INFORMATION TECHNOLOGY, COMPUTER SCIENCE AND MANAGEMENT
ИНФОРМАТИКА, ВЫЧИСЛИТЕЛЬНАЯ ТЕХНИКА И УПРАВЛЕНИЕ

UDC 004.738.5
https://doi.org/10.23947/2687-1653-2026-26-2-2228

Original Empirical Research

Efficiency and Prospects of the Experimental QUIC Protocol

Jahed Rahmani, Serafim P. Sukharev ✉
Moscow Technical University of Communication and Informatics, Moscow, Russian Federation
✉ s.p.suharev@mtuci.ru

EDN: CNWWGW

Abstract

Introduction. Ensuring and improving the availability of web resources on the Internet is a crucial task for developers of information systems. A critical role in the page accessibility via the Hypertext Transfer Protocol (HTTP) is played by the protocol version and its transport-level implementation. The QUIC (Quick User Datagram Protocol Internet Connections) protocol, developed by Google, provides an increase in resource loading speed through the use of User Datagram Protocol (UDP) in HTTP/3. However, QUIC has an experimental status, and existing research primarily focuses on theoretical aspects or general performance metrics in the global network. At the same time, the following aspects remain insufficiently studied: simultaneous comparison of three protocol versions under unified controlled conditions, practical complexities of configuration and the effect of congestion control algorithms on application-level metrics, labor costs for implementation and configuration, tuning efforts, and quantifying gains under controlled condition.

These gaps create a disconnect between theoretical expectations and practical implementation. Therefore, the objective of this study is to experimentally evaluate the applied performance of HTTP/3 (QUIC) under controlled conditions on a unified testbed, including a comparison of HTTP/1.1, HTTP/2, and HTTP/3, an analysis of the impact of the CUBIC and BBR congestion control algorithms, and documentation of the HTTP/3 server configuration procedure.

Materials and Methods. A testbed was deployed based on a virtual server running the Linux operating system (OS) and the nginx web server supporting HTTP/1.1, HTTP/3 (QUIC), and congestion control algorithms CUBIC and Bottleneck Bandwidth and Round-trip propagation time (BBR). The Google Chrome browser over a 4G network was used as the client. Performance was evaluated using the Time to First Byte (TTFB) metric, file download speed, and total web page load time. Measurements were performed multiple times using Chrome DevTools and client-side scripts. The paper provides a detailed description of the server configuration process for enabling HTTP/3.

Results. The experiments showed that using HTTP/3 (QUIC) reduced the time to first byte by 23.06% and accelerated full page load by 9.5% compared to HTTP/1.1. The theoretical model predicted a TTFB reduction of 71.43% due to the combined QUIC and TLS 1.3 handshake. The observed discrepancy was attributed to the specifics of UDP traffic processing by internet service providers, the experimental status of the implementation, and mobile channel instability. When downloading large files, the CUBIC and BBR algorithms provided comparable average speeds (≈13.12 MB/s and 12.75 MB/s, respectively). However, BBR transmitted 18.2% more data within the first three seconds, demonstrating faster ramp-up to operational speed and a more stable transfer profile.

Discussion. Practical results partially differed from theoretical estimates: the observed latency reduction was lower than expected due to Transport Layer Security (TLS) implementation features, UDP traffic processing by Internet providers, and hardware characteristics. It is shown that the advantages of QUIC/HTTP/3 are most noticeable under conditions of multiple short requests and high latency. The advantage of BBR over CUBIC is realized not in long-duration transfers, but when loading numerous small page resources — a typical web interaction scenario. To improve the reliability of performance evaluation, further experiments are planned under various network conditions, protocol implementations, and geographically distributed clients.

Conclusion. The study confirmed the advantages of HTTP/3 (QUIC): TTFB decreased by 23.06%, and page load time by 9.5%. However, the theoretical model predicted a greater reduction, indicating the influence of implementation and

network environment factors. The comparison of CUBIC and BBR revealed the advantage of BBR when transferring small-sized files. Despite the complexity of HTTP/3 configuration, the transition is justified for services with a significant number of resources. The experimental limitations indicate the need for further studies under different network scenarios.

Keywords: QUIC, HTTP/3, transport layer protocols, congestion control algorithms, BBR, CUBIC, TTFB, web performance

Acknowledgments. The authors sincerely acknowledge the Department of Network Information Technologies and Services, Moscow Technical University of Communications and Informatics, for their support in conducting this research. Special thanks are expressed to the journal editorial team for their attentive attitude to the manuscript, prompt support in the preparation of the publication, and organization of the peer-review process.

For Citation. Rahmani J, Sukharev SP. Efficiency and Prospects of the Experimental QUIC Protocol. *Advanced Engineering Research* (*Rostov-on-Don*). 2026;26(2):2228. https://doi.org/10.23947/2687-1653-2026-26-2-2228

Оригинальное эмпирическое исследование

Эффективность и перспективы экспериментального протокола Quick User Datagram Protocol Internet Connections

Д. Рахмани ✉, С.П. Сухарев ✉
Московский технический университет связи и информатики, г. Москва, Российская Федерация
✉ s.p.suharev@mtuci.ru

Аннотация

Введение. Обеспечение и повышение доступности веб-ресурсов в сети Интернет представляет собой актуальную задачу для разработчиков информационных систем. Критическую роль в доступности страниц по протоколу Hypertext Transfer Protocol (HTTP) играет версия протокола и его реализация на транспортном уровне. Протокол Quick User Datagram Protocol Internet Connections (QUIC), разработанный компанией Google, позволяет добиться прироста скорости загрузки ресурсов за счёт применения протокола User Datagram Protocol (UDP) в HTTP/3. Однако QUIC имеет статус экспериментального, а в существующих исследованиях основное внимание уделяется теоретическим аспектам или общим показателям в глобальной сети. При этом остаются недостаточно изученными: одновременное сопоставление трёх версий протокола в единых контролируемых условиях, практические сложности конфигурирования и влияние алгоритмов контроля перегрузок на прикладные метрики; трудозатраты на внедрение и настройку; усилия по настройке и количественная оценка выигрыша в контролируемых условиях. Указанные пробелы формируют разрыв между теоретическими ожиданиями и практической реализацией. Поэтому целью данного исследования явилась экспериментальная оценка прикладной эффективности HTTP/3 (QUIC) в контролируемых условиях на едином тестовом стенде, включающая сопоставление HTTP/1.1, HTTP/2, HTTP/3, анализ влияния алгоритмов контроля перегрузок CUBIC и BBR и документирование процедуры конфигурирования HTTP/3-сервера.

