DOI: 10.15514/ISPRAS-2025-37(3)-18

Обзор методов контроля перегрузки с использованием машинного обучения

[1,2] *И.А. Степанов, ORCID: 0009-0003-1964-5001 <ivan_mipt@ispras.ru>*
[1,4] *М.В. Попов, ORCID: 0009-0007-7774-2220 <mpopov@ispras.ru>*
[1,2,3,4] *А.И. Гетьман, ORCID: 0000-0002-6562-9008 <ever@ispras.ru>*
[1] *М.К. Иконникова, ORCID: 0000-0003-1530-5133 <mikonnikova@ispras.ru>*
[1,4] *А.А. Белеванцев, ORCID: 0000-0003-2817-0397 <abel@ispras.ru>*

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

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

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

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

Аннотация. Контроль перегрузки является ключевым аспектом современных сетей. Первые алгоритмы контроля перегрузки, такие как TCP Tahoe и TCP Reno, были разработаны в конце XX века, и их основные идеи остаются актуальными до сих пор. С развитием высокоскоростных сетей для них были созданы специализированные алгоритмы, например, TCP BIC и TCP CUBIC. Однако классические алгоритмы, основанные на определённых правилах, не всегда оказываются эффективными во всех сетевых условиях, и с развитием 4G, 5G и спутниковой связи задача контроля перегрузки стала более актуальной. Это привело к появлению решений этой задачи на основе машинного обучения и обучения с подкреплением, в частности таких, которые способны адаптироваться к динамически изменяющимся условиям сети. В статье представлены и рассмотрены как классические алгоритмы контроля перегрузки, так и наиболее популярные и новые алгоритмы, основанные на машинном обучении, а также некоторые реализации с использованием технологии multipath. Кроме того, выделены наиболее значимые проблемы алгоритмов на основе машинного обучения и обсуждены потенциальные направления будущих исследований в данной области.

Ключевые слова: контроль перегрузки в сети; обучение с подкреплением.

Для цитирования: Степанов И.А., Попов М.В., Гетьман А.И., Иконникова М.К., Белеванцев А.А. Обзор методов контроля перегрузки с использованием машинного обучения. Труды ИСП РАН, том 37, вып. 3, 2025 г., стр. 251–276. DOI: 10.15514/ISPRAS–2025–37(3)–18.

Machine Learning Dased Congestion Control Methods: a Survey

[1,2] *I.A. Stepanov, ORCID: 0009-0003-1964-5001 <ivan_mipt@ispras.ru>*
[1,4] *M.V. Popov, ORCID: 0009-0007-7774-2220 <mpopov@ispras.ru>*
[1,2,3,4] *A.I. Getman, ORCID: 0000-0002-6562-9008 <ever@ispras.ru>*
[1] *M.K. Ikonnikova, ORCID: 0000-0003-1530-5133 < mikonnikova@ispras.ru >*
[1,4] *A.A. Belevantsev, ORCID: 0000-0003-2817-0397 <abel@ispras.ru>*

[1] *Ivannikov Institute for System Programming of the Russian Academy of Sciences, 25, Alexander Solzhenitsyn st., Moscow, 109004, Russia.*

[2] *Moscow Institute of Physics and Technology (National Research University) 9 Institutskiy per., Dolgoprudny, Moscow Region, 141701, Russia.*

[3] *National Research University «Higher School of Economics» 20 Myasnitskaya ulitsa, Moscow 101000 Russia.*

[4] *Lomonosov Moscow State University 1 Leninskie Gory, Moscow, 119991 Russia.*

Abstract. Congestion control is a key aspect of modern networks. The first congestion control algorithms, such as TCP Tahoe and TCP Reno, were developed in the late 20th century, and their core concepts remain relevant to this day. With the development of high-speed networks, specialized algorithms such as TCP BIC and TCP CUBIC were created, which are adapted to these conditions. However, classical algorithms with predefined rules are not always effective in all network environments, and with the rise of 4G, 5G, and satellite communications, the congestion control issue has become increasingly relevant. This has led to the emergence of numerous works on machine learning-based congestion control algorithms, particularly reinforcement learning, which can adapt to dynamically changing network conditions. This paper presents and reviews both classical congestion control algorithms and the most popular and recent machine learning-based algorithms, along with some implementations using multipath. Additionally, it highlights the most significant challenges of machine learning-based algorithms and discusses potential directions for future research in this field.

Keywords: network congestion control; reinforcement learning.

For citation: Stepanov I.A., Popov M.V., Getman A.I., Ikonnikova M.K., Belevantsev A.A. Overview of congestion control methods using machine learning. Trudy ISP RAN/Proc. ISP RAS, vol. 37, issue 3, 2025. pp. 251-276 (in Russian). DOI: 10.15514/ISPRAS-2025-37(3)-18.

1. Введение

Алгоритмы контроля перегрузки являются необходимым компонентом, обеспечивающим возможность функционирования компьютерных сетей в настоящее время. Перегрузка в сети возникает, когда объём передаваемых данных превышает пропускную способность маршрутов, по которым они передаются, что приводит к невозможности сети эффективно обрабатывать трафик. Это вызывает деградацию качества обслуживания, увеличенные задержки, потерю пакетов и снижение общей производительности сети.

В 1988 году Ван Джейкобсон в своей работе подробно описал проблему сетевой перегрузки, что привело к разработке первого алгоритма контроля перегрузки – TCP Tahoe. Этот алгоритм стал основой для последующих исследований и разработок в области контроля перегрузок, заложив фундамент для новых подходов.

Со временем алгоритмы контроля перегрузки развивались. Изначальные методы (Reno, Cubic) регулировали скорость передачи данных на основе обнаружения потерь. Позже появились подходы, учитывающие задержки в сети (Veno, Vegas), которые стремились предсказать перегрузку на основе изменения времени приема-передачи (round trip time, RTT). Эти алгоритмы позволяли в некоторых случаях более эффективно управлять передачей данных, предотвращая потери ещё до их возникновения. Позднее появились гибридные

алгоритмы (BBR, Copa), комбинирующие различные метрики, такие как пропускная способность и задержка.

С развитием технологий и усложнением сетевых сценариев традиционные алгоритмы управления перегрузкой сталкиваются с серьёзными ограничениями, так как они зачастую разрабатывались для конкретных условий и не обладают достаточной адаптивностью для работы в динамично изменяющихся сетях.

В последнее время появились новые технологии и сети, такие как WiFi, 4G, 5G и спутниковая связь, что привело к большому разнообразию сетевых сценариев. Классические алгоритмы контроля перегрузки не могут эффективно работать во всех существующих сценариях, к примеру, в средах, содержащих как проводные, так и беспроводные соединения. Кроме того, трудности создают среды, динамически меняющие свои характеристики.

Для того, чтобы решить представленные выше проблемы, разрабатываются алгоритмы на основе машинного обучения и, в частности, на основе обучения с подкреплением (RL). Главное отличие алгоритмов контроля перегрузки на основе RL от классических заключается в способности принимать решения на основе данных о сети в реальном времени, что позволяет им лучше адаптироваться к различным сетевым структурам и изменению состояния сети. Однако у алгоритмов, основанных на машинном обучении, есть ряд недостатков, о которых будет сказано позднее.

Особое место в задаче контроля перегрузки занимает технология multipath, позволяющая клиенту (отправителю) посылать и принимать данные одновременно по нескольким сетевым маршрутам. Существуют две основные причины использования данной технологии. Первая заключается в объединении (агрегации) ресурсов нескольких путей для передачи данных по одному логическому соединению. Это важно для передачи больших объемов данных на устройствах, например, смартфонах. Вторая причина заключается в обеспечении устойчивости к неполадкам в сети: при невозможности передать данные по одному из путей данные могут быть переданы по другим. Задача контроля перегрузки в данном случае несколько усложняется в силу наличия нескольких путей, передачу данных по которым нужно эффективно оптимизировать, при этом критерий оптимизации может быть различным. Поведение классических алгоритмов контроля перегрузки в настоящий момент подробно изучено и, кроме того, существуют работы, описывающие область алгоритмов контроля перегрузки с помощью обучения с подкреплением [1, 2]. В данной работе, помимо обзора данных алгоритмов, уделена значительная часть технологии multipath, которая также связана с контролем перегрузки, и детально разобраны возможные недостатки алгоритмов, основанных на машинном обучении.

Далее статья структурирована следующим образом. В разделе II представлена постановка задачи и даны основные определения по теме. В разделе III описаны классические алгоритмы контроля перегрузки без использования машинного обучения и приведено их сравнение. Раздел IV содержит описание популярных методов обучения с подкреплением. В разделе V представлен анализ и описание работ по контролю перегрузки с помощью машинного обучения и приведено их сравнение в рамках рассматриваемой задачи. Раздел VI содержит анализ алгоритмов, реализующих технологию multipath. Раздел VII описывает проблемы и направление будущих исследований алгоритмов, основанных на машинном обучении в задаче контроля перегрузки.

2. Описание задачи

Существует широкое разнообразие формулировок и определений, описывающих одни и те же величины, связанные с контролем перегрузки. Поэтому необходимо ввести чёткие определения и описать формулировку задачи, исходя из них, что и будет проделано в данном разделе.

2.1 Определения

Рассмотрим некоторые базовые определения по нашей теме, необходимые для постановки задачи.

Ширина канала (bandwith) – предельный объем данных, который может проходить через заданный канал связи.

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.

BDP (Bandwidth-delay product) – произведение ширины канала на время приема-передачи (рис. 1). Этот коэффициент встречается во многих работах по данной теме и является максимальным объемом данных, которые могут находиться в сети в некоторый момент времени.

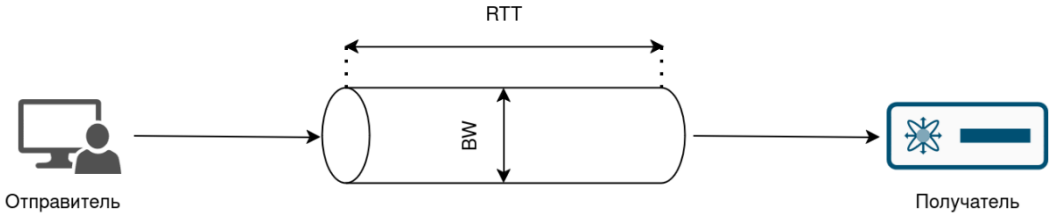

Рис. 1. Коэффициент BDP.
Fig. 1. The BDP coefficient.

