DOI: 10.15514/ISPRAS-2025-37(6)-16

Исследование эффективности планировщиков протокола MPQUIC в зависимости от алгоритмов контроля перегрузки

[1,2] *М.В. Попов, ORCID: 0009-0007-7774-2220 <mpopov@ispras.ru>*
[1,3] *И.А. Степанов, ORCID: 0009-0003-1964-5001 <ivan_mipt@ispras.ru>*
[1,2,3,4] *А.И. Гетьман, ORCID: 0000-0002-6562-9008 <ever@ispras.ru>*

[1] *Институт системного программирования им. В.П. Иванникова РАН, 109004, Россия, г. Москва, ул. А. Солженицына, д. 25.*

[2] *Московский государственный университет имени М.В. Ломоносова, 119991, Россия, г. Москва, Ленинские горы, д. 1.*

[3] *Московский физико-технический институт, 141700, Россия, Московская область, г. Долгопрудный, Институтский пер., 9.*

[4] *Национальный исследовательский университет «Высшая школа экономики», 101978, Россия, г. Москва, ул. Мясницкая, д. 20.*

Аннотация. В последние годы широкую популярность получил протокол QUIC, как альтернатива TCP. Кроме того, в настоящее время широко внедряется и исследуется технология Multipath, реализованная в протоколе MPQUIC. Центральным компонентом протокола MPQUIC является планировщик, принимающий решение по какому пути и в какой момент времени отправить следующие пакеты данных. Существуют реализации планировщиков как на основе эвристических правил, так и на основе обучения с подкреплением. На данный момент поведение планировщиков в различных, с точки зрения характеристик путей, средах изучено подробно. Однако вопрос их эффективности в зависимости от используемых алгоритмов контроля перегрузки недостаточно освящён. В данной работе представлена реализация различных планировщиков и исследование их эффективности в зависимости от алгоритма контроля перегрузки. Полученные результаты, на основе проведённых экспериментов, говорят о том, что планировщик может эффективно работать в сетевой среде с определённым алгоритмом контроля перегрузки, но при этом быть не эффективным в среде с другим алгоритмом контроля перегрузки.

Ключевые слова: Многопутевой планировщик; протокол QUIC; протокол MPQUIC; алгоритмы контроля перегрузки; обучение с подкреплением.

Для цитирования: Попов М.В., Степанов И.А., Гетьман А.И. Исследование эффективности планировщиков протокола MPQUIC в зависимости от алгоритмов контроля перегрузки. Труды ИСП РАН, том 37, вып. 6, часть 2, 2025 г., стр. 7–20. DOI: 10.15514/ISPRAS–2025–37(6)–16.

Research of the Effectiveness of MPQUIC Protocol Schedulers Depending on the Congestion Control Algorithms

[1,2] *M.V. Popov, ORCID: 0009-0007-7774-2220 <mpopov@ispras.ru>*
[1,3] *I.A. Stepanov, ORCID: 0009-0003-1964-5001 <ivan_mipt@ispras.ru>*
[1,2,3,4] *A.I. Getman, ORCID: 0000-0002-6562-9008 <ever@ispras.ru>*

[1] *Ivannikov Institute for System Programming of the Russian Academy of Sciences, 25, Alexander Solzhenitsyn st., Moscow, 109004, Russia.*
[2] *Lomonosov Moscow State University, 1 Leninskie Gory, Moscow, 119991 Russia.*
[3] *Moscow Institute of Physics and Technology (National Research University), 9 Institutskiy per., Dolgoprudny, Moscow Region, 141701, Russia.*
[4] *National Research University «Higher School of Economics», 20 Myasnitskaya ulitsa, Moscow 101000 Russia.*

Abstract. In recent years, the QUIC protocol has become widely popular as an alternative to TCP. In addition, Multipath technology implemented in the MPQUIC protocol is currently being widely implemented and researched. The central component of the MPQUIC protocol is the scheduler, which decides which path and at which time to send the next data packets. There are implementations of schedulers based on both heuristic rules and reinforcement learning. At the moment, the behavior of schedulers in various environments has been studied in detail in terms of path characteristics. However, the issue of their effectiveness, depending on the congestion control algorithms used, is not sufficiently sanctified. This paper presents the implementation of various schedulers and a study of their effectiveness depending on congestion control. The results obtained suggest that the scheduler can work effectively in a network environment with a certain congestion control algorithm, but it may not be effective in an environment with a different congestion control algorithm.

Keywords: Multipath scheduler; QUIC protocol; MPQUIC protocol; congestion control algorithms; reinforcement learning.

For citation: Popov M.V., Stepanov I.A., Getman A.I. Research of the effectiveness of MPQUIC protocol schedulers depending on the Congestion Control Algorithms. Trudy ISP RAN/Proc. ISP RAS, vol. 37, issue 6, part 2, 2025, pp. 7-20 (in Russian). DOI: 10.15514/ISPRAS-2025-37(6)-16.

1. Введение

Multipath (многопутевая передача данных) — это технология, позволяющая одновременно использовать несколько сетевых интерфейсов (4G, 5G, Wi-Fi и т.д.) для передачи данных. Главная цель данной технологии заключается в повышении производительности и отказоустойчивости. Первоначально данная технология была реализована для протокола TCP в виде многопутевого протокола MPTCP [1-3].

В последнее время протокол QUIC стал альтернативой протоколу TCP и получил популярность благодаря высокой производительности. Поэтому в последствии были реализованы технологии multipath для QUIC в виде протокола MPQUIC [4-5].

Принципиальное отличие протокола QUIC от протокола MPQUIC представлено на рис. 1.

