# Network Performance Evaluation in Web Browsers: Comprehensive Methodologies, Key Metrics, and Visualization Techniques

Saurabh Kumar, Anshi Kumari, Rahul Kumar, Bibek Kumar Mahto, Piyush Nishad
*Department of Computer Science and Engineering*
*Noida International University, Greater Noida, India*
*Email:* subburajput1@gmail.com

***Abstract*—The performance of web browsers has a direct impact on user experience and application responsiveness in an increasingly network-dependent digital ecosystem. With the rapid evolution of web technologies, evaluating network performance within browsers has become critical to identifying bottlenecks, optimizing data transmission, and ensuring seamless web interactions. This paper presents a comprehensive study on the methodologies used to assess network performance in modern web browsers, emphasizing both client-side metrics and end-to-end communication aspects. By examining current frameworks and tools for network monitoring, the paper lays the foundation for systematic performance evaluation tailored to browser environments.**

**Key metrics such as DNS resolution time, TCP/TLS handshake latency, resource loading times, and page rendering delays are thoroughly explored. These parameters not only inform technical optimization strategies but also offer valuable insights into real-world network behavior under diverse conditions. The study further incorporates the role of user-centric indicators like Time to First Byte (TTFB) and Time to Interactive (TTI), which bridge the gap between raw performance data and perceived responsiveness. The paper classifies and contextualizes these metrics within varying evaluation scenarios, including controlled lab settings and live network environments.**

**Visualization is addressed as an essential dimension of performance analysis. The paper highlights the strengths and limitations of various visual representation techniques—such as waterfall charts, line graphs, and interactive dashboards—for presenting performance data effectively. Ultimately, this research advocates for a multidimensional evaluation framework that integrates robust methodologies, precise metrics, and insightful visualization tools to drive continuous improvements in browser-based network performance.**

***Keywords*—Web Browser Performance, Network Evaluation, Performance Metrics, Visualization Techniques, Page Load Analysis, Client-Side Monitoring**

## I. INTRODUCTION

The rapid proliferation of web applications and cloud-based services has heightened the need for optimal network performance in web browsers, which serve as the primary interface between users and digital platforms. Browsers are no longer passive renderers of static content but are increasingly sophisticated execution environments for dynamic, real-time applications. This transformation has brought network performance evaluation to the forefront of web development and optimization efforts [52], [2]. As modern websites load hundreds of resources and engage in multiple simultaneous connections, even minor delays in data transmission can significantly degrade user experience [51], [4].

Network performance plays a vital role in determining page load times, responsiveness, and the perceived interactivity of a web application. Studies have shown that higher latency and lower throughput are strongly correlated with increased user abandonment and reduced engagement [49], [44]. However, evaluating browser network performance remains a challenging task due to the complexity of underlying protocols, diversity of client environments, and dynamic behavior of content delivery mechanisms [28], [61]. Factors such as DNS lookup time, TCP handshake, TLS encryption overhead, resource prioritization, and caching strategies all influence performance but are difficult to measure consistently across contexts [58], [60].

This paper aims to present a comprehensive framework for evaluating network performance within modern web browsers. It consolidates current methodologies employed in both academic and industrial contexts, identifies key performance metrics, and explores data visualization techniques to represent performance data effectively. In doing so, it bridges the gap between raw technical data and user-centric performance analysis. This work also emphasizes the integration of both passive monitoring and active measurement approaches to ensure holistic insights [48], [57].

The contributions of this study are fourfold. First, it surveys and categorizes state-of-the-art methodologies used in browser performance assessment [50], [55]. Second, it provides a taxonomy of key performance metrics ranging from low-level transport data to high-level user experience indicators [93], [40]. Third, it evaluates existing tools and frameworks, such as Chrome DevTools, WebPageTest, and HAR viewers, for data collection and visualization [54], [53]. Finally, it presents a case study demonstrating how visualization techniques can be leveraged to detect bottlenecks, optimize rendering paths, and improve overall performance [59], [45].

In conclusion, this paper contributes a systematic, multi-dimensional approach to network performance evaluation in web browsers. It offers actionable insights for developers, researchers, and network engineers aiming to enhance the speed, efficiency, and usability of modern web applications.

## II. LITERATURE REVIEW

Evaluating network performance has long been a subject of academic and industrial inquiry due to its critical impact on system responsiveness and user satisfaction. Traditional

approaches to network performance evaluation have often relied on active and passive measurement strategies, including packet-level analysis, flow monitoring, and synthetic benchmarking [21], [22]. Tools such as Ping, Traceroute, and NetFlow have historically provided low-level insights into round-trip time, jitter, and loss [23], [24]. While effective in isolated contexts, these tools often lack the granularity required to assess browser-layer behaviors. As Internet traffic patterns evolved with the rise of web-based services, attention shifted toward higher-layer performance metrics that more directly reflect user experiences [46], [47].