Материалы и методы. Для исследования был развёрнут тестовый стенд на основе виртуального сервера под управлением операционной системы (ОС) Linux и веб-сервера nginx с поддержкой HTTP/1.1, HTTP/3 (QUIC), а также алгоритмов контроля перегрузок CUBIC и Bottleneck Bandwidth and Round-trip propagation time (BBR). В качестве клиента использовался браузер Google Chrome в сети 4G. Производительность оценивалась по метрике Time to First Byte (TTFB), скорости загрузки файлов и времени полной загрузки веб-страницы. Замеры выполнялись многократно с использованием Chrome DevTools и клиентских скриптов. В статье подробно описан процесс настройки сервера для работы с HTTP/3.

Результаты исследований. Эксперименты показали, что применение HTTP/3 (QUIC) сокращает время до первого байта на 23,06 % и ускоряет полную загрузку страницы на 9,5 % по сравнению с HTTP/1.1. Теоретическая модель прогнозировала снижение TTFB на 71,43 % за счёт объединённого рукопожатия QUIC и TLS 1.3. Выявленное расхождение обусловлено особенностями обработки UDP-трафика операторами, экспериментальным статусом реализации и нестабильностью мобильного канала. При скачивании крупных файлов алгоритмы CUBIC и BBR обеспечили сопоставимую среднюю скорость (≈13,12 и 12,75 МБ/с соответственно). Однако BBR за первые три секунды передал на 18,2 % больше данных, демонстрируя более быстрый выход на рабочую скорость и стабильный профиль передачи.

Обсуждение. Практические результаты частично расходятся с теоретическими оценками: снижение задержки оказалось меньше прогнозируемого из-за особенностей реализации Transport Layer Security (TLS), обработки UDP-трафика провайдерами и характеристик оборудования. Показано, что преимущества QUIC/HTTP/3 наиболее заметны при множественных коротких запросах и высоких задержках. Преимущество BBR перед CUBIC

реализуется не на длительных передачах, а при загрузке множества небольших ресурсов страницы — типичного сценария веб-взаимодействий. Для повышения достоверности оценки производительности планируется расширение экспериментов с различными сетевыми условиями, реализациями протокола и географически распределёнными клиентами.

Заключение. Исследование подтвердило преимущество HTTP/3 (QUIC): TTFB снизился на 23,06 %, время загрузки страницы — на 9,5 %. При этом теоретическая модель прогнозировала более существенное сокращение, что указывает на влияние особенностей реализации и сетевой среды. Сравнение CUBIC и BBR выявило преимущество BBR при передаче файлов малого объёма. Несмотря на сложность конфигурирования HTTP/3, переход оправдан для сервисов со значительным количеством ресурсов. Ограничения эксперимента требуют расширения исследований в различных сетевых сценариях.

Ключевые слова: QUIC, HTTP/3, протоколы транспортного уровня, алгоритмы контроля перегрузок, BBR, CUBIC, TTFB, производительность веб-приложений

Благодарности. Авторы выражают искреннюю признательность сотрудникам кафедры «Сетевые информационные технологии и сервисы» Московского технического университета связи и информатики за поддержку в проведении исследования. Особая благодарность выражается редакционной команде журнала за внимательное отношение к рукописи, оперативное сопровождение подготовки публикации и организацию процесса рецензирования.

Для цитирования. Рахмани Д., Сухарев С.П. Эффективность и перспективы экспериментального протокола Quick User Datagram Protocol Internet Connections. *Advanced Engineering Research* (*Rostov-on-Don*). 2026;26(2):2228. https://doi.org/10.23947/2687-1653-2026-26-2-2228

Introduction. Today, available web resources constitute a valuable knowledge base and are an integral part of the operations of organizations across different sectors. The development of the online advertising market is a key factor in the growth of the entire advertising industry in the Russian Federation [1]. Providing comfortable interaction of the end user with information systems on the Internet is a priority task for companies of any scale. The research shows that even a slight increase in web page performance can have a significant positive impact on the financial performance of a business [2, 3].

Data is transmitted to the World Wide Web (WWW) via the HTTP protocol, which has three main versions [4]. Web server support for modern protocols can favorably affect the user experience when browsing websites [5]. HTTP/3 is based on the Quick User Datagram Protocol Internet Connections (QUIC) protocol [6], which was developed as an alternative to the Transmission Control Protocol (TCP) used in previous versions of HTTP [7], in order to increase the speed and security of Internet connections.

Analysis of the current literature identifies both the progress made and major discrepancies within the findings. In [8], the results of the large-scale deployment of QUIC in the Google infrastructure are presented: the delay in search service responses decreased by 8.0% for desktop platforms and 3.6% for mobile users, the frequency of video playback interruptions decreased by 18.0% and 15.3 %, respectively. The authors note that the gain grows with an increase in the Round-Trip Time (RTT) metric and packet loss, but it turns out to be lower than theoretically expected, including due to implementation limitations and the specifics of operators' network equipment. This means that transferring conclusions to moderate-scale deployments requires independent experimental verification.

The authors [9] compared QUIC and TCP in a laboratory environment with a locally installed server and found minimal differences between HTTP/2 and HTTP/3 in multithreaded transmission. In another article, [10], on the contrary, in a similar scenario with a loss rate of 12%, a five-fold advantage of HTTP/3 over HTTP/2 was recorded. This contradiction is due to the difference in network conditions. It remains unresolved in relation to regular deployment without simulated network disruptions. At the same time, the works do not include HTTP/1.1 as a starting point for comparisons and do not contain data on the TTFB metric.

In [11], client-server protocols were investigated from the standpoint of fault tolerance and security under real network conditions. It was found that when switching to QUIC, the actual performance gain varied depending on the configuration of the intermediate network equipment. This conclusion is consistent with the observation of the authors [8] about the discrepancy between the theoretically expected and practically achieved gains from the protocol.

Review paper on the prospects of HTTP/3 and QUIC [12] shows that the combined QUIC handshake and built-in TLS 1.3 encryption form the theoretical basis for significantly reducing latency. However, the researchers point to difficulties in practical deployment due to the immaturity of protocol support in server software. Moreover, the paper does not provide a specific configuration procedure.

Paper [13] has documented that 19.38% of requests in a large mobile web service fall into the slow-start phase, where the statically set initial congestion window size does not correspond to the actual bandwidth. This study thus confirms that the slow start mechanism of the CUBIC algorithm creates measurable delays when transferring small files, which are typically used to make up web pages, which serves as a basis for studying BBR as an alternative solution.

Paper [14] substantiated a BBR algorithm that used channel capacity and propagation delay estimation instead of packet loss as a congestion signal. Deployment of BBR in Google B4 backbone network resulted in throughput increases of 2–25 times compared to CUBIC. However, the paper analyzed aggregated throughput metrics, while the dynamics of the transmission rate over time (the sawtooth profile characteristic of CUBIC compared to the more stable BBR profile) were not experimentally visualized.

