DOI 10.37882/2223-2966.2026.03.09

МЕТОДЫ ПРОТОКОЛЬНОЙ ОПТИМИЗАЦИИ ВЕБ-РЕСУРСОВ: HTTP/2, HTTP/3 И ГЛОБАЛЬНОЕ РАСПРЕДЕЛЕНИЕ КОНТЕНТА

METHODS FOR PROTOCOL OPTIMIZATION OF WEB RESOURCES: HTTP/2, HTTP/3, AND GLOBAL CONTENT DISTRIBUTION

V. Zabelina
B. Goryachkin

Summary. The development of web technologies and the increasing volume of data delivered by web resources create an acute need for optimization of their performance. Traditional approaches to data transmission require evaluation of their impact on content delivery and reconsideration of architecture design in the context of multi-protocol and geographically distributed environments. This work addresses the problem of comprehensive optimization of web resources based on modern data transmission protocols, global content distribution systems, and caching strategies.

A detailed analysis of the evolution of data transmission protocols and their impact on web resource performance has been conducted. Practical methodologies for implementing HTTP/2 and HTTP/3 have been identified, with justification for selection based on target audience and network interaction conditions. The impact of CDN and Origin Shield usage on global performance indicators has been investigated, and optimal caching strategies for various content types have been proposed.

Keywords: HTTP/2, HTTP/3, QUIC, data transmission protocols, Content Delivery Network (CDN), Origin Shield, Core Web Vitals, Largest Contentful Paint (LCP), Interaction to Next Paint (INP), web resource performance, global content distribution, network latency, content caching, performance optimization.

Забелина Варвара Александровна
Московский государственный технический университет им. Н.Э. Баумана
varvara.zabelina@bmstu.ru
Горячкин Борис Сергеевич
Кандидат технических наук, Доцент,
Московский государственный технический университет им. Н.Э. Баумана
bsgor@mail.ru

Аннотация. Развитие веб-технологий и увеличение объемов, отдаваемых веб-ресурсами, создают острую необходимость в оптимизации их производительности. Традиционные подходы к передаче данных нуждаются в оценке их влияния на доставку контента и переосмысление построения архитектуры в контексте многопротокольной и географически-распределенной среды. В данной работе рассматривается проблема комплексной оптимизации веб-ресурсов на основе современных протоколов передачи данных, систем глобального распределения контента и кеширования.

Проведен детальный анализ эволюции протоколов передачи данных и их влияния на производительность веб-ресурсов. Определены практические методики применения HTTP/2 и HTTP/3 с обоснованием выбора в зависимости от целевой аудитории и условий сетевого взаимодействия. Исследовано влияние использования CDN и Origin Shield на глобальные показатели производительности, предложены оптимальные стратегии кеширования для различных типов контента.

Ключевые слова: HTTP/2, HTTP/3, QUIC, протоколы передачи данных, Content Delivery Network (CDN), Origin Shield, Core Web Vitals, Largest Contentful Paint (LCP), Interaction to Next Paint (INP), производительность веб-ресурсов, глобальное распределение контента, задержка сети, кеширование контента, оптимизация производительности.

Введение

Развитие веб-технологий на протяжении последних трёх десятилетий демонстрирует постоянную эволюцию подходов к передаче данных и оптимизации сетевых соединений. Веб-приложения стали содержать сотни ресурсов (скрипты, стили, изображения, шрифты), а объёмы пользовательских данных возросли экспоненциально. Из-за этого фокус сместился на динамическую адаптацию архитектуры передачи данных к условиям сетевого взаимодействия и географическому распределению пользователей — это именно то, что обеспечивают современные протоколы и глобальные сети доставки контента

Google, анализируя миллиарды сессий, определили три ключевых метрики, известные как Core Web Vitals, которые оценивают качество пользовательского опыта и коррелируют с конверсией [1]:

- Largest Contentful Paint (LCP) — время загрузки наиболее крупного видимого элемента (целевое значение: ≤ 2,5 сек)
- Cumulative Layout Shift (CLS) — индекс неожиданных смещений содержимого (целевое значение: ≤ 0,1)
- Interaction to Next Paint (INP) — время отклика на взаимодействие пользователя (целевое значение: ≤ 200 мс)

Согласно исследованиям, страницы, соответствующие всем порогам Core Web Vitals, имели на 24 % меньше отказов пользователей во время загрузки по сравнению со страницами, которые не соответствовали этим стандартам.

Достижение высоких показателей этих метрик требует комплексного подхода, включающего правильный выбор протоколов передачи данных и оптимальную архитектуру доставки контента. В статье рассматриваются методы протокольной оптимизации и глобального распределения контента, которые требуют минимальных капитальных вложений при максимальном эффекте улучшения производительности веб-ресурсов.

