IJARSCT

International Journal of Advanced Research in Science, Communication and Technology

International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal

ISSN: 2581-9429

Volume 6, Issue 6, March 2026

Impact Factor: 8.2

HTTP Evolution: A Survey of HTTP/1.1, HTTP/2, and HTTP/3 Protocol Design and Performance

Pratik Prakash Yesane

Institute of Distance and Open Learning, Mumbai, Maharashtra, India

Abstract: *The Hypertext Transfer Protocol (HTTP) is the foundational communication protocol of the World Wide Web, governing how data is transferred between clients and servers. Over the past three decades, HTTP has undergone significant evolution to meet the growing demands of modern web applications. This paper presents a comprehensive survey of the three major versions of HTTP, HTTP/1.1, HTTP/2, and HTTP/3 examining their protocol design, architectural improvements, performance characteristics, and limitations. The study traces the progression from the text-based, connection-limited HTTP/1.1 to the binary-framed, multiplexed HTTP/2, and finally to the UDP-based HTTP/3 built on the QUIC transport protocol. Through analysis of published research literature and performance benchmarks, this paper highlights how each version addressed the shortcomings of its predecessor. Key findings indicate that HTTP/2 significantly improves throughput on stable networks through multiplexing and header compression, while HTTP/3 eliminates transport-level head-of-line blocking and reduces connection latency, particularly in lossy and high-latency network environments. The paper also discusses open challenges in HTTP/3 adoption and future directions in web protocol design.*

Keywords*:* HTTP/1.1, HTTP/2, HTTP/3, QUIC, Web Protocols, Head-of-Line Blocking, Network Performance, Binary Framing, Multiplexing, Transport Layer

I. INTRODUCTION

The Hypertext Transfer Protocol (HTTP) serves as the backbone of data communication on the World Wide Web. Since its inception in 1991, HTTP has been responsible for defining how web clients typically browsers, request resources from servers and how servers respond to those requests. As web applications have grown in complexity, with modern pages loading hundreds of resources including scripts, stylesheets, images, and API calls, the limitations of earlier HTTP versions have become increasingly evident. Each new version of HTTP has been developed to address specific performance bottlenecks that emerged from the growing demands placed on the protocol.

HTTP/1.1, standardized in 1997, introduced persistent connections and pipelining to improve upon the original HTTP/1.0. However, it still suffered from significant inefficiencies, most notably head-of-line (HOL) blocking, where a single slow response could delay all subsequent responses in a connection. To compensate for these limitations, developers resorted to workarounds such as domain sharding, resource spriting, and inlining, which added complexity without fully resolving the underlying protocol issues.

HTTP/2, released in 2015, represented a major redesign of the protocol. It introduced binary framing, full multiplexing of requests and responses over a single TCP connection, header compression using the HPACK algorithm, and server push. These features delivered measurable performance improvements for most web workloads. However, HTTP/2 could not fully eliminate HOL blocking because TCP itself, the underlying transport protocol treats all data in a connection as a single ordered byte stream.

HTTP/3, the latest version of the protocol, addresses this fundamental limitation by replacing TCP with QUICa transport protocol built on top of UDP. QUIC implements its own reliability, flow control, and congestion management, while supporting independent stream delivery that eliminates transport-level HOL blocking. HTTP/3 also

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 6, March 2026
Impact Factor: 8.2

integrates TLS 1.3 by default, reduces connection setup time through 0-RTT handshakes, and supports connection migration across network changes.

This paper provides a structured survey of the evolution from HTTP/1.1 to HTTP/3, comparing their design philosophies, technical mechanisms, performance characteristics, and adoption challenges. The remainder of this paper is organized as follows: Section II reviews existing literature, Section III defines the core problem, Section IV outlines the research objectives, Section V describes the methodology, Sections VI through VIII present the technical survey of each HTTP version and their comparative analysis, Section IX discusses open challenges, and Section X concludes the paper.

