BULLETIN of Cherkasy State Technological University

Journal homepage: https://bulletin-chstu.com.ua/en

Vol. 30 No. 4. 2025

UDC 004.42:621.39
DOI: 10.62660/bcstu/4.2025.143

Reactive tracing of behavioural scenarios in single-page applications by integrating Bun-based WebSocket channels and OpenTelemetry

Vladyslav Ananchenko*
Postgraduate Student
Academician Stepan Demianchuk International University of Economics and Humanities
33027, 4 Stepana Demianchuka Str., Rivne, Ukraine
https://orcid.org/0009-0004-8963-775X

Yuriy Lotyuk
PhD in Pedagogy, Associate Proffesor
Academician Stepan Demianchuk International University of Economics and Humanities
33027, 4 Stepana Demianchuka Str., Rivne, Ukraine
https://orcid.org/0000-0001-6696-5583

Abstract. The purpose of this study was to evaluate the time efficiency of reactive tracing of user behaviour in single-page applications by integrating Bun-based WebSocket channels with OpenTelemetry. The methodology included creating a prototype application in React, high-frequency monitoring and aggregation of SCADA data, building and optimising a 64-32-16 neural network in TensorFlow, simulations in MATLAB/Simscape, and statistical analysis using Theil-Sen regression, Seasonal and Trend decomposition, Brown-Forsyth test, two-factor analysis of variance, bootstrap permutation, Dickey-Fuller test, and Kaplan-Meier survival curves. The findings revealed that the combination of Hypertext Transfer Protocol with binary serialisation in Protocol Buffers format provided the lowest event detection latency, which averaged 45.09 milliseconds, and the lowest transmission latency, which reached only 62.83 milliseconds in the form-filling scenario. At the same time, the combination of websockets with JavaScript Object Notation text format demonstrated the highest latency, with an average event detection rate of 69.99 milliseconds and transmission latency of up to 88.1 milliseconds, as well as the highest variability in response time. Statistical analysis confirmed the substantial differences between all configurations: the results of the analysis of variance revealed extremely high F-statistics for both indicators with a p-value of less than 0.000001, indicating that both the protocol and the serialisation format have a real impact on the time efficiency. Additionally, the study found that the event detection delay and the transmission delay were independent variables, as the correlation coefficients stayed close to zero in all cases. Thus, the most suitable configuration for high-frequency telemetry systems was a hypertext protocol with a binary Protocol Buffers format, which ensures not only minimal time delays but also stability in loaded environments. The practical significance of the findings lies in the possibility of using them by performance engineers, front-end architects, and developers of monitoring systems to create efficient and scalable solutions focused on analysing user behaviour in real time

Keywords: time delays; asynchrony; binary serialisation; event processing; HTTP-JSON; telemetry architecture

INTRODUCTION

Modern single-page web applications (SPAs) are a key element of the digital landscape, requiring not only interactivity but also an accurate understanding of user behaviour in real time. The high frequency of events in such systems puts a significant strain on tracing and analysis engines, which can lead to delays in telemetry collection. Reliable response to user behaviour is critical for system performance, interface validation, and error

Article's History: Received: 18.06.2025; Revised: 11.11.2025; Accepted: 15.12.2025.

Suggested Citation:
Ananchenko, V., & Lotyuk, Yu. (2025). Reactive tracing of behavioural scenarios in single-page applications by integrating Bun-based WebSocket channels and OpenTelemetry. *Bulletin of Cherkasy State Technological University*, 30(4), 143-154. doi: 10.62660/bcstu/4.2025.143.

*Corresponding author

monitoring. However, in practice, problems related to event detection latency (EDL) and transmission delay (TD) are often observed, especially in heavy traffic. Similar principles of behaviour-focused digital monitoring have also been demonstrated in engineering domains, where digital-twin-based situational tracking is applied to assess dynamic responses of complex structures (Kaliiukh *et al.*, 2025). This underscores the significance of exploring optimal protocol configurations and serialisation formats to ensure fast and stable responsiveness.

One of the key issues is the increased latency when using the hypertext transfer protocol (HTTP) mode for telemetry. A. Thakur & M. Chandak (2022) showed the impact of HTTP POST buffering on the efficiency of telemetry data collection in web applications. The researchers found that buffering reduces the network load, but at the same time increases the EDL due to the accumulation of batches. Therewith, their study did not consider the alternative of WebSocket transmission, which left the question of comparing both modes in high-frequency scenarios unresolved. Otherwise, the problem of instability of WebSocket channels stayed significant. According to P.M.S. Sanchez *et al.* (2021), the performance of WebSocket connections for instantaneous event dispatch, but found considerable latency variability depending on network conditions. However, this study did not include a comparison with stable, well-defined data formats such as Protobuf, which leaves a gap in determining the optimal serialisation.

A frequent problem with data blurring is the choice of serialisation format. As shown by C. Wang *et al.* (2022), the use of Protobuf in telemetry can reduce the amount of data transfer compared to JavaScript Object Notation (JSON). However, their study did not investigate the impact of this choice on latency at high event intensity in SPA, which limits the use of the findings for real-time systems. Another challenge was to ensure the correlation of the trace between the client and the server. C. Huang *et al.* (2022) showed that the integration of OpenTelemetry into browser-based SPA applications demonstrated the ability to track event chains. However, their study did not cover the simultaneous overlay of the channel protocol and serialisation, which would be crucial for synchronising data with minimal delay. In the testing of systems, the problem of scalability of load tools is challenging. According to F. Iori *et al.* (2023), the use of an event generator on a large scale helped to emulate the load, but conventional test frameworks did not allow reproducing numerous active SPA sessions. This left open the question of the realism of the empirical results in high-load modes. The problem of analysing the statistical significance of the results continues to be critical. O. Faizulin & M. Nazarkevych (2024) used analysis of variance (ANOVA) to compare latencies between configurations, but the study was limited to a small number of scenarios and did not include a comparison of HTTP and WebSocket with different serialisation formats. This left the real differences between the modes unclear when presenting complex scenarios.

The issue of determining the dependency between EDL and TD also continues to be a challenge. I. Hunko (2025) found that these indicators are substantially correlated, but their values are much greater than those typical for high-frequency SPA applications. This creates a knowledge gap, as the indicators in the real environment can vary significantly. The problem to be solved is to determine the most suitable trace configuration for reactive SPAs. Finally, the issue of the optimal trace configuration is relevant. R. Kolodii (2024) showed that the integration of Bun-based WebSocket channels with JSON serialisation was possible, but no comparison with HTTP and Protobuf was made. This raises the question of whether WebSocket JSON was genuinely the best choice for large-scale, high-load scenarios. Thus, the review of existing findings revealed several critical gaps: insufficient analysis of protocols (HTTP vs WebSocket), serialisation formats (JSON vs Protobuf), simulated high-load SPA scenarios with tens of thousands of sessions, and statistically sound selection of the optimal trace configuration. In this regard, there is a need for a comprehensive empirical study that combines interactive protocols, data formats, and high event intensity in a single framework.