Центральным элементом протокола MPQUIC является планировщик – компонент, отвечающий за распределение данных между несколькими доступными сетевыми путями таким образом, чтобы обеспечивать высокую производительность, отказоустойчивость и минимизацию задержек. Планировщик взаимодействует с алгоритмом контроля перегрузки. Стоит отметить, что эти два компонента решают одну общую задачу – оптимизацию производительности, но используют разные методы. Планировщик решает, как распределить данные между несколькими путями, а алгоритмы контроля перегрузки (Reno, Cubic, BBR) управляют интенсивностью передачи данных на каждом пути, чтобы избежать перегрузок сети.

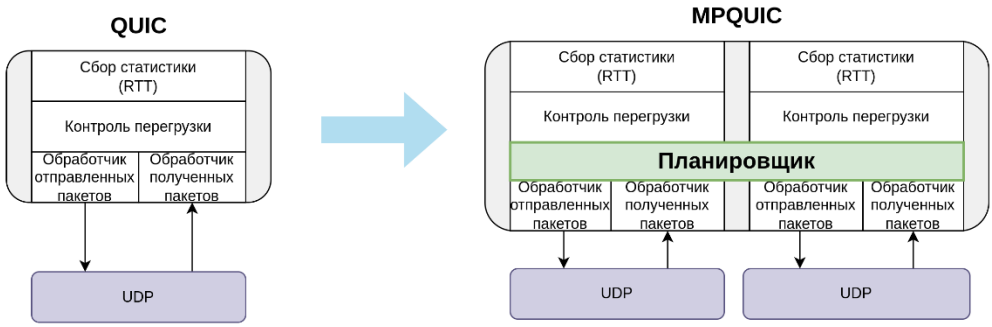

Рис. 1. протокол QUIC и протокол MPQUIC.
Fig. 1. QUIC protocol and MPQUIC protocol.

В настоящее время существующие планировщики можно разбить на два класса: основанные на эвристических правилах и основанные на обучении с подкреплением. К недостаткам первых относится неэффективная работа в динамически меняющихся средах, к недостаткам вторых - высокая вычислительная сложность и возможное переобучение. Стоит отметить, что на работу планировщика может значительно влиять алгоритм контроля перегрузки, который управляет скоростью передачи данных. Однако в известных нам работах не представлены исследования эффективности реализованных планировщиков в зависимости от алгоритмов контроля перегрузки.

В данной работе представлены реализации планировщиков как на основе обучения с подкреплением, так и на основе эвристических правил, и представлено сравнение их эффективности в зависимости от использованного алгоритма контроля перегрузки.

Остальная часть работы организована следующим образом. В разделе 2 представлена постановка задачи и даны основные определения по данной теме. В разделе 3 описаны популярные планировщики, основанные как на эвристических правилах, так и на обучении с подкреплением. Раздел 4 содержит описание планировщиков на основе обучения с подкреплением, реализованных в данной работе. В разделе 5 представлены эксперименты эффективности реализованных планировщиков в различных сетевых сценариях и при использовании различных алгоритмов контроля перегрузки. В разделе 6 подводятся итоги на основе полученных результатов и обсуждаются направления дальнейших исследований.

2. Постановка задачи

В данном разделе будут рассмотрены базовые определения, связанные с данной темой и необходимые для постановки задачи.

2.1 Определения

Задержка (delay) – время, необходимое для передачи данных от отправителя к получателю.

Время приема-передачи (Round trip time, RTT) – время, необходимое для передачи данных от отправителя к получателю и получения подтверждения, что данные были получены.

SRTT (Сглаженное RTT): $SRTT_n = \frac{7}{8} SRTT_{n-1} + \frac{1}{8} RTT_{last}$, RTT_{last} — последнее значение данного параметра.

CWND (окно перегрузки): максимальное количество данных (в байтах или сегментах), которое отправитель может передать получателю через сеть до получения подтверждения о доставке этих данных без учёта окна приёма RWND.

RWND (окно получателя): окно приёма, которое указывает, сколько данных готов принять получатель (определяется буфером получателя).

SWND (окно отправления): максимальное количество данных (в байтах), которые отправитель может передать получателю до получения подтверждений (ACK). SWND определяется как $min(CWND, RWND)$.

Bytes in Flight (P, байты в полёте): число отправленных ещё не подтверждённых байт.

Пропускная способность (throughput) – отношение объёма данных, проходящих через сеть (канал) за заданный интервал времени, к данному интервалу времени. Пропускная способность является изменяемой величиной и показывает скорость отправки данных, в то время как **ширина канала (bandwidth)** – это величина неизменная, являющаяся максимально возможной пропускной способностью.

Capacity (мера свободы канала) – отношение пропускной способности к ширине канала в данный момент времени.

Loss (потери) пакетов – число потерянных пакетов за некоторый интервал времени.

Loss_rate (скорость потери пакетов) – отношение числа потерянных пакетов за заданный интервал времени к данному интервалу времени.

2.2 Постановка задачи Multipath

Как уже отмечалось ранее, технология Multipath предназначена для повышения производительности и отказоустойчивости. Более строго постановку данной задачи можно записать следующим образом. Пусть существует N доступных путей, каждый из которых имеет следующие характеристики:

- RTT_i- RTT i-го пути
- L_i - процент потерь i-го пути
- r_i - пропускная способность i-го пути
- C_i - ширина i-го канала

Тогда задача технологии Multipath выглядит следующим образом:

$$\alpha \sum_{i=1}^{N} r_i - \beta \sum_{i=1}^{N} RTT_i - \gamma \sum_{i=1}^{N} L_i \to max \quad (1)$$
$$r_i < C_i \forall i \in [1 \dots N]$$

α, β, γ - некоторые числовые коэффициенты.

Стоит отметить, что задачи максимизации пропускной способности и минимизации RTT могут быть ортогональными. Так как для увеличения пропускной способности необходимо максимально использовать доступную ширину канала сети. Это достигается за счёт более агрессивного заполнения буферов маршрутизаторов и коммутаторов, что позволяет передавать больше данных. Это в свою очередь приводит к возникновению очередей, что увеличивает RTT. Поэтому в данной постановке задачи реализованный алгоритм должен соблюдать баланс между минимизацией RTT и максимизацией пропускной способности.