Развитие веб-технологий на протяжении последних трёх десятилетий демонстрирует постоянную эволюцию подходов к передаче данных и оптимизации сетевых соединений. Когда в 1989 году Тим Бернерс-Ли создал Всемирную паутину с протоколом HTTP, веб был статичным — простые HTML-страницы, минимальные изображения, никакой интерактивности. HTTP/1.1, стандартизированный в 1997 году, был спроектирован именно для этих условий: медленные сети, простые документы, последовательная загрузка ресурсов.

Однако к началу 2010-х годов произошла кардинальная трансформация. Веб-приложения стали содержать сотни ресурсов (скрипты, стили, изображения, шрифты), а объёмы пользовательских данных возросли экспоненциально. HTTP/1.1 с его архитектурой одного соединения на ресурс и проблемой head-of-line blocking стал узким местом производительности. В 2015 году появился HTTP/2, решивший эти проблемы мультиплексированием потоков и бинарной протоколизацией. Но этого было недостаточно для мобильных сетей с высокой задержкой и потерей пакетов.

В 2022 году HTTP/3, основанный на протоколе QUIC, кардинально пересмотрел фундаментальный транспортный уровень, перейдя с TCP на UDP.

В Таблице 1 приведено рассмотрено базовое сравнение характеристик.

HTTP/1.1 — проблемы архитектуры

HTTP/1.1 использует один TCP-поток на соединение. Проблема возникает, когда требуется загрузить множество файлов: браузер должен открывать отдельное соединение для каждого ресурса или ждать завершения предыдущего запроса. Время ожидания в очереди составляет значительную часть от общего времени загрузки.

Расчёт времени загрузки при HTTP/1.1 ведется следующим образом:

$$t_{HTTP/1.1} = \sum_{i=1}^{n} \left(t_{queue_i} + t_{connection_i} + t_{download_i} \right) \qquad (1)$$

где: — t_{queue_i} — время в очереди запроса — $t_{connection_i}$ — время установления соединения — $t_{download_i}$ — время скачивания файла *i*

Таблица 1.
Сравнение характеристик HTTP/1.1, HTTP/2 и HTTP/3

Характеристика	HTTP/1.1	HTTP/2	HTTP/3
Базовый транспортный протокол	TCP	TCP	UDP (QUIC)
Год стандартизации	1997	2015	2022
Мультиплексирование	Нет (последовательное)	Да (одно соединение)	Да (оптимизированное)
Head-of-Line Blocking	Да (критическая проблема)	Да (на уровне TCP)	Нет
Компрессия заголовков	Нет	HPACK	QPACK
Ручное управление приоритетами	Нет	Да	Да (более гибкое)
Шифрование	Опционально	Опционально (TLS 1.2)	Обязательно (TLS 1.3)
Установление соединения (RTT)	1 RTT	1 RTT + TLS	0-RTT (0-RTT resumption)
Поддержка миграции соединения	Нет	Нет	Да
Скорость decompression в браузере	N/A	Базовая	До 64 % быстрее, чем gzip
Браузерная поддержка	100 %	97 %	73 % (быстро растёт)
Поддержка CDN (Content Delivery Network)	100 %	100 %	85 % и выше

HTTP/2 — решение на уровне приложения

HTTP/2 решает проблему head-of-line blocking (устранение проблемы блокировки начала очереди) на уровне приложения путём использования фреймов и потоков (streams). Множество потоков могут использовать одно TCP-соединение, что значительно сокращает количество операций установки соединения и очередей.

Ключевые улучшения [4]:

- Бинарный протокол — в HTTP/1.1 заголовки передаются как текстовые строки с повторяющимися именами и значениями в каждом запросе, для этого нужны разделители (\r\n, двоеточия, пробелы), которые занимают байты, но не несут полезной нагрузки. В бинарном фрейме часть этой структуры заменяется фиксированными полями и флагами, которые можно упаковать в несколько байт, без лишних символов (ожидаемое уменьшение размера на 10 %–20 %).

- Мультиплексирование — одновременная передача нескольких потоков
- Server Push — сервер может отправить ресурсы до того, как клиент их запросит
- HPACK компрессия заголовков — ожидаемое сжатие заголовков на 30–50% благодаря тому, что повторяющиеся поля передаются как небольшие ссылки на ранее отправленные, экономя трафик, чем повторяющиеся текстовые строки в HTTP/1.1.

Практическое улучшение времени загрузки:

$$t_{HTTP/2} = \\ = \max\left(t_{connection} + t_{handshake}\right) + \sum_{i=1}^{n} t_{download_i} + \Delta t_{overhead} \quad (2)$$

Ожидаемое сокращение времени загрузки: в среднем на 30–50 % [5] для типового веб-сайта с 50+ ресурсами.

HTTP/3 — устранение проблем TCP

HTTP/3 решает фундаментальную проблему TCP — head-of-line blocking на уровне транспорта. TCP доставляет все пакеты в строгом порядке; потеря одного пакета задерживает доставку всех последующих. QUIC (Quick UDP (User Datagram Protocol) Internet Connections), используемый в HTTP/3, использует UDP в качестве базового протокола и может обрабатывать потерю пакетов независимо для каждого потока.