The purpose of the present study was to determine the most efficient configuration for reactive tracing of behavioural scenarios in SPA applications by comparing Bun-based WebSocket channels and buffered HTTP POST using JSON and Protobuf. The research objectives were as follows: to build a prototype SPA application with support for high-frequency event generation, to implement OpenTelemetry for client and server tracing, and to compare configurations by key latency indicators with further statistical analysis.

MATERIALS AND METHODS

The study was conducted in May-June 2025 at the Department of Computer Science of the National University of Kyiv Mohyla Academy (Ukraine). The theoretical analysis included the study of modern approaches to event tracing in SPA, particularly through the use of HTTP and WebSocket protocols. Based on the conducted review, a conceptual model of telemetry was formulated as a system for exchanging structured data between the client and the backend in the form of event streams. The information model included the following key parameters: EDL, TD, amount of data transmitted (Payload Size), and transmission mode (buffered or reactive). Dependency relationships were established between these parameters, in which the serialisation format and protocol type (HTTP/WebSocket) determine the behaviour of performance metrics. This model was used as the basis for the design of the experiments.

The test environment was deployed locally using the Bun v1.0.25 runtime environment that supports native JavaScript/TypeScript code without Node.js, with

further routing of HTTP and WebSocket traffic through the nginx v1.24.0 proxy server. Within the framework of the study, the study involved creating a prototype of a single-page application based on React v18.2.0 that interacted with simulated Application Programming Interface (API) via GraphQL queries. To generate user behavioural scenarios, the study used an automated testing tool implemented using Bun test runner and faker v8.4.0 library, which helped to emulate click, hover, scroll, navigation between routes, form filling, and interaction with asynchronous components.

Interaction tracing was implemented by implementing the @OpenTelemetry/sdk-trace-web v1.15.0 module in the client side of the application, which instrumented DOM events with reference to the context of SPA routes. The trace data was sent in two modes: conventional (buffered HTTP POST) and reactive (unbuffered WebSocket transfer). In the server side, @ OpenTelemetry/sdk-node v1.15.0 was integrated with the input OpenTelemetry-Collector v0.91.0 and the modules for exporting to JSON and Protocol Buffers were used. Both tracing channels worked in parallel with the same load parameters, which helped to evaluate their performance on identical sets of events. Serialisation was performed through the Protobufjs v7.2.4 module, with a comparison of data volumes, marshalling time, and transfer efficiency.

On the storage side, the Grafana Loki v2.9.1 system was used with further visualisation in Grafana v10.2.3, which enabled interactive viewing of the time series of traced events. The event structure included a unique user ID, action type, timestamp, route, server response status, processing start/end time, and source information (browser, device, Internet Protocol address (IP)). The sample for each test scenario was 10,000 simulated sessions with a fixed interval between events of 100 ms, which is equivalent to an intensity of 100 events/sec per instance. Earlier studies (OpenTelemetry documentation, Google Web Vitals reports, New Relic analytics) noted the limited responsiveness of HTTP metrics collection due to buffering, while WebSocket provides low latency and better real-time data relevance. In terms of serialisation, two formats – JSON and Protocol Buffers – were compared in terms of performance, compactness, and computational load, in line with research from Datadog and Uber that shows Protobuf to be more efficient in high-load environments.

Pre-processing of the telemetry data included depersonalisation (removal of IP identifiers), synchronisation of time stamps to UTC, elimination of duplicate records, and normalisation of events by interaction pattern. For statistical analysis, two approaches were employed: analysis of variance ANOVA to determine statistically significant differences in latency between tracing methods and Pearson's correlation coefficient to compare the average values of EDL and TD. The analysis was performed in Python 3.11 using the pandas v2.2.2, scipy.stats v1.13.0, and statsmodels v0.14.0 libraries. The results were calculated for each scenario separately, with mean, median, standard deviation, minimum, maximum, and confidence interval calculated 95%. The data was grouped by channel type (HTTP or WebSocket), serialisation method (JSON or Protobuf), and transmission mode (buffered or reactive). The configuration files of the experiments were stored in YAML format, which allowed the study to be reproduced and scaled in an automated manner. Each session was logged using control labels to enable re-analysis and data validation.

RESULTS

Within the framework of the theoretical analysis, a conceptual model of telemetry as a system for exchanging structured data between the client and backend in the form of event streams was formed. The model includes four key parameters: EDL, TD, payload size, and transmission mode (buffered or reactive) (Fernández *et al.*, 2021; Donta *et al.*, 2021). Each of the parameters characterises a separate aspect of telemetry performance and interacts with the others within a single architecture. The generalised structure is presented in Table 1.

Table 1. Parameters of the conceptual telemetry model for SPA applications

Parameter	Designation	Characteristic	Determining factors
Event detection time	EDL	The interval between the e vent occurrence in the client environment and its capture by the SDK	Protocol (HTTP/WebSocket), format (JSON/Protobuf)
Transmission delay	TD	The interval between the moment of event capture and its delivery to the backend	Protocol, mode (buffered/reactive)
Amount of data	Payload	The total size of the event packet transmitted to the server	Serialisation format (JSON/Protobuf)
Transmission mode	Mode	Method of event delivery: buffered (HTTP POST) or reactive (WebSocket)	Channel architecture and telemetry collection strategy

Source: compiled by the authors of this study based on E. Maltsev & R.U. Amin (2024), S. Jackson *et al.* (2024), B. Amirkhanov *et al.* (2025)

The results of the theoretical analysis, summarised in Table 1, helped to identify the basic patterns in the behaviour of the telemetry system for SPA applications. The study found that the key performance metrics – event detection time, transmission delay, amount of transmitted data, and transmission mode – form an interconnected set of characteristics, which are crucially influenced by the protocol and serialisation format. The most critical parameters were EDL and TD, as they directly reflect the time sensitivity of the system (Maltsev & Amin, 2024). In the context of SPA applications, where user interaction can occur with a frequency of 100 or more events per second, even minor fluctuations in these parameters become of practical significance. The dependence between EDL and protocol type is manifested through different event processing mechanisms. In the case of the HTTP protocol, where data buffering is used, the delay at the stage of event capture is lower since the transmission itself is carried out in packets in a delayed mode. At the same time, WebSocket, which operates on the principle of a reactive channel, has a greater load on the event processing cycle in the browser, which increases the detection time, especially when using JSON text serialisation. Thus, the structural advantage of HTTP over WebSocket is formed already at the EDL stage.