С точки зрения обучения с подкреплением агент, который является планировщиком, выбирает действие на основе некоторой политики. Действием обычно является выбор некоторого пути или ожидания освобождения окна перегрузки. После выбора действия среда генерирует новое состояние и награду. Состоянием является некоторое описание характеристики сети в данный момент. Наградой выступают метрики, значения которых оптимизируются в формуле 1.

2.3 Алгоритмы контроля перегрузки

Как уже отмечалось ранее в данной работе большое внимание уделено исследованию технологии Multipath для различных алгоритмов контроля перегрузки, таких как CUBIC, BBRv1 и BBRv2. Упомянутые алгоритмы являются наиболее популярными на данный момент, поэтому были выбраны в качестве исследований. Данные алгоритмы были впервые

реализованы для протокола TCP, однако впоследствии появились реализации для QUIC, учитывающие особенность данного протокола.

New Reno (1999 г.) [6]. Данный алгоритм является улучшенной версией классического алгоритма TCP Reno (1990 г.), который включал в себя экспоненциальное или линейное увеличение окна перегрузки в случае работы без потерь и уменьшение окна перегрузки в случае потерь. Хоть New Reno крайне стабильный алгоритм, однако, зачастую, использует ширину канала не столь эффективно, как алгоритмы, появившиеся позднее, и поэтому в данной работе он подробно не исследуется.

Cubic (2005 г.) [7]. Вместо линейного увеличения окна перегрузки, данный алгоритм использует кубическую функцию, которая зависит от времени, прошедшего с момента последней потери пакета. Таким образом CUBIC эффективен в каналах с высокой пропускной способностью и на данный момент является одним из самых популярных алгоритмов контроля перегрузки.

BBRv1 (2016 г.) [8]. В отличие от предыдущих алгоритмов, определяющих перегрузку по потери пакетов, данный алгоритм использует модель, основанную на оценке ширины канала и минимальной задержке для определения оптимального размера окна. Этот размер окна соответствует идеальному объему данных, который может быть передан по сети без перегрузки и без необходимости дожидаться потерь пакетов для корректировки скорости передачи. Таким образом BBRv1 использует ширину канала максимально эффективно, однако может не обеспечивать равномерное распределение пропускной способности.

BBRv2 (2018 г.). Вторая версия алгоритма BBR стремится к лучшей справедливости при разделении полосы пропускания с другими соединениями, устраняя таким образом недостатки первой версии. Однако BBRv2 в силу своей справедливости использует ширину канала не столь эффективно.

3. Обзор существующих решений

В данном разделе будут рассмотрены популярные планировщики, основанные как на эвристических правилах, так и на обучении с подкреплением.

3.1 Планировщики, основанные на эвристических правилах

3.1.1 Round Robin (RR). Один из самых простых планировщиков, который последовательно и циклично отправляет пакеты по доступным путям. Данный планировщик не требует сложных вычислений состояний пути, что неизменно является его достоинством. Однако стоит отметить существенные недостатки данного метода. Во-первых, в том случае, если один путь заведомо лучше других, планировщик не будет учитывать данное обстоятельство и эффективно использовать лучший путь. Во-вторых, в случае динамически меняющейся среды Round Robin не может адаптироваться к новым условиям. Так как данный планировщик не учитывает состояние сетевой среды, в которой работает, то в дальнейшем рассматриваться Round Robin не будет.

3.1.2 MinRTT. RTT является важной характеристикой пути, указывающей на задержку при передаче данных. Планировщик minRTT учитывает эту характеристику, отправляя пакет в путь с минимальным значением RTT, в том случае, если путь доступен. Это может быть важно для приложений, чувствительных к задержкам (видеоприложения, VoIP и т.д.). Данный планировщик может быстро реагировать на изменения в состоянии сети, в том случае если RTT путей увеличивается или уменьшается. Однако использование в качестве метрики эффективности только RTT может быть недостаточным в ряде случаев. К примеру, в том случае, если первый путь имеет низкий RTT и низкую пропускную способность, а второй путь имеет RTT чуть выше и очень высокую пропускную способность, алгоритм будет передавать пакеты по первому пути, что ограничивает общую производительность, так как второй более эффективный путь не используется. Вторая проблема планировщика

minRTT при выборе пути с минимальным RTT заключается в том, что если в какой-то момент быстрый путь становится недоступным, то пакеты отправляются в медленный путь. Поэтому может возникнуть ситуация, при которой быстрый путь будет свободен, однако пакеты были отправлены по медленному пути.

3.1.3 BLEST. Планировщик BLEST [9] отчасти решает проблему планировщика minRTT следующим образом: в том случае, если быстрый путь недоступен, BLEST оценивает будущее заполнение окон перегрузки на уровне соединения. В том случае, если оценка покажет, что пути будут заполнены, планировщик ожидает, а не отправляет пакеты в более медленный второй путь. Стоит отметить, что в своих оценках BLEST помимо RTT использует также CWND и число байт в полёте для оценки эффективности пути.

3.1.4 ECF. Данный планировщик [10] выбирает путь с минимальным RTT. В том случае, если данный путь не доступен, то ECF выбирает путь с минимальным ожидаемым временем доставки. Оценка ожидаемого времени доставки также основана на информации об RTT и CWND, что отличает данный планировщик от minRTT с точки зрения оценки эффективности путей.

3.2 Планировщики, основанные на обучении с подкреплением

Главным недостатком планировщиков, основанных на эвристических правилах, является их низкая адаптируемость к динамически меняющимся условиями среды в силу того, что они работают на основе определённых правил. В данном подразделе будут рассмотрены популярные планировщики на основе RL и выделены их особенности с точки зрения задачи обучения с подкреплением.