Улучшения HTTP/3 над HTTP/2:

1. Нулевая задержка при повторном подключении (0-RTT)
- Клиент может отправлять данные при установлении соединения, если он подключался ранее
- Сокращение времени на 50–100 мс для повторных визитов

2. Миграция соединения
- Connection ID в QUIC позволяет серверу идентифицировать клиента без IP-адреса.
- В TCP при смене сети меняется и IP-адрес клиента из-за чего соединение разрывается и приходится устанавливать новое. В HTTP/3 даже после смены IP, клиент продолжает отправлять пакеты с тем же Connection ID, что позволяет пользователю перейти с Wi-Fi на 4G без разрыва соединения

3. Независимая обработка потоков
- Потеря пакета в одном потоке не влияет на другие
- На нестабильных сетях улучшение производительности примерно на 12–20 % по сравнению с предыдущим протоколом [7].

Несмотря на преимущества и развитие протоколов перед нами встает вопрос правильного их применения. Точный процент постоянно меняется, но примерно половина всего интернет-трафика все еще использует HTTP/1.1 из-за его широкого распространения. Однако не стоит рассматривать его использование не только из-за появления более современных методов, но и из-за наличия угрозы безопасности — HTTP desync-атаки [9] все еще возможны на данной версии протокола. На основе проведенного выше анализа протоколов подведем итог, когда стоит остаться на HTTP/2, а когда можно перейти на HTTP/3.

Лучше использовать HTTP/2 следует при следующих факторах:

- веб-сайты, работающие на всех серверах (Apache, Nginx);
- требуется максимальная совместимость (99%+ браузеров);
- стабильное проводное соединение;
- нет критичности к микросекундным задержкам.

HTTP/3 лучше использовать, если:

- мобильные приложения с частой сменой сети;
- потоковые сервисы (видео, аудио);
- API с критичным требованием к задержке (latency);
- целевая аудитория — современные браузеры (Chrome, Firefox, Safari).

Такие рекомендации по выбору обусловлены следующими ограничениями:

1. Использование HTTP/2 предполагает обязательное применение TLS, что при первоначальном подключении может добавлять порядка сотен миллисекунд к времени установления соединения.
2. Отдельные устаревшие сетевые устройства и фильтрующая инфраструктура могут ограничивать или блокировать трафик QUIC (UDP на порту 443).
3. Динамическое сжатие контента в режиме реального времени для HTTP/3 в ряде сценариев оказывается более ресурсоёмким по сравнению с обслуживанием уже закешированных сжатых объектов.

В связи с вышеизложенным можно предложить следующие решения ограничений:

1. Для снижения накладных расходов при установлении защищённых соединений целесообразно использовать TLS 1.3 с поддержкой режима 0-RTT возобновления сессии.
2. Для обеспечения доступности сервиса при неоднородной сетевой инфраструктуре рекомендуется реализовать механизм автоматического отката на HTTP/2 в случаях недоступности QUIC, а также периодически тестировать его проходимость.
3. Для уменьшения вычислительной нагрузки и задержек при передаче контента имеет смысл кешировать результаты компрессии на уровне CDN и по возможности отдавать пользователям заранее сжатые версии ресурсов.

На графиках на рисунках 1 и 2 представлена статистика по запросам используемым и поддерживающим протоколы HTTP/2 и HTTP/3 за 2019–2025 год.

Для анализа метрик производительности с применением разных протоколов был использован источник HTTP Archive 2024 [11][12], обеспечивающий репрезентативную выборку данных по реальным веб-страницам (в том числе Alexa Top-1000 — наиболее посещаемые сайты). На основе опубликованных данных формируется профиль классического высоконагруженного веб-ресурса и распределение используемых технологий. HTTP Archive предоставляет агрегированную статистику по реальным загрузкам страниц, включая медианные