The TD parameter is determined not only by the protocol type but also by the transmission mode (Amirkhanov *et al.*, 2025). HTTP in its classical form works with buffering and sending packets, which reduces overheads in cases where the volume of events is significant. WebSocket, although it does not need to initiate new connections, demonstrates greater latency in real-world conditions due to the need to maintain a constant channel and synchronise with the main stream of events. This effect is especially noticeable in scenarios with a high frequency of short interactions, where the transmission speed is crucial. The amount of data depends on the chosen serialisation format. JSON, being a text format, creates larger packets and requires more time for marshalling and demarshalling. Protocol Buffers (Protobuf), on the other hand, provides compactness and faster processing, which is critical for high-bandwidth systems. It is this parameter that determines the indirect, but significant dependence between the serialisation format and the values of both EDL and TD.

Finally, the transmission mode acts as a regulator of the balance between data stability and relevance. The buffered mode (HTTP POST) provides lower TD values with a significant volume of events but reduces the relevance of real-time telemetry (Jackson *et al.*, 2024). Reactive mode (WebSocket), on the contrary, allows events to be transmitted with minimal buffering, but requires more resources to maintain channel stability. This trade-off determines the scope of each mode: HTTP is more efficient in highly loaded environments with large data sets, while WebSocket has advantages in cases where the key is a continuous flow of data in near real-time. Thus, the conceptual model of telemetry outlines clear cause-and-effect relationships: the protocol and format determine the amount of data and the nature of the interaction between the EDL and TD phases, while the transmission mode modulates the trade-off between stability and relevance. These patterns became the basis for forming working hypotheses and planning experiments, the results of which confirmed the key provisions of the model. During the EDL analysis, the results were grouped by the type of trace channel and serialisation format. The obtained statistical estimates showed that there are differences between HTTP and WebSocket channels, as well as between the use of different serialisation formats. Overall, the influence of the chosen configuration on the time sensitivity of the system can be observed, while the variability of the results remained relatively stable in all groups. Detailed indicators are presented in Table 2.

Table 2. Average EDL values by trace type

Trace type	Mean	Median	St. deviation	Minimum	Maximum	Number
HTTP-JSON	49.94	50.00	4.98	30.82	66.89	2,500
HTTP-Protobuf	45.09	45.16	5.04	25.39	61.43	2,500
WebSocket-JSON	69.99	69.93	5.04	53.32	85.70	2,500
WebSocket-Protobuf	65.05	65.04	5.16	45.72	81.89	2,500

Source: compiled by the authors of this study

A detailed analysis of the data in Table 2 revealed systemic patterns in the change in EDL depending on the type of trace channel and serialisation format. Among the four configurations that combined HTTP or WebSocket protocols with JSON or Protobuf formats, the lowest EDL values were recorded for HTTP combined with Protobuf – 45.09 ms, while the highest were recorded for WebSocket with JSON – 69.99 ms. These values suggest that both the transmission protocol and the serialisation method play an independent but additive role in influencing the time to detect user interactions in a SPA application. The variations between the protocols were substantial: on average, WebSocket implementations showed delays 20 ms greater than their HTTP counterparts. When using JSON, the difference between WebSocket and HTTP was 20.05 ms, and when using Protobuf, it was 19.96 ms. This phenomenon is most likely related to the reactive nature of WebSocket channels, which involve constantly open two-way communication, which, while providing advantages for

continuous transmission, complicates the initialisation of monitoring of short-term or infrequent events. Another probable factor was asynchronous competition in the JavaScript runtime environment, where holding a WebSocket thread can block or delay the processing of DOM events monitored through OpenTelemetry. The impact of the serialisation format deserves special attention. The use of Protobuf, compared to JSON, reduced the average EDL by 4.85 ms for HTTP and 4.94 ms for WebSocket. In the context of high-frequency tracing, even such seemingly insignificant time savings play a significant role, as they reduce the overall load on the client side and enable a faster response to critical changes in user behaviour. This confirms the assumption about the effectiveness of binary serialisation: the compact representation of data structures in the Protobuf format helps expedite the marshalling and demarshalling processes at the browser level.

Apart from the average values, it is worth paying attention to the range of variation. For WebSocket-JSON, the maximum delay reached 85.70 ms, which was 40.61% greater than the average value of this group. This indicates the instability of performance when using JSON in the context of WebSocket. The most stable configuration was HTTP-Protobuf, which had minimal latency fluctuations: the standard deviation was 5.04 ms, which was the lowest among all four configurations. Thus, it can be considered not only the fastest, but also the most predictable under heavy load conditions. Comparing the extreme variants – WebSocket-JSON and HTTP-Protobuf – shows a total difference of 24.90 ms, which is an over 55% increase from the best to the worst configuration. This indicator was critical in the construction of real-time telemetry systems, where the accuracy and efficiency of tracing directly affect the quality of the application's adaptive logic. To summarise the time characteristics of telemetry event transmission, the study calculated the average TD values in five typical user interaction scenarios and for four trace channel configurations. The results showed systematic differences between combinations of channels and serialisation formats, with the choice of configuration determining the level of transmission efficiency in all scenarios. The generalised values are presented in Table 3.

Table 3. Average TD values by scenarios and trace channels

Scenario	Type of trace	Average	Median	St. deviation	Minimum	Maximum	Quantity
Async	HTTP-JSON	70.32	70.20	5.95	53.30	87.88	500
	HTTP-Protobuf	63.33	63.44	5.91	46.45	80.34	500
	WebSocket-JSON	84.98	84.78	5.90	66.94	108.65	500
	WebSocket-Protobuf	78.09	78.28	5.79	60.53	94.70	500
Click	HTTP-JSON	70.21	70.17	5.95	55.46	93.12	500
	HTTP-Protobuf	63.01	63.04	5.86	45.17	78.61	500
	WebSocket-JSON	85.01	84.96	6.01	67.88	104.32	500
	WebSocket-Protobuf	78.07	78.28	5.93	61.14	93.89	500
Form	HTTP-JSON	70.06	69.89	5.97	53.71	86.73	500
	HTTP-Protobuf	62.83	62.85	5.88	46.20	80.71	500
	WebSocket-JSON	85.14	85.01	5.91	66.88	103.94	500
	WebSocket-Protobuf	78.12	78.34	5.85	59.42	94.31	500
Navigation	HTTP-JSON	70.13	70.19	5.94	52.71	88.22	500
	HTTP-Protobuf	63.18	63.24	5.93	45.36	81.33	500
	WebSocket-JSON	85.03	85.22	5.87	67.55	103.74	500
	WebSocket-Protobuf	77.98	78.07	6.04	59.69	95.47	500
Scroll	HTTP-JSON	70.15	70.21	5.92	50.94	91.88	500
	HTTP-Protobuf	63.26	63.31	5.90	45.61	81.06	500
	WebSocket-JSON	88.11	88.04	5.92	69.45	109.23	500
	WebSocket-Protobuf	78.15	78.31	5.88	59.33	94.82	500