Взвешенное скользящее среднее или усредненная круговая задержка (SRTT): $SRTT = \alpha \cdot SRTT + (1 - \alpha) \cdot RTT$.

Отклонение RTT: $RTTVAR = \beta \cdot RTTVAR + (1 - \beta) \cdot |SRTT - RTT|$.

Интервал ожидания: $RTO = SRTT + 4 \cdot RTTVAR$.

V (объём передаваемых данных) – число байт, переданных отправителем получателю за некоторое время.

Пропускная способность (throughput) – отношение объема данных, проходящих через сеть (канал) за заданный интервал времени, к этому интервалу. Пропускная способность является изменяемой величиной и показывает скорость отправки данных, в то время как ширина канала – это величина неизменная и является характеристикой канала.

CWND (окно перегрузки) – число байт, которые могут находиться в сети от каждого отправителя во время работы алгоритма контроля перегрузки.

Bytes in flight (байты в «полёте») – количество отправленных неподтвержденных байт.

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

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

MSS (Maximum segment size) – максимальный размер полезного блока данных в байтах для TCP-пакета.

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

2.2 Постановка задачи контроля перегрузки

Классическая задача контроля перегрузки выглядит следующим образом (рис. 2): существует отправитель и получатель, которые обмениваются данными по некоторому пути, состоящему из разных каналов. Каналы сети имеют некоторые характеристики, главной из которых является ширина канала. Алгоритм контроля перегрузки должен стремиться передавать

данные таким образом, чтобы общий объем данных был равен значению BDP или близко к нему. С другой стороны, алгоритм должен избегать ситуаций, когда данных так много, что задержка становится критичной.

Таким образом, задачу контроля перегрузки можно сформулировать так:

$$V \to BDP$$

$$delay \to min$$

Однако стоит понимать, что данная задача имеет ряд трудностей: маршрут от отправителя к получателю может содержать большое число путей с разной пропускной способностью, тип связи может быть различен (беспроводная связь, оптоволокно и т.д.), у отправителя может быть несколько каналов передачи данных (multipath) и т.д.

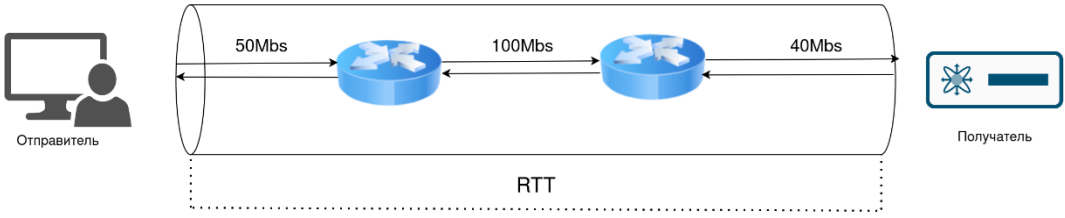

Рис. 2. Постановка задачи контроля перегрузки.
Fig. 2. Setting the task of overload control.

Более детально, различные алгоритмы контроля перегрузки решают схожие задачи, которые можно сформулировать следующим образом:

Максимизация пропускной способности (throughput). Высокая пропускная способность означает высокую степень использования канала, то есть высокую эффективность передачи данных. Алгоритмы контроля перегрузки направлены на решение данной задачи. Однако высокая пропускная способность может приводить к созданию длинных очередей, что может значительно увеличить задержку и время приема-передачи (RTT).

Минимизация времени приема-передачи (RTT) и задержки. Алгоритмы контроля перегрузки стараются уменьшить время приема-передачи, чтобы повысить эффективность передачи данных. Зачастую задача минимизации этого времени является взаимоисключающей к задаче максимизации пропускной способности, что приводит к поиску оптимального значения между решениями двух данных задач.

Минимизация скорости потерь (loss_rate). Высокая скорость потерь пакетов приводит к нестабильной работе сети и, соответственно, к уменьшению скорости передачи данных и увеличению времени RTT. Поэтому алгоритмы контроля перегрузки должны уменьшать данную скорость потерь.

Быстрота реагирования на перегрузку. Высокая скорость реагирования позволяет оперативно решить проблему перегрузки, не уменьшая эффективность передачи данных. Однако стоит понимать, что высокая скорость реагирования достигается путём частой смены значений cwnd, что может быть ресурсозатратно. Именно поэтому алгоритмы контроля перегрузки стараются соблюдать баланс, выбирая оптимальную частоту обновления размеров окна cwnd.

Справедливость. Высокая справедливость требует, чтобы алгоритмы контроля перегрузки одинаково распределяли ресурсы между различными получателями и отправителями, использующими одну и ту же часть пути для передачи данных.

Основной способ управлять перегрузкой – это увеличение или уменьшение размера окна cwnd, в то время как перегрузку можно определять разными способами. Классические алгоритмы контроля перегрузки делятся на три группы: основанные на потерях, основанные на задержке и гибридные.

Алгоритмы, **основанные на потерях**, определяют факт перегрузки после её возникновения, что в некоторых случаях может быть недостатком. При этом индикатором перегрузки является истечение интервала ожидания RTO (повторной передачи, retransmission) до получения подтверждения или получение трех повторяющихся подтверждений. Стоит заметить, что в случае истёкшего интервала поведение алгоритма будет отличаться от случая трёх подтверждений.

Алгоритмы, **основанные на задержках**, определяют перегрузку до её возникновения, полагаясь на время приема-передачи RTT. Однако измерить это время в сложных сетях бывает проблематично, что может влиять на корректность работы алгоритма. Во-вторых, данный класс алгоритмов в силу своей природы использует доступную ширину канала не столь эффективно, как алгоритмы, основанные на потерях.

Гибридные алгоритмы стараются достичь баланса между двумя классами, представленными ранее. Их основная идея заключается в следующем: значительное увеличение скорости передачи данных в том случае, если полоса пропускания загружена незначительно и более мягкое поведение в ситуациях, когда канал почти заполнен. К главным недостаткам данного класса стоит отнести плохую совместимость с другими алгоритмами контроля перегрузки, о чём будет сказано в дальнейшем.

3. Классические алгоритмы контроля перегрузки

В этом разделе рассмотрены наиболее популярные алгоритмы контроля перегрузки без использования машинного обучения. В конце раздела представлена сравнительная таблица описанных алгоритмов.

3.1 Алгоритмы, основанные на потерях

TCP Tahoe (1988) [3] был первым алгоритмом контроля перегрузки в сети. Tahoe использует два ключевых механизма для управления перегрузкой: медленный старт и предотвращение перегрузки. Медленный старт начинает свою работу с минимальным значением размера окна. Значение cwnd увеличивается экспоненциально при подтверждении сегмента, то есть через каждый интервал приема-передачи RTT. Этот процесс продолжается до тех пор, пока размер окна cwnd не достигнет порогового значения (ssthresh), после чего начинается фаза предотвращения перегрузки, или пока не произойдет событие потери, то есть истечение интервала ожидания RTO, после которого пороговое значание ssthresh устанавливается равным половине текущего cwnd, размер окна принимается равным минимальному значению, и медленный старт начинается заново.

Предотвращение перегрузки линейно увеличивает размер cwnd за один интервал RTT. При обнаружении потери пакетов Tahoe уменьшает ssthresh вполовину от текущего значения, резко снижает cwnd до минимального значения и вновь начинает процесс с фазы медленного старта. Этот механизм интегрирует принципы подхода AIMD, обеспечивая аддитивное увеличение (additive increase) размера окна до момента обнаружения перегрузки, после чего происходит мультипликативное уменьшение (multiplicative decrease).

Основной недостаток данного алгоритма заключается в том, что после обнаружения потери пакета Tahoe сбрасывает размер окна cwnd до минимального значения и начинает процесс увеличения заново, что приводит к снижению общей производительности сети.

TCP Reno (1990) [4] усовершенствует процесс обнаружения потерь, вводя механизмы быстрой повторной передачи и быстрого восстановления. Вместо того, чтобы инициировать повторную передачу потерянного сегмента после истечения интервала ожидания, как это делает Tahoe, Reno запускает повторную передачу быстрее, реагируя на получение трех дубликатных подтверждений (ACK) для одного и того же сегмента данных. Это позволяет быстрее восстанавливаться после потерь, не дожидаясь истечения таймера повторной

передачи. После того, как отправитель получает три дубликатных ACK для одного сегмента, TCP Reno входит в фазу быстрого восстановления.

В этой фазе Reno уменьшает порог ssthresh вдвое от текущего размера окна перегрузки и устанавливает окно cwnd равным ssthresh + 3*MSS. После уменьшения размера окна перегрузки Reno продолжает отправку данных, используя новый размер окна. За каждый последующий дубликатный ACK, полученный в этой фазе, значение cwnd увеличивается на размер одного блока MSS, позволяя отправлять дополнительный сегмент данных за каждое подтверждение, которое предположительно указывает на то, что другой сегмент был успешно доставлен.

Фаза быстрого восстановления завершается, когда получено подтверждение о доставке нового сегмента данных, и алгоритм переходит в фазу предотвращения перегрузки – или когда происходит тайм-аут, и алгоритм переходит в фазу медленного старта.

TCP NewReno (1999) [5] вводит улучшение по сравнению с алгоритмом Reno, заключающееся в обработке частичных подтверждений. В отличие от Reno, NewReno остается в фазе быстрого восстановления, пока не получит подтверждения всех оставшихся пакетов. Если приходит частичный ACK, который подтверждает только часть пакетов, алгоритм рассматривает это как признак потери и инициирует повторную передачу. Это позволяет NewReno эффективно обрабатывать множественные потери без необходимости выхода из фазы быстрого восстановления, а также избежать многократного уменьшения cwnd. Поэтому по сравнению с Reno алгоритм NewReno более эффективен при обработке множественных потерь пакетов.