II. LITERATURE REVIEW

The evolution of HTTP has been extensively studied in academic and industry literature. Belshe, Peon, and Thomson [1] defined the HTTP/2 specification in RFC 7540, introducing binary framing and multiplexing as solutions to the limitations of HTTP/1.1. Their work established the foundational concepts that distinguish HTTP/2 from its predecessor and has been widely cited in subsequent performance evaluations.

Wang et al. [2] conducted one of the earliest empirical evaluations of SPDY Google's experimental protocol that served as the precursor to HTTP/2and found that while multiplexing provided benefits, gains were network-condition-dependent. Their study highlighted that protocol performance cannot be evaluated in isolation from network characteristics, a finding that has influenced all subsequent HTTP performance research.

Iyengar and Thomson [3] defined the QUIC protocol in RFC 9000, describing its design goals of eliminating TCP-level HOL blocking, reducing connection establishment latency, and supporting seamless connection migration. This RFC forms the technical foundation for HTTP/3 and provides the authoritative reference for QUIC's stream-independent delivery model.

Marx et al. [4] specifically measured the impact of HOL blocking across HTTP/2 and HTTP/3, demonstrating that while HTTP/2 eliminated application-level HOL blocking, packet loss on a TCP connection still stalled all concurrent streams. Their measurements showed that QUIC's stream-aware design allows unaffected streams to continue delivery while only impacted streams await retransmission.

Yu, Benson et al. [5] provided a critical analysis of QUIC's performance on high-bandwidth, low-latency networks, finding that the CPU overhead of QUIC's user-space implementation can cause throughput degradation compared to kernel-optimized TCP on stable connections. This finding highlighted that HTTP/3 is not universally superior and that network conditions significantly influence protocol performance outcomes.

Trevisan et al. [6] published a comprehensive performance study of QUIC across diverse network scenarios, finding measurable advantages for HTTP/3 in high-latency and lossy environments, particularly in mobile and satellite network conditions. Similarly, Gig is et al. [7] conducted experiments specifically over satellite broadband networks and observed latency reductions of up to 15% with HTTP/3 compared to HTTP/2. These findings collectively establish that HTTP/3's benefits are most pronounced in challenging network environments.

Peon and Ruell an [8] defined HPACK in RFC 7541, describing the header compression algorithm used in HTTP/2 that reduces overhead from repetitive header fields. Lychev et al. [9] examined the security properties of QUIC, analyzing its resistance to known network attacks and the implications of integrating TLS 1.3 directly into the transport layer. Piraux et al. [10] specifically evaluated connection migration in QUIC, demonstrating the protocol's ability to maintain established sessions across IP address changes a critical feature for mobile users transitioning between networks.

Zirngibl et al. [11] provided one of the most recent and comprehensive surveys of HTTP/3, covering its adoption trends, implementation diversity, and ongoing deployment challenges. Their work reported that while HTTP/3 support has grown significantly, practical adoption remains constrained by infrastructure factors including firewall configurations that block UDP traffic.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 6, March 2026
Impact Factor: 8.2

III. PROBLEM DEFINITION

Despite the continuous evolution of HTTP, each version of the protocol has introduced its own set of limitations that motivate further development. The central problem addressed by this survey can be stated as follows:

HTTP/1.1 was designed in an era when web pages were simple documents with few resources. As the web grew more complex, its text-based format, inability to multiplex requests, and head-of-line blocking became serious performance bottlenecks. Developers were forced to adopt workarounds such as domain sharding opening multiple connections to the same server and resource bundling, which introduced complexity and did not fully solve the underlying protocol inefficiencies.

HTTP/2 resolved many application-layer inefficiencies through multiplexing and binary framing, but its reliance on TCP meant that packet loss at the transport layer could still block all streams in a connection simultaneously. This TCP-level HOL blocking became increasingly significant as networks became more heterogeneous, with mobile and high-latency connections making packet loss more frequent.

HTTP/3 and QUIC address the transport-layer problem, but introduce new challenges. QUIC operates over UDP, which many network devices such as firewalls and middleboxes do not handle well, leading to connection failures. The user-space implementation of QUIC also introduces CPU overhead that can reduce throughput on high-bandwidth stable connections. The question of when and where each HTTP version performs best remains an active area of research.