Study [15] documented that in QUIC implementations, the CUBIC algorithm remained the default, while BBR was provided as an option. Experiments conducted by the authors in the ns-3 simulator showed that BBR provided higher throughput compared to CUBIC at high channel delays, but at low bandwidth it was inferior to it in terms of the number of lost packets. However, the paper does not analyze the dynamics of the transmission rate over time, which leaves open the question of the stability of the speed profile of each of the algorithms.

This review highlights three interrelated gaps. Existing papers do not compare HTTP/1.1, HTTP/2, and HTTP/3 simultaneously under consistent, controlled conditions: most studies are limited to comparing HTTP/2 and HTTP/3. Protocol performance data varies significantly depending on testing conditions, and benchmarking under normal conditions without proxy servers or artificially introduced packet loss in an emulated network environment is underrepresented. Furthermore, the practical procedure for configuring a standard web server for HTTP/3 is not documented in the scientific literature. Thus, the research gap lies in the lack of studies that, within a unified methodological framework, simultaneously: compare HTTP/1.1, HTTP/2, and HTTP/3 under controlled conditions; document the procedure for configuring a standard web server to support HTTP/3; evaluate the impact of the CUBIC and BBR congestion control algorithms on applied web access metrics. Addressing this gap determines the objective and tasks of this study.

The research objective was to experimentally evaluate the application performance of HTTP/3 (QUIC) under controlled conditions on a single testbed through comparing HTTP/1.1, HTTP/2 and HTTP/3, analyze the impact of the CUBIC and BBR congestion control algorithms, and document the HTTP/3 server configuration procedure.

To achieve this objective, the following tasks were completed:

– deploy a testbed supporting HTTP/1.1, HTTP/2, and HTTP/3;

– document the web server configuration sequence for HTTP/3;

– measure and compare TTFB and full page load time for different HTTP versions;

– compare the behavior of the CUBIC and BBR algorithms when transferring a large file;

– interpret the results obtained taking into account theoretical expectations and published research.

Materials and Methods

Equipment. To conduct the experiments, a testbed was prepared consisting of a virtual machine rented from the IXcellerate data center in Moscow. The following specifications were used: one 2.4 GHz Virtual Central Processing Unit (vCPU), 2 GB of RAM, and a personal workstation running Ubuntu 22.04.5 Long-Term Support (LTS). Both devices had internet access. The virtual machine had a static Internet Protocol (IP) address with the domain name serafimdev.com.

Research Plan. Three experiments were planned for this study: comparison of connection establishment speed using HTTP/1.1, HTTP/2, and HTTP/3; comparison of test page loading speed using HTTP/1.1, HTTP/2, and HTTP/3; and comparison of data transfer speed using the CUBIC and BBR congestion control protocols.

The experiments were performed in the following sequence. On a virtual machine, a web server (nginx) was installed and configured with three subdomains, each responsible for serving traffic over one protocol version: HTTP/1.1, HTTP/2, and HTTP/3. To assess connection speed, the TTFB metric was measured on each subdomain. A web page template consisting of 32 files was then uploaded to the server, and the time it took to fully load was measured. For an experiment

comparing congestion control algorithms, a 3 GB file was uploaded to the server. Downloads were performed sequentially with the CUBIC algorithm active (default configuration) and after switching to BBR, in both cases recording the transfer rate profile. Subsequent analysis of the research findings included a comparison of the experimental data with theoretical values obtained from the model presented below.

When using HTTP/1.1 with TLS 1.2 encryption, the total time to send the first byte of data consisted of several connection initialization stages (Table 1), each of which took a certain number of round trips between network nodes. Thus, the expected initialization time for a secure connection via HTTP/1.1 was obtained using formula (1).

Table 1

HTTP/1.1 Connection Initialization Steps

Step	Assignment	Contribution to initialization time
1	Three-way Handshake of TCP protocol	1.5 RTT
2	ClientHello/ServerHello message exchange with protocol cipher negotiation TLS [16]	1 RTT
3	Transferring certificate and session keys for TLS protocol [16]	1 RTT

$$T_{TCP} = 1.5\,RTT + 1\,RTT + 1\,RTT = 3.5\,RTT. \quad (1)$$

The QUIC protocol combined the transport and cryptographic handshake based on TLS 1.3, reducing the total cost to 1 RTT. From formula (2), we obtained the theoretical gain in terms of the TTFB metric.

$$\mu = \frac{T_{TCP} - T_{QUIC}}{T_{TCP}} \cdot 100\% = \frac{3.5\,RTT - 1\,RTT}{3.5\,RTT} \approx 71.43\%. \quad (2)$$

Tooling. All measurements were performed using a 4G mobile internet connection. Nginx version 1.27.4 was installed on the virtual machine, as support for HTTP/3 connections was added to the Nginx web server starting with version 1.25.0[1].

For the experiments, three subdomains were configured: http1.serafimdev.com, http2.serafimdev.com, and http3.serafimdev.com, with Let's Encrypt certificates for the HTTP/1.1, HTTP/2, and HTTP/3 protocols, respectively. Certbot, a software tool for automating certificate management, was used for the configuration.

The server settings were verified using Google Chrome developer tools. When exchanging data over HTTP/3, the protocol column should display h3 values (Fig. 1). These tools also provided information for assessing web page performance and component loading speeds. They were used to measure full page load times.

Resource	Status	Protocol
images.html	200	h3
1.jpg	200	h3
2.jpg	200	h3
3.jpg	200	h3
4.jpg	200	h3
5.jpg	200	h3

Fig. 1. HTTP/3 protocol in developer tools

The TTFB metric was obtained using a JavaScript script that performed server requests and recorded the metric values. A similar approach was applied to collect the data needed to visualize graphs of resource utilization rates by various overload control algorithms. When making measurements, it was important to disable query caching in the developer's tools to eliminate the impact of the local cache on the results. The collected data was processed using the numpy and matplotlib libraries for Python. To integrate the non-standard BBR overload control algorithm into the operating system kernel — Ubuntu 22.04.5 LTS was used in this study — it was necessary to download the corresponding module — sudo modprobe tcp_bbr and change the value of the variable — sudo sysctl -w net.ipv4.tcp_congestion_control=bbr.

[1] Mukhin S. *How to Enable HTTP/3 in NGINX.* SergeyMukhin.com. URL: https://sergeymukhin.com/blog/kak-vklyuchit-http3-v-nginx (accessed: 11.10.2025). (In Russ.)