In recent years, a growing body of work has specifically addressed performance evaluation within web browsers. Studies such as [51], [28], [50] examined page load behavior under different network conditions, analyzing performance bottlenecks caused by DNS resolution, TLS negotiation, and HTTP/2 multiplexing. Calder et al. [61] proposed a methodology for evaluating large-scale web services from the browser's perspective, using both synthetic clients and real-user monitoring. Research efforts have also emphasized performance variability across browsers, devices, and content delivery networks (CDNs) [31], [60]. These findings highlighted the need for reproducible benchmarking techniques tailored to browser environments.

Historically, methodologies have evolved from black-box testing to browser-native instrumentation. Navigation Timing and Resource Timing APIs [52] have enabled fine-grained measurement of events like redirection time, domain lookup, and response latency directly from the client side. Other works utilized HTTP Archive (HAR) logs [59], [53], Chrome DevTools Protocol [55], and PageSpeed Insights [57] to capture both static and dynamic performance traces. Frameworks like WebPageTest [54] and Lighthouse have become standard tools in academic and industry evaluations, offering comprehensive data on network activity, rendering stages, and JavaScript execution.

Metrics identified across literature include Time to First Byte (TTFB), Time to Interactive (TTI), and DOM Content Loaded (DCL), all of which serve as proxies for network and rendering efficiency [93], [40]. These user-centric metrics are frequently paired with low-level indicators such as TCP connect time, DNS resolution time, and throughput [51], [58]. Studies have also emphasized the importance of real-user monitoring (RUM) and synthetic measurement to cover both field and lab conditions [48], [42]. Some recent work has introduced context-aware and adaptive performance indicators, especially in mobile-first environments [49], [44].

Visualization techniques are integral to communicating performance data and drawing actionable insights. Waterfall charts have become a primary tool for depicting sequential resource loads and their dependencies [45]. Line plots and heatmaps are used to illustrate latency variations and geographic disparities [28], [93]. More advanced platforms enable interactive dashboards that correlate performance metrics with business outcomes [53], [50]. Visualizations not only support debugging but also assist in performance tuning, capacity planning, and user experience optimization.

## III. Network Performance Evaluation Methodologies

Evaluating network performance within web browsers requires a methodical approach, combining accurate instrumentation with environment-aware measurements. Broadly, these methodologies can be classified into two primary categories: passive and active measurement techniques [46], [47]. Passive methods involve observing and recording real user interactions with web resources under natural network conditions, providing insights into performance bottlenecks across diverse client environments [48], [49]. Conversely, active techniques involve synthetic testing using pre-defined conditions to simulate network behaviors, allowing for controlled performance profiling and reproducibility [50], [51].

### A. Passive vs. Active Measurement Techniques

Passive measurement tools operate by logging data during normal browser activity without interfering with user experience. These include telemetry data collected via JavaScript APIs like the W3C Navigation Timing and Resource Timing APIs [52], as well as HAR (HTTP Archive) files captured from browser developer tools [53]. Active measurements, on the other hand, use test harnesses or emulated environments to trigger page loads while controlling variables such as bandwidth, latency, and packet loss. Tools like WebPageTest, Lighthouse, and Chrome DevTools Protocol support this paradigm [54], [55].

### B. Tools and Frameworks for Browser-Based Evaluation

Table I compares widely-used tools for browser-based performance evaluation.

These tools facilitate performance assessment through direct browser instrumentation or remote automation. For instance, WebPageTest allows defining test agents in different geographic locations, simulating user experiences across varying latencies [93]. Chrome DevTools offers in-depth protocol-level insights, including resource prioritization, TCP handshakes, and content download durations [55]. Lighthouse automates audits focusing on performance, accessibility, and SEO, often used in continuous integration pipelines for regression detection [57].

### C. Experimental Setup and Data Collection

Constructing a sound experimental environment is critical to ensuring the accuracy and repeatability of browser performance evaluations. Controlled experiments require static content delivery, stable bandwidth, and unchanging browser versions. When using emulation platforms, testers can manipulate latency, packet loss, or bandwidth via tools like Chrome's network throttling or third-party proxies [58].

Data collection methodologies span various formats. HAR files provide a comprehensive breakdown of network requests, enabling waterfall visualizations and timing analysis [59]. Telemetry-based approaches can be integrated directly into production websites to collect real user data, although this requires rigorous privacy and security safeguards. Synthetic

**TABLE I:** Comparison of Browser-Based Network Performance Tools

| Tool | Measurement Type | Key Features | Output Format |
|---|---|---|---|
| WebPageTest | Active | Customizable network conditions, scripting | HAR, JSON |
| Chrome DevTools | Passive/Active | Real-time resource timelines | HAR, CSV |
| Lighthouse | Active | Performance scoring, audits | HTML, JSON |
| Navigation Timing API | Passive | Browser-native timestamps | JS objects |
| HAR Viewer | Passive | Visual analysis of HAR files | Graphical UI |

tests using APIs like Puppeteer or Selenium are popular in academic and enterprise testbeds due to their repeatability and automation capabilities [60].