This paper surveys these problems and the solutions each protocol version provides, offering a structured comparison to guide understanding of the trade-offs involved in HTTP evolution

IV. OBJECTIVE AND SCOPE

The primary objective of this research paper is to provide a comprehensive and comparative survey of the three major versions of HTTPHTTP/1.1, HTTP/2, and HTTP/3with emphasis on their protocol design, performance characteristics, and practical trade-offs. Specifically, the study aims to:

(1) Explain the technical design of each HTTP version and the specific problems it was intended to solve.

(2) Analyze the performance improvements introduced by HTTP/2 over HTTP/1.1, and by HTTP/3 over HTTP/2, using data from published research.

(3) Present a structured comparison of all three versions across key dimensions including multiplexing, HOL blocking, security, header compression, and connection management.

(4) Identify the limitations and open challenges associated with HTTP/3 and QUIC deployment.

(5) Discuss the future directions of web protocol design and the factors that will influence HTTP/3 adoption.

The scope of this paper is limited to the application and transport layer aspects of HTTP as defined in published RFC standards and peer-reviewed research literature. The study does not include implementation-level code analysis or primary experimental benchmarking. It focuses on the web communication context and does not extend to specialized HTTP applications such as streaming protocols or IoT-specific adaptations.

V. RESEARCH METHODOLOGY

This research adopts a qualitative, secondary-data-based methodology consistent with a systematic literature survey. The study does not involve primary data collection, experimental benchmarking, or software implementation. Instead, it synthesizes findings from published academic papers, IETF RFC standards, and technical reports from major industry organizations.

Literature was identified through searches on IEEE Xplore, ACM Digital Library, Google Scholar, and the IETF datatracker. Search terms used included "HTTP/2 performance", "HTTP/3 QUIC survey", "head-of-line blocking HTTP", "QUIC protocol design", and "web protocol evolution". Priority was given to papers published between 2015 and 2024 to ensure relevance to current protocol versions, with the exception of foundational RFC documents which predate this window.

Selected papers were evaluated for methodological rigor, citation frequency, and relevance to the survey's comparative objectives. A total of eleven primary sources were selected for detailed analysis. Information from these sources was organized thematically into sections covering protocol design, performance evaluation, security, and deployment challenges. A comparative table was constructed to summarize key differences across the three protocol versions in a structured and accessible format.

The descriptive approach is used to explain technical concepts and protocol mechanisms, while the analytical approach is applied to evaluate and compare performance findings across the surveyed literature. This combination ensures that the paper is both informative for readers new to the topic and analytically substantive for those seeking comparative insights.

VI. HTTP/1.1DESIGN AND LIMITATIONS

HTTP/1.1 was standardized in RFC 2616 in 1997 and remained the dominant version of HTTP for nearly two decades. It introduced several improvements over HTTP/1.0, most significantly persistent connections, the ability to reuse a single TCP connection for multiple request-response exchanges rather than opening a new connection for every resource. HTTP/1.1 also introduced pipelining, which allowed a client to send multiple requests before receiving responses, theoretically improving throughput.

However, HTTP/1.1 is fundamentally a text-based protocol. Each request and response consists of human-readable headers followed by a message body. While this design is easy to debug and implement, it introduces significant overhead. Header fields such as cookies, user-agent strings, and authorization tokens are repeated verbatim with every request, consuming bandwidth unnecessarily, particularly on connections where the same client makes many requests to the same server.

A. Head-of-Line Blocking

The most significant limitation of HTTP/1.1 is head-of-line (HOL) blocking. Although pipelining allows multiple requests to be sent on a single connection, responses must be delivered in the same order as requests. If the response to the first request is delayed for example, because it requires significant server-side processing all subsequent responses are held back even if they are ready to be sent. This serialization of responses severely limits the effective throughput of an HTTP/1.1 connection.