Procedures. The Internet connection is often unstable; therefore, a statistical analysis of the results was applied to avoid the impact of outliers on the results of the study. In experiments to measure the connection initialization time and page loading speed, measurements for each protocol were performed 20 times. Then, the interquartile range method was applied to identify outliers, and the arithmetic mean — to aggregate the remaining measurements into the final result. The upper and lower limits of emissions were determined from formulas (3) and (4).

$$\text{Lower limit} = Q1 - 1.5 \times IQR; \quad (3)$$

$$\text{Lower limit} = Q3 + 1.5 \times IQR, \quad (4)$$

where $Q1$ — 25th percentile, $Q3$ — 75th percentile, $IQR = Q3 - Q1$.

When analyzing the transfer rate, the JavaScript XMLHttpRequest module was used for measurements, which provided the number of bytes downloaded every unequal number of milliseconds. Due to the small gap between measurements and the instability of the network, the raw data contained a large number of outliers. To obtain more visual graphs, the results were smoothed using a moving mean with a window of 31 dimensions.

Research Results

HTTP/3 Server Setup Procedure. Documentation of the testbed configuration process was an independent research task. To organize access to web content via HTTP/3, a server with an operating system and access to it was required. The study used a server with the Ubuntu 22.04.5 LTS operating system with remote SSH access. During the setup process, a key difficulty was identified when organizing an HTTP/3 web server — the need to use the nginx mainline repository instead of the standard one.

The default Ubuntu 22.04.5 LTS repository contains nginx version 1.18, while HTTP/3 support has been available since version 1.25.0. To install the latest version, it was necessary to add the main repository with the latest software versions to the system using the commands shown in Figure 2.

```
curl https://nginx.org/keys/nginx_signing.key | gpg --dearmor \
  | sudo tee /usr/share/keyrings/nginx-archive-keyring.gpg >/dev/null

echo "deb [signed-by=/usr/share/keyrings/nginx-archive-keyring.gpg] \
http://nginx.org/packages/mainline/ubuntu `lsb_release -cs` nginx" \
  | sudo tee /etc/apt/sources.list.d/nginx.list
```

Fig. 2. Adding the nginx mainline repository to the operating system's package sources list

These instructions write the key used to sign packages from the new repository to the system's trusted keystore: /usr/share/keyrings and add the address of the nginx mainline repository to the package sources list /etc/apt/sources.list.d, specifying the path to the package signing location. After completing these steps, installing nginx via the apt package manager will use the repository that contains all the latest versions of the service.

Since encryption is built into the QUIC protocol, a certificate and domain name are mandatory requirements when using HTTP/3. In previous versions of the HTTP protocol, TLS encryption is a separate layer, which adds latency when establishing a connection but allows for unsecured data exchange. Unencrypted HTTP is suitable for networks where encryption is already provided via a Virtual Private Network (VPN). Therefore, using HTTP/3 in such networks necessitates the configuration of certificates. The testbed used issued certificates from Let's Encrypt using certbot, a certificate automation software.

Additionally, changes must be made to the site configuration file (Fig. 3). The key lines in the configuration are lines 3, 10, 12, and 16. Line 12 contains an instruction enabling the use of the HTTP/3 protocol. Line 16 specifies the Alt-Svc header. If this header is present in the server's response, the client switches to using HTTP/3 (if supported). Therefore, lines 2 and 3 specify support for standard TCP connections and UDP over QUIC. Subsequently, the header value is cached, and no header exchange occurs over TCP. Since QUIC operates on TLS version 1.3, this version is explicitly specified on line 10.

```
server {
    listen 443 ssl;
    listen 443 quic reuseport;

    server_name http3.serafimdev.com;

    ssl_certificate     /etc/letsencrypt/live/http3.serafimdev.com/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/http3.serafimdev.com/privkey.pem;

    ssl_protocols TLSv1.3;

    http3 on;
    quic_gso on;
    quic_retry on;

    add_header Alt-Svc 'h3=":443"; ma=86400';

    root /var/www/http3.serafimdev.com;
    index index.html;

    location / {
        try_files $uri $uri/ =404;
    }
}

server {
    listen 80;
    server_name http3.serafimdev.com;
    return 301 https://$host$request_uri;
}
```

Fig. 3. Nginx configuration file for HTTP/3 site

TTFB. TTFB is a metric that shows how long it takes from sending a request to the start of resource transfer. A JavaScript script was used to conduct the measurements. The resulting values are presented in Table 2.

Table 2

TTFB Measurements for Different HTTP Protocol Versions

HTTP/1,1, ms	HTTP/2, ms	HTTP/3, ms
20.5	80.3	69.4
28.3	28.4	26.8
38.3	24.4	29.1
27.0	30.4	16.3
41.3	29.1	21.1
24.8	25.7	17.5
30.1	33.2	18.4
26.7	30.4	23.2
30.9	46.0	22.4
24.7	15.6	23.1
29.2	26.0	24.1
33.3	23.6	19.7
25.9	26.1	24.3
34.7	26.3	23.7
33.3	25.1	23.5
33.3	32.5	25.4
25.9	26.5	26.4
29.4	19.7	22.6
25.0	26.9	24.5
22.7	28.6	15.8

The following values were obtained:

– HTTP/1.1: no outliers, mean 29.27 ms;

– HTTP/2: three outliers, mean 27.23 ms;

– HTTP/3: one outlier, mean 22.52 ms.

$$\mu = \frac{29.27 - 22.52}{29.27} \cdot 100\% \approx 23.06\%. \quad (5)$$

According to calculations from formula (5), the experiment showed that the transition from HTTP/1.1 to HTTP/3 with the QUIC protocol provided a reduction in time to first byte (TTFB) by 23.06%.

Page Load Time. To study the impact of using the QUIC protocol on the loading speed of a typical web page, a page template consisting of 32 files, totaling 728 kilobytes, was found in open sources. The values obtained are presented in Table 3**.**

Table 3

Page Load Time for Different Versions of HTTP Protocol

HTTP/1,1, ms	HTTP/2, ms	HTTP/3, ms
1140	1060	781
826	794	699
1160	953	630
726	851	733
731	940	622
757	814	652
746	846	669
815	908	978
791	747	764
834	832	917
709	1060	725
830	868	601
951	744	890
1020	785	643
677	725	906
806	711	739
1100	695	647
743	922	687
775	750	652
739	658	678

The following values were obtained:

– HTTP/1.1: 3 outliers, mean 792.7 ms;

– HTTP/2: no outliers, mean 833.15 ms;

– HTTP/3: 1 outlier, mean 717.63 ms.

$$\mu = \frac{792.7 - 717.63}{792.7} \cdot 100\% \approx 9.5\%. \quad (6)$$

Calculation from formula (6) shows a reduction in page load time of 9.5% when using HTTP/3 and QUIC compared to HTTP/1.1 and TCP.