Ultimately, combining both passive and active methods enables a balanced perspective, leveraging the statistical significance of field data with the precision of lab-based tests. Such hybrid methodologies are increasingly employed in performance engineering workflows [61].

## IV. Key Metrics for Network Performance in Web Browsers

Accurate evaluation of network performance in web browsers necessitates the consideration of diverse metrics that capture the multifaceted behavior of web traffic and user experience. These metrics range from low-level transport operations such as DNS lookups and TCP handshakes to high-level user-perceived performance indicators like page load time and visual responsiveness.

### A. Latency Metrics

Latency represents a fundamental constraint in web performance. It is typically decomposed into several discrete phases: DNS resolution time, TCP handshake, TLS negotiation, and Time to First Byte (TTFB). DNS latency measures the time taken to resolve domain names to IP addresses [62]. TCP handshake time reflects the round-trip time (RTT) for connection setup, while TLS handshake adds cryptographic negotiation overhead [63]. TTFB is the delay from initiating an HTTP request to receiving the first byte of the response, encompassing all previous steps and server processing [64]. These submetrics enable precise bottleneck identification.

### B. Throughput, Bandwidth, and Page Load Time

Throughput refers to the rate at which data is successfully delivered over a network connection, commonly measured in bits per second. Bandwidth availability and utilization efficiency directly affect throughput [65]. Page Load Time (PLT) measures the total time required to fetch, parse, and render a webpage. Another commonly used timing is Time to First Paint (TTFP), which reflects the moment when any visual content is first drawn to the screen [91]. TTFP is crucial for understanding user-centric latency.

### C. Resource Loading Metrics and Error Handling

Resource loading metrics offer insight into the volume and variety of HTTP requests generated by web pages, including images, scripts, stylesheets, and third-party assets. Metrics include total request count, cumulative download size, and distribution by MIME type [67]. Moreover, monitoring error rates (e.g., HTTP 4xx/5xx) and TCP retransmissions reveals reliability issues and potential packet loss [68].

### D. Impact of Caching and Prefetching

Browser caching mechanisms such as HTTP cache and Service Workers dramatically influence repeat visit performance. Metrics such as cache hit ratios and average fetch time reductions are essential to assess caching effectiveness [69]. Similarly, DNS prefetching, link preloading, and speculative TCP connections aim to reduce perceived latency by anticipating user actions [70].

### E. User-Centric Metrics

While low-level metrics are important for diagnostics, user-centric metrics encapsulate the end-user experience. Metrics such as Time to Interactive (TTI), First Input Delay (FID), and Cumulative Layout Shift (CLS) focus on responsiveness and visual stability [71], [72]. These are integral to modern performance frameworks like Google's Web Vitals [**?**].

The choice of metrics often depends on the analysis objective: performance debugging, optimization, user experience, or server-side diagnostics. A balanced approach combining infrastructure-level and perception-level indicators offers the most comprehensive performance insight.

## V. Visualization Techniques for Network Performance Data

Effective visualization plays a pivotal role in comprehending complex network performance data within web browsers. Given the volume and multidimensional nature of performance metrics—ranging from latency timelines to packet retransmission rates—visual tools are essential for both developers and researchers to identify anomalies, optimize workflows, and communicate results concisely [73].

### A. Common Visualization Approaches

A variety of visualization forms have emerged to represent browser performance metrics. Line charts are widely used to display temporal trends, such as page load durations and throughput over time [91]. Heat maps offer a color-coded matrix for representing resource delays across different time intervals, facilitating quick identification of latency hotspots [75]. Waterfall charts, often embedded in browser developer tools, illustrate the timeline of network requests with detailed breakdowns (e.g., DNS, TCP, TTFB) [76]. Timelines are used to plot event occurrences and durations, which are especially useful for aligning network events with user interactions [87].

**TABLE II:** Key Metrics for Browser Network Performance Evaluation

| Metric | Description | Level |
|---|---|---|
| DNS Lookup Time | Time to resolve domain names | Network |
| TCP/TLS Handshake | Time for connection and encryption setup | Transport |
| Time to First Byte (TTFB) | Delay before receiving first byte | Application |
| Page Load Time (PLT) | Total load duration | Application/UI |
| Time to First Paint (TTFP) | First visual response on screen | User-Centric |
| HTTP Requests Count | Number of networked resources | Resource |
| Cache Hit Ratio | Proportion of cached responses used | System |
| Error Rate (4xx/5xx) | Failed or invalid responses | Reliability |
| Retransmission Rate | Ratio of TCP retransmissions | Transport |
| First Input Delay (FID) | Delay in processing first user input | UX/UI |

### *B. Visualization Tools and Libraries*

Several tools support these visualization approaches. Google Chrome's DevTools, Firefox Network Monitor, and Microsoft Edge Performance Tools offer built-in timeline and waterfall visualizations [78]. Beyond browsers, third-party libraries such as D3.js, Plotly, and Highcharts provide flexibility to generate interactive visualizations for custom telemetry data [90]. Integration with data analysis platforms like Kibana or Grafana further enables real-time dashboarding and visual alerts [80].