3.2.1 Peekaboo. Данный планировщик [11] основан на алгоритме Linear Upper Confidence Bound, являющимся улучшением классического алгоритма UCB. В качестве состояний авторами выбраны нормализованные значения отношений CWND, SWND и числа байт в полёте (P) к RTT для каждого из двух путей. Таким образом, вектор состояний имеет 6 элементов. Награда представляет собой сумму наград за каждое действие с учётом коэффициента дисконтирования. Награда за каждое действие (отправки пакета) определяется как отношение между размером пакета в байтах и временем, прошедшим с момента передачи пакета до получения подтверждения (ACK). Набор действий зависит от доступности путей. Авторы рассматривают два случая. Если доступен только один путь, набор действий включает в себя: передачу по этому пути или ожидание до тех пор, пока другой путь снова не станет доступным; если доступны оба пути, то набор действий включает в себя: передачу пакета по первому пути или же передачу пакета по второму пути. Авторами был проведён анализ работы алгоритма в различных сетевых средах с точки зрения ширины канала, задержки, дисперсии RTT и уровня потерь по сравнению с классическими планировщиками, такими как: RR, minRTT, BLEST и ECF.

3.2.2 M-Peekaboo. Данный планировщик [12] был представлен как расширение планировщика Peekaboo для сетей 5G. С точки зрения обучения с подкреплением M-Peekaboo отличался от своего предшественника незначительно. Главным вкладом авторов было исследование планировщика в сценарии сетей 5G, то есть сетей со следующими параметрами: ширина канала (Bandwidth) = 1100Mbps, RTT = 27.4 ± 6.4 ms, доля потерь пакетов = 0.01.

3.2.3 Планировщик MPQUIC на основе DQN. Данный планировщик [13] основан на глубоком Q-обучении и был исследован в сценарии потоковой передачи видео. Одним из особенностей данного планировщика является принятие решений не для каждого отдельного пакета, а для всех в течении определённого интервала времени (50ms). Это сделано по двум причинам: во-первых, частое принятие решений требует высоких вычислительных затрат, во-вторых, из-за задержки в сети требуется время, чтобы узнать, был ли успешно передан запланированный пакет, то есть было ли успешным то или иное действие. Авторы расширили

пространство состояний, включив помимо классических RTT, CWND, также SRTT, размер окна отправки (SWND) и т.д. В качестве действий выбрано случайное действие с некоторой вероятностью или же выбор пути с минимальным RTT. Реализованный планировщик исследовался в следующих сетевых сценариях: сеть без потерь и с разнообразным RTT для путей, сеть с потерями и одинаковыми путями с точки зрения RTT.

3.2.4 Планировщик MPQUIC на основе DQN для 5G. Данный планировщик [14] был предложен для сетей 5G и использовал классические состояния: SRTT, CWND и число байт в полёте. Итоговая награда определялась как средняя пропускная способность в мегабитах в секунду. Кроме того, в качестве вознаграждения был предусмотрен штраф при достижении некоторого количества пакетов без подтверждения или повторных пакетов. В качестве действий авторами было выбрана отправка в лучший путь (с точки зрения RTT) или ожидание. Основой планировщика является нейронная сеть, представленная классической архитектурой для задачи DQN: небольшое число скрытых слоёв с функцией активацией ReLU. К сожалению, эксперименты были проведены в узком диапазоне сетевых характеристик и не в реальных сетевых условиях, но в тоже время, в данной работе было представлено сравнение эффективности передачи данных при использовании технологии Multipath с передачей данных без использования данной технологии.

3.2.5 FALCON. Ещё один планировщик [15], разработанный для сценария 5G. Данный планировщик основан на DQN и является продолжение планировщика M-Peekaboo, однако главным его отличием является парадигма адаптированного обучения. Использование адаптированного обучения повышает эффективность планировщика в различных сетевых средах, рассматриваемых авторами: в средах с высокой пропускной способностью, низкой пропускной способностью и так далее. Сетевые среды исследовались с точки зрения характеристик сетевых каналов, однако исследования сред с точки зрения алгоритмов контроля перегрузки выполнено не было. Сравнение планировщиков на основе обучения с подкреплением представлено в табл. 1.

4. Реализованные планировщики на основе обучения с подкреплением

Помимо классических планировщиков minRTT, ECF и BLEST в данной работе были реализованы планировщики на основе Q-learning, DQN и LinUCB. В данном разделе подробно описаны архитектуры последних трёх.

4.1 Q-learning

Планировщик, основанный на Q-обучении, вначале своей работы определяет путь с минимальным RTT (*path_f*). Затем планировщик определяет путь с минимальным RTT, который является доступным (*path_s*). Доступность определяется заполненностью окна перегрузки у пути. В том случае, если *path_f* не равен *path_s* запускается алгоритм Q-обучения.

Состояния: для каждого из двух активных путей вычисляется класс производительности (низкий, средний, высокий), полученный из оценки качества пути.

- **Низкий класс** – высокий RTT или большое количество срабатываний PTO или высокие потери.
- **Средний класс** – средний RTT, низкая пропускная способность пути или умеренные потери.
- **Высокий класс** – RTT низкий, потери малы, PTO не срабатывает.

Так как каждый путь может иметь 3 класса, а пути всего два, то, следовательно, среда имеет всего 9 состояний.

Действия: действие1 – отправить пакет по первому пути, действие2 — отправить пакет по второму пути.

Награда: $\alpha(throughtput_1 + throughtput_2) - \beta(RTT_1 + RTT_2)$.

При этом значения каждой из 4 переменных были нормализованы.

Табл. 1. Сравнение различных алгоритмов Multipath.
Table 1. Comparing different learning algorithms Multipath.