Transfer Speed. A 3 GB file was used for the experiment, transmitted via the HTTP/2 protocol. The choice of HTTP/2 for this experiment was driven by methodological considerations: the comparison aimed to isolate the effect of the congestion control algorithm on the throughput profile, without additional effects from the transport protocol. The HTTP/3 (QUIC) protocol implements its own congestion control stack on top of UDP, while HTTP/2 uses the standard TCP stack of the operating system kernel, in which switching between CUBIC and BBR algorithms was performed directly through Linux kernel parameters without changing the application-layer protocol logic. A file was downloaded using the CUBIC and BBR congestion control algorithms. Figures 4 and 5 show the initial fragments of the download speed graphs using CUBIC and BBR, respectively.

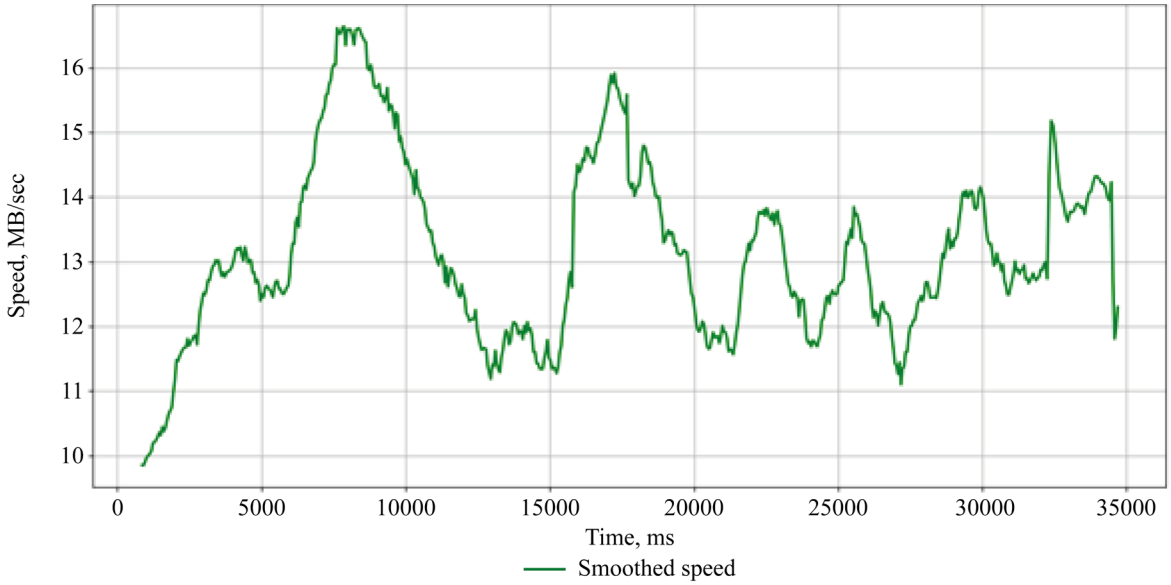

Fig. 4. Download speed when using CUBIC

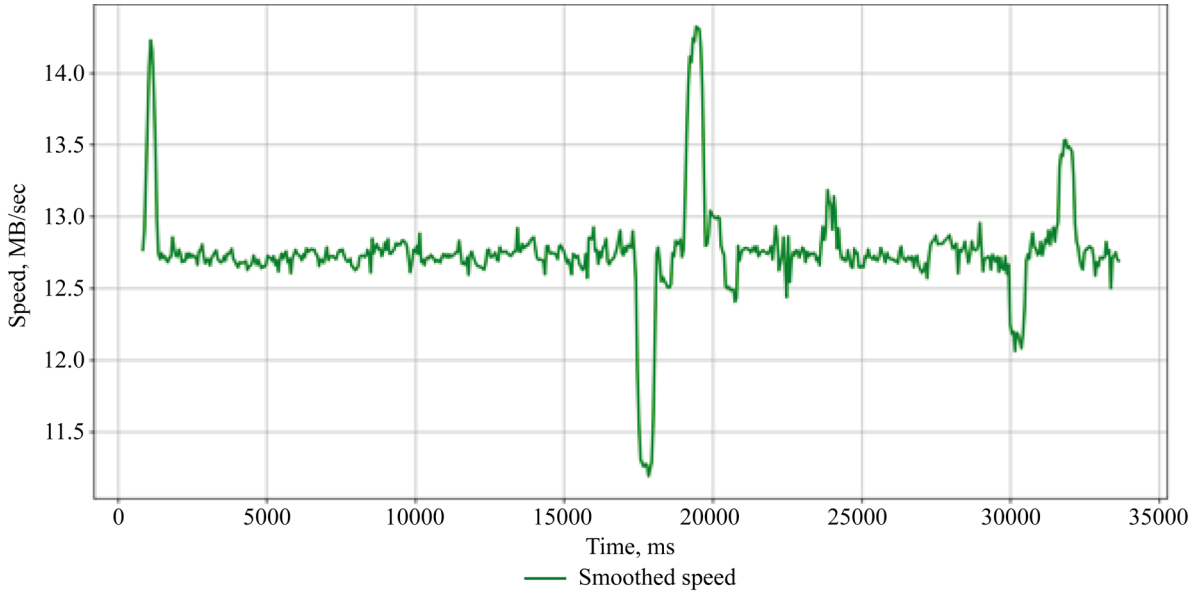

Fig. 5. Download speed when using BBR

When using the BBR algorithm, speed remained constant most of the time. Average download speed was 13.12 MB/s for CUBIC, and 12.75 MB/s for BBR.

The area under the graph curve reflects the amount of information transmitted. It was calculated from formula (7). In the first 3 seconds of transmission, 23.1568 MB of data were transmitted using CUBIC, and 27.3622 MB using BBR.

$$S = \int_0^{3000} v(t)\,dt \approx \sum_{i=0}^{3000} \frac{v_i + v_{i+1}}{2}\left(t_{i+1} - t_i\right). \quad (7)$$

Discussion. The results obtained show that the practical gain from the implementation of HTTP/3 (QUIC) is confirmed experimentally, but its value turns out to be significantly lower than theoretically expected. This requires interpretation taking into account the characteristics of the network environment, software implementation, and measurement methodology.

Configuration Complexities and Operational Risks. The described configuration procedure presents a number of practical challenges that must be considered when planning a production HTTP/3 deployment.

The first challenge is the inability to use standard virtual machine and container images for automated deployment. Most official Ubuntu images and popular Nginx container images based on Alpine Linux or Debian include a version of Nginx from the distribution repository that does not support HTTP/3. When using such images in orchestration systems (Kubernetes, Docker Compose), each node requires either manually adding the mainline repository and reinstalling nginx, or building a custom image with the required version. This complicates automation and increases operational costs for infrastructure maintenance [17].