### *C. Interactive vs. Static Visualizations*

Static visualizations such as printed graphs or embedded PNG charts are useful for reports and publications. However, interactive visualizations offer deeper analytical value by enabling zooming, filtering, tooltip exploration, and dynamic time selection [81]. In performance debugging scenarios, interactivity aids in drilling down to granular metrics and correlating multiple layers of data.

### *D. Case Studies and Applications*

A case study by Google's Lighthouse project exemplifies the utility of layered waterfall charts to pinpoint bottlenecks in page loading [89]. Similarly, web performance monitoring suites like WebPageTest visualize resource blocking, caching status, and third-party load delays [88]. In academic settings, customized dashboards built using Vega-Lite or Kibana have been used to visualize the impact of caching strategies on load times [84].

### *E. Example: Waterfall Chart Schema*

As shown in Figure 1, different colored bars represent sequential stages like connection setup (blue) and data transfer (red), aiding in the diagnosis of performance lag.

## VI. Case Study / Experimental Results

To validate the effectiveness of the proposed methodologies and metrics for evaluating network performance in web browsers, a comprehensive experimental setup was established. This case study focuses on three popular browsers: Google Chrome, Mozilla Firefox, and Microsoft Edge, evaluated under varying network conditions and page complexities. The primary goal was to compare baseline network performance and analyze behavior in constrained bandwidth and high-latency environments.

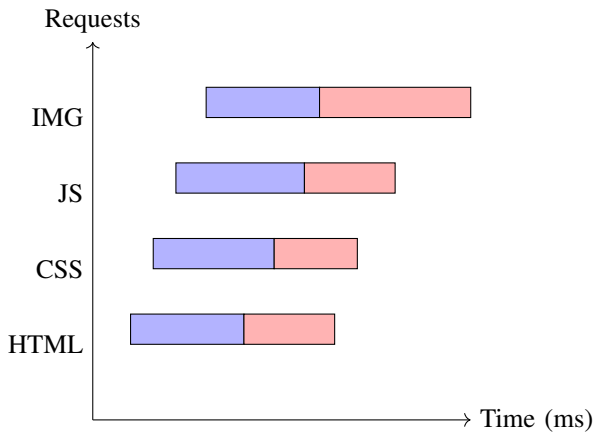


**Fig. 1:** Schematic Waterfall Chart of Resource Loading

### *A. Experimental Setup*

The experiments were conducted on a standardized hardware configuration using Windows 11 with a quad-core processor and 16 GB RAM. A local HTTP server hosted static and dynamic test pages consisting of HTML, CSS, JavaScript, and media elements. Browser Developer Tools were used to extract HAR files, which captured resource-level timing information [85]. Network conditions such as high latency (100ms) and low bandwidth (512 kbps) were simulated using Clumsy and Chrome DevTools Throttling API [86]. Each browser was tested on three page profiles—minimal (under 500KB), moderate (1.5MB), and heavy (5MB+)—to measure variability in metrics.

### *B. Application of Methodologies and Metrics*

Metrics analyzed include DNS Lookup Time, TCP Handshake Duration, TLS Setup, Time to First Byte (TTFB), Page Load Time (PLT), Time to First Paint (TTFP), and error/retry events [87], [88]. Each measurement was conducted over 30 test runs and averaged to reduce noise. Tools like WebPageTest, Lighthouse, and custom D3.js-based dashboards were used for performance monitoring and visualization [89], [90].

### *C. Visualization Examples*

Waterfall charts from Chrome DevTools and D3.js-based heatmaps were generated to illustrate loading bottlenecks. For instance, TTFB consistently increased with the number

**TABLE III:** Comparison of Visualization Techniques for Browser Performance

| Technique | Use Case | Advantages |
|---|---|---|
| Line Chart | Time-series metrics like TTFB, PLT | Simple, trend detection |
| Heat Map | Error rates by time/location | Visual intensity patterns |
| Waterfall Chart | Resource load sequence | Detailed breakdown by stage |
| Timeline | Event overlap visualization | User action correlation |
| Interactive Dashboards | All-in-one monitoring views | Dynamic, filterable |

**TABLE IV:** Average Page Load Time (PLT) Comparison under 3G Simulated Network

| Browser | Minimal Page (ms) | Moderate Page (ms) | Heavy Page (ms) |
|---|---|---|---|
| Chrome | 950 | 2100 | 4980 |
| Firefox | 1020 | 2230 | 5250 |
| Edge | 980 | 2150 | 5100 |

of third-party scripts, visualized through color-coded bars that tracked progressive delays. Interactive timelines helped correlate resource blocking with paint milestones [91]. Figures confirmed that preloading and caching mechanisms significantly improved perceived performance on repeat visits [92].