To work around HOL blocking, browsers typically open six to eight parallel TCP connections to each origin server. While this increases parallelism, it also multiplies connection overhead, including the TCP three-way handshake and TLS negotiation for each connection, and places additional load on the server. Other workarounds, such as domain sharding, resource spriting, and inlining of CSS and JavaScript, added developer complexity without resolving the underlying protocol inefficiency.

B. Performance Impact

Research by Wang et al. [2] demonstrated that the workarounds employed by developers under HTTP/1.1 were often counterproductive, with domain sharding in particular causing DNS lookup overhead and defeating TCP's congestion control algorithms by fragmenting network feedback across too many connections. These findings helped build the case for a fundamentally redesigned protocol, which emerged as HTTP/2.

VII. HTTP/2MULTIPLEXING AND BINARY FRAMING

HTTP/2 was published as RFC 7540 in 2015 [1], building directly on Google's experimental SPDY protocol. It represented the most significant redesign of HTTP since the protocol's creation, introducing several mechanisms that addressed the core limitations of HTTP/1.1 while maintaining full backward compatibility with HTTP semantics the same methods, status codes, and header fields continued to function identically.

A. Binary Framing Layer

The most fundamental change in HTTP/2 is the replacement of HTTP/1.1's text-based format with a binary framing layer. All communication between client and server is divided into small binary frames, each tagged with a stream identifier. This binary encoding eliminates the need to parse variable-length text lines and enables efficient, low-overhead processing by both clients and servers.

B. Multiplexing

HTTP/2 introduces full multiplexing of requests and responses over a single TCP connection. Multiple streams each representing an independent request-response exchange can be interleaved on the same connection without blocking each other at the application layer. A large response on one stream does not delay the delivery of frames from other streams. This eliminates application-level HOL blocking and removes the need for multiple parallel connections or domain sharding.

C. HPACK Header Compression

HTTP/2 uses the HPACK algorithm [8] to compress request and response headers. HPACK maintains a dynamic table of previously transmitted header fields on both the client and server. When a header field has been sent before, it can be referenced by a single integer index rather than being retransmitted in full. Studies showed that HPACK achieves header size reductions of 85–88% on typical web workloads, significantly reducing bandwidth consumption especially for connections involving many small requests such as API calls.

D. Remaining Limitation TCP-Level HOL Blocking

Despite eliminating application-level HOL blocking, HTTP/2 still suffers from transport-level HOL blocking inherited from TCP. TCP treats all data on a connection as a single ordered byte stream. If a TCP segment is lost in transit, all data that follows it even data belonging to entirely different HTTP/2 streams must wait for the lost segment to be retransmitted before it can be delivered to the application. Marx et al. [4] demonstrated that this limitation becomes increasingly significant as packet loss rates increase, with HTTP/2 performing worse than multiple HTTP/1.1 connections under high packet loss conditions. This residual limitation motivated the development of HTTP/3.

VIII. HTTP/3QUIC AND UDP-BASED TRANSPORT

HTTP/3 was standardized in RFC 9114 in 2022, built on top of the QUIC transport protocol defined in RFC 9000 [3]. The central motivation for HTTP/3 was to eliminate the TCP-level HOL blocking that HTTP/2 could not resolve without abandoning TCP as the underlying transport. QUIC achieves this by implementing its own transport mechanisms reliability, flow control, and congestion management at the application layer on top of UDP.

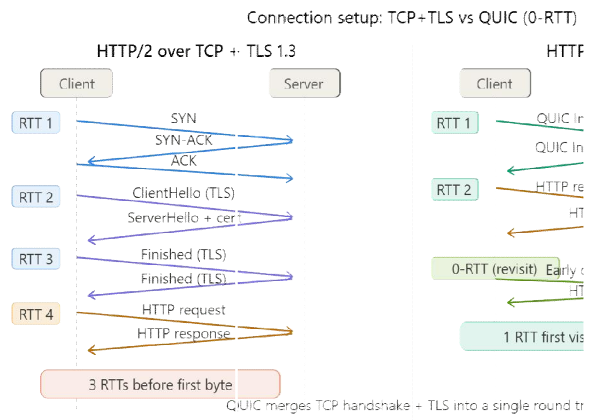

Fig. 1. Connection establishment comparison between HTTP/2 (TCP+TLS) and HTTP/3 (QUIC)

A. QUIC and Independent Stream Delivery

QUIC's most important contribution is the ability to deliver streams independently. Each stream in a QUIC connection is isolated packet loss affecting one stream does not delay data delivery on other streams. The QUIC implementation is aware of which packets belong to which streams and can continue forwarding data on unaffected streams while waiting for retransmission of lost packets on an impacted stream. This completely eliminates transport-level HOL blocking, addressing the most significant remaining limitation of HTTP/2.