N	Год	Состояния	Награды	Действия
11	2020	CWND, SWND, lnP, RTT	throughtput	1. ожидание свободного пути 2. передача по быстрому пути
12	2021	CWND, RTT, lnP	throughtput	1. ожидание свободного пути 2. передача по быстрому пути
13	2022	CWND, SRTT, P и ещё 7 состояний	подтверждённые пакеты	1. ожидание свободного пути 2. передача по быстрому пути
14	2019	CWND, SRTT, P и ещё 3 состояния	throughput	1. отправка пакета в путь 1 2. отправка пакета в путь 2
15	2022	CWND, SWND, RTT, P	throughput	1. отправка пакета в путь 1 2. отправка пакета в путь 2

4.2 Deep Q-learning

В данном планировщике табличное представление функции качества Q(s,a), применявшееся в базовом Q-learning, заменено аппроксимацией на основе полносвязной нейронной сети. Полносвязная нейронная сеть имеет 6 входных признаков, два скрытых слоя по шесть нейронов и два выходных нейрона. В качестве функции активации использовалась ReLU.

Планировщик, основанный на глубоком Q-обучении, вначале своей работы определяет путь с минимальным RTT (*path_f*). Затем планировщик определяет путь с минимальным RTT, который является доступным (*path_s*). Доступность определяется заполненностью окна перегрузки у пути. В том случае, если *path_f* не равен *path_s* запускается алгоритм глубокого Q-обучения.

Состояния: $CWND_1$, $CWND_2$, $SRTT_1$, $SRTT_2$, Bytes in $Flight_1$, Bytes in $Flight_2$.

Действия: действие1 – ожидать освобождения первого (наиболее быстрого) пути, действие2 – отправить пакет по второму пути.

Награда: $\alpha(throughtput_1 + throughtput_2) - \beta(RTT_1 + RTT_2)$ (компоненты награды нормализованы).

4.3 Linear Upper Confidence Bound (UCB)

Основная идея данного планировщика была взята из [11]. Алгоритм принимает решение, по какому из двух активных путей отправить пакет, рассматривая это как задачу Linear Upper

Confidence Bound (LinUCB) с двумя действиями. Планировщик, основанный на LinUCB, вначале своей работы определяет путь с минимальным RTT (*путь F*) и путь с большим RTT, чем у пути F, но меньшим, чем у всех остальных (*путь S*). Если у *пути F* ещё есть свободное окно CWND, пакет сразу отправляется по нему, иначе запускается алгоритм обучения LinUCB.

Состояния x: для каждого из *путей F и S* рассчитывается отношение текущего окна CWND к усреднённому RTT, отношение RWND к усреднённому RTT и количество байт в полёте к усреднённому RTT соответствующего пути.

Действия: действие1 – отправить пакет по *пути F*, действие2 – отправить пакет по *пути S*. Для каждого действия хранится матрица A размера 6 × 6 и вектор b длиной 6 – вектор накопленных наград.

Награда: $\alpha(throughtput_1 + throughtput_2) - \beta(RTT_1 + RTT_2)$.

Основные этапы работы планировщика заключаются в следующем:

1. Для текущего вектора состояний x вычисляется прогноз эффективности каждого действия: $val_F = (A_F^{-1}b_F)^T x$ и $val_S = (A_S^{-1}b_S)^T x$.
2. В том случае, если $val_S < val_F$, то со случайной вероятностью 0.3 выбирается *путь S*, а в другом случае происходит ожидание *пути F*.
3. В том случае, если $val_S \geq val_F$, то со случайной вероятностью 0.9 выбирается *путь S*, а в другом случае происходит ожидание *пути F*.
4. Рассчитывается накопленная награда на основе предыдущей формулы.
5. Расчёт $theta_F = A_F^{-1} \cdot b_F$ и $theta_F = A_S^{-1} \cdot b_S$ следующим образом: $\theta[i] = \theta[i] + lr \cdot (reward - val)$.

Таким образом, данный планировщик **реализует** идеи эвристических методов, используя вероятностный выбор, и обучения с подкреплением, используя Linear Upper Confidence Bound (LinUCB).

5. Результаты

В данном разделе содержатся основные результаты, полученные при проведении экспериментов.

5.1 Экспериментальный стенд

Для проведения исследований использовалась среда Mininet [16], с помощью которой была создана топология, состоящая из MPQUIC-клиента и MPQUIC-сервера, каждый из которых имел два сетевых интерфейса. Эти интерфейсы были подключены к двум независимым маршрутам передачи данных, каждый из которых мог иметь различные характеристики. Топология сети представлена на рис. 2.

В ходе эксперимента клиент отправлял серверу файлы различного размера, используя протокол QUIC. В качестве конкретной реализации протокола QUIC была взята реализация XQUIC [17]. Стоит отметить, что в XQUIC присутствует всего один планировщик – minRTT. Поэтому в данной работе были реализованы также следующие планировщики: ECF, BLEST, основанный на Q-обучении, основанный на DQN, основанный на LinUCB. Для каждого сценария и каждого алгоритма контроля перегрузки было выполнено 10 запусков и замеров времени передачи файлов.

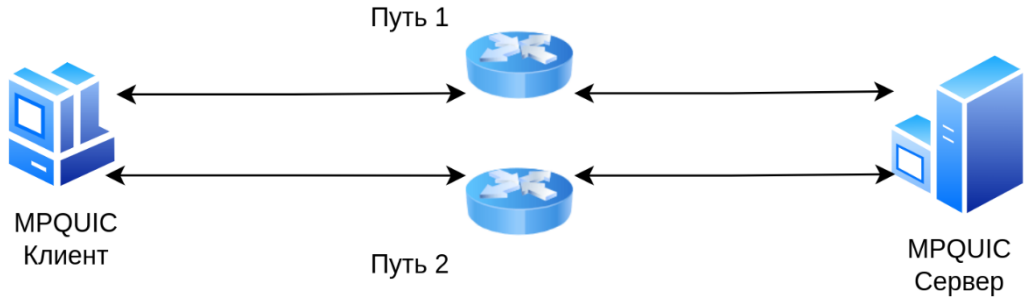

Рис. 2. Топология сети.
Fig. 2. Network topology.