Scalable TCP (2003) [6] базируется на подходе MIMD (Multiplicative Increase, Multiplicative Decrease). При обнаружении потери пакетов Scalable TCP уменьшает размер окна перегрузки мультипликативно, то есть на определённый процент от текущего размера окна. Scalable TCP гораздо более эффективен, чем Reno, в сетях с высокой задержкой. Быстрее и агрессивней восстанавливается после потери пакета. Может проявлять несправедливость по отношению к другим потокам данных, использующим стандартный TCP, в смешанных сетевых средах. Его агрессивное увеличение окна перегрузки может привести к тому, что потоки Scalable TCP будут занимать большую часть доступной пропускной способности за счет других потоков.

Highspeed TCP (2003) [7] – это вариант протокола TCP, разработанный для улучшения производительности в сетях с высоким BDP и с высокой задержкой, таких как дальнемагистральные и широкополосные сети. В HSTCP изменение размера окна при наличии или отсутствии перегрузки сети зависит от определяющих размер изменения функций $a(cwnd)$ и $b(cwnd)$, которые в свою очередь зависят от текущего размера окна. Когда перегрузка не обнаружена, размер окна cwnd увеличивается через каждое RTT на величину $a\,(cwnd)/cwnd$. При обнаружении перегрузки размер окна cwnd уменьшается на $b(cwnd) \cdot cwnd$. В сетях с высоким BDP HSTCP ведет себя эффективно и агрессивно, однако может привести к несправедливому распределению сетевых ресурсов между потоками.

TCP BIC (2004) [8] решает проблему несправедливости, характерную для HSTCP. Помимо медленного старта и аддитивного увеличения окна, BIC использует механизм бинарного поиска. Бинарный поиск позволяет найти максимальный размер cwnd, при котором сеть может работать без перегрузок. BIC TCP быстро увеличивает размер окна до тех пор, пока не достигнет порога. После достижения порога алгоритм переключается на бинарный поиск, чтобы найти новое оптимальное значение окна перегрузки. Если потери не обнаруживаются, BIC продолжает увеличивать размер окна, сужая диапазон поиска между текущим минимальным размером окна (последним известным стабильным значением) и размером окна перед последней потерей. Алгоритм устанавливает cwnd как среднее между данными значениями и проверяет наличие потерь, постепенно уменьшая разницу между ними до тех пор, пока не будет найдено оптимальное значение. Это помогает более плавно адаптироваться к состояниям перегрузки. Таким образом, алгоритм может эффективно

работать в высокоскоростных сетях с большой задержкой. При возникновении перегрузки в сети BIC использует мультипликативное уменьшение для коррекции cwnd.

CUBIC TCP (2008) [9] управляет окном перегрузки, заменяя сложные алгоритмы BIC на кубическую функцию. При каждом подтверждении (ACK) окно изменяется следующим образом:

$$W(t) = W_{max} + C(t - K)^3$$

$$K = \sqrt[3]{\frac{W_{max} \cdot \beta}{C}}$$

- t – время с момента последней потери,
- C– параметр, определяющий скорость увеличения окна,
- β– начальная константа,
- W_{max}– размер окна перед последней потерей,
- W– текущий размер окна перегрузки.

При обнаружении потери пакета:

$$W_{max} = W$$

$$W = W \cdot (1 - \beta)$$

Кубическая функция позволяет резко увеличивать размер окна сразу после потери и после длительного времени работы без потерь. В то же время окно перегрузки меняется незначительно в состоянии, близком к предыдущей потере, что обеспечивает более стабильную работу алгоритма в сетях с высокой шириной канала.

CUBIC на данный момент является одним из самых широко используемых алгоритмов управления перегрузкой.

3.2 Алгоритмы, основанные на задержке

Как уже говорилось ранее, данный класс кардинально отличается от алгоритмов, основанных на потерях с точки зрения определения перегрузки.

TCP Vegas (1994) [10] основан на использовании информации о RTT. Данный алгоритм постоянно измеряет RTT для текущего соединения. Vegas вычисляет ожидаемую пропускную способность как отношение текущего размера окна к базовому RTT (BASERTT), который является минимальным измеренным RTT для соединения, и предполагает, что это отношение будет поддерживаться, если в сети нет перегрузки.

Фактическая пропускная способность вычисляется как отношение размера окна к текущему RTT. Если разница между ожидаемой и фактической пропускной способностью (Differ) меньше заданного порога, то Vegas увеличивает размер окна на один MSS. Если разница больше некоторого другого порога, то Vegas уменьшает размер окна. Уменьшение размера окна в Vegas происходит более мягко, чем мультипликативное уменьшение в других алгоритмах.

Основываясь на идеях Vegas, исследователи разработали ряд его модификаций [11-12], которые в основном отличались способами измерения и предсказания RTT для прогнозирования перегрузки. Однако данный алгоритм не стал популярным в высокоскоростных сетях, так как зачастую использует доступную полосу пропускания не полностью.

TCP Veno (2003) [13] комбинирует методы Reno и Vegas, используя элементы контроля перегрузки Reno и механизм оценки состояния сети на основе задержек из Vegas. В любой момент времени, если

$$BASERTT \cdot Differ \geq \beta - \text{перегрузка;}$$

$$BASERTT \cdot Differ < \beta \text{ - перегрузки нет.}$$

Если обнаружена потеря пакета, когда соединение находится в состоянии перегрузки, то TCP Veno использует стандартную схему AIMD TCP Reno для уменьшения cwnd в режиме предотвращения перегрузки. В отличие от Reno, при отсутствии перегрузки Veno уменьшает окно cwnd на 20% вместо стандартных 50%.

3.3 Гибридные алгоритмы

TCP Westwood (2001) [14] был разработан для улучшения производительности передачи данных в сетях с высокой пропускной способностью и высокими скоростями потерь. Алгоритм постоянно оценивает пропускную способность и измеряет значения RTT при отсутствии нагрузки.

Эти оценки затем используются для адаптации размера окна перегрузки cwnd и порога медленного старта ssthresh после обнаружения потери пакетов, что позволяет более точно и гибко реагировать на изменения в сетевых условиях.

Вместо фиксированного сброса, как в TCP Reno, Westwood минимизирует снижение производительности, пропорционально снижая размер окна с учетом текущей оценки пропускной способности и RTT. В случае отсутствия потерь Westwood увеличивает окно перегрузки аналогично традиционным алгоритмам TCP через фазы медленного старта и предотвращения перегрузки.

BBR (2016) [15] использует модель, основанную на оценке ширины канала BtlBw и минимальной задержке RTprop 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. Поэтому BBR стремится поддерживать окно передачи на уровне оцененного BDP, используя канал максимально эффективно и с минимальной задержкой, не полагаясь на обнаружение потерь пакетов.

Алгоритм BBR регулярно измеряет сеть, чтобы адаптировать скорость передачи и выявлять изменения в пропускной способности или задержке. Для этого BBR использует механизм gain cycling, который чередует фазы изменения коэффициента передачи, позволяя эффективно использовать ширину канала.

Однако BBR может не обеспечивать идеальное справедливое распределение пропускной способности с алгоритмами контроля перегрузки, основанными на потерях. Данная проблема отчасти была решена в более поздних версиях BBR [16, 17].

Copa (2018) [18] превосходит BBRv1 по способности поддерживать высокую пропускную способность с более низкой задержкой. Целевая скорость передачи данных определяется как $\frac{1}{(\delta \cdot d_q)}$, где d_q – измеренная задержка в очереди, а δ – параметр, который регулирует баланс между задержкой и пропускной способностью: чем выше δ, тем выше акцент на снижении задержки, а при меньших значениях – на увеличении пропускной способности. Это позволяет гибко адаптировать скорость передачи в зависимости от условий сети. Алгоритм Copa динамически адаптирует cwnd: увеличивает его при низкой скорости и уменьшает при превышении целевой. При конкуренции с другими потоками алгоритм изменяет δ по принципу AIMD для поддержания пропускной способности и стабильности соединения.

Краткое описание представленных выше алгоритмов дано в табл. 1.

Табл.1. Классические алгоритмы контроля перегрузки.
Table 1. Classical congestion control algorithms.

№	Год	Название	Тип алгоритма	Особенности
3	1988	TCP Tahoe	На потерях	Первый алгоритм контроля перегрузки. Использует медленный старт и предотвращение перегрузки.
4	1990	TCP Reno	На потерях	Объединяет идеи TCP Tahoe и механизмов быстрой повторной передачи и быстрого восстановления.
5	1999	TCP New Reno	На потерях	Улучшение TCP Reno с обработкой частичных подтверждений.
6	2003	Scalable TCP	На потерях	Увеличивает cwnd относительно текущего размера окна.
7	2003	Highspeed TCP	На потерях	Предназначен для сетей с высоким BDP, агрессивно увеличивает окно.
8	2004	TCP BIC	На потерях	Поиск оптимального размера окна с помощью бинарного поиска.
9	2008	TCP CUBIC	На потерях	Использует кубическую функцию для изменения окна.
10	1994	TCP Vegas	На задержках	Определяет перегрузку, основываясь на измерении разницы RTT.
13	2003	TCP Veno	На задержках	Предсказывает перегрузку, используя идеи Vegas. Изменяет размеры окна, используя идеи Reno.
14	2001	TCP Westwood	Гибридный	Адаптирует окно в зависимости от оценок RTT и пропускной способности.
15	2016	BBR	Гибридный	Стремится использовать максимально доступную пропускную способность узких мест без перегрузок.
18	2018	Copa	Гибридный	Изменяет cwnd в соответствии с целевой скоростью, с параметром δ для регулировки приоритетов задержки и пропускной способности.

4. Обучение с подкреплением

Алгоритмы обучения с подкреплением можно разделить на два класса: на основе ценностей и на основе политик. В контексте контроля перегрузки наиболее рассматриваемыми алгоритмами являются:

- Q-обучение (на основе ценностей);
- глубокое Q-обучение (на основе ценностей);
- Actor-Critic (на основе ценностей и политики);
- Proximal Policy Optimization (PPO) (на основе ценностей и политики).