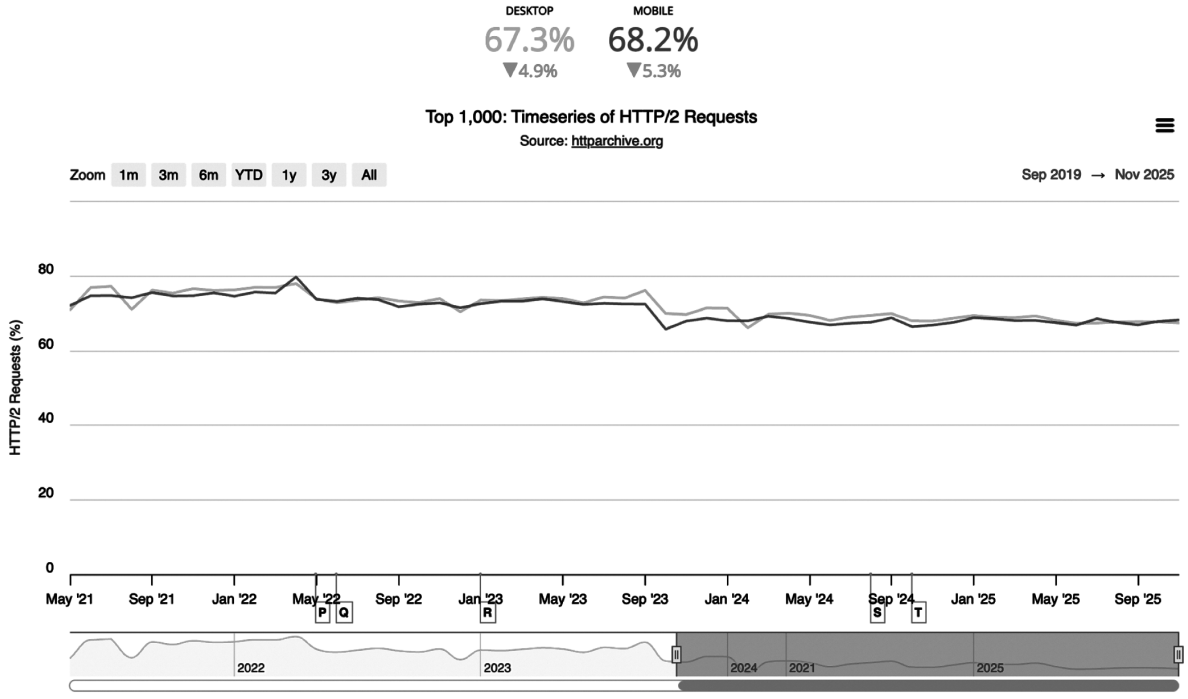

Рис. 1. Использование HTTP/2 в запросах

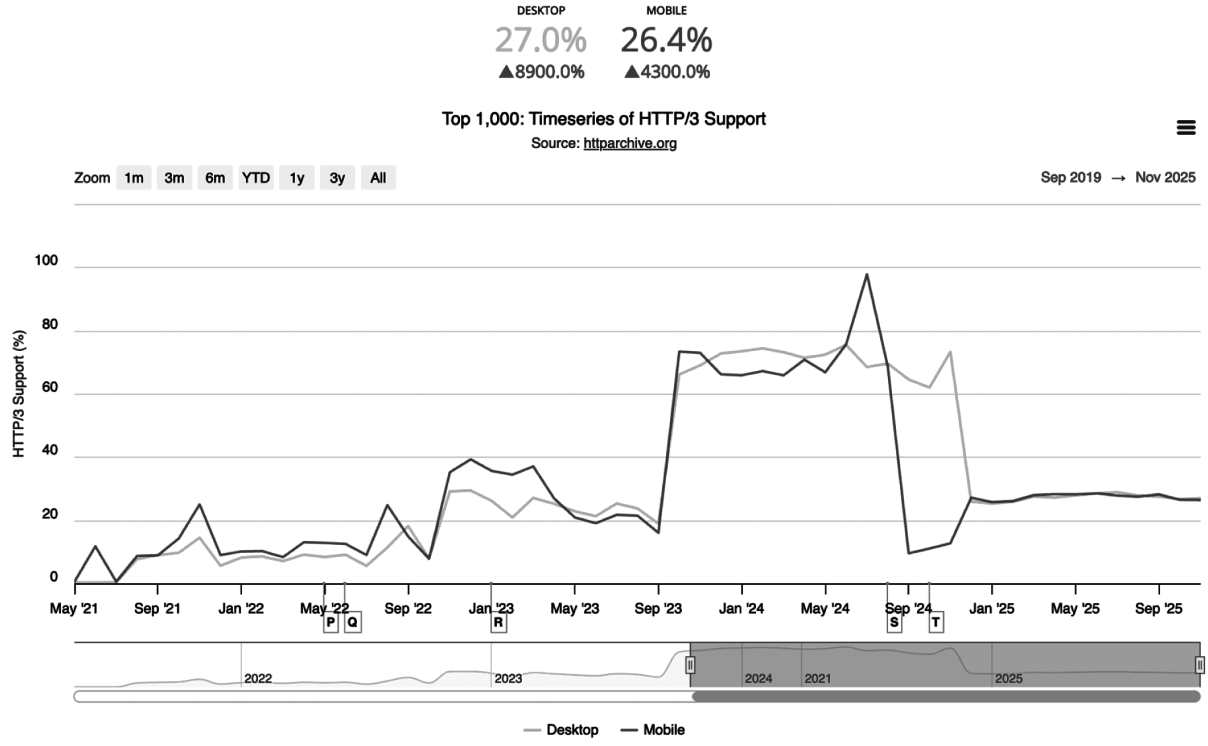

Рис. 2. Поддержка HTTP/3 для запросов

значения ключевых метрик веб-производительности для различных сетевых условий и протоколов. Относительные эффекты от перехода на HTTP/2 и HTTP/3 оценены на основе опубликованных сравнительных бенчмарков протоколов в контролируемых условиях (в частности, обзора Catchpoint по HTTP/3 vs HTTP/2 [3]).