5.2 Характеристики экспериментов

Наиболее важными с точки зрения характеристик путей являются: ширина канала, задержка, процент потерь и дисперсия задержки. Характеристики путей в различных экспериментах представлены в табл. 2.

Табл. 2. Характеристики путей.
Table 2. Paths characteristics.

N	Ширина каналов (Mb/s)		Задержка (мс)		Процент потерь (%)		Дисперсия задержки ($мс^2$)	
	Путь №1	Путь №2	Путь №1	Путь №2	Путь №1	Путь №2	Путь №1	Путь №2
1	70	20	40	10	0.5	1	5	5
2	70	20	10	40	0.5	1	5	5
3	70	70	15	15	0	0	0	0

Сценарий № 1 соответствует ситуации, при которой первый путь имеет высокую ширину канала и высокую задержку относительно второго пути. При этом, каждый путь имеет определенную дисперсию задержки и число потерь. Данный сценарий моделирует 4G или Wi-Fi соединение, при котором наблюдается нестабильное соединение. Также стоит отметить, что в таком случае сложно определить какой из двух путей является лучшим с точки зрения эффективности.

Сценарий № 2 соответствует ситуации, при которой первый путь имеет высокую ширину канала и низкую задержку относительно второго пути. При этом, каждый путь имеет определенную дисперсию задержки и число потерь. Данный сценарий также моделирует 4G или Wi-Fi соединение, при котором наблюдается нестабильное соединение. Главное отличие заключается в том, что в таком случае существует путь (путь 1), который является предпочтительнее с точки зрения эффективности.

Сценарий № 3 соответствует ситуации, при которой оба пути имеют одинаковые характеристики. При этом каждый путь не имеет дисперсию задержек и потерь. Данный сценарий моделирует 4G или Wi-Fi соединение, при котором наблюдается стабильное соединение.

5.3 Полученные результаты

Результаты первого эксперимента представлены на рис. 3. В качестве Q-learning, DQN и UCB планировщика выступали планировщики, подробно описанные в разделе 4.

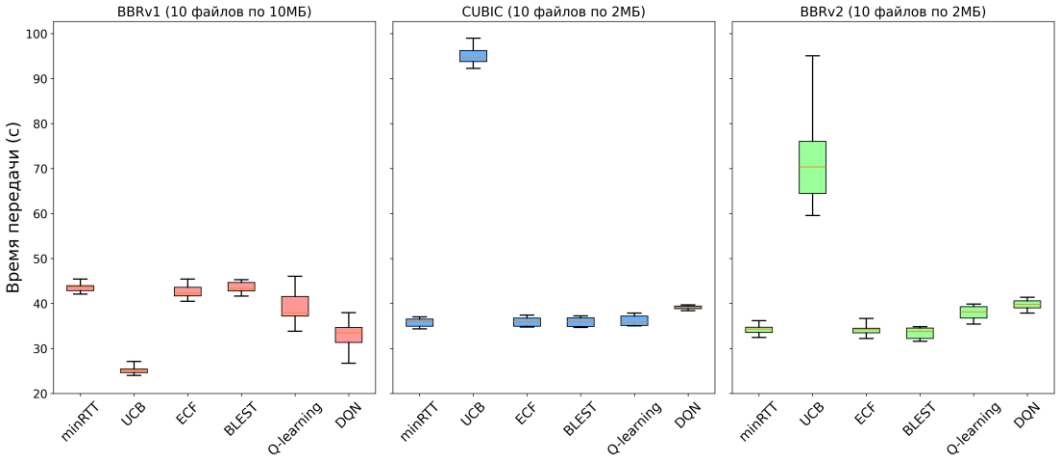

Рис. 3. Результаты экспериментов в сценарии №1.
Fig. 3. Results of experiments in scenario 1.

Планировщик UCB показал результат значительно лучше остальных в сценарии BBRv1, но в тоже время его эффективность в двух других сценариях оказалась крайне низкой. Отчасти это связано с его эвристической природой, в результате которой планировщик UCB был крайне эффективен в сценарии BBRv1 и не эффективен в двух других сценариях.

Планировщик, основанный на глубоком Q-обучении, показал результат лучше, чем minRTT, ECF и BLEST, а также Q-обучение в сценарии BBRv1. Это связано с динамически меняющимися условиями среды, так как планировщики, основанные на эвристических правилах, не всегда могут определить какой из двух путей является лучшим с точки зрения эффективности. В тоже время в сценарии CUBIC и BBRv2 планировщик, основанный на глубоком Q-обучении, показал результат немного хуже, чем minRTT, ECF и BLEST, а также Q-обучение. Это объясняется тем, что DQN был обучен в среде BBRv1.

Результаты второго эксперимента представлены на рис. 4. В качестве Q-learning, DQN и UCB планировщика выступали планировщики, подробно описанные в разделе 4.

В данном эксперименте поведение планировщиков в случае CUBIC и BBRv2 аналогично первому эксперименту. Однако результаты в случае BBRv1 несколько отличаются. Планировщик UCB работает лишь немного лучше других. Это связано с тем, что в данной среде существует путь (путь 1), который является предпочтительнее с точки зрения эффективности. Поэтому планировщики, основанные на эвристических правилах и Q-learning работают не так плохо по сравнению с UCB. В тоже время поведение и результаты планировщика, основанного на глубоком Q-обучении, аналогичны сценарию № 2.