Основная схема работы обучения с подкреплением представлена на рис. 3. Агент выбирает некоторое действие a_t из своей стратегии. Среда реагирует на это и выдаёт следующие состояние и награду s_{t+1}, r_{t+1}. Основываясь на полученных результатах, агент корректирует свою стратегию. В задаче контроля перегрузки состояниями обычно являются: скользящее среднее (SRTT), число подтверждённых пакетов и т.д. Наградой выступают величины, которые мы хотим оптимизировать (пропускная способность, RTT и т.д). Действиями – уменьшение или увеличение окна перегрузки. Выбор состояний, действий и функции награды в данной задаче будет представлен в следующих разделах.

4.1 Q-обучение

Q-обучение (Q-learning) – это один из методов обучения с подкреплением [19], который обучает агента с помощью функции ценности действия (Q-функции). Формально, Q-функция

– ценность действия s в состоянии a – определяет ожидаемое вознаграждение следующим образом:

$$Q(s_t, a_t) = E[R_t \vee s_t = s, a_t = a]$$

- R_t – общее вознаграждение, полученное от времени t до конца эпизода;
- a_t – действие, выбранное в состоянии;
- E – математическое ожидание.

Для каждого t выбирается действие, среда генерирует награду и следующее состояние. Затем происходит обновление Q-функции:

$$Q(s_t, a_t) \rightarrow Q(s_t, a_t) + \alpha(r_t + \gamma max_a, Q(s_{t+1}, a') - Q(s_t, a_t))$$

- $Q(s_t, a_t)$ – текущее значение Q-функции для заданного состояния и действия;
- r_t – вознаграждение, полученное после выполнения действия a_t;
- α – скорость обучения (learning rate);
- γ – коэффициент дисконтирования, который определяет влияние будущих вознаграждений.

Одним из преимуществ данного метода является отсутствие необходимости предварительного знания о динамике среды, что может быть очень полезно в случае контроля перегрузки в сети с динамическими параметрами.

Однако, с другой стороны, Q-обучение может быть чувствительным к шуму и нестабильным параметрам в динамических средах, где вознаграждения могут существенно изменяться от одного эпизода к другому, что может приводить к низкой эффективности.

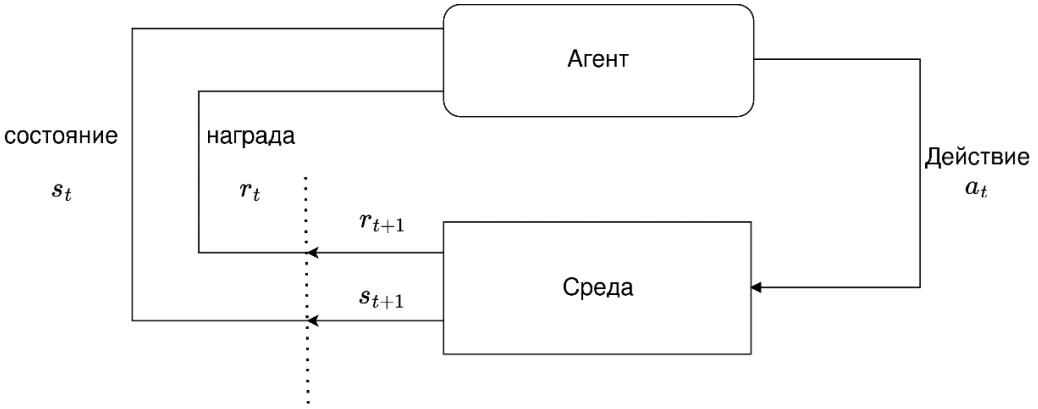

Рис. 3. Обучение с подкреплением.
Fig. 3. Reinforcement learning.

4.2 Глубокое Q-обучение

Для проведения глубокого Q-обучения (DQL) Q-функция представляется в виде нейронной сети [20]. В процессе работы алгоритма происходит обучение нейронной сети на основе истории о состояниях, действиях и награды. Как правило, награда зависит от цели алгоритма: достижения высокой пропускной способности и уменьшения RTT. В контексте задачи контроля перегрузки входом нейронной сети являются состояния, а выходом – действия (изменения размера окна cwnd).

DQL способен работать с высокоразмерными пространствами, то есть с объектами, содержащими большое число признаков, что может быть полезно в случае контроля перегрузки при использовании в качестве признаков последовательностей RTT, cwnd и др.

Другим преимуществом является высокая обобщающая способность нейронных сетей, которая позволяет улучшить обучение в новых, ранее незнакомых состояниях, то есть сетевых средах.

Однако стоит понимать, что использование нейронных сетей требует значительно большего объема вычислений и памяти, что может быть критично в том случае, если алгоритм контроля перегрузки должен с высокой частотой выдавать решение об изменении размера окна cwnd. Кроме того, для надежного обучения требуется большое количество собранных данных, что может увеличить время обучения и привести к некорректной работе алгоритма, особенно в начале.

4.3 Метод Actor-Critic

Метод Actor-Critic (AC) сочетает в себе идеи как из алгоритмов на основе ценностей, так и из алгоритмов на основе политик [21]. Актор отвечает за выбор действий. Он представляет собой политику, которая определяет, какое действие выполнять в зависимости от текущего состояния. Политика может быть детерминированной или стохастической. Актор обновляет свои параметры, используя информацию о вознаграждении, полученную от критика.

Критик, в свою очередь, оценивает действие, выбранное Актором, и предоставляет обратную связь в виде оценки (обычно функции ценности Q). Основной задачей критика является регрессия к целевому значению, что позволяет Актору получать качественную оценку своей текущей политики.

Помимо классического алгоритма AC существуют Advantage AC (A2C) [22] и Asynchronous AC (A3C). Данные методы основаны на функции преимущества:

$$A(s_t, a_t) = r_t + \gamma V(s_{t+1}, \theta_v) - V(s_t, \theta_v)$$

- V – функция ценности;
- r_t – вознаграждение, полученное после выполнения действия a_t;
- θ_v – параметры политики;
- γ – коэффициент дисконтирования, который определяет влияние будущих вознаграждений.

A3C использует несколько копий одного и того же агента для одновременного и независимого изучения среды. Поэтому такой метод позволяет быстрее исследовать среду и следовательно, даёт возможность быстрее сходиться к оптимальному значению, что очень важно в рамках задачи контроля перегрузки.

4.4 Proximal Policy Optimization

Proximal Policy Optimization (PPO) является методом, основанным на градиентах политик [23], и использует подход актор-критик, обладая при этом некоторыми усовершенствованиями и ограничивая диапазон целевой функции. В данном случае задача выглядит следующим образом:

$$L^{clip} = Emin(r(\theta) \cdot A^{\pi}(s,a), clip(r(\theta), 1-\epsilon, 1+\epsilon) \cdot A^{\pi}(s,a))$$

- $clip$ – функция, обрезающая значения, ограничивая его в заданных границах;
- ϵ – некоторое положительное, близкое к нулю число.

Данный метод может быть эффективен в силу своей устойчивости к средам, выходящим за рамки его обучения, что крайне полезно в случае контроля перегрузки, когда характеристики сетевой среды могут постоянно меняться.

5. Машинное обучение для контроля перегрузки

Большинство работ, посвящённых контролю перегрузки на основе машинного обучения, можно разделить на две группы: алгоритмы контроля перегрузки с помощью обучения с учителем и алгоритмы контроля перегрузки с помощью обучения с подкреплением.

5.1 Обучение с учителем

Обучение с учителем предполагает собранный набор данных, на котором и учится алгоритм машинного обучения. Существует две популярные задачи контроля перегрузки, которые могут быть решены с помощью обучения с учителем: классификация типа потерь и прогнозирование задержки или RTT.

Алгоритмы контроля перегрузки, основанные на потерях, указывают лишь на факт перегрузки, но ничего не говорят о её типе, что бывает полезно в некоторых случаях. Поэтому имеет смысл использовать алгоритмы машинного обучения для определения типа перегрузки. Данная задача, с точки зрения машинного обучения с учителем, является задачей классификации на n классов, где признаками могут являться некоторые характеристики сети, а классом – тип перегрузки.

Так, в [24] авторы, используя в качестве признаков одностороннюю задержку и время между передачами пакетов, классифицируют потери на два класса (потеря связи или перегрузка) в сценарии беспроводных сетей. В качестве алгоритма классификации используется градиентный бустинг решающих деревьев – алгоритм машинного обучения, последовательно строящий ансамбль деревьев, каждое из которых корректирует ошибки предыдущих. В качестве признаков – задержки пакета до потери и три задержки трех пакетов после потери. Обучающий набор классификатора представлял собой набор данных из 25 000 событий потерь, а встроенный классификатор использовался авторами в TCP NewReno.

Более полное исследование с точки зрения числа алгоритмов машинного обучения было представлено в [25]. Сценарий был тем же – сценарий беспроводных сетей; классами являлись потеря связи или перегрузка сети. В качестве признаков выступали различные статистические значения интервалов между пакетами, а в качестве алгоритмов – модели, основанные на решающих деревьях, и полносвязные нейронные сети. В работе авторы показали, что классическое решающее дерево даёт оптимальный результат при небольших вычислительных затратах.

В [26] авторы используют скрытые марковские модели (HMM) для классификации потерь на два класса (потеря связи или перегрузка) в случае проводных сетей. В качестве признаков использовались количества подтверждённых пакетов между двумя потерями.

Как уже отмечалось ранее, алгоритмы, основанные на задержках, прогнозируют перегрузку, оценивая время RTT. Именно поэтому в некоторых случаях может быть полезным прогноз времени приёма-передачи на основе набора признаков. Кроме того, алгоритмы, основанные на потерях, косвенно используют время RTT, так как сигналом перегрузки служит истечение таймера RTO. Следовательно, задача прогнозирования RTT в ряде случаев является актуальной. С точки зрения машинного обучения – это задача регрессии, где предсказываемой величиной является следующие значения RTT.

Так, в [27] авторы используют ансамбль для предсказания времени RTT в сценарии сетей, чувствительных к задержкам. Авторы подтверждают полученные результаты экспериментами в реальном времени, реализовав предложенный алгоритм в ядре Linux. Данные эксперименты показывают более высокую точность оценки RTT с помощью машинного обучения: по сравнению со стандартным протоколом TCP улучшение более чем на 40%. В качестве признаков используется линейная комбинация максимальных и минимальных предыдущих значений RTT.