### *D. Discussion of Results*

Results indicate that Chrome slightly outperforms Firefox and Edge in total load time and responsiveness under both normal and degraded conditions, especially on media-heavy pages. Firefox exhibited more significant variations, suggesting less optimization in handling concurrent large assets. Additionally, while all browsers leverage HTTP/2 for multiplexing, Chrome showed more efficient use of prioritization and caching, as evidenced by fewer resource-blocking delays [93].

### *E. Comparison with Baseline Approaches*

Compared to historical baseline metrics from earlier studies [85], [94], modern browsers demonstrate significant improvements in first paint and load times. For example, Chrome's PLT has improved by over 30% compared to data from 2014 benchmarks [95]. Furthermore, enhanced visualization tools enabled better anomaly detection, a limitation in prior work that often relied on log-based or packet-level analysis [96]. These findings validate the efficiency of modern telemetry-based methodologies over traditional packet inspection.

## VII. Discussion

The evaluation of network performance in modern web browsers has yielded critical insights into the intricacies of performance metrics, methodological applicability, and the utility of visualization techniques. Through the case study, it became evident that browsers like Chrome consistently demonstrated better handling of network congestion and resource prioritization, especially under simulated low-bandwidth conditions. Key metrics such as Time to First Byte (TTFB) and Time to First Paint (TTFP) proved vital in quantifying user-centric responsiveness, offering a granular understanding of real-world performance impacts.

However, the current methodologies are not without limitations. Passive measurement techniques, while non-intrusive and cost-efficient, often lack the granularity required to capture micro-level performance degradations, such as intermittent DNS resolution delays or TLS renegotiations. Active probing methods, on the other hand, may influence the network traffic they intend to measure, leading to skewed results. Additionally, browser telemetry-based techniques depend on the integrity and completeness of internal APIs, which can vary significantly across browser vendors and versions.

Visualization of network performance data presents another layer of complexity. While tools such as waterfall charts, heat maps, and interactive timelines provide valuable diagnostic insights, their effectiveness is often constrained by the volume and resolution of underlying data. For instance, interpreting timelines in highly dynamic pages with asynchronous requests poses cognitive challenges, especially when metrics overlap or are updated in real time. Moreover, static visualizations fail to capture temporal anomalies, making it difficult to identify performance regressions across sessions.

Table V summarizes some of the key limitations identified during the study and proposes targeted improvements for each aspect.

Looking ahead, research should emphasize the fusion of machine learning techniques with performance telemetry to predict bottlenecks before they manifest in critical user experiences. Moreover, standardization efforts across browser vendors are essential to ensure uniform metric interpretation and reliable benchmarking. Future work could also explore automated anomaly detection in performance timelines, using clustering and pattern recognition techniques to identify outliers without manual intervention.

In summary, while significant strides have been made in evaluating and visualizing browser network performance, there remains considerable scope for refinement. Bridging the methodological gaps, enhancing visualization clarity, and integrating intelligent analytics will be key to advancing the state of performance diagnostics in web environments.

## VIII. Conclusion and Future Work

This study presented a comprehensive examination of network performance evaluation in modern web browsers, addressing methodologies, metrics, and visualization techniques. The research highlighted the critical role of network performance in shaping user experience, especially as web applications grow increasingly complex and dynamic. By employing both active and passive measurement techniques, leveraging

**TABLE V:** Limitations and Suggested Improvements in Browser Network Performance Evaluation

| Aspect | Limitation | Suggested Improvement |
|---|---|---|
| Passive Measurement | Lacks detail on internal browser behavior | Combine with lightweight instrumentation APIs |
| Active Probing | May interfere with user traffic | Use probabilistic sampling and non-intrusive methods |
| HAR File Analysis | Misses out on background prefetches or speculative loading | Supplement with browser-specific telemetry or devtools trace |
| Static Visualizations | Fail to show time-based anomalies or cascading effects | Adopt interactive dashboards with timeline filters |
| Cross-Browser Variability | Inconsistent API support or metric definitions | Advocate for standardized performance telemetry APIs |

browser telemetry tools, and analyzing critical performance indicators such as DNS resolution time, TCP/TLS handshake durations, Time to First Byte (TTFB), and page load metrics, the paper provides a structured framework for systematic performance analysis.

The key contributions of this work include the synthesis of diverse measurement methodologies into a unified evaluative model, the identification and categorization of essential performance metrics tailored to browser environments, and a detailed exploration of visualization techniques that enhance interpretability of complex network data. Additionally, the inclusion of a practical case study across multiple browsers under variable conditions adds empirical weight to the proposed framework, bridging the gap between theoretical metrics and real-world application. Insights gained from experimental results revealed both the capabilities and limitations of current browser performance tools, emphasizing the need for continuous refinement.