The second complication stems from the protocol negotiation mechanism via the Alt-Svc header. The first connection between the client and the server is always established over TCP (HTTP/1.1 or HTTP/2), and only after receiving the Alt-Svc header in the response does the browser switch to QUIC for subsequent requests. If there is an intermediary node between the client and the server — a load balancer, reverse proxy, or CDN node — that does not support QUIC or blocks UDP traffic on port 443, the Alt-Svc header either does not reach the client, or the client is unable to establish a UDP connection. This behavior complicates diagnostics and creates the risk of a hidden lack of effect from protocol implementation. According to the QUIC core specification [18], if UDP transport is unavailable, the client should fall back to a TCP connection without explicitly signaling an error. That makes the absence of HTTP/3 in real traffic difficult to diagnose without explicit instrumental verification. To verify the use of HTTP/3, explicit verification via browser developer tools or server logs with protocol version information are required.

A prerequisite for establishing an HTTP/3 connection on the client side is a browser with QUIC support enabled. In Google Chrome, support for the protocol has been enabled by default since version 87, but in corporate environments, it can be disabled via group policies. With QUIC disabled, the browser will not establish HTTP/3 connections, regardless of the correct server configuration and the presence of the Alt-Svc header.

Interpretation of TTFB Results. The combined QUIC and TLS 1.3 handshake, specified in RFC 9001 [19] and based on the TLS 1.3 specification [20], theoretically reduces initialization to 1 RTT, but the practical efficiency of this mechanism is sensitive to the implementation details of the cryptographic stack on specific hardware and in a specific software version. A 23.06% reduction in TTFB when switching from HTTP/1.1 to HTTP/3 experimentally confirms the advantage of the unified QUIC handshake, which eliminates the additional RTT, typical of separate TCP and TLS initialization. However, the measured gain is significantly lower than the theoretically expected value of 71.43%, which is explained by a combination of factors.

Processing of UDP packets on the provider's intermediate network equipment is often less optimized compared to TCP traffic: routers and firewalls are traditionally configured for the TCP load profile, and some operators apply additional inspection or throttling of UDP flows [8, 11]. QUIC implementations in nginx and Google Chrome remained experimental at the time of the experiment, which manifested itself in suboptimal stack parameter values and additional overhead at the application level. The 4G mobile channel introduces an unstable variable component of latency, which offsets some of the gain from the reduced RTT during connection establishment. A similar discrepancy between theory and practice was recorded in Google large-scale deployment of QUIC, where the actual search latency reduction was 8.0% for desktop users and 3.6% for mobile users, instead of the theoretically expected values [8], and was also confirmed in domestic studies of real-world network conditions [11]. This finding also aligns with the data from ITSumma[2], which observed a 12.4% decrease in TTFB. Collectively, this points to a systemic discrepancy between the theoretical and practical performance of the protocol at its current stage of evolution.

Page Load Time Interpretation. The 9.5% speedup in full test page load time using HTTP/3 compared to HTTP/1.1 is primarily due to multiplexing requests within a single QUIC connection and eliminating Head-of-Line blocking at the transport layer. When loading a test page consisting of 32 files, HTTP/3 processes all requests in parallel without deadlocking, whereas HTTP/1.1 is limited by the number of parallel TCP connections.

A notable anomaly in the HTTP/2 results is its average load time (833.15 ms) exceeding that of HTTP/1.1 (792.7 ms). This effect is an artifact of the statistical methodology used, rather than an indicator of any actual protocol inefficiency. In the HTTP/1.1 sample, the interquartile range method excluded three values exceeding the upper bound, significantly lowering the resulting mean. The average load times for HTTP/1.1 and HTTP/2 across all twenty measurements without removing outliers are 843.8 ms and 833.15 ms, respectively. This confirms HTTP/2's superior load performance compared to HTTP/1.1. The HTTP/2 data had no upper-bound outliers, indicating that the protocol achieved a more even distribution of response times by multiplexing requests and prioritizing streams within a single TCP connection. Thus, the comparison of cleaned means does not correctly reflect the real relationship between protocols: HTTP/2 shows higher stability and less variability, while the final mean value after removing outliers turned out to be statistically lower for HTTP/1.1.

[2] ITsumma. *HTTP/3 and QUIC: Future of Fast Internet*. Habr. URL: https://habr.com/ru/companies/itsumma/articles/497520 (accessed: 11.03.2026). (In Russ.)

Interpretation of CUBIC and BBR. Average transfer rates for a 3 GB file using CUBIC (13.12 MB/s) and BBR (12.75 MB/s) are comparable: the 3% difference is within the measurement error for an unstable mobile channel. This result is predictable for long-term transfers of a single file: over hundreds of seconds, both algorithms reach a steady-state channel load, and differences in initialization strategies no longer determine the final average speed.

The fundamental difference between the algorithms is evident in the transmission profile. CUBIC exhibits a characteristic sawtooth speed profile (Fig. 4): it increases monotonically from starting values of less than 10 MB/s to the operating level in approximately four seconds. After that, it cyclically decreases during periods of packet loss. BBR, which uses Bandwidth-Delay Product (BDP) throughput estimation, maintains a stable speed from the first seconds.

When working with the Web, most requested files are small. Assume that the size of a single resource file for a web page is 63 KB, the initial slow-start window size is 10 kilobytes (the default setting on Linux systems), and the maximum window size is 64 KB. With slow-start, the CUBIC algorithm uses an exponential function to increase the congestion window [21].

$$A = \left\lfloor log_2 \frac{cwnd_{max}}{cwnd_0} \right\rfloor = \left\lfloor log_2 \frac{64}{10} \right\rfloor = 3. \quad (8)$$

The calculation result obtained from formula (8) shows that transferring the file requires tripling the congestion window size, which takes 3 RTT. Had the transmission started with a congestion window of 64, the transfer could have been accomplished in just 1 RTT.

It is this initial period that determines the practical significance of BBR for web services. A typical page resource — a script, stylesheet, font — is transferred in one or two RTT and fits within CUBIC slow-start phase, during which the algorithm has not yet reached its congestion window. In this scenario, BBR ensures immediate access to optimal speed, which ultimately reduces the time it takes to download multiple small resources in parallel. Thus, BBR advantage over CUBIC is realized not during long transfers, but during page resource downloads, accompanied by multiple short requests immediately after establishing a connection.

Quantitative confirmation of the advantage of BBR in the initial phase of transmission is provided by a comparison of the areas under the curves of the speed graphs calculated from formula (7) by the trapezoidal method. In the first three seconds of transmission, the CUBIC algorithm ensured the transfer of 23.16 MB of data, while BBR provided the transfer of 27.36 MB, which is 18.2% more. This result is consistent with the experimental results in [15], where higher BBR throughput was recorded at high RTT values, which are typical for intercontinental connections.