Значения в таблице представляют собой медианные результаты для каждого протокола, основанные на опубликованных исследованиях и данных вышеуказанных источников. Оценка производительности сводится к вычислениям следующих трех ключевых метрик (таблица 2):

1. Время подключения (Connection Establishment Time) — интервал от начала установления соединения до возможности передачи данных приложения.
2. First Contentful Paint (FCP) — момент первого появления значимого содержимого (текст, изображения) в окне браузера.
3. Относительное падение производительности при потере пакетов в нестабильной 3G-сети — показатель деградации скорости загрузки при фиксированном уровне потерь.

Таблица 2.

Результаты применения разных протоколов

Метрика	HTTP/1.1	HTTP/2	HTTP/3
Время подключения (Connection Establishment Time)	250–300 мс (6–8 соединений × 40–50 мс)	80–120 мс (1 соединение) (60%ускорение)	20–40 мс (0-RTT при повторном подключении) (88 % ускорение относительно HTTP/1.1)
Время загрузки First Contentful Paint (FCP)	3.2 сек (медиана для мобильной 4G)	1.8 сек (44 % улучшение)	1.4 сек (56 % улучшение относительно HTTP/1.1)
Потеря пакетов на нестабильной сети (3G)		падение на 25–35 %	падение на 5–8 %

Content Delivery Network (CDN) — это система географически распределённых серверов, которые кешируют и доставляют контент пользователю с минимальной задержкой.

Стандартная архитектура:

- Origin Server — основной сервер, где хранится оригинальный контент
- Edge Nodes — серверы CDN, распределённые по миру
- PoP (Point of Presence) — точки присутствия CDN в разных регионах
- Intelligent Routing — система, которая направляет пользователя на ближайший edge node

Метрики производительности был получены на основе обобщения региональных бенчмарков CDN, опубликованных провайдерами CDN-услуг BlazingCDN [13] и Cloudflare [14]. В таблице 3 приведены агрегированные показатели для пяти регионов:

- медианная задержка до первого байта (First-Byte Latency) — время от отправки HTTP-запроса до получения первого байта ответа от ближайшего узла CDN; измеряется в миллисекундах;
- медианная пропускная способность (Median Throughput) — средняя скорость передачи данных при загрузке контента из CDN в данном регионе; измеряется в мегабитах в секунду (Mbps);
- доля запросов, обслуживаемых из кеша CDN (Cache Hit Ratio, CHR) — процент запросов, которые обслуживаются из кеша региональных edge-узлов без обращения к исходному серверу;
- параметр «Лучший CDN» отражает провайдера, показавшего минимальную задержку и/или максимальную пропускную способность в соответствующем регионе в опубликованных бенчмарках [13].

Таблица 3.

Меры задержки по регионам (2024)

Регион	Медианная задержка до первого байта	Медианная пропускная способность	Доля попаданий в кеш (Cache Hit Ratio)	Лучший CDN
North America	37 ms	146 Mbps	94 %	Fastly
Europe	42 ms	132 Mbps	94 %	BlazingCDN
Asia-Pacific	67 ms	98 Mbps	88 %	CloudFlare
Latin America	85 ms	76 Mbps	82 %	Limelight
Africa	120 ms	45 Mbps	76 %	Akamai

Cache Hit Ratio (CHR) — процент запросов, которые обслуживаются из кеша edge node, без обращения к origin server.

Среднее время ответа через CDN представляется как математическое ожидание двух исходов: запрос обслуживается из кеша edge-узла с вероятностью CHR или перенаправляется на origin-сервер с вероятностью (1-CHR). Тогда среднее время отклика записывается:

$$t_{avg} = \left(CHR \times t_{edge}\right) + \left(\left(1 - CHR\right) \times t_{origin}\right) \qquad (3)$$

где: — t_{edge} — время отклика от edge, — t_{origin} — время отклика от origin

Рассмотрим пример расчёта влияния на производительность. Для типичного глобального сценария при-

нимаем ориентировочные диапазоны t_{edge}=35–50 мс и t_{origin}=150–300 мс, основанные на опубликованных региональных бенчмарках CDN (BlazingCDN [13], Cloudflare [14]) и усреднённых измерениях задержки до ближайшего узла CDN и до удалённых origin-серверов. Эти значения используются как репрезентативные оценки, позволяющие показать порядок влияния роста CHR на среднее время отклика.

При CHR = 94 % и среднем трафике:

$$t_{avg} = (0.94 \times 40) + (0.06 \times 200) = \\ = 37.6 + 12 = 49.6\,\text{ms} \quad (4)$$

Без CDN (всё с origin):

$$t_{avg} = 200\,\text{ms}$$

Процент улучшения считается как относительное уменьшение времени отклика:

$$\text{Улучшение} = \frac{t_{origin} - t_{CDN}}{t_{origin}} \cdot 100\% = \\ = \frac{200 - 49{,}6}{200} \cdot 100\% \approx 75{,}2\% \quad (5)$$

Эффективная работа CDN зависит не столько от мощности инфраструктуры, сколько от правильной классификации контента и применения адаптированной стратегии кеширования. Разные типы контента имеют радикально отличающиеся характеристики изменчивости, размера и значения для пользовательского опыта. Понимание этих различий позволяет максимизировать Cache Hit Ratio и минимизировать нагрузку на origin server.