С внедрением видеоприложений реального времени и беспроводных сетей появилась актуальность контроля перегрузки в этом сценарии. Одной из первых работ в данной области был алгоритм [28], основанный на теореме Байеса.

Однако классические алгоритмы машинного обучения могут быть не столь эффективны в случае предсказания ряда RTT, в отличие от рекуррентных нейронных сетей (RNN). В [29] авторы используют последовательность из 6 интервалов между пакетами для предсказания нейронной сетью 4 следующих значений RTT в сценарии проводной сети. В качестве оценки работы используется метрика SNR, и её значение на тестовом наборе данных равно 0.13.

Чуть позже появились исследования с большим числом объектах в наборе данных. Так, в [30] авторы в качестве набора данных использовали 2160 объектов. Признаками выступали средние значения RTT за минуту, и на их основе рекуррентных нейронная сеть предсказывала от одного до 30 значений следующих RTT. MSE при этом составила на тестовом наборе данных от 0.004 до 0.01 в зависимости от числа предсказанных значений RTT.

Однако средние значения RTT за длительный интервал могут быть не информативны. В [31] авторы, используя каждое RTT, с помощью k-NN выделяли ряд признаков (10-15), которые затем подавались на вход рекуррентной нейронной сети. Удалось предсказать большое число следующих значений RTT: от 320 до 960. Размер набора данных составлял 10000, и метрика NRMSE на тестовом наборе данных составила 0.26-0.32 в зависимости от числа предсказанных значений RTT.

Предсказание RTT в мобильной сети (4G) представлено в [32]. В силу особенности сценария авторами выбраны в качестве признаков не только предыдущие значения RTT, но и уровень принимаемого сигнала, густонаселенность района сигнала и день недели, так как это может влиять на нагрузку сети. Кроме того, данная работа характерна предсказанием не конкретного значения RTT, а лишь его серьёзного изменения, что на практике может быть более полезно для контроля перегрузки.

Краткий обзор алгоритмов обучения с учителем для предсказания типа перегрузки и RTT представлен в табл. 2.

5.2 Обучение с подкреплением

С точки зрения контроля перегрузки алгоритмы обучения с подкреплением основаны на следующей идее: агент использует состояния, наблюдаемые из окружающей сетевой среды, и на основе этих состояний принимает решения об увеличении скорости отправки или размера окна перегрузки.

QTCP [33] был одной из первых работ по контролю перегрузки с помощью обучения с подкреплением. В качестве метода обучения выступало Q-обучение, функция награды зависела от пропускной способности и RTT. Состоянием была следующая тройка: средний интервал между отправкой двух пакетов, средний интервал между получением двух последовательных подтверждений, среднее RTT. На основе состояний и функций награды представленный алгоритм совершал всего три действия: увеличивал окно на 10 байт, уменьшал окно на один байт, оставлял окно без изменений. Авторы сравнивали представленный алгоритм лишь с NewReno, но по отношению к нему получили значительное улучшение, как с точки зрения скорости передачи, так и с точки зрения минимизации RTT.

В **Aurora [34]** функция награды зависела от пропускной способности, задержки, RTT и числа потерь. Агент DRL обучался с помощью Proximal Policy Optimization (PPO). Состоянием была скорость передачи данных и доля потерь, а действиями – уменьшение или увеличение скорости отправки. Авторы изучали работу алгоритма в сети с каналами с шириной от 1,2 до 6 Мб/с и задержками от 50 до 500 миллисекунд. Предложенный алгоритм показал высокую эффективность в такой сетевой среде по сравнению с TCP CUBIC, но возникает вопрос работоспособности алгоритма в средах с высокой пропускной способностью.

Табл. 2. Сравнение различных алгоритмов обучения с учителем.
Table 2. Comparing different learning algorithms with a teacher.

№	Год	Сценарий	Признаки	Модель	Задача
24	2005	Беспроводная сеть	Задержки	Градиентный бустинг	Потеря связи или перегрузка (классификация)
25	2004	Беспроводная сеть	Интервалы между пакетами	Решающие дерево	Потеря связи или перегрузка (классификация)
26	2008	Проводная сеть	Подтверждённые пакеты между потерями	HMM	Потеря пакетов или перегрузка (классификация)
27	2014	Сети, чувствительные к задержкам	Время приема-передачи RTT	Ансамбль алгоритмов	Предсказание времени RTT для контроля перегрузки (регрессия)
28	2018	Беспроводная сеть (видео)	Задержки	теорема Байеса	Предсказание времени RTT для контроля перегрузки (регрессия)
29	2002	Проводная сеть	RTT	RNN	Предсказание следующего значения RTT (регрессия)
30	2007	Проводная сеть	RTT	RNN	Предсказание следующих значений RTT (регрессия)
31	2009	Проводная сеть	Разложения RTT	RNN	Предсказание следующих значений RTT (регрессия)
32	2021	Мобильная сеть (4G)	RTT и косвенные характеристики сети	Градиентный бустинг	Предсказание увеличения RTT (классификация)

R3Net [35] ориентирован на потоковое видео и приложения для конференцсвязи в реальном времени. Функция награды зависела от пропускной способности, задержки, RTT и числа потерь. Агент обучается с помощью Proximal Policy Optimization (PPO). В состояние входили скорость передачи, средний интервал между пакетами, потери пакетов и среднее значение RTT, действиями были уменьшение или увеличение скорости отправки. Авторы исследовали работу алгоритма в сети с низкой задержкой, сильно отличающейся от реальных сетей. Кроме того, в работе не были представлены сравнения с другими классическими алгоритмами.

Orca [36] в отличие от двух предыдущих алгоритмов регулировал не скорость отправки, а размер окна перегрузки. При этом диапазон действий был достаточно широк: умножение размера окна на 2^a, где a может принимать значения от -2 до 2. Функция награды равнялась отношению текущей пропускной способности к ширине канала, а пространство состояний было шире, чем у многих других алгоритмов: пропускная способность, потери, задержка пакетов, количество подтверждений, SRTT, окно перегрузки cwnd, максимальная скорость передачи, минимальная задержка пакетов. Алгоритм обучения, используемый Orca, представляет собой глубокий детерминированный градиент политики (DDPG), который сочетает в себе элементы Q-обучения и метода актор-критик, что также отличает его от двух предыдущих работ.

MVFST-RL [37] был разработан для контроля перегрузки в случае протокола QUIC, то есть для потокового видео. Состояние – это вектор из 20 значений (последнее значение времени приема-передачи RTT, минимальное значение RTT в течении эпизода, усредненная круговая задержка SRTT, дисперсия RTT, количество байт, отправленных с момента последнего подтверждения, количество байт, полученных момента последнего подтверждения и др.). Пространство действий – это 5 вариантов: уменьшить cwnd на 10, увеличить cwnd на 10, оставить cwnd без изменений, увеличить или уменьшить cwnd в 2 раза. Стоит отметить, что данные изменения cwnd (в несколько раз) достаточно агрессивны и используются не часто, так как, с одной стороны, способны увеличить скорость передачи за короткое время, но при этом существует большой риск вызвать перегрузку. Награда же представлена классической

комбинацией пропускной способности и задержки, но без учёта потерь. В 2023 году появилась схожая работа [38] по контролю перегрузки в протоколе QUIC. Функция награды имеет классическую зависимость от пропускной способности, RTT и потерь пакетов, а действиями являются уменьшение или увеличение cwnd в некоторое количество раз, наподобие двоичного поиска в алгоритме BIC. Кроме того, отличительной особенностью является использование алгоритма BBR для вычисления некоторых характеристик, которые уже затем используются для обучения.

Spine [39]. Функция награды не всегда является простой комбинацией пропускной способности, задержки и потерь. К примеру, в Spine авторы используют более сложную конструкцию, учитывая в функции награды только высокие значения задержки. Вектор состояний выбран довольно классический: пропускная способность, задержка, cwnd и др. (всего 8 показателей). Также в данной работе представлены сравнения с другими классическими алгоритмами контроля перегрузки и алгоритмами, основанными на машинном обучении (Aurora, Orca, Cubic, BBR). Результаты показывают, что в ряде случаев Spine работает несколько лучше с характеристиками сети из диапазона bandwidth (40-200 Mbps), delay (10-100ms) и loss (0-1%).

TCP-PPO2. В [40] была предложена ещё одна реализация алгоритма. В состояние входили: время с момента начала работы, текущий размер cwnd, количество отправленных, но ещё не подтверждённых байт, количество подтверждённых пакетов, RTT, скорость передачи данных, число потерянных пакетов. Cwnd менялось согласно следующей формуле: cwnd = cwnd + n * MSS, где высокие значения n должны быть использованы при низкой заполненности канала, n = 1 при почти заполненном канале, n является отрицательным значением в случае перегрузки. В качестве моделирования компьютерной сети авторы используют Mininet, являющийся популярным эмулятором компьютерных сетей, встречающийся во многих работах. Результаты показывают, что политика TCP-PPO2 снижает задержку на 11,7–87,5% по сравнению с классическим CUBIC.

С развитием IoT (Интернета вещей) появилась необходимость контроля перегрузки в данных сетях, где устройства пользователей в жилых домах решают конкретные задачи. В 2024 году появилась работа [41], в которой механизм контроля перегрузки адаптирован для приложений умных сетей электроснабжения (smart grid), использующих ненадежные транспортные протоколы. Примером такого протокола является UDP, который, в отличие от TCP, не имеет встроенного контроля перегрузки, что может значительно снижать производительность сети. В силу нестандартности задачи авторы используют для состояния всего два неклассических показателя: долю пакетов, успешно принятых принимающим узлом, по отношению к общему количеству пакетов, отправленных со всех узлов сети; долю пакетов, успешно принятых принимающим узлом, по отношению к общему количеству пакетов, отправленных к данному узлу. Действием является не изменение значения cwnd, а решение о том, передавать или не передавать данные.

Проанализировав популярные алгоритмы контроля перегрузки с помощью обучения с подкреплением, можно выделить следующие закономерности (рис. 4).