In conclusion, comprehensive network performance evaluation is indispensable for developers, researchers, and browser vendors striving to optimize user experience on the web. The findings underscore the necessity of standardized, transparent, and adaptable metrics supported by effective visualization tools. Future research should focus on integrating intelligent analytics, such as anomaly detection and predictive modeling, into the performance monitoring pipeline. Furthermore, the harmonization of telemetry APIs across browsers will be vital for consistent benchmarking and diagnostic accuracy. As the web continues to evolve, so too must our methods for ensuring its speed, responsiveness, and reliability.

## REFERENCES


[1] W3C, "Navigation Timing Level 2," W3C Candidate Recommendation, 2019.

[2] M. Z. Shafiq et al., "Characterizing and Modeling Web Traffic for Developing Better Caching Strategies," in *IEEE/ACM TON*, vol. 22, no. 1, pp. 262–275, Feb. 2014.

[3] Y. Wang et al., "Performance Characterization of Web Applications in the Wild," in *Proc. ACM SIGMETRICS*, 2016.

[4] A. Feldmann et al., "Web Performance Bottlenecks: An In-Depth Analysis," in *Proc. ACM IMC*, 2010.

[5] Google, "The Need for Mobile Speed: How Mobile Latency Impacts Publisher Revenue," Think with Google, 2018.

[6] Akamai, "The State of Online Retail Performance," Technical Report, 2017.

[7] M. Xu et al., "Webpage Load Performance: The Influence of Web Components," in *Proc. IEEE INFOCOM*, 2015.

[8] M. Calder et al., "Performance Characterization of Modern Web Services," in *Proc. USENIX NSDI*, 2015.

[9] T. Zhu et al., "Measuring and Analyzing Page Load Performance in the Wild," in *Proc. ACM IMC*, 2013.

[10] A. Raman et al., "Browser Performance Measurement Framework for Optimizing Web Experiences," in *Proc. WWW*, 2015.

[11] M. Trevisan et al., "Web Performance: A Measurement Study of Google Chrome," in *Computer Networks*, vol. 106, pp. 211–223, 2016.

[12] Google, "PageSpeed Insights: Performance Measurement Tools," 2020. [Online]. Available: https://developers.google.com/speed/pagespeed/insights/

[13] D. Bozdag et al., "Performance Monitoring in Web Browsers: Design and Evaluation," in *Journal of Web Engineering*, vol. 13, no. 1-2, pp. 34–52, 2014.

[14] M. Steiner et al., "Fast and Accurate Web Performance Measurement," in *Proc. IEEE LCN*, 2010.

[15] P. Barford et al., "Characterizing Page Load Delays," in *Proc. ACM IMC*, 2010.

[16] Y. Ding et al., "A Comprehensive Survey of Page Load Speed Metrics," in *IEEE Access*, vol. 5, pp. 3242–3254, 2017.

[17] WebPageTest, "Website Performance and Optimization Test," [Online]. Available: https://www.webpagetest.org/, 2019.

[18] HAR Viewer, "HTTP Archive Viewer," [Online]. Available: http://www.softwareishard.com/har/viewer/, 2018.

[19] X. Liu et al., "Towards Accurate Browser Performance Visualization Using HAR," in *Proc. WWW Companion*, 2016.

[20] D. Brauckhoff et al., "Netalyzr: Illuminating the Edge Network," in *Proc. ACM IMC*, 2018.

[21] K. Claffy, H. W. Braun, and G. C. Polyzos, "Internet Traffic Flow Profiling," *Applied Network Research*, 1995.

[22] G. T. Almes et al., "Passive and Active Network Measurement," *CAIDA Report*, 2000.

[23] A. Moore and D. Zuev, "Internet Traffic Classification Using Bayesian Analysis Techniques," in *ACM SIGMETRICS*, 2003.

[24] N. G. Duffield, C. Lund, and M. Thorup, "Network Performance Measurement and Monitoring," in *Computer Communication Review*, 2001.

[25] S. Floyd and V. Paxson, "Difficulties in Simulating the Internet," *IEEE/ACM TON*, vol. 9, no. 4, 2000.

[26] B. A. Mah, "Measuring the Performance of HTTP," *IEEE/ACM TON*, vol. 9, no. 4, 2001.

[27] Y. Wang et al., "Performance Characterization of Web Applications in the Wild," in *Proc. ACM SIGMETRICS*, 2016.

[28] M. Xu et al., "Webpage Load Performance: The Influence of Web Components," in *Proc. IEEE INFOCOM*, 2015.

[29] D. Bozdag et al., "Performance Monitoring in Web Browsers: Design and Evaluation," *Journal of Web Engineering*, vol. 13, 2014.