Статический контент (CSS, JS, изображения):

Такой контент можно охарактеризовать следующими признаками:

- Изменяется редко (обычно только при развёртывании новой версии приложения)
- Может быть кеширован агрессивно без риска подачи устаревшей информации
- Часто имеет versioning в имени файла (app.abc123def.js, style.hash456.css)
- Размер: от килобайт до нескольких мегабайт на файл

Сценарий для конфигурации кеширования:

TTL (Time to Live): 30 дней-1 год.

Cache Header: Cache-Control: public, max-age=31536000.

CHR: 95%+.

Для неизменяемого статического контента (CSS, JS, изображения), как правило, используется стратегия «versioned assets»: имена файлов включают хеш версии (например, app.abc123.js). При таком подходе CDN может кешировать эти файлы на edge-узлах с максимально возможным временем жизни (TTL) — вплоть до года при заголовке Cache-Control: public, max-age=31536000. Даже в случае кратковременной недоступности origin-сервера пользователи продолжают получать уже закэшированные копии ресурсов. Обновление приложения приводит к генерации файлов с новым именем (например, app.def789.js), что автоматически заставляет браузер и CDN загрузить и закешировать свежую версию.

В условиях типового веб-приложения с десятками статических ресурсов такой режим даёт очень высокий коэффициент попаданий в кеш (CHR ожидается 95% и выше). Это означает, что из 100 000 запросов к статике в день лишь около 5 000 доходят до origin-сервера, тогда как остальные обслуживаются с ближайших edge-узлах, где задержка ответа обычно составляет 35–50 мс вместо 150–300 мс при обращении к origin из статистики приведенной выше.

HTML страницы:

HTML-страницы можно охарактеризовать следующими признаками:

- Содержат динамический контент (дата обновления, отсчёт продаж, число пользователей онлайн)
- Часто персонализированы на уровне сегмента пользователей (устройство, язык, регион)
- Регулярно обновляются, но изменения не критичны в реальном времени
- Размер: 50–300 KB на страницу после сжатия

Сценарий для конфигурации кеширования:

TTL: 5-60 минут.

Cache Header: Cache-Control: public, max-age=300, s-maxage=3600.

CHR: 70–80%.

Для HTML-страниц обычно используется более консервативная политика кеширования. Cache-Control: max-age=300, s-maxage=3600 задаёт два разных интервала: max-age=300 ограничивает время жизни копии в браузерном кеше примерно пятью минутами, тогда как s-maxage=3600 разрешает прокси-кешам и CDN-узлам хранить ту же страницу до одного часа. По истечении этих интервалов включается механизм валидации: клиент или edge-узел отправляет условный запрос с заголовком If-None-Match (на основе ETag) или If-Modified-Since (на основе Last-Modified). Если контент на origin-сервере не изменился, он возвращает статус 304 Not Modified, и кешированная версия продолжает использоваться без повторной передачи полного тела ответа. В практике это позволяет достичь коэффициента попаданий в кеш (CHR) порядка 70–80 % для HTML-страниц, то есть боль-

шая часть запросов обслуживается с edge-узлов, что уменьшает добавочную задержку по сравнению с обращением к origin-серверу на сотни миллисекунд.

API ответы (JSON):

API ответов можно охарактеризовать следующими признаками:

- Часто содержат real-time данные (курсы валют, цены акций, активные пользователи)
- Могут быть полностью идентичны для группы пользователей (публичные данные) или уникальны для каждого (приватные данные)
- Размер: от 1 KB до нескольких сотен KB
- Высокая вариативность — один API может давать разные ответы в зависимости от параметров запроса

TTL: 1–5 минут.
Cache Header: Cache-Control: public, max-age=300.
CHR: 50–70 %.

В случае API-эндпоинтов кеширование на уровне CDN обычно настраивается с существенно меньшим временем жизни, чем для статического контента, поскольку ответы чаще меняются. При этом важным условием является привязка кеша к полному URL, включая строку запроса: запросы вида /api/products?category=electronics и /api/products?category=clothing должны рассматриваться как разные объекты кеша. Если интерфейс поддерживает сортировку, пагинацию и сложные фильтры, каждая уникальная комбинация параметров приводит к появлению отдельной записи в кеше, что в предельном случае может приводить к эффекту «взрыва кеша», когда число кешируемых объектов быстро растёт из-за большого количества вариантов параметров.