- Функция награды зависит от пропускной способности, RTT и потерь. При этом чем выше скорость передачи, тем награда выше и, наоборот, чем выше RTT или потери, тем награда меньше.
- Пространство состояний разнообразно от работы к работе и задаётся различными характеристиками (от 2 до 20).
- Пространство действий также отличается от работы к работе, однако существуют две общие идеи: увеличение или уменьшение cwnd на некоторое фиксированное число байт или же увеличение или уменьшение в 2^a, где a – некоторое число.

Краткий обзор алгоритмов обучения с подкреплением для контроля перегрузки представлен в табл. 3 и табл. 4. В табл. 4 представлены награды данных алгоритмов.

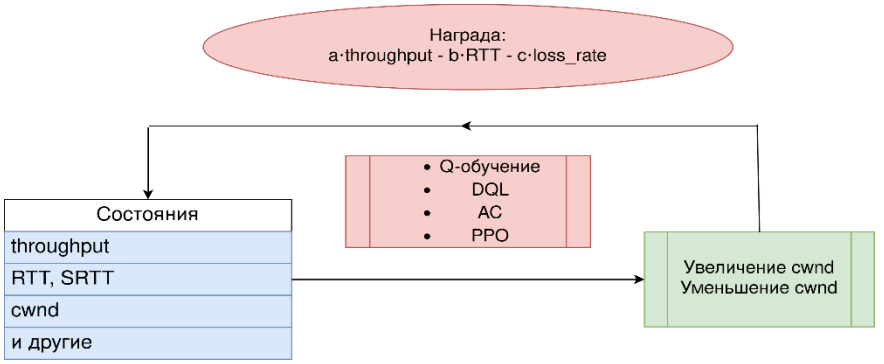

Рис. 4. Контроль перегрузки с помощью обучения с подкреплением.
Fig. 4. Cognestion control using reinforcement learning.

Табл. 3. Сравнение различных алгоритмов обучения с подкреплением.
Table 3. Comparison of various reinforcement learning algorithms.

№	Год	Сценарий	Состояния	Действия
33	2018	Проводная сеть	1. интервалы между пакетами 2. RTT	1. cwnd = cwnd + 2 MSS 2. cwnd = cwnd — MSS 3. cwnd = cwnd
34	2019	Сеть с низким bw	1. пропускная способность 2. число потерь	Уменьшение или увеличение скорости отправки
35	2019	Потоковое видео	1. пропускная способность 2. интервал между пакетами 3. потери пакетов 4. среднее значение RTT	Уменьшение или увеличение скорости отправки
36	2020	Потоковое видео	8 различных состояний (cwnd, RTT, …)	Умножение cwnd на 2^a, а от -2 до 2
37	2019	Потоковое видео	20 различных состояний (cwnd, RTT и т.д.)	1. cwnd = cwnd ± 10 MSS 2. cwnd = cwnd/2 3. cwnd = 2 cwnd 4. cwnd = cwnd
38	2023	Потоковое видео	–	Умножение cwnd на 2^a, а от -1 до 1
39	2022	Проводная сеть	8 различных состояний (cwnd, RTT, …)	Умножение cwnd на 2^a, а от -1 до 1
40	2023	Проводная сеть	8 различных состояний (cwnd, RTT, время с начала работы и т.д.)	1. cwnd = cwnd + n * MSS 2. cwnd = cwnd + MSS 3. cwnd = cwnd - n * MSS
41	2024	IoT	Доля успешно принятых пакетов	1. передавать данные 2. не передавать данные

Табл. 4. Функции награды.
Table 4. Reward function.

№	Функция награды	Алгоритм
33	$a \cdot \log(throughput) - b \cdot \log(RTT)$	Q-обучение
34	$10 \cdot throughput - 1000 \cdot delay - 2000 \cdot loss$	PPO
35	$0.6 \cdot \ln(4 \cdot throughput) - RTT - 10 \cdot loss$	PPO
36	$throughput - b \cdot delay$	DDPG
37	$a \cdot throughput - b \cdot max(delay)$	AC
38	$a \cdot throughput - b \cdot loss$	PPO
40	$(a \cdot \frac{throughput}{throughput_{max}}) - (1 - a) \cdot \frac{delay_{min}}{delay}$	PPO

6. Алгоритмы контроля перегрузок в технологии multipath

Как уже отмечалось ранее, технология multipath позволяет клиенту (отправителю) посылать и принимать данные одновременно по нескольким сетевым маршрутам для увеличения эффективности работы алгоритма контроля перегрузки. Помимо стабильной работы технологии multipath, можно выделить два главных требования к этой технологии:

- соединению, состоящему из нескольких путей, необходимо обеспечивать пропускную способность не меньше, какую оно могло бы получить при использовании классического протокола TCP на лучшем пути (с самой высокой шириной канала);
- необходимо обеспечивать меру свободы канала (capacity) таким образом, чтобы ее значение для каждого пути был не больше, чем при использовании классического протокола TCP на лучшем пути.

Классической реализацией технологии multipath для TCP является протокол MPTCP [42-44]. Решение о выборе пути, по которому отправить пакет, принимает планировщик, выбирая путь с минимальным значением RTT. Однако как отмечают некоторые исследователи [45], протокол MPTCP в определённых сценариях обеспечивает пропускную способность ниже, чем при использовании классического TCP. Причиной такого поведения является Head-of-line blocking (HOL-блокировка) [46] – явление, ограничивающее пропускную способность (производительность), которое возникает, когда пакеты задерживаются первым пакетом в очереди. Некоторые улучшения, решающие данную проблему, представлены в работе [42].

Помимо реализации технологии multipath для TCP, существует несколько реализаций для протокола QUIC [47, 48]. Протокол MPQUIC реализован на прикладном уровне, а не на уровне ядра, что сильно упрощает его обновления. Кроме того, в данном случае не наблюдается проблема HOL-блокировки, характерной для протокола MPTCP.

Перечисленные выше алгоритмы технологии multipath имеют в своей основе планировщики, основанные на определённых наборах правил. Однако некоторый набор правил может быть эффективен для работы в одной среде, но менее эффективен в другой. С другой стороны, планировщики, основанные на обучении с подкреплением, могут учитывать особенности сетевой среды, в которой работают [49]. Так, в [50] авторами был предложен алгоритм multipath для QUIC на основе обучения с подкреплением. В качестве награды выступает число подтверждённых пакетов, состояниями являются cwnd, SRTT и другие классические параметры. Действия же направлены на выбор пути – выбор пути с некоторой вероятностью или выбор пути с минимальным RTT.

Существуют реализации технологии multipath с помощью обучения с подкреплением и для TCP [51]. Характеристики состояния – cwnd каждого из путей, а действия – отправление следующего пакета по каналу Wi-Fi или каналу сотовой связи. В [52] представлена реализация технологии multipath для 5G, что в последние годы является популярным направлением. Используя идеи контроля перегрузки с помощью обучения с подкреплением в случае одного пути, авторы [53] реализуют данную технологию для сценария беспроводных сетей. Характеристики состояния – это классические cwnd, SRTT, а в качестве действий выбрано регулирование окна каждого пути, что отличает данную работу от других работ по рассматриваемой теме, где действиями служат выбор конкретного пути для отправки конкретного пакета.

Спутниковая связь имеет высокую задержку и высокую скорость передачи данных. Авторы в [54] предлагают алгоритм multipath без использования машинного обучения в данном сценарии и представляют результаты, которые показывают, что в случае использования нескольких путей с высокой задержкой обеспечивается время загрузки веб-страницы на таком же уровне, как и при использовании одного пути с низкой задержкой. В [55, 56] представлен алгоритм в похожем сценарии, но уже с использованием обучения с подкреплением.

Исследования в данной области продолжаются и сейчас. В [57] был предложен алгоритм для протокола QUIC в сценарии 5G, главным отличием которого является использование метода fuzzy Q-learning [58] вместо классического Q-learning или DQL.

Проанализировав популярные алгоритмы технологии multipath с помощью обучения с подкреплением, можно выделить следующие закономерности:

- функция награды зависит от скорости передачи, RTT и потерь. При этом чем выше скорость передачи, тем награда выше и, наоборот, чем выше RTT или потери, тем награда меньше;
- пространство состояний зачастую представлено только cwnd и SRTT, в отличие от задач, рассмотренных в табл. 3, где существует широкое разнообразие состояний;
- пространство действий включает либо выбор пути для отправки пакета, либо изменение cwnd конкретного пути.

Краткий обзор алгоритмов multipath, реализованных с помощью обучения с подкреплением, представлен в табл. 5 и табл. 6.

Табл. 5. Сравнение различных алгоритмов multipath.
Table 5. Comparison of different multipath algorithms.

№	год	сценарий	состояния	действия
50	2022	Беспроводная сеть	1. cwnd 2. SRTT 3. Другие состояния	1. Выбор пути с некоторой вероятностью 2. Выбор пути с минимальным RTT
51	2019	Беспроводная сеть	cwnd каждого из путей	1. Выбор пути Wi-Fi 2. Выбор пути мобильной связи
52	2019	Мобильная связь 5G	1. cwnd 2. SRTT 3. Байты в «полёте»	1. Выбор первого пути 2. Выбор второго пути
53	2021	Беспроводная сеть	1. cwnd 2. SRTT	Регулировка cwnd каждого пути
49	2017	Беспроводная сеть	Каждый канал находится в «хорошем» или «плохом состоянии».	Отправка пакета в определённый путь
55	2019	Спутниковая связь	1. cwnd 2. SRTT 3. число подтверждений 4. сумма throughtput всех каналов	Регулировка cwnd каждого пути
56	2023	Спутниковая связь	–	Регулировка cwnd каждого пути
57	2024	Мобильная связь 5G	1. cwnd 2. RTT 3. ln(loss_rate)	1. Регулировка cwnd каждого пути

Табл. 6. Сравнение различных алгоритмов multipath.
Table 6. Comparison of different multipath algorithms.