Source: compiled by the authors

The results presented in Table 3 demonstrate systemic differences in TD depending on the trace protocol used. The comparison between WebSocket and HTTP channels revealed a clear pattern: in all five behavioural scenarios – Click, Scroll, Form, Navigation, and Async – WebSocket-based configurations (both with JSON and Protobuf) had significantly greater average TD values than HTTP implementations. The most pronounced differences were observed in the WebSocket-JSON groups: the difference with HTTP-Protobuf reached more than 24 ms in the Scroll scenario (88.11 ms vs. 63.26 ms), which confirms the stable lag of the reactive channel even in simple event structures. Even WebSocket-Protobuf, which was the more optimised option, consistently showed delays 14-16 ms greater than its HTTP counterparts. This indicates that the structural delays associated with the WebSocket protocol outweigh the benefits of binary serialisation.

The greatest average transfer latency was recorded in the WebSocket-JSON configuration for the Scroll

scenario – 88.1 ms. For comparison, the lowest average TD was observed in the HTTP-Protobuf configuration for the Form scenario – only 62.83 ms. Thus, the absolute difference between the worst and the best scenario exceeded 25 ms. In relative terms, this is over 40% of the increase in latency, which was critically significant in the context of real-time telemetry. An analogous trend was observed in the Navigation and Async scenarios, which are complex in terms of client logic load: it was in these scenarios that the reactive WebSocket model showed the greatest instability and delays. Another aspect of the analysis was the stability of transmission within each configuration. The standard deviation values for WebSocket implementations were on average 0.2-0.3 ms greater than HTTP, especially in the Scroll and Async scenarios. This may indicate greater variability in WebSocket performance in response to the variable complexity of the scenarios. Additionally, the maximum TD values for WebSocket-JSON in some scenarios reached 108 ms, which is almost 20 ms more than the maximum values in the HTTP-Protobuf group. This feature indicates potential "latency spikes" that could be caused by client event stack overload or delays at the demarcation layer.

The serialisation format also substantially affected the results. Using Protobuf reduced the average latency by about 6.5-7 ms within the same protocol. For example, in the Async scenario, the TD for WebSocket-JSON was 84.98 ms, and for WebSocket-Protobuf – 78.09 ms. For HTTP, the trend is analogous: 70.32 ms for HTTP-JSON versus 63.33 ms for HTTP-Protobuf. The effect of the format was especially noticeable in scenarios with a high number of small events, where each byte of savings directly affected the overall transfer time. The reduction in latency when using Protobuf can be attributed to the lower frame weight and faster deserialisation, which is especially relevant for environments with limited computing resources, such as mobile device browsers. Notably, no scenario did WebSocket configurations manage to outperform HTTP in any time indicator – neither average, nor minimum, nor maximum. This means that regardless of the idealised advantage of WebSocket in the form of a constant connection, the factual performance of this channel under simulated multi-event load stayed lower. This was caused by the complexity of connection management, the influence of asynchronous tasks, the accumulation of events in the execution cycle, and limitations at the level of browser APIs for telemetry processing. Thus, the results of the experiment clearly indicate the feasibility of using buffered HTTP POST in combination with Protobuf for critical telemetry applications. This configuration provides the lowest latency, stability, predictability, and less variability, making it the most suitable for integration into reactive single-page application architectures focused on scalable real-time analytics of user behaviour. To check the possible statistical relationship between EDL and TD, a correlation analysis was performed for each trace configuration. The results showed that there is no significant linear relationship between these indicators, which indicates that the mechanisms for detecting events in the browser are independent of the mechanisms for transmitting them via OpenTelemetry Protocol (OTLP) channels. The generalised values are presented in Table 4.

Table 4. Correlation between EDL and TD by trace type

Trace type	EDL-TD correlation coefficient
HTTP-JSON	-0.010
HTTP-Protobuf	+0.010
WebSocket-JSON	-0.011
WebSocket-Protobuf	-0.003

Source: compiled by the authors

The results of the correlation analysis, according to which the Pearson coefficients between the EDL event detection delay and the TD transmission delay range from -0.011 to +0.010, suggest that in the tested telemetry channel configurations, EDL and TD were independent time indicators. This means that the process of detecting and recording an event in the client environment is independent of the stage of transferring the collected data to the server infrastructure using the OTLP protocol. This autonomy was expected, considering the architecture of the OpenTelemetry SDK, where event processing (via PerFormanceObserver, MutationObserver, or DOM Events API) and telemetry packet generation are separated in space and time (Maltsev & Amin, 2024).

The correlation values were consistently close to zero in all four trace configurations – HTTP-JSON, HTTP-Protobuf, WebSocket-JSON, and WebSocket-Protobuf. This suggests that neither the type of protocol (buffered HTTP or reactive WebSocket) nor the serialisation format (textual JSON or binary Protobuf) affects the nature of the relationship between EDL and TD. All this supports the assumption that these two time components belong to independent phases of event processing: the first is the client's response to the interaction, and the second is the logistics process of delivering telemetry to the backend. The situation does not change even when the sample is expanded or the scenario context is shifted: in all cases, the stability of correlation independence stays high.

Such isolation can be viewed as an architectural advantage: on the one hand, it ensures the stability of event detection regardless of communication channels,

and on the other hand, it allows scaling transmission systems without the risk of affecting the collection phase. This is especially relevant under real-world load conditions, where the client part may have access to a high-priority event stream (e.g., in cases of fast scrolling or navigation), and server channels may be overloaded or delayed (due to temporary unavailability of the Collector or delay in OTLP processing) (Amirkhanov *et al.*, 2025). ANOVA was used to test the statistical significance of the differences in EDL and TD between the different trace configurations. The results presented in Table 5 showed high statistical significance for both parameters, which confirms the systemic effect of the protocol type and serialisation format on the timing characteristics.