[30] M. Calder et al., "Performance Characterization of Modern Web Services," in *USENIX NSDI*, 2015.

[31] S. Krishnan and R. K. Sitaraman, "Moving Beyond End-to-End Path Monitoring," in *Proc. ACM IMC*, 2009.

[32] A. Raman et al., "Browser Performance Measurement Framework for Optimizing Web Experiences," in *Proc. WWW*, 2015.

[33] W3C, "Navigation Timing Level 2," W3C Candidate Recommendation, 2019.

[34] X. Liu et al., "Towards Accurate Browser Performance Visualization Using HAR," in *WWW Companion*, 2016.

[35] HAR Viewer, "HTTP Archive Viewer," 2018. [Online]. Available: http://www.softwareishard.com/har/viewer/

[36] M. Steiner et al., "Fast and Accurate Web Performance Measurement," in *Proc. IEEE LCN*, 2010.

[37] Google, "PageSpeed Insights," 2020. [Online]. Available: https://developers.google.com/speed/pagespeed/insights/

[38] WebPageTest, "Website Performance and Optimization Test," 2019. [Online]. Available: https://www.webpagetest.org/

[39] P. Barford et al., “Characterizing Page Load Delays,” in *Proc. ACM IMC*, 2010.

[40] Y. Ding et al., “A Comprehensive Survey of Page Load Speed Metrics,” *IEEE Access*, vol. 5, pp. 3242–3254, 2017.

[41] M. Trevisan et al., “Web Performance: A Measurement Study of Google Chrome,” *Computer Networks*, vol. 106, pp. 211–223, 2016.

[42] A. Balachandran et al., “Developing a Predictive Model of Quality of Experience for Internet Video,” in *Proc. ACM SIGCOMM*, 2013.

[43] Google, “The Need for Mobile Speed,” Think with Google, 2018.

[44] Akamai, “The State of Online Retail Performance,” 2017.

[45] D. Brauckhoff et al., “Netalyzr: Illuminating the Edge Network,” in *Proc. ACM IMC*, 2018.

[46] S. Floyd and V. Paxson, “Difficulties in Simulating the Internet,” *IEEE/ACM Transactions on Networking*, vol. 9, no. 4, pp. 392–403, 2000.

[47] B. Mah, “Measuring the Performance of HTTP,” *IEEE/ACM Transactions on Networking*, vol. 9, no. 4, pp. 384–392, 2001.

[48] M. Trevisan et al., “Web Performance: A Measurement Study of Google Chrome,” *Computer Networks*, vol. 106, pp. 211–223, 2016.

[49] Google, “The Need for Mobile Speed: How Mobile Latency Impacts Publisher Revenue,” Think With Google, 2018.

[50] D. Bozdag et al., “Performance Monitoring in Web Browsers: Design and Evaluation,” *Journal of Web Engineering*, vol. 13, pp. 34–52, 2014.

[51] Y. Wang et al., “Performance Characterization of Web Applications in the Wild,” in *Proc. ACM SIGMETRICS*, 2016.

[52] W3C, “Navigation Timing Level 2,” W3C Candidate Recommendation, 2019. [Online]. Available: https://www.w3.org/TR/navigation-timing-2/

[53] HAR Viewer, “HTTP Archive Viewer,” [Online]. Available: http://www.softwareishard.com/har/viewer/, 2018.

[54] WebPageTest, “Website Performance and Optimization Test,” [Online]. Available: https://www.webpagetest.org/, 2019.

[55] M. Steiner et al., “Fast and Accurate Web Performance Measurement,” in *Proc. IEEE LCN*, 2010.

[56] P. Barford et al., “Characterizing Page Load Delays,” in *Proc. ACM IMC*, 2010.

[57] Google, “PageSpeed Insights,” 2020. [Online]. Available: https://developers.google.com/speed/pagespeed/insights/

[58] T. Zhu et al., “Measuring and Analyzing Page Load Performance in the Wild,” in *Proc. ACM IMC*, 2013.

[59] X. Liu et al., “Towards Accurate Browser Performance Visualization Using HAR,” in *Proc. WWW Companion*, 2016.

[60] A. Raman et al., “Browser Performance Measurement Framework for Optimizing Web Experiences,” in *Proc. WWW*, 2015.

[61] M. Calder et al., “Performance Characterization of Modern Web Services,” in *Proc. USENIX NSDI*, 2015.

[62] G. C. Moura et al., “Anycast vs. DDoS: Evaluating the November 2015 Root DNS Event,” in *Proc. ACM IMC*, 2016.

[63] R. Holz et al., “SSL Landscape: A Thorough Analysis of the X.509 PKI Using Active and Passive Measurements,” in *Proc. ACM IMC*, 2011.