№	Функция награды	Алгоритм
50	$throughput$	Глубокое Q-обучение
51	$packet_{ack} - packet_{loss}$	Глубокое Q-обучение
52	$throughput$	Глубокое Q-обучение
53	$a \cdot throughput - b \cdot delay - c \cdot loss_{rate}$	Глубокое Q-обучение
55	$a \cdot cwnd - b \cdot RTT - c \cdot packet_{retransmissions}$	DDPG

Сравнительная схема наиболее популярных алгоритмов, представленных в данной работе изображена на рис. 5. Таким образом, все алгоритмы и технологии для контроля перегрузки можно разбить на следующие группы:

- классические алгоритмы контроля перегрузки, основанные на потерях, на задержках и гибридные;
- алгоритмы контроля перегрузки, основанные на обучении с учителем;
- алгоритмы контроля перегрузки, основанные на обучении с подкреплением;
- алгоритмы, реализующие технологию multipath без использования машинного обучения;
- алгоритмы, реализующие технологию multipath с использованием обучения с подкреплением.

На потерях	На задержках	Обучение с учителем	Multipath
TCP Tahoe [3]	TCP Vegas [10]	LCC [28]	MPTCP [42]
TCP Reno [4]	TCP Vegas-A [11]	PCC Vivace [60]	MPTCP-LA [43]
TCP New Reno [5]	TCP Veno [13]	PCC Allegro [61]	MPTCP [44]
Scalable TCP [6]	TCP Vegas-V [59]	**Обучение с подкреплением**	MPQUIC [47]
Highspeed TCP [7]	**Гибридные**	QTCP [33]	MPQUIC [48]
TCP BIC [8]	TCP Westwood [14]	Aurora [34]	**Multipath RL**
TCP CUBIC [9]	BBR [15]	R3Net [35]	MPQUIC [50]
	BBR2 [16]	Orca [36]	MPTCP [51]
	Copa [18]	MVFST-RL [37]	MPQUIC [52]
		PBQ-Enhanced QUIC[38]	CMT-DRL [53]
		Spine [39]	MPTCP in Satellites [55]
		TCP-PPO2 [40]	mobile MPQUIC [57]

Рис. 5. Алгоритмы, представленные в работе.
Fig. 5. The algorithms presented in the work.

7. Заключение: проблемы алгоритмов, основанных на обучении с подкреплением

Сравнение эффективности различных алгоритмов. Как уже отмечалось ранее, сетевые среды могут быть очень различны с точки зрения ширины каналов, задержек и потерь. В большинстве работ авторы сравнивают эффективность реализованных алгоритмов лишь в узком диапазоне характеристик сети. При этом сравнения в широком диапазоне характеристик зачастую не осуществляются в силу их высокой трудоёмкости.

Кроме того, существует различные способы эмуляции сетей: как с помощью mininet [62], так и с помощью NS3 [63]. Данный факт также вносит неоднородность в результаты различных экспериментов и не позволяет достоверно сравнить эффективность работы различных алгоритмов.

Сбор набора данных. Данная проблема более характерна для алгоритмов обучения с учителем. Существует несколько готовых решений для записи трафика и получения его характеристик [64, 65] для дальнейшего обучения модели на собранном наборе данных. Однако собранные с помощью них признаки не всегда могут быть достаточно точны, чтобы использовать их для классификации типа потерь или предсказания RTT.

Вычислительная сложность. Алгоритмы, в основе которых лежат нейронные сети, могут обладать высокой точностью, однако их использование приводит к дополнительным

вычислительным затратам на определение оптимального действия. Таким образом, может произойти задержка в выполнении действий по контролю перегрузки сети, что значительно снижает эффективность работы алгоритмов. Кроме того, высокие затраты на обучение могут сделать алгоритм непригодным в силу большого времени отклика.

Низкая скорость обучения. Алгоритмы RL, в основе которых лежат глубокие нейронные сети, могут не сразу достигать оптимальных действий. Это может привести к непредвиденной работе алгоритма контроля перегрузки, особенно в начале.

Выбор состояний, действий и награды. Состояния, действительно влияющие на контроль перегрузки, могут быть различными в силу разнородности сетевых сред. Выбор состояний является нетривиальной задачей, поэтому в представленных работах можно заметить их широкий набор. Действия также могут быть различны и направлены как на резкое увеличение cwnd, так и на более плавное. Награда же обычно зависит от пропускной способности, RTT и потерь, так как они являются главными показателями эффективности работы алгоритма. Таким образом, можно утверждать, что состояния и действия, оптимальные для одной среды, могут не быть таковыми для другой, что резко уменьшает преимущество алгоритмов обучения с подкреплением – эффективно работать в различных средах.

Выбор гиперпараметров. Проблема, характерная для многих алгоритмов RL, проявляет себя и в данной задаче. Чтобы получить нужный результат, зачастую требуется очень тонкая настройка гиперпараметров модели, что приводит к долгим поискам этих параметров.

Совместимость с другими алгоритмами. На данный момент плохо изучен вопрос совместимости представленных алгоритмов контроля перегрузки на основе машинного обучения с классическими алгоритмами контроля перегрузки (NewReno, CUBIC, BBR и др.). Однако, как известно, алгоритмы с агрессивными поведением (BBR) могут быть несправедливы при работе с другими алгоритмами (NewReno), занимая большую часть доступной полосы пропускания. Именно поэтому задачи исследования совместимости новых RL алгоритмов является актуальной.

Внедрение алгоритмов. Контроль перегрузок в классическом варианте находится на транспортном уровне и в случае TCP является частью ядра операционной системы (New Reno, CUBIC). Алгоритмы, основанные на RL, может быть проблематично встраивать в ядро операционной системы, что не способствует их широкому внедрению.

Большое разнообразие реализаций протокола QUIC. В последние годы появилось множество реализаций протокола QUIC, каждый из которых имеет особенности в работе. К примеру, в [66] было выделено 11 популярных реализаций QUIC и проанализировано их поведение в классических алгоритмах контроля перегрузки (CUBIC, Reno и т.д.). Результаты показывают, что некоторые реализации значительно отличаются. Данные отличия могут создать определённые проблемы при реализации алгоритма контроля перегрузки с помощью обучения с подкреплением для конкретного протокола QUIC, так как для каждой реализации QUIC придётся создавать свой алгоритм RL, который учитывает ее особенности.

Список литературы / References

[1]. Jiang H. et al. When machine learning meets congestion control: A survey and comparison //Computer Networks. – 2021. – Т. 192. – С. 108033.

[2]. Wei W., Gu H., Li B. Congestion control: A renaissance with machine learning //IEEE network. – 2021. – Т. 35. – №. 4. – С. 262-269.

[3]. Jacobson V. Congestion avoidance and control //ACM SIGCOMM computer communication review. – 1988. – Т. 18. – №. 4. – С. 314-329.

[4]. Mo J. et al. Analysis and comparison of TCP Reno and Vegas //IEEE INFOCOM'99. Conference on Computer Communications. Proceedings. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. The Future is Now (Cat. No. 99CH36320). – IEEE, 1999. – Т. 3. – С. 1556-1563.

[5]. Floyd S., Henderson T. The NewReno modification to TCP's fast recovery algorithm. – 1999. – №. rfc2582.
[6]. Kelly T. Scalable TCP: Improving performance in highspeed wide area networks //ACM SIGCOMM computer communication Review. – 2003. – Т. 33. – №. 2. – С. 83-91.
[7]. Floyd S. HighSpeed TCP for large congestion windows. – 2003. – №. rfc3649.
[8]. Xu L., Harfoush K., Rhee I. Binary increase congestion control (BIC) for fast long-distance networks //IEEE INFOCOM 2004. – IEEE, 2004. – Т. 4. – С. 2514-2524.
[9]. Ha S., Rhee I., Xu L. CUBIC: a new TCP-friendly high-speed TCP variant //ACM SIGOPS operating systems review. – 2008. – Т. 42. – №. 5. – С. 64-74.
[10]. Brakmo L. S., O'malley S. W., Peterson L. L. TCP Vegas: New techniques for congestion detection and avoidance //Proceedings of the conference on Communications architectures, protocols and applications. – 1994. – С. 24-35.
[11]. Srijith K. N., Jacob L., Ananda A. L. TCP Vegas-A: Solving the fairness and rerouting issues of TCP Vegas //Conference Proceedings of the 2003 IEEE International Performance, Computing, and Communications Conference, 2003. – IEEE, 2003. – С. 309-316.
[12]. Srijith K. N., Jacob L., Ananda A. L. TCP Vegas-A: Improving the performance of TCP Vegas //Computer communications. – 2005. – Т. 28. – №. 4. – С. 429-440.
[13]. Fu C. P., Liew S. C. TCP Veno: TCP enhancement for transmission over wireless access networks //IEEE Journal on selected areas in communications. – 2003. – Т. 21. – №. 2. – С. 216-228.
[14]. Mascolo S. et al. TCP Westwood: Bandwidth estimation for enhanced transport over wireless links //Proceedings of the 7th annual international conference on Mobile computing and networking. – 2001. – С. 287-297.
[15]. Cardwell N. et al. Bbr: Congestion-based congestion control: Measuring bottleneck bandwidth and round-trip propagation time //Queue. – 2016. – Т. 14. – №. 5. – С. 20-53.
[16]. Cardwell N. et al. BBRv2: A model-based congestion control performance optimization //Proc. IETF 106th Meeting. – 2019. – С. 1-32.
[17]. Zhang Y., Cui L., Tso F. P. Modest BBR: Enabling better fairness for BBR congestion control //2018 IEEE symposium on computers and communications (ISCC). – IEEE, 2018. – С. 00646-00651.
[18]. Arun V., Balakrishnan H. Copa: Practical {Delay-Based} congestion control for the internet //15th USENIX Symposium on Networked Systems Design and Implementation (NSDI 18). – 2018. – С. 329- 342.
[19]. Watkins C. J. C. H., Dayan P. Q-learning //Machine learning. – 1992. – Т. 8. – С. 279-292
[20]. Gu S. et al. Continuous deep q-learning with model-based acceleration //International conference on machine learning. – PMLR, 2016. – С. 2829-2838.
[21]. Konda V., Tsitsiklis J. Actor-critic algorithms //Advances in neural information processing systems. – 1999. – Т. 12.
[22]. Labao A. B., Martija M. A. M., Naval P. C. A3C-GS: Adaptive moment gradient sharing with locks for asynchronous actor–critic agents //IEEE Transactions on Neural Networks and Learning Systems. – 2020. – Т. 32. – №. 3. – С. 1162-1176.
[23]. Schulman J. et al. Proximal policy optimization algorithms //arXiv preprint arXiv:1707.06347. – 2017.
[24]. El Khayat I., Geurts P., Leduc G. Improving TCP in wireless networks with an adaptive machine-learnt classifier of packet loss causes //NETWORKING 2005. Networking Technologies, Services, and Protocols; Performance of Computer and Communication Networks; Mobile and Wireless Communications Systems: 4th International IFIP-TC6 Networking Conference, Waterloo, Canada, May 2-6, 2005. Proceedings 4. – Springer Berlin Heidelberg, 2005. – С. 549-560.
[25]. Geurts P., El Khayat I., Leduc G. A machine learning approach to improve congestion control over wireless computer networks //Fourth IEEE International Conference on Data Mining (ICDM'04). – IEEE, 2004. – С. 383-386.
[26]. Jayaraj A., Venkatesh T., Murthy C. S. R. Loss classification in optical burst switching networks using machine learning techniques: improving the performance of tcp //IEEE Journal on Selected Areas in Communications. – 2008. – Т. 26. – №. 6. – С. 45-54.
[27]. Arouche Nunes B. A. et al. A machine learning framework for TCP round-trip time estimation //EURASIP Journal on Wireless Communications and Networking. – 2014. – Т. 2014. – С. 1-22.
[28]. Dai T., Zhang X., Guo Z. Learning-based congestion control for internet video communication over wireless networks //2018 IEEE International Symposium on Circuits and Systems (ISCAS). – IEEE, 2018. – С. 1-5.