Результаты третьего эксперимента представлены на рис. 5. В качестве Q-learning, DQN и UCB планировщика выступали планировщики, подробно описанные в разделе 4.

В данном эксперименте наблюдается примерно одинаковое поведение различных планировщиков с точки зрения эффективности. Это вполне объяснимо тем, что

характеристики путей являются идентичными и при этом каждый путь не имеет дисперсию задержек и потерь. Стоит также заметить, что планировщик, основанный на UCB, работает несколько хуже в силу эвристических правил.

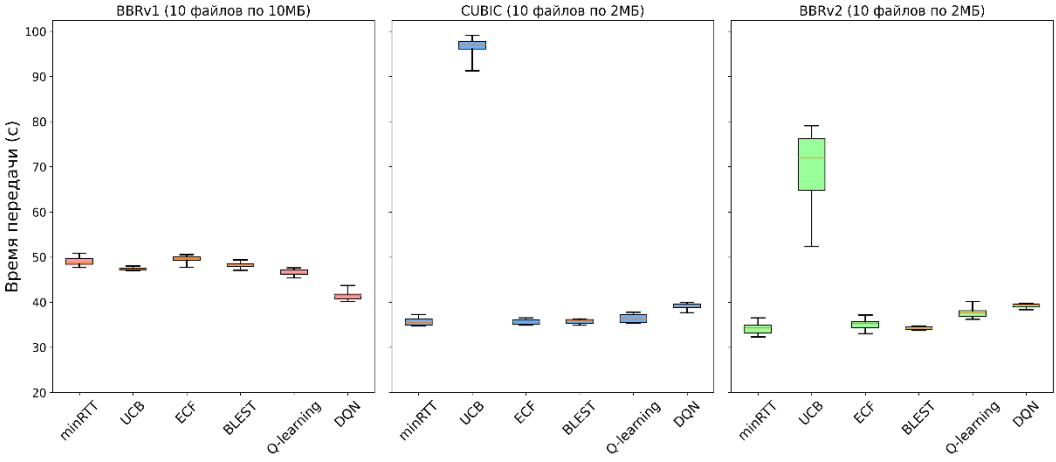

Рис. 4. Результаты экспериментов в сценарии №2.
Fig. 4. Results of experiments in scenario № 2.

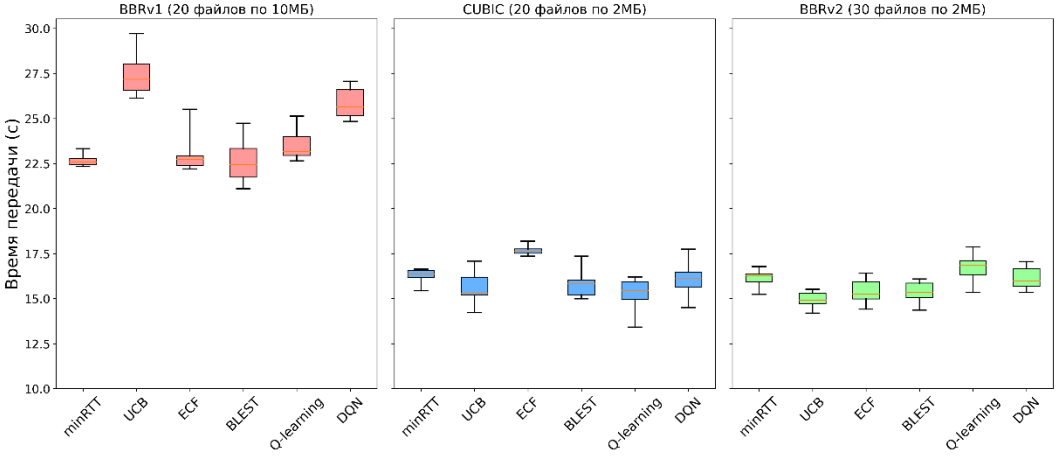

Рис. 5. Результаты экспериментов в сценарии №3.
Fig. 5. Results of experiments in scenario № 3.

5.4 Итоги

Итоговые результаты эффективности (средняя скорость в Мбит/с) планировщиков в зависимости от алгоритма контроля перегрузки представлены на рис. 6.

На основе полученных результатов необходимо отметить следующие:

1. Ни один из представленных планировщиков не оказался эффективным во всех сценариях;
2. DQN продемонстрировал самую стабильную работу;
3. UCB оказался эффективным в первом сценарии и крайне неэффективным в других;
4. Эффективность minRTT, ECF и BLEST примерно одинакова.

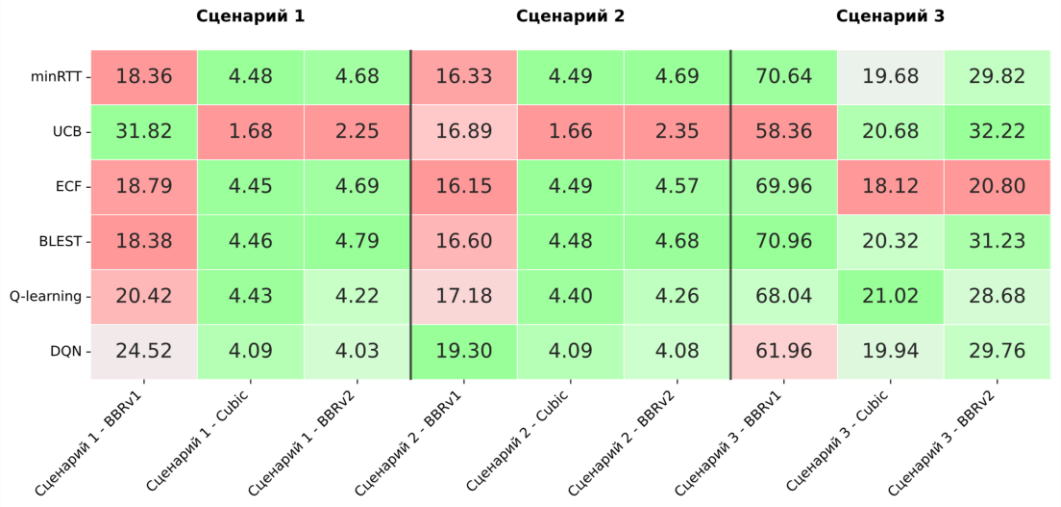

Рис. 6. Сравнительная таблица работы планировщиков.
Fig. 6. The comparative table of schedulers performance.