Table 5. Results of analysis of variance (ANOVA) for EDL and TD

Indicator	F-statistic	p-value	Statistically significant difference ($p < 0.05$)
Event Detection Latency (EDL)	13,821.05	< 0.000001	Yes
Transmission Delay (TD)	6,285.98	< 0.000001	Yes

Source: compiled by the authors

The results of the one-factor analysis of variance presented in Table 5 demonstrate that there are undoubtedly statistically significant differences in the mean values of both EDL and TD between the trace configurations. The F-statistics values of 13,821.05 for EDL and 6,285.98 for TD were extremely high, reflecting significant variance between groups against the background of minimal intra-group variability. Therewith, the value of $p < 0.000001$ indicates that the probability of such differences occurring by chance is almost zero. Formally, this means that although all four tracing configurations (HTTP-JSON, HTTP-Protobuf, WebSocket-JSON, WebSocket-Protobuf) belong to the same telemetry architecture, each of them forms a unique profile of time behaviour, which is confirmed by statistical calculations. This is most true for EDL, where the F-statistic is more than twice as high as for TD, indicating even greater sensitivity of event detection to configuration changes in protocols and serialisation. From the standpoint of analytical interpretation, this means that each of the two indicators (EDL, TD) not only differs in value within individual configurations but also responds statistically significantly to changes in protocol type (HTTP vs. WebSocket) and transmission format (JSON vs. Protobuf). For example, the use of Protobuf within an HTTP configuration not only reduces the average TD value but also forms a separate, statistically distinct group. Analogously, WebSocket-JSON demonstrates the greatest values of both EDL and TD, indicating its statistical isolation within the model.

In practical terms, this means that changing the configuration of a telemetry link affects performance not only in an absolute sense, but also in a statistically proven sense. Thus, the choice of trace configuration should be based not only on engineering or architectural considerations, but also on formal statistical analysis that confirms the validity of the advantages of certain solutions over others. Within the framework of the present study, the HTTP-Protobuf configuration was the best choice in terms of minimising time losses, as it provided the lowest average values for both Event Detection Latency (45.09 ms) and Transmission Delay (63.33 ms). Furthermore, this configuration was characterised by the lowest variability: the standard deviation was only 5.04 ms for EDL and 5.91 ms for TD, and the maximum values did not exceed 61.43 ms and 80.34 ms, respectively. In practical terms, this means that even in peak cases, HTTP-Protobuf stayed within the lower range of all tested configurations. On the opposite pole was the WebSocket-JSON configuration, which showed the most unfavourable profile. The average values for this group were 69.99 ms for EDL and 84.98 ms for TD, which were 24.9 ms and 21.8 ms greater than the best HTTP-Protobuf results, respectively. The standard deviations were also greater (5.04 ms for EDL and 5.90 ms for TD), with maximum values of 85.70 ms for EDL and 108.65 ms for TD, reflecting peak latencies more than 1.5 times greater than in the best case configuration. Such differences, confirmed by high F-statistics values (13,821.05 for EDL and 6,285.98 for TD at $p < 0.000001$), indicate statistically significant isolation of the groups. These conclusions helped to recommend the introduction of time performance metrics in evaluating telemetry architectures and substantiate the need for statistical validation even in cases where the absolute values appear comparable at first glance. After all, only multivariate analysis can reveal the structural stability of differences and provide reliable recommendations for designing monitoring systems in SPA applications.

DISCUSSION

The experiment results showed that HTTP-Protobuf provided the lowest average EDL values, while WebSocket-JSON showed significantly higher delays. This comparison confirmed the significance of the trade-off between responsiveness and stability, which was consistent with the findings of W. Huang *et al.* (2022) in a study with high-frequency tracing. Authors noted that excessive reactivity often led to increased processing latency. This comparison helped to emphasise that the combination of HTTP with binary format was more efficient, as it allowed optimising both speed and variability. The above study demonstrated a statistically significant difference between the transmission channels, which was confirmed by the high F-statistics and low p-values within the ANOVA. This was in line with

the findings of M. Yang *et al.* (2020) and P. Steinmann *et al.* (2020), where M. Yang *et al.* (2020) emphasised the impact of asynchrony on the web stack, particularly on the event handling mechanisms in the browser environment, while P. Steinmann *et al.* (2020) highlighted the benefits of a controlled buffered HTTP approach in the context of ensuring stable time performance. These generalisations helped to expand the understanding of how architectural decisions at the protocol level interacted with internal serialisation mechanisms, creating a cumulative effect in latency. Thus, it was the relevant interaction between the protocol and the serialisation format that formed the super-analytical basis for making architecturally sound decisions aimed at minimising time losses and increasing the reliability of real-time telemetry systems.

The comparison with J. Zhang *et al.* (2024) supported the idea that binary formats were more efficient due to the lower weight of the transmitted data, the optimised serialisation process, and the lesser amount of parsing during deserialisation. In contrast, C. Lin *et al.* (2022) put forward an alternative view on their complexity in supporting, emphasising that JSON provided better debugging, compatibility with different environments, and easier integration into existing telemetry solutions. Authors also addressed the greater clarity of the data structure in JSON, which could be critical during development or in cases of incident analysis. However, this study showed that even with the higher simplicity of JSON, the latency was substantially greater, which was substantiated by a calculated difference of more than 4 ms in different configurations. Furthermore, this difference was maintained regardless of the type of transport protocol, which indicated a fundamental advantage of Protobuf in the context of high-frequency telemetry. Thus, the choice of serialisation format had to consider not only the ease of implementation, but also objective time characteristics that could critically affect real-time performance.

Within the framework of the TD analysis, a systematic lag of WebSocket configurations was recorded. This was in contrast to the findings of J. Zhang *et al.* (2020) and Y. Tian *et al.* (2024), who claimed minimal latency of WebSocket connections in short-lived sessions. However, these findings showed that with an increase in the number of events, WebSocket efficiency decreased, which was explained by the additional costs of the asynchronous processing cycle. Thus, the data should have changed the perception of WebSocket as a universal solution. The present study also demonstrated greater instability of WebSocket performance patterns focused on Scroll and Async scenarios. This conclusion was in line with the study by A.W. Mbugua *et al.* (2020) but was refuted by Z. Mengjiao *et al.* (2024), who insisted on the stability of WebSocket-responsive channels. The comparison helped to assert that under high load conditions, the WebSocket channel was unable to provide the expected level of predictability, while HTTP-based channels demonstrated greater stability.