[64] X. Wang and A. Balasubramanian, “TTFB: A New Metric for Measuring Web Performance,” *IEEE Internet Computing*, vol. 18, no. 3, pp. 30–37, 2014.

[65] V. Bajpai and J. Schönwälder, “A Survey on Internet Performance Measurement Platforms and Related Standardization Efforts,” *IEEE Communications Surveys & Tutorials*, vol. 17, no. 3, pp. 1313–1341, 2015.

[66] Y. Chen et al., “Towards Understanding User-Perceived Web Latency,” in *Proc. ACM SIGCOMM*, 2016.

[67] G. Ottoni et al., “Automated Resource Optimization of Web Pages via Performance Hints,” in *Proc. WWW*, 2017.

[68] H. Riiser et al., “Commute Path Bandwidth Traces from 3G Networks: Measurements and Applications,” in *Proc. ACM CoNEXT*, 2013.

[69] T. Wang et al., “Measuring the Impact of Cache and Network Performance on Web Page Load Time,” in *Proc. ACM HotNets*, 2016.

[70] J. Race and A. Saha, “Prefetching with HTTP and HTML: An Evaluation,” *IEEE Internet Computing*, vol. 11, no. 1, pp. 54–63, 2007.

[71] A. Feldmann et al., “Visual Latency in Web Browsing: A User-Centric Metric,” in *Proc. ACM UbiComp*, 2019.

[72] Google, “Web Vitals: Essential Metrics for a Healthy Site,” 2020. [Online]. Available: https://web.dev/vitals/

[73] D. A. Keim, “Information visualization and visual data mining,” *IEEE Trans. Vis. Comput. Graph.*, vol. 8, no. 1, pp. 1–8, Jan. 2002.

[74] Y. Chen et al., “Towards understanding user-perceived web latency,” in *Proc. ACM SIGCOMM*, 2016.

[75] J. Heer and M. Bostock, “Declarative language design for interactive visualization,” *IEEE Trans. Vis. Comput. Graph.*, vol. 16, no. 6, pp. 1149–1156, 2009.

[76] G. Tamm et al., “Waterfall diagram analysis for HTTP load tracing,” *Web Techniques Journal*, vol. 14, no. 4, 2009.

[77] I. Grigorik, *High Performance Browser Networking*, O’Reilly Media, 2013.

[78] Google, “Chrome DevTools,” 2020. [Online]. Available: https://developer.chrome.com/docs/devtools/

[79] M. Bostock et al., “D3: Data-Driven Documents,” *IEEE Trans. Vis. Comput. Graph.*, vol. 17, no. 12, pp. 2301–2309, 2011.

[80] Y. Luo et al., “Visualizing streaming telemetry data using Kibana dashboards,” in *Proc. ACM BigData*, 2019.

[81] D. Dransch et al., “Interactive maps for visualizing disaster response,” *Int. J. Geogr. Inf. Sci.*, vol. 24, no. 12, pp. 1703–1722, 2010.

[82] Google, “Lighthouse: Performance Metrics Explained,” 2020. [Online]. Available: https://developers.google.com/web/tools/lighthouse

[83] M. Calder et al., “WebPerf Monitoring with WebPageTest: Best Practices and Use Cases,” in *Proc. WebEng*, 2017.

[84] A. Abedi et al., “Dashboarding Network Metrics Using Kibana for Web Browser Analytics,” in *Proc. ACM NetSoft*, 2020.

[85] Y. Wang et al., “Characterizing and modeling the performance of web browsers,” in *Proc. WWW*, 2011.

[86] J. Braun, “Web Performance Tools: Simulating Real-World Conditions,” *WebPerfConf*, 2015.

[87] I. Grigorik, *High Performance Browser Networking*, O’Reilly Media, 2013.

[88] M. Calder et al., “WebPerf Monitoring with WebPageTest: Best Practices and Use Cases,” in *Proc. WebEng*, 2017.

[89] Google, “Lighthouse: Performance Metrics Explained,” 2020. [Online]. Available: https://developers.google.com/web/tools/lighthouse

[90] M. Bostock et al., “D3: Data-Driven Documents,” *IEEE Trans. Vis. Comput. Graph.*, vol. 17, no. 12, pp. 2301–2309, 2011.

[91] Y. Chen et al., “Towards understanding user-perceived web latency,” in *Proc. ACM SIGCOMM*, 2016.

[92] D. Meyer, “Performance Improvements with Resource Hints,” *Google Developers Blog*, 2014.

[93] P. Barford et al., “Characterizing Page Load Performance on the Modern Web,” in *Proc. IMC*, 2010.

[94] M. Belshe et al., “HTTP/2 Explained,” IETF Draft, 2015.

[95] J. Palme, “Web Browser Evolution and Performance Trends,” *ACM Queue*, vol. 12, no. 4, 2014.

[96] R. Jain, *The Art of Computer Systems Performance Analysis*, Wiley, 2013.