6. Выводы и направления будущих исследований

В ходе данной работы были исследованы и реализованы планировщики как основанные на эвристических правилах, так и на обучении с подкреплением. Реализованные планировщики исследовались не только с точки зрения характеристик путей, но и с точки зрения алгоритма контроля перегрузки. Полученные результаты говорят о том, что планировщик может эффективно работать в сетевой среде с определённым алгоритмом контроля перегрузки, но при этом быть неэффективным в среде с другим алгоритмом контроля перегрузки. Решение данной проблемы является открытым и представляет собой направление будущих исследований: разработка планировщика, работающего эффективно независимо от используемого алгоритма контроля перегрузки.

Список литературы / References

[1]. Wischik D. et al. Design, implementation and evaluation of congestion control for multipath {TCP} // 8th USENIX Symposium on Networked Systems Design and Implementation (NSDI 11). 2011.
[2]. Nguyen K. et al. An approach to reinforce multipath TCP with path-aware information // Sensors. 2019. Vol. 19, issue 3, p. 476.
[3]. Chao L. et al. A brief review of multipath tcp for vehicular networks // Sensors. 2021. Vol. 21, issue 8, p. 2793.
[4]. De Coninck Q., Bonaventure O. Multipath quic: Design and evaluation // Proceedings of the 13th international conference on emerging networking experiments and technologies. 2017, pp. 160-166..
[5]. Viernickel T. et al. Multipath QUIC: A deployable multipath transport protocol // 2018 IEEE International Conference on Communications (ICC). IEEE, 2018, pp. 1-7.
[6]. Floyd S., Henderson T. The NewReno modification to TCP's fast recovery algorithm. 1999. Issue rfc2582.
[7]. Ha S., Rhee I., Xu L. CUBIC: a new TCP-friendly high-speed TCP variant // ACM SIGOPS operating systems review. 2008. Vol. 42, issue 5, pp. 64-74.
[8]. Cardwell N. et al. Bbr: Congestion-based congestion control: Measuring bottleneck bandwidth and round-trip propagation time // Queue. 2016. Vol. 14, issue 5, pp. 20-53.
[9]. Simone Ferlin и др. “BLEST: Blocking estimation-based MPTCP scheduler for heterogeneous networks”. In: 2016 IFIP networking conference (IFIP networking) and workshops. IEEE. 2016, pp. 431-439.
[10]. Yeon-sup Lim и др. “ECF: An MPTCP path scheduler to manage heterogeneous paths”. In: Proceedings of the 13th international conference on emerging networking experiments and technologies. 2017, pp. 147-159.

[11]. Wu H. et al. Peekaboo: Learning-based multipath scheduling for dynamic heterogeneous environments // IEEE Journal on Selected Areas in Communications. 2020. Vol. 38, issue 10, pp. 2295-2310.
[12]. Wu H. et al. Multipath scheduling for 5G networks: Evaluation and outlook // IEEE Communications Magazine. 2021. Vol. 59, issue 4. pp. 44-50.
[13]. Lee S., Yoo J. Reinforcement learning based multipath QUIC scheduler for multimedia streaming // Sensors. 2022. Vol. 22, issue 17, p. 6333.
[14]. Roselló M. M. Multi-path scheduling with deep reinforcement learning // 2019 European Conference on Networks and Communications (EuCNC). IEEE, 2019, pp. 400-405.
[15]. Wu H. et al. Falcon: Fast and accurate multipath scheduling using offline and online learning // arXiv preprint arXiv:2201.08969. 2022.
[16]. Mininet [Электронный ресурс]. – URL: https://mininet.org/ (дата обращения: 15.05.2025).
[17]. XQUIC [Электронный ресурс]. – URL: https://github.com/alibaba/xquic (дата обращения: 15.05.2025).

Информация об авторах / Information about authors

Максим Владимирович ПОПОВ – старший лаборант ИСП РАН, студент магистратуры факультета ВМК МГУ. Сфера научных интересов: анализ сетевого трафика, алгоритмы контроля перегрузок.

Maxim Vladimirovich POPOV – graduate student at the Faculty of Computational Mathematics and Cybernetics of Moscow State University. Research interests: network traffic analysis, congestion control algorithms.

Иван Александрович СТЕПАНОВ – аспирант ИСП РАН, стажёр-исследователь ИСП РАН, ассистент кафедры информатики и вычислительной математики МФТИ. Сфера научных интересов: анализ сетевого трафика с помощью машинного обучения.

Ivan Alexandrovich STEPANOV – postgraduate student of the ISP RAS, intern researcher at ISP RAS, an assistant at the Department of Computer Science and Computational Mathematics at MIPT. Research interests: network traffic analysis using machine learning.

Александр Игоревич ГЕТЬМАН – кандидат физико-математических наук, старший научный сотрудник ИСП РАН, ассистент ВМК МГУ и МФТИ, доцент ВШЭ. Сфера научных интересов: анализ бинарного кода, восстановление форматов данных, анализ и классификация сетевого трафика.

Aleksandr Igorevich GETMAN – Cand. Sci. (Phys.-Math.), senior researcher at ISP RAS, assistant at CMC MSU and MIPT, associate professor at HSE. Research interests: binary code analysis, data format recovery, network traffic analysis and classification.