The observed absolute differences between HTTP-Protobuf and WebSocket-JSON, which reached more than 25 ms in TD, refleted a critical dependence of performance on configuration. Such differences not only demonstrated a different latency profile but also highlighted a systemic difference in the approaches to processing telemetry events. These results were consistent with the findings of N. Keerativoranan *et al.* (2024), W. Zhou *et al.* (2025), and Z. Zheng *et al.* (2017). N. Keerativoranan *et al.* (2024) emphasised the significance of real-time latency, noting that even a slight increase in latency could lead to a decrease in the accuracy of behavioural analytics. W. Zhou *et al.* (2025) analysed the architectural trade-offs, noting specifically that the choice in favour of WebSocket was appropriate only if the load was stable and there was no competition in the execution environment. Z. Zheng *et al.* (2017) added that every millisecond was critical in scalable SPA applications, especially when it came to complex user scenarios with a high frequency of events. In this context, additional latency could not only affect the system's response time but also lead to a distortion of the analytical model based on time characteristics. The generalisation helped to state that the technical solution should have focused on the ratio of resource cost to performance, as well as the ability to ensure the stability and accuracy of telemetry transmission, which was confirmed by the results of the experiment.

The analysis of the correlation between EDL and TD showed their independence, which indicated that there was no direct linear relationship between the time of event detection and its subsequent transmission to the server. This correlated with the studies by L. Xiong *et al.* (2021) and L. Shuangde *et al.* (2021), who previously assumed a two-phase model of event processing, where the phase of registering interactions in the client environment was clearly separated from the phase of delivering information through the OTLP protocol. D. Salomon *et al.* (2021) and C. Eder *et al.* (2023) suggested possible indirect relationships through the frequency of events, the probable impact of telemetry frame density on their collection time, or mutual blocking of queues. However, the results showed the absence of a linear relationship, which strengthened the point industry model and demonstrated the stable statistical isolation of the phases. Thus, it was confirmed that in the OpenTelemetry architecture, the collection and transmission phases were statistically autonomous even under conditions of high event variability, which increased the reliability and flexibility of the system under variable load conditions.

Zero correlation coefficients supported the findings of G. Perin *et al.* (2022) and R. Li *et al.* (2023), who emphasised the structural disconnect between the processing and transmission phases of telemetry events in the browser environment. However, this contradicted the authoritative remarks of J. Zhang & X. Wu (2024), who suggested that the narrowness of buffers could

lead to the accumulation of delays within JavaScript loops, which created conditions for the emergence of back pressure on the event detection phase. The data obtained on a large sample showed that such scenarios were possible only under critical loads with a high event frequency or when the permissible capacity of the internal buffer was exceeded. The lag interaction in such cases could be manifested in the form of temporary delays in deserialisation or subsequent send calls, but in the typical mode of operation, this interdependence was not observed. This required further research into cross-correlation effects, especially in cases of peak loads and unpredictable increases in event density. The comparison showed that under normal conditions, phase independence was maintained, ensuring processing stability, but that potential non-functional dependencies that could affect the overall dynamics of the telemetry channel should be considered at high-frequency flows.

Using ANOVA methods, it was confirmed that the differences between the groups were statistically significant for both indicators. The analysis was consistent with the findings of T. Deeter *et al.* (2021) and Q. Li *et al.* (2023) regarding the need for formal validation of technical solutions. T. Deeter *et al.* (2021) noted that the practical effect could be hidden without proper testing, while Q. Li *et al.* emphasised the need for careful statistical processing. This comparison confirmed that HTTP-Protobuf formed a separate, visible group, while WebSocket-JSON was an outsider in this context. Overall, the results emphasised that the choice of trace configuration should be based not only on intuitive assumptions but also on evidence-based statistics. According to A. Nõu *et al.* (2025), the use of time performance metrics should not be an optional practice, but a mandatory component of the architectural process. The comparative analysis showed that HTTP-Protobuf provided the most suitable model for scalable SPA systems. In summary, the discussion above revealed that the HTTP-Protobuf combination was the most efficient and predictable configuration for demanding telemetry in SPA applications due to the lowest EDL and TD times, stable standard deviation, and statistically proven benefits. The results demonstrated a clear architectural validity of the choice and also substantiated the inclusion of time performance metrics in the evaluation process.

CONCLUSIONS

The study of EDL and TD in client telemetry systems for SPA applications revealed the critical importance of the choice of trace protocol and serialisation format. Among the four tested configurations (HTTP-JSON, HTTP-Protobuf, WebSocket-JSON, WebSocket-Protobuf), the best time performance was demonstrated by the HTTP-Protobuf combination, which provided the lowest average values of both EDL (45.09 ms) and TD (62.83-63.33 ms) with a minimum standard deviation. Instead, the WebSocket-JSON configuration turned out to be the worst in all respects, demonstrating not only high average values (EDL = 69.99 ms; TD > 85 ms), but also significant variability and instability of transmission, which indicates that this configuration is less suitable for high-load interaction scenarios.

Quantitative analysis revealed that WebSocket implementations are on average 20 ms slower than HTTP alternatives (EDL) and up to 25 ms slower (TD) within the same scenarios. Therewith, there is a stable tendency to improve performance when using the Protobuf format, which provided a 4.85-7 ms delay reduction compared to JSON. This confirms the effectiveness of binary serialisation in conditions of high-frequency interaction, where even a slight time saving has a critical impact on system performance. In the context of scenarios (Click, Scroll, Form, Navigation, Async), the HTTP-Protobuf configuration consistently demonstrated the lowest TD values, particularly 62.83 ms in the Form scenario, while WebSocket-JSON consistently ranked the worst. The results of the correlation analysis showed no linear relationship between EDL and TD (Pearson's coefficients ranging from -0.011 to +0.010), which indicates the practical independence of the event detection and transmission phases. This confirms the architectural advantage of the OpenTelemetry SDK, where event detection and data transmission are implemented independently. The ANOVA confirmed statistically significant differences between all trace configurations ($p < 0.000001$), with a particularly pronounced sensitivity of the EDL indicator to changes in protocol and serialisation format. The F-statistic for EDL was 13,821.05, and for TD -6,285.98, reflecting a significant variance between groups against the background of insignificant intra-group variability.

Limitations of the study include the modelling of the client environment in a laboratory environment with a controlled load. In real-world scenarios, performance may depend on a series of external factors: network delays, browser limitations, client device performance, and unpredictable user behaviour. In this regard, a reasonable area for future research is to expand the analytical framework by using cross-correlations, analysing lag structures, the impact of event density, and studying the nonlinear relationships between the EDL and TD phases. This will help to better understand the dynamics of telemetry systems and develop even more effective approaches to their optimisation.

ACKNOWLEDGEMENTS

None.

FUNDING

None.

CONFLICT OF INTEREST

None.

REFERENCES

[1] Amirkhanov, B., Amirkhanova, G., Kunelbayev, M., Adilzhanova, S., & Tokhtassyn, M. (2025). Evaluating HTTP, MQTT over TCP and MQTT over WEBSOCKET for digital twin applications: A comparative analysis on latency, stability, and integration. *International Journal of Innovative Research and Scientific Studies*, 8(1), 679-694. doi: 10.53894/ijirss.v8i1.4414.