Для смягчения этой проблемы современные CDN-провайдеры (например, Fastly и Cloudflare) позволяют нормализовать строку запроса: исключать из неё малозначимые параметры, упорядочивать их или группировать запросы с близкими наборами параметров, тем самым уменьшая количество уникальных ключей кеша и повышая вероятность повторного использования уже сохранённых ответов. На практике грамотная конфигурация такого кеширования для API позволяет достигать коэффициента попаданий в кеш порядка 50–70 %, то есть обслуживать значительную долю запросов непосредственно с edge-узлов и тем самым уменьшать нагрузку на backend и сокращать время ответа на десятки-сотни миллисекунд по сравнению с прямым обращением к исходному серверу.

Динамический контент (user-specific):

Под динамическим контентом в данном контексте понимаются ответы, которые формируются индивидуально для конкретного пользователя: личный профиль, история заказов, персональные рекомендации и т.п. Такие данные:

- отличаются для разных пользователей и могут содержать персональную информацию;
- часто изменяются (новые заказы, изменения профиля, обновление рекомендаций);
- обычно доступны только после аутентификации и зависят от контекста сессии (cookie, токен).

В силу этих причин динамический контент, как правило, не кешируется на общих edge-узлах CDN, чтобы не допустить выдачу персональных данных одному пользователю из кеша другого. Запросы этого типа обрабатываются только origin-сервером, где уже может использоваться внутренний кеш в памяти (например, Redis или Memcached) с ключами вида user_id:profile либо более сложными комбинациями параметров (например, user_id:recommendations:category:sort_order). Для такого класса ответов коэффициент попаданий в CDN-кеш (CHR), по сути, равен нулю, поскольку каждый запрос требует обращения к origin-логике приложения.

Origin Shield представляет собой дополнительный уровень кеша между edge-узлами CDN и исходным сервером и используется для консолидации запросов к origin. В сценариях с высокой повторяемостью контента это позволяет кратно (в несколько раз) сократить число обращений [18] к исходному серверу и заметно повысить суммарный коэффициент попаданий в кеш (CHR) на уровне всей сети — вплоть до значений порядка 95–98% для наиболее часто запрашиваемых ресурсов.

Приведённый расчёт демонстрирует потенциальный эффект по достижению сокращения нагрузки (таблица 4):

Таблица 4.

Без Origin Shield	С Origin Shield
100 000 пользователей в день	
Edge CHR: 92 %	Shield CHR: 98 %
8 000 запросов проходят до origin	До origin: 160 запросов
Origin processing: 500 мс/запрос = 4000 сек нагрузки	Origin processing: 80 сек нагрузки

Процент сокращения нагрузки рассчитывается как относительное уменьшение суммарного времени обработки запросов на origin:

$$\text{Относительное сокращение нагрузки} = \frac{t_{NoShield} - t_{shield}}{t_{NoShield}} \cdot 100\% = \frac{4000 - 80}{4000} \cdot 100\% = 98\% \quad (6)$$

Для иллюстрации экономического эффекта CDN рассмотрим условный сценарий e-commerce-сайта с глобальной аудиторией. Исходные параметры выбирались

на основе типичных значений, рассмотренных выше в работе и приводимых в отраслевых отчётах о производительности и конверсии. Далее предполагаем, что подключение CDN уменьшает Time to First Byte (TTFB) для удалённых регионов с 250–300 мс до порядка 45 мс и сокращает полное время загрузки страницы примерно вдвое.

Исходные данные, взятые на основе средних оценок, рассчитанных выше:

- 100 GB статического контента
- 10 млн запросов в месяц
- 30 % из них — из дальних регионов (Africa, Asia)
- Среднее время загрузки без CDN (на основе статистик собранной HTTP Archive): 3.5 сек
- Bounce rate (показатель отказов) из-за скорости загрузки сайта [19]: 8 %

Предполагаем из медианных данных, без CDN Time to First Byte для дальних регионов:

- Network latency: 150–200 ms
- Server processing: 100 ms
- Итого: 250–300 ms
- Full page load: 3.5–4.5 сек

С учетом исходных данных: 8 % от 10 млн = 800 000 пользователей — уходят после открытия страницы и не совершают никаких действий. Тогда потеря выручки при среднем заказе в 20$: 800 000 × $20 = $16 млн в год.

Предполагаем на основе рассмотренных выше данных, что с подключением CDN (CloudFlare Pro: $200/месяц) Time to First Byte:

- Edge latency: 35–50 ms
- Cache hit: 94 %
- Итого TTFB: 45 ms (в 6х быстрее)
- Full page load: 1.2–1.8 сек (в 2х быстрее)

В данном случае теряется только: 2.5 % от 10 млн = 250 000 пользователей. Таким образом с подключением CDN предотвращена потеря: 800 000–250 000 = 550 000 пользователей. Получаем, что при среднем заказе в 20$ восстановление выручки будет: 550 000 × $20 = $11 млн в год

Годовой эффект от внедрения CDN оценивается как восстановленная выручка за счёт снижения доли отказов. Для этого высчитывается метрика ROI — это показатель окупаемости инвестиций, то есть насколько выгодным оказался проект с учётом вложенных в него средств. При стоимости CDN 200 USD в месяц расчётный ROI можно записать в виде:

$$\begin{aligned} \text{ROI} &= \frac{\text{Доп. выручка - Расходы на CDN}}{\text{Расходы на CDN}} = \\ &= \frac{11000000 - (200 \times 12)}{200 \times 12} = \\ &= \frac{11000000 - 2400}{2400} = 4582 \end{aligned} \tag{7}$$

Это означает, что каждый доллар, вложенный в CDN, приносит более 4 500 долларов дополнительной выручки в год, и затраты на сервис окупаются практически мгновенно на масштабе годового трафика.