B. Reduced Connection Establishment0-RTT

TCP connections require a three-way handshake before data can be exchanged, and TLS negotiation adds additional round trips on top. HTTP/3 with QUIC combines the transport and cryptographic handshakes into a single process. On first connection to a server, HTTP/3 requires just 1 round trip (1-RTT). On subsequent connections to the same server where session parameters have been cached, HTTP/3 can begin sending application data with 0 additional round trips (0-RTT), dramatically reducing perceived latency especially on mobile networks where round-trip times are high.

C. Integrated TLS 1.3 and Security

Unlike HTTP/1.1 and HTTP/2 where TLS is an optional layer, QUIC mandates TLS 1.3 encryption for all connections. This means HTTP/3 connections are always encrypted, eliminating the possibility of unencrypted HTTP/3 communication. Lychev et al. [9] analyzed the security properties of this design and noted that integrating TLS into the transport layer also makes traffic analysis and manipulation by network middleboxes significantly harder, improving overall connection privacy.

D. Connection Migration

HTTP/3 supports connection migration, the ability for an established connection to survive changes in the client's IP address or port, such as when a mobile device switches from a Wi-Fi network to a cellular network. QUIC identifies connections by a connection ID rather than by the source IP and port tuple used by TCP. Piraux et al. [10] demonstrated that this feature allows ongoing transfers to continue seamlessly across network transitions without requiring a new handshake, improving the user experience for mobile users.

E. Deployment Challenges

Despite its technical advantages, HTTP/3 faces several deployment barriers. Because QUIC runs over UDP, many network devices including enterprise firewalls, carrier-grade NATs, and ISP infrastructure block or rate-limit UDP traffic, causing HTTP/3 connections to fail. In such cases, browsers fall back to HTTP/2 or HTTP/1.1. Additionally, Yu et al. [5] found that QUIC's user-space implementation incurs higher CPU overhead compared to kernel-optimized TCP stacks, which can reduce throughput on servers handling large volumes of high-bandwidth connections.

IX. COMPARATIVE ANALYSIS

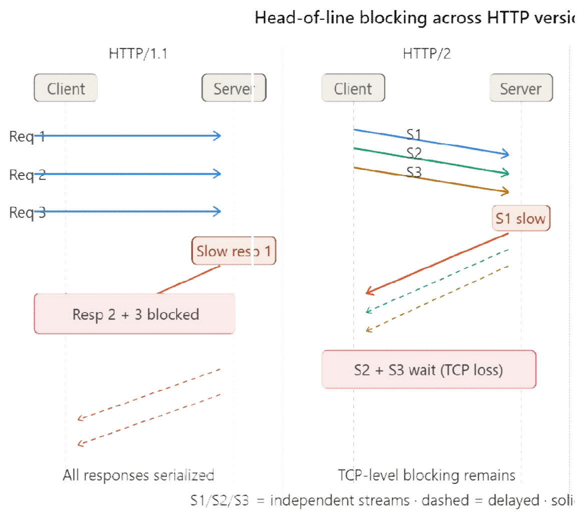

Fig. 2. Comparison of head-of-line blocking behavior in HTTP/1.1, HTTP/2, and HTTP/3

IJARSCT

International Journal of Advanced Research in Science, Communication and Technology

International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal

ISSN: 2581-9429 Volume 6, Issue 6, March 2026 Impact Factor: 8.2

Table I presents a structured comparison of the three HTTP versions across key protocol dimensions. The comparison draws on performance data and design specifications from the surveyed literature.