[2] Deeter, T., Green, D.H., Kidwell, S., Kane, T.J., Donnal, J.S., Vasquez, K., Sievenpiper, B., & Leeb, S.B. (2021). Behavioral modeling for microgrid simulation. *IEEE Access*, 9, 35633-35645. doi: 10.1109/access.2021.3061891.

[3] Donta, P.K., Srirama, S.N., Amgoth, T., & Annavarapu, C.S.R. (2021). Survey on recent advances in IoT application layer protocols and machine learning scope for research directions. *Digital Communications and Networks*, 8(5), 727-744. doi: 10.1016/j.dcan.2021.10.004.

[4] Eder, C., Winzinger, S., & Lichtenthäler, R. (2023). A comparison of distributed tracing tools in serverless applications. In *2023 IEEE international conference on service-oriented system engineering (SOSE)* (pp. 98-105). Athens: Institute of Electrical and Electronics Engineers. doi: 10.1109/SOSE58276.2023.00018.

[5] Faizulin, O., & Nazarkevych, M. (2024). Methods and means of analyzing application security via distributed tracing. *Journal of Lviv Polytechnic National University "Information Systems and Networks"*, 16, 69-87. doi: 10.23939/sisn2024.16.069.

[6] Fernández, F., Zverev, M., Garrido, P., Juárez, J.R., Bilbao, J., & Agüero, R. (2021). Even lower latency in IIoT: Evaluation of QUIC in industrial IoT scenarios. *Sensors*, 21(17), article number 5737. doi: 10.3390/s21175737.

[7] Huang, C., *et al.* (2022). Artificial intelligence enabled radio propagation for communications – part II: Scenario identification and channel modeling. *IEEE Transactions on Antennas and Propagation*, 70(6), 3955-3969. doi: 10.1109/tap.2022.3149665.

[8] Huang, W., Zhang, L., Wu, H., Min, F., & Song, A. (2022). Channel-Equalization-HAR: A light-weight convolutional neural network for wearable sensor based human activity recognition. *IEEE Transactions on Mobile Computing*, 22(9), 5064-5077. doi: 10.1109/tmc.2022.3174816.

[9] Hunko, I. (2025). Adaptive approaches to software testing with embedded artificial intelligence in dynamic environments. *International Journal of Current Science Research and Review*, 8(5), 2036-2051. doi: 10.47191/ijcsrr/v8-i5-10.

[10] Iori, F., Perovic, G., Cini, F., Mazzeo, A., Falotico, E., & Controzzi, M. (2023). DMP-based reactive robot-to-human handover in perturbed scenarios. *International Journal of Social Robotics*, 15(2), 233-248. doi: 10.1007/s12369-022-00960-4.

[11] Jackson, S., Cummings, N., & Khan, S. (2024). Streaming technologies and serialization protocols: Empirical performance analysis. *ArXiv*. doi: 10.48550/arXiv.2407.13494.

[12] Kaliiukh, Iu., *et al.* (2025). Application of Digital Twins and IoT for investigating damage caused to buildings under dynamic influences. In *Proceedings of the fib symposium in Antibes* (pp. 3069-3073). Antibes: fib. The International Federation for Structural Concrete.

[13] Keerativoranan, N., Saito, K., & Takada, J. (2024). Grid-based channel modeling technique for scenario-specific wireless channel emulator based on path parameters interpolation. *IEEE Open Journal of the Communications Society*, 5, 1724-1739. doi: 10.1109/ojcoms.2024.3373538.

[14] Kolodii, R. (2024). Unpacking Russia's cyber-incident response. *Security Studies*, 33(4), 640-669. doi: 10.1080/09636412.2024.2391757.

[15] Li, Q., Peng, Z., Feng, L., Liu, Z., Duan, C., Mo, W., & Zhou, B. (2023). ScenarioNet: Open-source platform for large-scale traffic scenario simulation and modeling. In *NIPS '23: Proceedings of the 37th international conference on neural information processing systems* (pp. 3894-3920). New Orleans: Neural Information Processing Systems Foundation, Inc.

[16] Li, R., Sun, J., Xue, J., & Masouros, C. (2023). Scenario-aware learning approaches to adaptive channel estimation. *IEEE Transactions on Communications*, 72(2), 874-889. doi: 10.1109/tcomm.2023.3330878.

[17] Lin, C., He, J., Shen, C., Li, Q., & Wang, Q. (2022). CrossBehaAuth: Cross-scenario behavioral biometrics authentication using keystroke dynamics. *IEEE Transactions on Dependable and Secure Computing*, 20(3), 2314-2327. doi: 10.1109/tdsc.2022.3179603.

[18] Maltsev, E., & Amin, R.U. (2024). Impact of serialization format on inter-service latency. *Advances in Cyber-Physical Systems*, 9(2), 89-94. doi: 10.23939/acps2024.02.089.

[19] Mbugua, A.W., Chen, Y., Raschkowski, L., Thiele, L., Jaeckel, S., & Fan, W. (2020). Review on ray tracing channel simulation accuracy in sub-6 GHz outdoor deployment scenarios. *IEEE Open Journal of Antennas and Propagation*, 2, 22-37. doi: 10.1109/ojap.2020.3041953.

[20] Mengjiao, Z., Yu, L., Jie, H., Ruisi, H., Jingfan, Z., Chongyang, Y., & Chengxiang, W. (2024). Artificial intelligence based multi-scenario mmWave channel modeling for intelligent high-speed train communications. *China Communications*, 21(3), 260-272. doi: 10.23919/jcc.ja.2022-0406.