[29]. Parlos A. G. Identification of the internet end-to-end delay dynamics using multi-step neuro-predictors //Proceedings of the 2002 International Joint Conference on Neural Networks. IJCNN'02 (Cat. No. 02CH37290). – IEEE, 2002. – Т. 3. – С. 2460-2465.
[30]. Bui V. et al. Long horizon end-to-end delay forecasts: A multi-step-ahead hybrid approach //2007 12th IEEE Symposium on Computers and Communications. – IEEE, 2007. – С. 825-832.
[31]. Belhaj S., Tagina M. Modeling and Prediction of the Internet End-to-end Delay using Recurrent Neural Networks //J. Networks. – 2009. – Т. 4. – №. 6. – С. 528-535.
[32]. Ahmed A. H. et al. Predicting high delays in mobile broadband networks //IEEE Access. – 2021. – Т. 9. – С. 168999-169013.
[33]. Li W. et al. QTCP: Adaptive congestion control with reinforcement learning //IEEE Transactions on Network Science and Engineering. – 2018. – Т. 6. – №. 3. – С. 445-458.
[34]. Jay N. et al. A deep reinforcement learning perspective on internet congestion control //International Conference on Machine Learning. – PMLR, 2019. – С. 3050-3059.
[35]. Fang J. et al. Reinforcement learning for bandwidth estimation and congestion control in real-time communications //arXiv preprint arXiv:1912.02222. – 2019.
[36]. Abbasloo S., Yen C. Y., Chao H. J. Classic meets modern: A pragmatic learning-based congestion control for the internet //Proceedings of the Annual conference of the ACM Special Interest Group on Data Communication on the applications, technologies, architectures, and protocols for computer communication. – 2020. – С. 632-647.
[37]. Sivakumar V. et al. Mvfst-rl: An asynchronous rl framework for congestion control with delayed actions //arXiv preprint arXiv:1910.04054. – 2019.
[38]. Zhang Z. et al. PBQ-Enhanced QUIC: QUIC with Deep Reinforcement Learning Congestion Control Mechanism //Entropy. – 2023. – Т. 25. – №. 2. – С. 294.
[39]. Tian H. et al. Spine: An efficient DRL-based congestion control with ultra-low overhead //Proceedings of the 18th International Conference on emerging Networking Experiments and Technologies. – 2022. – С. 261-275.
[40]. Shi H., Wang J. Intelligent TCP congestion control policy optimization //Applied Sciences. – 2023. – Т. 13. – №. 11. – С. 6644.
[41]. Andrade-Zambrano A. R. et al. A Reinforcement Learning Congestion Control Algorithm for Smart Grid Networks //IEEE Access. – 2024.
[42]. 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.
[43]. Nguyen K. et al. An approach to reinforce multipath TCP with path-aware information //Sensors. – 2019. – Т. 19. – №. 3. – С. 476.
[44]. Chao L. et al. A brief review of multipath tcp for vehicular networks //Sensors. – 2021. – Т. 21. – №. 8. – С. 2793.
[45]. Deng S. et al. WiFi, LTE, or both? Measuring multi-homed wireless internet performance //Proceedings of the 2014 Conference on Internet Measurement Conference. – 2014. – С. 181-194.
[46]. Scharf M., Kiesel S. NXG03-5: Head-of-line Blocking in TCP and SCTP: Analysis and Measurements //IEEE Globecom 2006. – IEEE, 2006. – С. 1-5.
[47]. De Coninck Q., Bonaventure O. Multipath quic: Design and evaluation //Proceedings of the 13th international conference on emerging networking experiments and technologies. – 2017. – С. 160-166.
[48]. Viernickel T. et al. Multipath QUIC: A deployable multipath transport protocol //2018 IEEE International Conference on Communications (ICC). – IEEE, 2018. – С. 1-7.
[49]. Wang S. et al. Deep reinforcement learning for dynamic multichannel access //International Conference on Computing, Networking and Communications (ICNC). – 2017. – С. 257-265.
[50]. Lee S., Yoo J. Reinforcement learning based multipath QUIC scheduler for multimedia streaming //Sensors. – 2022. – Т. 22. – №. 17. – С. 6333.
[51]. Luo J., Su X., Liu B. A reinforcement learning approach for multipath TCP data scheduling //2019 IEEE 9th Annual Computing and Communication Workshop and Conference (CCWC). – IEEE, 2019. – С. 0276-0280.
[52]. Roselló M. M. Multi-path scheduling with deep reinforcement learning //2019 European Conference on Networks and Communications (EuCNC). – IEEE, 2019. – С. 400-405.
[53]. Yu C. et al. Reliable cybertwin-driven concurrent multipath transfer with deep reinforcement learning //IEEE Internet of Things Journal. – 2021. – Т. 8. – №. 22. – С. 16207-16218.
[54]. Deutschmann J., Hielscher K. S. J., German R. An ns-3 model for multipath communication with terrestrial and satellite links //Measurement, Modelling and Evaluation of Computing Systems: 20th International

GI/ITG Conference, MMB 2020, Saarbrücken, Germany, March 16–18, 2020, Proceedings 20. – Springer International Publishing, 2020. – С. 65-81.
[55]. Mai T. et al. Self-learning congestion control of MPTCP in satellites communications //2019 15th International Wireless Communications & Mobile Computing Conference (IWCMC). – IEEE, 2019. – С. 775-780.
[56]. Xing Z. et al. A multipath routing algorithm for satellite networks based on service demand and traffic awareness //Frontiers of Information Technology & Electronic Engineering. – 2023. – Т. 24. – №. 6. – С. 844-858.
[57]. Lv G. et al. Chorus: Coordinating Mobile Multipath Scheduling and Adaptive Video Streaming //Proceedings of the 30th Annual International Conference on Mobile Computing and Networking. – 2024. – С. 246-262.
[58]. Berenji H. R. Fuzzy Q-learning: a new approach for fuzzy dynamic programming //Proceedings of 1994 IEEE 3rd International Fuzzy Systems Conference. – IEEE, 1994. – С. 486-491.
[59]. Zhou W. et al. TCP Vegas-V: Improving the performance of TCP Vegas //International Conference on Automatic Control and Artificial Intelligence (ACAI 2012). – IET, 2012. – С. 2034-2039.
[60]. Dong M. et al. {PCC} vivace: {Online-Learning} congestion control //15th USENIX Symposium on Networked Systems Design and Implementation (NSDI 18). – 2018. – С. 343-356.
[61]. Dong M. et al. {PCC}: Re-architecting congestion control for consistent high performance //12th USENIX Symposium on Networked Systems Design and Implementation (NSDI 15). – 2015. – С. 395-408.
[62]. De Oliveira R. L. S. et al. Using mininet for emulation and prototyping software-defined networks //2014 IEEE Colombian conference on communications and computing (COLCOM). – Ieee, 2014. – С. 1-6.
[63]. Riley G. F., Henderson T. R. The ns-3 network simulator //Modeling and tools for network simulation. – Berlin, Heidelberg: Springer Berlin Heidelberg, 2010. – С. 15-34.
[64]. Beale J., Orebaugh A., Ramirez G. Wireshark & Ethereal network protocol analyzer toolkit. – Elsevier, 2006.
[65]. Deri L. et al. ndpi: Open-source high-speed deep packet inspection //2014 International Wireless Communications and Mobile Computing Conference (IWCMC). – IEEE, 2014. – С. 617-622.
[66]. Mishra A., Leong B. Containing the Cambrian Explosion in QUIC Congestion Control //Proceedings of the 2023 ACM on Internet Measurement Conference. – 2023. – С. 526-539.

Информация об авторах / Information about authors

Иван Александрович СТЕПАНОВ – аспирант 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, an assistant at the Department of Computer Science and Computational Mathematics at MIPT. Research interests: network traffic analysis using machine learning.

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.

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.

Мария Кирилловна ИКОННИКОВА – младший научный сотрудник ИСП РАН. Сфера научных интересов: анализ сетевого трафика, машинное обучение.

Maria Kirillovna IKONNIKOVA is a junior researcher at the ISP RAS. Research interests: network traffic analysis, machine learning.

Андрей Андреевич БЕЛЕВАНЦЕВ – доктор физико-математических наук, ведущий научный сотрудник ИСП РАН, профессор МГУ. Сфера научных интересов: статический анализ программ, оптимизация программ, параллельное программирование.

Andrey Andreevich BELEVANTSEV – Dr. Sci. (Phys.-Math.), Prof., leading researcher at ISP RAS, Professor at MSU. Research interests: static analysis, program optimization, parallel programming.