A promising direction for further research is to experimentally test the benefits of BBR directly in a web scenario: measuring the full page load time of a page consisting of multiple small resources using the CUBIC and BBR algorithms in combination with each version of the HTTP protocol. Such an experiment will allow us to quantify the extent to which the accelerated speed of BBR translates into reduced page load time in a real web scenario, as well as determine which combinations of protocol version and congestion management algorithm produce the most pronounced practical effect.

Study Limitations and Practical Recommendations. While the results obtained are practically valuable for evaluating the gains of switching to HTTP/3, several limitations of the experimental testbed restrict the applicability of the conclusions.

The setup included a single virtual machine with one vCPU and a single client connected via a 4G mobile network. The mobile network introduces unstable, variable latency and inconsistent bandwidth, increasing measurement variability and making it difficult to isolate protocol effects from channel effects. Transport protocol performance is significantly dependent on network topology, RTT values, packet loss rates, and intermediate equipment policies; therefore, transferring the results to other network scenarios requires independent experimental verification [22, 23].

The limited sample size (20 measurements per protocol) provides sufficient accuracy for an initial assessment, but is insufficient for constructing statistically robust confidence intervals. The lack of geographically distributed clients precludes assessing the impact of high RTT — the conditions under which QUIC advantage is theoretically most pronounced.

Based on the data obtained, the following practical recommendations can be formulated. Switching to HTTP/3 is primarily advisable for services with a large number of small resources per page, high-latency audiences (mobile users, remote regions), and requirements for minimizing TTFB. For services that predominantly transfer large files over stable wired connections, the practical gain from switching will be minimal. BBR is recommended for use alongside HTTP/3: although it achieves a comparable average throughput on long-lived transfers, it offers a stable rate profile and shortens the time to reach full operating speed — a critical advantage precisely in the web scenario of numerous short requests.

To improve the reliability of performance assessment in subsequent studies, a comparison of alternative QUIC server implementations (in addition to nginx) is justified, as well as an analysis of the impact of proxy servers on performance metrics when using QUIC [10].

Conclusion. The study has confirmed that implementing the QUIC protocol provides practical gain in terms of web interaction speed. This resulted in a 23.06% reduction in TTFB and a 9.5% reduction in test web page load time. The model presented in the article predicts a 71.43% reduction in TTFB, not 23.06%. This discrepancy reveals the significant impact of the protocol's experimental status on its availability on network infrastructure, as well as the critical role of other factors discussed in the article for the protocol full-scale implementation.

The paper compares the CUBIC (standard) and BBR (most common in HTTP/3) congestion control algorithms, and presents a theoretical justification for the advantage of BBR when transferring small files, due to its higher initial speed and the absence of a slow-start mechanism. Similar advantages of BBR were documented in foreign studies. In particular, article [15] noted the efficiency of this algorithm in channels with high RTT (intercontinental connections) due to the modeling of the communication channel throughput.

The QUIC protocol in combination with HTTP/3 at the current stage of development is associated with certain implementation difficulties, but it has significant potential for building high-performance information systems, which is confirmed by the results of other studies [24], including in relation to systems with requirements for reliable delivery of real-time traffic [25].

Setting up an HTTP/3 server requires loading additional modules into the operating system kernel. This negatively impacts the adoption of the protocol, but if an organization has sufficient qualified personnel to support such an infrastructure, switching to HTTP/3 can provide significant gains. The improved TTFB and the elimination of Head-of-Line blocking can theoretically significantly speed up page resource loading. The performance gains can be particularly noticeable on pages containing numerous small resources.

In addition to performance issues, the transition to QUIC and HTTP/3 poses specific challenges for the information security infrastructure: built-in encryption complicates deep packet inspection (DPI) and the functioning of intrusion detection systems, which is discussed in detail in [26] and requires separate consideration when planning production implementation.

Limitations of the study include a simplified testbed configuration (one virtual machine and a 4G mobile channel) and a small sample size. The results obtained have practical value for assessing the feasibility of switching to HTTP/3, but require confirmation across a wider range of scenarios, as the performance of transport protocols is significantly dependent on network topology, latency, and loss. The subject of congestion control algorithms can be addressed in more depth in subsequent publications, in particular, it seems appropriate to compare various algorithms in combination with different versions of the HTTP protocol. In modern information systems, proxy servers are widely used for traffic management. A promising direction is to study the impact of such intermediate nodes on performance metrics when using QUIC [10].

References

1. Gorokhova PA. Russian Advertising Market: State, Structure, Trends and Development Prospects. *State and Municipal Management. Scholar Notes.* 2020;(1):297–302. https://doi.org/10.22394/2079-1690-2020-1-1-297-302

2. Arapakis I, Park S, Pielot M. Impact of Response Latency on User Behaviour in Mobile Web Search. In: *Proc. ACM SIGIR Conference on Human Information Interaction and Retrieval (CHIIR '21).* Canberra, Australia: ACM; 2021. P. 279–283. https://doi.org/10.1145/3406522.3446038

3. Rahmani J, Baranov MD, Kuzmin DA. Development of a Web Application Optimization Method to Improve Performance. *International Journal of Humanities and Natural Sciences.* 2025;102(3–1):237–241. https://doi.org/10.24412/2500-1000-2025-3-1-237-241

4. Wendroth J, Jaeger B. A Brief Overview on HTTP. In: *Proc. Seminar on Innovative Internet Technologies and Mobile Communications (IITM).* Munich: Technical University of Munich; 2022. P. 59–69. https://doi.org/10.2313/NET-2022-11-1_11

5. Rahmani J, Rogov ID. Trends in the Development of Network Technologies in 2022. In: *Proc. XVI International Industry Scientific and Technical Conference*. Moscow: Izdatel'skii dom Media pablisher; 2022. P. 30–31. (In Russ).

6. Bishop M (ed). *HTTP/3. RFC 9114.* AMS LLC: RFC Editor; 2022. https://doi.org/10.17487/RFC9114

7. Amet M, Thomas L, Ye-Qiong Song. A Performance Evaluation of QUIC in Real-Time Networks. In: *Proc. 32nd International Conference on Real-Time Networks and Systems (RTNS 2024).* New York, NY: ACM; 2024. P. 255–265. https://doi.org/10.1145/3696355.3699698

8. Langley A, Riddoch A, Wilk A, Vicente A, Krasic C, Zhang D, et al. The QUIC Transport Protocol: Design and Internet-Scale Deployment. In: *Proc. ACM SIGCOMM Conference.* New York, NY: ACM; 2017. P. 183–196. https://doi.org/10.1145/3098822.3098842

9. Fan Liu, Crowley P. Security and Performance Characteristics of QUIC and HTTP/3. In: *Proc. 10th ACM Conference on Information-Centric*. New York, NY: ACM; 2023. p. 124–126. https://doi.org/10.1145/3623565.3623757

10. Fan Liu, Farkiani B, Dehart J, Parwatikar J, Crowley P. Performance Comparison of HTTP/3 and HTTP/2 with Proxy Integration. *arXiv preprint*. 2024. https://doi.org/10.48550/arXiv.2409.16267

11. Andreyanov NS, Ryabikov AY. Research on Client-Server Protocols to Improve Fault Tolerance and Connection Security. *International Journal of Humanities and Natural Sciences.* 2024;97(10–1):128–132. https://doi.org/10.24412/2500-1000-2024-10-1-128-132

12. Smirnov IA, Mikhailov MYu, Lapteva MG. HTTP/3 and the QUIC Protocol: Future of High-Performance Web Services. *Vestnik Nauki.* 2025;2(6):1780–1788.

