rescorla E. The Transport Layer Security (TLS) Protocol Version 1.3. – Internet Engineering Task Force (IETF), 2018. – 160 p. – doi.org/10.17487/RFC8446.

Маклашин Денис Игоревич – аспирант	**Maklashin Denis Igorevich –** postgraduate student
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
densuper2005@gmail.com	

Received 14.09.2025