Проведённое исследование показало, что современные протоколы передачи данных и глобальные системы распределения контента являются критически важными компонентами архитектуры веб-ресурсов. Переход с HTTP/1.1 на HTTP/2 может дать улучшение производительности на 30–50 %, а использование HTTP/3 может обеспечить дополнительное ускорение на 12–20 % на нестабильных сетях.

Применение CDN позволяет снизить Time to First Byte во многих случаях на 75 % и существенно улучшить метрики Core Web Vitals для глобальной аудитории. Использование дополнительного слоя Origin Shield может снизить нагрузку на origin server до 98 %.

Ключевой вывод исследования заключается в том, что выбор протокола и архитектуры доставки контента должен быть адаптирован к профилю целевой аудитории, типу контента и условиям сетевого взаимодействия. Комбинированное применение HTTP/2 и/или HTTP/3 с CDN и Origin Shield обеспечивает оптимальное соотношение между капитальными вложениями и эффектом оптимизации.

ЛИТЕРАТУРА

1. Understanding Core Web Vitals and Google search results // Google Search Central. — URL: https://developers.google.com/search/docs/appearance/core-web-vitals.
2. Горячкин Б.С., Ханмурзин Т.И., 2022. Повышение эффективности работы с веб-ресурсом за счет инструментария системного программиста. Динамика сложных систем-XXI век, 16(3), p.26.
3. HTTP/3 vs HTTP/2: A Detailed Comparison // Catchpoint. — 2024. — URL: https://www.catchpoint.com/http3-vs-http2.
4. Протоколы HTTP/1.1 и HTTP/2.0: эволюция, архитектура, сравнение и практические примеры // Proselyte Software Engineering. — 2025. — URL: https://proselyte.net/http-evolution/.
5. Modern HTTP: HTTP/1.1 vs HTTP/2 For Performance // DebugBear. — 2024. — URL: https://www.debugbear.com/blog/http1-vs-http2.
6. Belshe Mike, Roberto Peon, and Martin Thomson. Hypertext transfer protocol version 2 (HTTP/2). No. rfc7540. 2015.

7. Iyengar Jana, and Martin Thomson. «QUIC: A UDP-based multiplexed and secure transport.» In RFC 9000. 2021.
8. Fielding Roy, and Julian Reschke. Hypertext transfer protocol (HTTP/1.1): Message syntax and routing. No. rfc7230. 2014.
9. HTTP/1.1 must die: the desync endgame // Port Swigger. — 2025. — URL: https://portswigger.net/research/http1-must-die.
10. Web Vitals Report 2024 // Google Search Central Documentation. — 2024. — URL: https://web.dev/vitals/.
11. Reports with dashboards. // HTTP Archive. — URL: httparchive.org.
12. HTTP. 2024 Web Almanac // HTTP Archive. — 2024. — URL: https://almanac.httparchive.org/en/2024/http.
13. Content Delivery Network Performance Benchmarks Across Regions // BlazingCDN. — 2025. — URL: https://blog.blazingcdn.com/en-us/content-delivery-network-performance-benchmarks-across-regions.
14. Network performance update: Birthday Week 2024 // Cloudflare. — 2024. — URL: https://blog.cloudflare.com/network-performance-update-birthday-week-2024/.
15. CDN Performance Metrics: Strategies for Superior Content Delivery // IoRiver. — 2024. — URL: https://www.ioriver.io/blog/cdn-performance-metrics.
16. CDN Optimization & Multi CDN Strategies // Dynadot. — 2025. — URL: https://www.dynadot.com/blog/global-cdn-strategies.
17. Cloudflare Performance Benchmarks 2024 // Cloudflare. — 2024. — URL: https://blog.cloudflare.com/radar-2024-year-in-review/.
18. AWS CloudFront Documentation: Origin Shield // Amazon Web Services Documentation. — 2024. — URL: https://docs.aws.amazon.com/AmazonCloudFront/latest/DeveloperGuide/origin-shield.html.
19. Breaking down the impact of web performance on ecommerce conversions // Noibu. — 2023. — URL: https://www.noibu.com/blog/the-impact-of-web-performance-on-ecommerce-conversions.

Журнал «Современная наука: актуальные проблемы теории и практики»