13. Jia Zhang, Haixuan Tong, Enhuan Dong, Xin Qian, Mingwei Xu, Xiaotian Li, et al. Cold Start or Hot Start? Robust Slow Start in Congestion Control with A Priori Knowledge for Mobile Web Services. In: *Proc. ACM Web Conference*. New York, NY: ACM; 2024. P. 2870–2878. https://doi.org/10.1145/3589334.3645393

14. Cardwell N, Yuchung Cheng, Gunn CS, Hassas Yeganeh S, Jacobson V. BBR: Congestion-Based Congestion Control. *Communications of the ACM*. 2017;60(2):58–66. https://doi.org/10.1145/3009824

15. Yi Han, Mengjie Zuo, Huijun Yuan, Yi Zhong, Zhenhui Yuan, Ting Bi. A QoS-Based Fairness-Aware BBR Congestion Control Algorithm Using QUIC. *Mathematical Problems in Engineering.* 2022;2022:7222030. https://doi.org/10.1155/2022/7222030

16. Dierks T, Rescorla E. The Transport Layer Security (TLS) Protocol Version 1.2. *RFC 5246.* New York: IETF; 2008. 101 p. https://doi.org/10.17487/RFC5246

17. Voronova AG. Typification of Projects for the Transition to Cloud Services. *Advanced Engineering Research (Rostov-on-Don)*. 2024;24(3):274–282. https://doi.org/10.23947/2687-1653-2024-24-3-274-282

18. Iyengar J, Thomson M (eds). *QUIC: A UDP-Based Multiplexed and Secure Transport. RFC 9000*. AMS LLC: RFC Editor; 2021. https://doi.org/10.17487/RFC9000

19. Thomson M, Turner S (eds). *Using TLS to Secure QUIC. RFC 9001*. AMS LLC: RFC Editor; 2021. https://doi.org/10.17487/RFC9001

20. Rescorla E. *The Transport Layer Security (TLS) Protocol Version 1.3. RFC 8446.* New York, NY: IETF; 2018. https://doi.org/10.17487/RFC8446

21. Rhee I, Xu L, Ha S, Zimmermann A, Eggert L, Scheffenegger R. *CUBIC for Fast and Long-Distance Networks. RFC 8312.* AMS LLC: RFC Editor; 2018. https://doi.org/10.17487/RFC8312

22. Linets G.I., Voronkin R.A., Slyusarev G.V., Govorova S.V. Optimization Problem for Probabilistic Time Intervals of Quasi-Deterministic Output and Self-Similar Input Data Packet Flow in Telecommunication Networks. Advanced Engineering Research (Rostov-on-Don). 2024;24(4):424-432. https://doi.org/10.23947/2687-1653-2024-24-4-424-432

23. Samoylenko V.V. Concept of a Multilevel Network Infrastructure for Monitoring Agricultural Facilities Based on Wireless Sensor Networks. *Advanced Engineering Research (Rostov-on-Don)*. 2025;25(4):371–382. https://doi.org/10.23947/2687-1653-2025-25-4-2238

24. Gupta A, Bartos R. Improving Web Content Delivery with HTTP/3 and Non-Incremental EPS. In: *Proc. Conference on Networking and Internet Architecture*. Durham, NH: University of New Hampshire; 2024. P. 1–10. https://doi.org/10.48550/arXiv.2404.13460

25. Antonova VM, Buzhin IG, Grechishkina NA, Korochkin ED, Kuznetsov NA. QUIC Transport Protocol as Reliable Transmission Method for Real-Time Traffic. *Information Processes*. 2025;25(3):490–500. https://doi.org/10.53921/18195822_2025_25_3_490

26. Selivanov MA. New Generation Network Protocols and Their Analysis from the Point of View of Information Security. *Young Researcher of the Don.* 2022;(1(34):56–62.

About the Authors:

Jahed Rahmani, Senior Lecturer of the Department of Network Information Technologies and Services, Moscow Technical University of Communications and Informatics (8a, Aviamotornaya Str., Moscow, 111024, Russian Federation), SPIN-code, ORCID, j.rahmani@mtuci.ru

Serafim P. Sukharev, Technician of the Research and Innovation Department "Center for Artificial Intelligence and Advanced Projects", Moscow Technical University of Communications and Informatics (8a, Aviamotornaya Str., Moscow, 111024, Russian Federation), SPIN-code, ORCID, ResearchGate, ResearcherID, s.p.suharev@mtuci.ru

Claimed Contributorship:

J Rahmani: conceptualization, methodology, supervision, writing & editing.

SP Sukharev: data curation, formal analysis, investigation, software, visualization, writing – original draft preparation.

Conflict of Interest Statement: **the authors declare no conflict of interest.**

All authors have read and approved the final version of manuscript.

Об авторах:

Джахед Рахмани, старший преподаватель кафедры «Сетевые информационные технологии и сервисы» Московского технического университета связи и информатики (111024, Российская Федерация, г. Москва, ул. Авиамоторная, 8 а), SPIN-код, ORCID, j.rahmani@mtuci.ru

Серафим Павлович Сухарев, техник НИО «Центр искусственного интеллекта и перспективных проектов» Московского технического университета связи и информатики (111024, Российская Федерация, г. Москва, ул. Авиамоторная, 8 а), SPIN-код, ORCID, ResearchGate, ResearcherID, s.p.suharev@mtuci.ru

Заявленный вклад авторов:

Д. Рахмани: разработка концепции и методологии, научное руководство, написание рукописи, редактирование**.**

С.П. Сухарев: курирование данных, формальный анализ, проведение исследования, разработка программного обеспечения, визуализация, написание черновика рукописи.

Конфликт интересов: **авторы заявляют об отсутствии конфликта интересов.**

Все авторы прочитали и одобрили окончательный вариант рукописи.

Received / Поступила в редакцию 02.03.2026
Reviewed / Поступила после рецензирования 02.04.2026
Accepted / Принята к публикации 09.04.2026