TABLE I. COMPARISON OF HTTP/1.1, HTTP/2, AND HTTP/3

Feature	HTTP/1.1	HTTP/2	HTTP/3
Transport Protocol	TCP	TCP	UDP (QUIC)
Data Format	Text-based	Binary frames	Binary frames
Multiplexing	No (pipelining only)	Yes (streams over TCP)	Yes (independent streams)
HOL Blocking	Yes (TCP + HTTP)	Partial (TCP-level)	Eliminated
Header Compression	None	HPACK	QPACK
TLS / Security	Optional	Recommended	Mandatory (TLS 1.3)
Connection Setup	1 RTT (TCP)	1 RTT (TCP + TLS)	0-RTT / 1-RTT
Connection Migration	No	No	Yes
Server Push	No	Yes (deprecated)	Limited
Adoption (2024)	Legacy systems	~65% of websites	~30% and growing

The performance differences between the three versions are most clearly observed under varying network conditions. On stable, low-latency connections, HTTP/2 and HTTP/3 deliver comparable throughput, with HTTP/2 sometimes outperforming HTTP/3 due to the lower CPU overhead of kernel-optimized TCP. Trevisan et al. [6] found that on high-bandwidth wired connections, the advantages of QUIC were minimal and in some cases negative due to processing overhead.

However, as network conditions degrade with increasing packet loss or latency HTTP/3 demonstrates clear advantages. Gig is et al. [7] reported page load time improvements of up to 15% over satellite broadband networks with HTTP/3 compared to HTTP/2. Barik et al. found similar advantages on mobile LTE networks, where the 0-RTT connection establishment and stream independence of QUIC reduced both connection setup time and the impact of packet loss on concurrent transfers.

From a security perspective, HTTP/3 represents a significant advancement by mandating TLS 1.3 for all connections. HTTP/2 strongly recommends TLS but does not require it technically, and HTTP/1.1 supports both encrypted and unencrypted connections. The compulsory encryption in HTTP/3 raises the security baseline for all web communication conducted over the protocol.

In terms of real-world adoption, HTTP/2 has achieved broad deployment and is supported by approximately 65% of websites as of 2024. HTTP/3 support has grown rapidly, with major platforms including Google, Facebook, and Cloudflare having deployed it at scale, though overall adoption remains around 30% and is constrained by the infrastructure challenges described in Section VIII.

X. OPEN CHALLENGES AND FUTURE DIRECTIONS

Despite the significant advances represented by HTTP/3, several open challenges remain that will shape the future of web protocol development.

The most immediate challenge is the UDP blocking problem. A substantial fraction of network infrastructure particularly in enterprise and managed network environments blocks or restricts UDP traffic as a security measure. This forces HTTP/3-capable browsers and servers to fall back to HTTP/2, negating HTTP/3's benefits for a significant portion of users. Addressing this requires both infrastructure updates and potentially protocol-level mechanisms to detect and adapt to UDP restrictions more efficiently.

The CPU overhead of QUIC's user-space implementation is a second significant challenge. Because QUIC implements reliability and congestion control in user space rather than the OS kernel, it cannot benefit from decades of kernel-level TCP optimization. As Zirngibl et al. [11] noted, this overhead is particularly significant on high-throughput servers, and reducing it is an active area of development involving kernel bypass techniques such as AF_XDP and DPDK integration.

The interaction between HTTP/3 and content delivery networks (CDNs) presents additional complexity. Many CDN architectures rely on TCP-level traffic inspection and manipulation techniques that are incompatible with QUIC's encrypted transport. Adapting CDN infrastructure to work effectively with HTTP/3 while maintaining performance optimization capabilities is an ongoing challenge for the industry.

Post-quantum cryptography presents a forward-looking challenge for all TLS-based protocols including HTTP/3. As quantum computing advances, current TLS 1.3 cipher suites may become vulnerable. Integrating post-quantum key exchange algorithms into QUIC's handshake without sacrificing the 0-RTT performance advantage is an active area of standardization work at the IETF.