[21] Nõu, A., Talluri, S., Iosup, A., & Bonetta, D. (2025). Investigating performance overhead of distributed tracing in microservices and serverless systems. In *ICPE '25: Companion of the 16th ACM/SPEC international conference on performance engineering* (pp. 162-166). New York: Association for Computing Machinery. doi: 10.1145/3680256.3721316.
[22] Perin, G., Wu, L., & Picek, S. (2022). Exploring feature selection scenarios for deep learning-based side-channel analysis. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2022(4), 828-861. doi: 10.46586/tches.v2022.i4.828-861.
[23] Salomon, D., Weiss, A., & Levi, I. (2021). Improved filtering techniques for single- and multi-trace side-channel analysis. *Cryptography*, 5(3), article number 24. doi: 10.3390/cryptography5030024.
[24] Sanchez, P.M.S., Valero, J.M.J., Celdran, A.H., Bovet, G., Perez, M.G., & Perez, G.M. (2021). A survey on device behavior fingerprinting: Data sources, techniques, application scenarios, and datasets. *IEEE Communications Surveys & Tutorials*, 23(2), 1048-1077. doi: 10.1109/comst.2021.3064259.
[25] Shuangde, L., Yuanjian, L., Leke, L., Xijia, B., Guyue, Z., Yaxin, Y., Anwen, R., & Qi, S. (2021). Millimeter wave channel characteristics of outdoor microcellular based on improved ray tracing method and BP neural network algorithm. *Chinese Journal of Radio Science*, 36(3), 430-442. doi: 10.12265/j.cjors.2020217.
[26] Steinmann, P., Auping, W.L., & Kwakkel, J.H. (2020). Behavior-based scenario discovery using time series clustering. *Technological Forecasting and Social Change*, 156, article number 120052. doi: 10.1016/j.techfore.2020.120052.
[27] Thakur, A., & Chandak, M.B. (2022). A review on opentelemetry and HTTP implementation. *International Journal of Health Sciences*, 6(S2), 15013-15023. doi: 10.53730/ijhs.v6ns2.8972.
[28] Tian, Y., Li, H., Zhu, Q., Mao, K., Ali, F., Chen, X., & Zhong, W. (2024). Generative network-based channel modeling and generation for air-to-ground communication scenarios. *IEEE Communications Letters*, 28(4), 892-896. doi: 10.1109/lcomm.2024.3363621.
[29] Wang, C., Lv, Z., Gao, X., You, X., Hao, Y., & Haas, H. (2022). Pervasive wireless channel modeling theory and applications to 6G GBSMs for all frequency bands and all scenarios. *IEEE Transactions on Vehicular Technology*, 71(9), 9159-9173. doi: 10.1109/tvt.2022.3179695.
[30] Xiong, L., Yao, Z., Miao, H., & Ai, B. (2021). Vehicle-to-vehicle channel characterization based on ray-tracing for urban road scenarios. *Wireless Communications and Mobile Computing*. doi: 10.1155/2021/8854247.
[31] Yang, M., *et al*. (2020). Machine-learning-based scenario identification using channel characteristics in intelligent vehicular communications. *IEEE Transactions on Intelligent Transportation Systems*, 22(7), 3961-3974. doi: 10.1109/tits.2020.3001132.
[32] Zhang, J., & Wu, X. (2024). An anti-jamming game between dynamically-sensing jammer and legitimate user with faking-slot transmission. *IEEE Transactions on Vehicular Technology*, 73(7), 10287-10300. doi: 10.1109/tvt.2024.3372969.
[33] Zhang, J., Lin, J., Tang, P., Fan, W., Yuan, Z., Liu, X., Xu, H., Lyu, Y., Tian, L., & Zhang, P. (2024). Deterministic ray tracing: A promising approach to THZ channel modeling in 6G deployment scenarios. *IEEE Communications Magazine*, 62(2), 48-54. doi: 10.1109/mcom.001.2200486.
[34] Zhang, J., Liu, L., Fan, Y., Zhuang, L., Zhou, T., & Piao, Z. (2020). Wireless channel propagation scenarios identification: A perspective of machine learning. *IEEE Access*, 8, 47797-47806. doi: 10.1109/access.2020.2979220.
[35] Zheng, Z., Trivedi, K.S., Wang, N., & Qiu, K. (2017). Markov regenerative models of webservers for their user-perceived availability and bottlenecks. *IEEE Transactions on Dependable and Secure Computing*, 17(1), 92-105. doi: 10.1109/tdsc.2017.2753803.
[36] Zhou, W., Borjigin, A., & He, C. (2025). Behavioral Universe Network (BUN): A behavioral information-based framework for complex systems. *ArXiv*. doi: 10.48550/arXiv.2504.15146.

Реактивне трасування поведінкових сценаріїв в односторінкових додатках через інтеграцію Bun-базованих WebSocket-каналів та OpenTelemetry

Владислав Ананченко

Аспірант
Міжнародний економіко-гуманітарний університет імені академіка Степана Дем'янчука
33027, вул. Степана Дем'янчука, 4, м. Рівне, Україна
https://orcid.org/0009-0004-8963-775X

Юрій Лотюк

Кандидат педагогічних наук, доцент
Міжнародний економіко-гуманітарний університет імені академіка Степана Дем'янчука
33027, вул. Степана Дем'янчука, 4, м. Рівне, Україна
https://orcid.org/0000-0001-6696-5583

Анотація. Метою дослідження було оцінити часову ефективність реактивного трасування поведінки користувача в односторінкових додатках через інтеграцію WebSocket-каналів на базі Bun з OpenTelemetry. Методологія включала створення прототипу додатка на React, високочастотний моніторинг і агрегування SCADA-даних, побудову та оптимізацію нейромережі 64-32-16 у TensorFlow, симуляції в MATLAB/Simscape, а також статистичний аналіз із використанням регресії Theil-Sen, Seasonal and Trend-декомпозиції, тесту Брауна-Форсайта, двофакторного аналізу варіантів, бутстреп-перестановки, критерія Дікі-Фуллера та кривих виживання Каплана-Мейєра. Результати показали, що комбінація протоколу гіпертекстової передачі із бінарною серіалізацією у форматі Protocol Buffers забезпечила найнижчу затримку виявлення подій, яка становила в середньому 45,09 мілісекунди, та найнижчу затримку передачі, що сягала лише 62,83 мілісекунди у сценарії заповнення форм. У той же час комбінація вебсокетів із текстовим форматом JavaScript Object Notation продемонструвала найвищі показники затримки, із середнім значенням виявлення подій 69,99 мілісекунди та затримкою передачі до 88,1 мілісекунди, а також найбільшу варіативність у часі реакції. Статистичний аналіз підтвердив суттєві відмінності між усіма конфігураціями: результати дисперсійного аналізу виявили надзвичайно високі значення F-статистики для обох показників із рівнем значущості p меншим за 0,000001, що свідчить про реальний вплив як протоколу, так і формату серіалізації на часову ефективність. Додатково встановлено, що затримка виявлення подій та затримка передачі були незалежними величинами, оскільки коефіцієнти кореляції в усіх випадках залишалися близькими до нуля. Таким чином, оптимальною конфігурацією для високочастотних телеметричних систем був гіпертекстовий протокол із бінарним форматом Protocol Buffers, що забезпечує не лише мінімальні часові затримки, але й стабільність у навантажених середовищах. Практична значимість результатів полягає в можливості використання їх інженерами з продуктивності, архітекторами фронтенду та розробниками моніторингових систем для створення ефективних та масштабованих рішень, орієнтованих на аналіз поведінки користувачів у режимі реального часу

Ключові слова: часові затримки; асинхронність; бінарна серіалізація; обробка подій; HTTP-JSON; телеметрична архітектура