BAB 1

Hypertext Transfer Protocol (HTTP) has been a foundational protocol for web technology. As web technologies have evolved, so too has HTTP, undergoing several revisions to meet increasingly diverse demands. HTTP/1.1, standardized in 1997, introduced persistent connections, allowing a connection between a client and server to remain open after the server responded to a client request, which significantly improved efficiency over its predecessor, HTTP/1.0 [1].

However, as the web grew and modern web technologies demanded more, HTTP/1.1's limitations became apparent. To address these limitations, HTTP/2 was standardized in 2015, introducing new features such as multiplexing and header compression. These enhancements aimed to improve performance by allowing multiple requests and responses to be sent simultaneously over a single connection, reducing the overhead of multiple connections [2]. Yet, even with these improvements, the underlying Transmission Control Protocol (TCP) continued to pose challenges, particularly in environments with poor network conditions, where issues like packet loss and head-of-line blocking could severely degrade performance.

Recognizing these limitations, HTTP/3, standardized in 2022, adopted the QUIC protocol (Quick UDP Internet Connections) to replace TCP. QUIC was designed to address the deficiencies of TCP by providing more reliable and faster connections, particularly in adverse network conditions [3]. This marked a significant shift, as QUIC operates over UDP (User Datagram Protocol), which allows for faster connection setups and more efficient data transfer, especially in scenarios with high latency or packet loss.

The strengths, weaknesses, and design considerations for using different HTTP versions in specific cases, or comparisons to previous versions, have been discussed in several pieces of literature. For example, Min & Lee (2019) in “An experimental view on fairness between HTTP/1.1 and HTTP/2” found that while HTTP/2 improves web performance compared to HTTP/1.1, bandwidth allocation is not always consistent between these two versions when implemented on the same server [4].

Other studies have also explored different aspects of HTTP performance. A study by Trevisan et al. (2021) titled "Measuring HTTP/3: Adoption and Performance" offers key insights into the adoption and performance benefits of HTTP/3 compared to its predecessors. The research highlights that while HTTP/3 provides noticeable performance improvements, these benefits are most significant under specific network conditions, particularly in scenarios with high latency or low bandwidth. The study also found that performance gains with HTTP/3 vary considerably depending on the infrastructure hosting the websites, with larger benefits observed for websites that limit connections to a smaller set of third-party domains and avoid legacy protocols [5].

Abhinav Gupta and Radim Bartos, in their study "User experience evaluation of the HTTP/3 in real-world deployment scenarios," explores the Quality of Experience (QoE) differences between HTTP/2 and HTTP/3 under various network conditions. The research shows that while HTTP/3 generally performs better in challenging network environments, such as those with high latency and packet loss, its performance does not consistently surpass HTTP/2 in all scenarios. Specifically, HTTP/3 demonstrates superior First Contentful Paint (FCP) times in scenarios with higher latency, but its throughput advantages are not as clear-cut [6]. Similarly, Naoki Oda and Saneyasu Yamaguchi, in "HTTP/2 performance evaluation with latency and packet losses," examined performance differences between HTTP/1.1 and HTTP/2 under varying conditions of latency and packet loss. The study finds that HTTP/2 generally offers comparable performance to HTTP/1.1 when multiple connections are

used. However, in scenarios with high latency and high packet loss, HTTP/2's performance significantly deteriorates due to its reliance on a single TCP connection [7].

The paper "Performance evaluation of the HTTP/3 client implementations" by Ján Balažia and Pavel Čičák explored the performance differences between HTTP/3 and HTTP/2 clients using cURL [8]. Furthermore, Robin Marx, Maarten Wijnants, Peter Quax, Axel Faes, and Wim Lamotte provided an in-depth comparison between HTTP/2 and HTTP/1.1, from specifications to performance, in their paper "Web performance characteristics of HTTP/2 and comparison to HTTP/1.1" [9].

Additionally, Xaver Zak, Juraj Machaj, and Lukas Sevcik, in "A Comparative Analysis of HTTP/2 and HTTP/3 Web Server Performance," compared the performance of Nginx and Caddy web servers using containerization when utilizing HTTP/2 and HTTP/3 [10]. Lastly, Nikola Kirilov and E. Bischoff, in their study "Networking Aspects of the Electronic Health Records: Hypertext Transfer Protocol Version 2 (HTTP/2) vs HTTP/3," discussed the performance differences between HTTP/2 and HTTP/3 in the context of Electronic Health Records (EHRs) [11].

Despite several comprehensive studies on the impact and differences of HTTP versions on frontend web application performance, there remains a significant research gap regarding their impact on backend applications or servers. This distinction is crucial as the differing behaviors of frontend and backend applications may render previous findings inapplicable to both contexts. The majority of existing literature focuses on how different HTTP versions enhance user experience by improving page load speed and efficiency. However, questions remain about how these changes in HTTP versions affect the technical aspects and performance of applications on the server or backend side.

While previous studies, such as “Performance Evaluation of the HTTP/3 Client Implementations” [8] and “A Comparative Analysis of HTTP/2 and HTTP/3 Web Server Performance” [10], have explored various aspects of HTTP protocols, they primarily investigate performance differences between HTTP/3 clients or compare HTTP/2 and HTTP/3 implementations in web servers under low-resource systems. These studies provide valuable insights into how different HTTP versions perform in general environments and under resource constraints. However, most of this research focuses on broader performance evaluations without a specific focus on backend applications. Although these studies examine web server performance and protocol efficiency, they often do not address the unique challenges faced by backend systems. Backend applications differ significantly from frontend applications in that they must handle large volumes of concurrent requests, maintain low error rates, and optimize resource usage across diverse network conditions. Existing studies generally overlook how different HTTPversions impact backend-specific metrics such as CPU and memory consumption, throughput under high concurrency, and the robustness of backend services in distributed environments.

This study aims to fill this gap by exploring the impact of using different HTTP versions—HTTP/1.1, HTTP/2, and HTTP/3—on backend application performance, specifically how these advantages translate to backend systems, where network conditions, resource consumption, and server-side performance—such as resource usage, throughput (requests per second), error rate, and response time under varying network conditions—play a more critical role in determining overall application efficiency.