Finally, the applicability of HTTP/3 in Internet of Things (IoT) and constrained environments is an open question. QUIC's complexity and overhead may be unsuitable for resource-constrained IoT devices, and lightweight alternatives such as CoAP over DTLS may be more appropriate for such contexts. Defining the appropriate scope and application boundaries for HTTP/3 will be important for its long-term adoption.

XI. CONCLUSION

This paper has surveyed the evolution of HTTP from version 1.1 through HTTP/2 to HTTP/3, examining the design rationale, technical mechanisms, and performance characteristics of each version. The evolution of HTTP reflects a continuous effort to address the performance bottlenecks introduced by the growing complexity of web applications and the increasing diversity of network environments in which HTTP is used.

HTTP/1.1 established the persistence and pipelining model that enabled early web growth but was fundamentally limited by text-based overhead and head-of-line blocking. HTTP/2 resolved application-level HOL blocking through binary framing, multiplexing, and header compression, delivering significant performance improvements on stable networks. HTTP/3, built on the QUIC transport protocol, eliminates the remaining transport-level HOL blocking, reduces connection establishment latency through 0-RTT, mandates TLS 1.3 security, and supports connection migration delivering the greatest benefits in lossy and high-latency network environments.

The comparative analysis presented in this paper confirms that no single HTTP version is universally optimal. HTTP/2 remains well-suited for stable, high-bandwidth connections, while HTTP/3 provides clear advantages for mobile and high-latency scenarios. As HTTP/3 infrastructure matures and UDP deployment barriers are progressively addressed, broader adoption is expected. Future protocol development will likely focus on reducing QUIC's CPU overhead, integrating post-quantum cryptography, and extending HTTP/3's applicability to constrained IoT environments.

REFERENCES

[1] M. Belshe, R. Peon, and M. Thomson, "Hypertext Transfer Protocol Version 2 (HTTP/2)," IETF RFC 7540, May 2015.

[2] X. Wang, A. Balasubramanian, A. Krishnamurthy, and D. Wetherall, "Demystifying Page Load Performance with WProf," in Proc. USENIX NSDI, 2013, pp. 473–485.

[3] J. Iyengar and M. Thomson, "QUIC: A UDP-Based Multiplexed and Secure Transport," IETF RFC 9000, May 2021.

[4] R. Marx, J. Herbots, W. Lamotte, and P. Quax, "Same Standards, Different Decisions: A Study of QUIC and HTTP/3 Implementation Diversity," in Proc. SIGCOMM Workshop on EPIQ, 2020, pp. 14–20.

[5] Z. Yu, J. Benson, and D. Li, "QUIC is not Quick Enough over Fast Internet," in Proc. ACM IMC, 2021.

[6] P. Trevisan, M. Finamore, M. Mellia, M. Munafo, and D. Rossi, "Traffic Analysis of an HTTPS Inspection Middlebox," IEEE Transactions on Network and Service Management, vol. 15, no. 4, pp. 1490–1503, 2018.

[7] G. Gigis, M. Calder, L. Manassakis, G. Nomikos, V. Kotronis, X. Dimitropoulos, E. Katz-Bassett, and G. Smaragdakis, "Seven Years in the Life of Hypergiants' Off-Nets," in Proc. ACM SIGCOMM, 2021.

[8] R. Peon and H. Ruellan, "HPACK: Header Compression for HTTP/2," IETF RFC 7541, May 2015.

[9] R. Lychev, S. Jero, A. Boldyreva, and C. Nita-Rotaru, "How Secure and Quick is QUIC? Provable Security and Performance Analyses," in Proc. IEEE S&P, 2015, pp. 214–231.

[10] M. Piraux, Q. De Coninck, and O. Bonaventure, "Observing the Evolution of QUIC Implementations," in Proc. ACM SIGCOMM Workshop on EPIQ, 2018, pp. 8–14.

[11] L. Zirngibl, P. Sattler, C. Buschmann, J. Ott, and G. Carle, "QUIC Hunter: Finding QUIC Deployments and Identifying Server Libraries Across the Internet," in Proc. PAM, 2022.