Volume XXVI 2023
ISSUE no.2
MBNA Publishing House Constanta 2023

Scientific Bulletin of Naval Academy

SBNA PAPER • OPEN ACCESS

Optimization of sensor data gathering interface for Platmarisc project

To cite this article: A. Atodiresei, A. Băutu and S. Diaconu, Scientific Bulletin of Naval Academy, Vol. XXVI 2023, pg. 136-143.

Submitted: 22.05.2023
Revised: 05.09.2023
Accepted: 10.10.2023

Available online at www.anmb.ro

ISSN: 2392-8956; ISSN-L: 1454-864X

doi: 10.21279/1454-864X-23-I2-016

Optimization of sensor data gathering interface for Platmarisc project

Anca Atodiresei[1], Andrei Băutu[2], and Simona Diaconu[3]

[1] "Mircea cel Bătrân" Naval Academy, Romania, anca.atodiresei@anmb.ro
[2] "Mircea cel Bătrân" Naval Academy, Romania, marian.cata@anmb.ro
[3] Coremar SA, simona.diaconu@coremar.ro

Abstract. Software interfaces enable communication between different system components, regardless of platform, language, or hardware. They integrate existing components and make development easier. Protocols and technologies like XML, JSON, and WebSockets facilitate this communication. As technology evolves, interfaces will become more complex and specialized for real-time communication, particularly in IoT. Effective interfaces are essential for modern software development. In this article, we analyzed various communication protocols considered for the implementation of the Platmarisc project (including REST, XML-RPC, RPC, SOAP, JSON-RPC, GraphQL, HTTP POST, WebSockets, and MQTT).

1. Introduction

Software interfaces are a key element in the development of complex software applications involving multiple systems and services. They allow connection and communication between different system components, regardless of the platform, programming language or hardware they are running on. This makes the development of applications and services easier and more efficient, as it allows the integration of already existing components instead of creating a new solution from scratch.

The protocols and technologies that enable this communication are varied and tailored to the specific needs of applications. These can be protocols based on XML, JSON, HTTP or specialized protocols for real-time communication such as WebSockets or MQTT. There are also various frameworks and libraries that facilitate the creation and implementation of these protocols within applications and services.

As technology continues to evolve, we expect the use and development of software interfaces to become even more important. They will become increasingly complex and efficient, allowing applications and services to interact with more systems and services than ever before. We will also likely see an increase in the use of specialized protocols for IoT devices and other applications that require real-time communication.

In general, software interfaces play a crucial role in the development of modern software applications and are essential to enable efficient and scalable communication between various system components. By using the right protocols and technologies, developers can create powerful and efficient solutions that meet the specific needs of applications and services.

In this article, we'll examine the pros and cons of several communication protocols used in software interfaces, including REST, XML-RPC, RPC, SOAP, JSON-RPC, GraphQL, HTTP POST, WebSockets, MQTT, and the like.

2. XML-based protocols

XML-based protocols have been widely used in the past to enable communication between various applications and services. However, in recent years they have been largely replaced by other lighter and more efficient protocols such as REST and JSON.

Although XML-based protocols are not as popular as they once were, they can still be useful in certain situations, such as when you need to call a method from another system that uses one of these protocols. Also, some older or more complex applications may still use these protocols for communication.

In general, the use of XML-based protocols can be justified in some situations, but in most cases the use of other lighter and more efficient protocols such as REST and JSON is recommended. The following subsections outline the main advantages and disadvantages of the major XML-based protocols, XML-RPC and SOAP,

2.1. XML-RPC

XML-RPC (Remote Procedure Call) is a protocol that allows a remote method to be called via an HTTP call to an XML-RPC server [1][2]. This protocol is simple and easy to implement, but it is not as efficient as other protocols due to the large amount of XML code required to transmit the information [3][4][5][6].

Benefits of XML-RPC include:

• XML-RPC can be used in a variety of programming languages because XML is a standard format that is supported by most programming languages.

• XML-RPC is a simple protocol that does not require much configuration or advanced knowledge to implement.

• XML-RPC can be used to make procedure calls on a server, which makes it ideal for creating applications that need to access various resources on a server.

Disadvantages of XML-RPC include:

• XML-RPC uses XML, which is a rather verbose format and can be difficult to read and process manually.

• XML-RPC is not as flexible as other protocols such as REST because it relies on procedure calls and does not allow for a resource-oriented approach.

• XML-RPC may be less efficient than other protocols such as JSON-RPC because the XML format is larger and more complex than the JSON format.

2.2. SOAP

SOAP (Simple Object Access Protocol) is another XML-based protocol that allows remote methods to be called via an HTTP call to a SOAP server [7]. This is a more complex and powerful protocol than XML-RPC, but it is also more cumbersome due to the large amount of XML code required to transmit the information. Also, SOAP is less flexible than REST or JSON protocols because it is stricter in terms of message format and allowed operations.

Benefits of SOAP include:

• Interoperability: SOAP is a platform and language independent protocol so applications can communicate in a standardized and interoperable way.

• Security: SOAP provides multiple levels of security, such as encryption and authentication, to protect transmitted data.

• Flexibility: SOAP allows the use of a large number of data types and complex operations to perform various functions.

• Transaction Support: The SOAP protocol provides transaction support, thereby allowing applications to work with multiple resources and maintain data consistency.

Disadvantages of SOAP include:

• Complexity: SOAP is a complex protocol and can be difficult to understand and implement.

• Performance: Using the SOAP protocol can affect system performance because there is an overload of network resources and a large amount of data sent.

• File size: SOAP files can be quite large, which can lead to slow response times and network congestion.

• Compatibility issues: Using different versions of the SOAP protocol between applications can lead to compatibility issues and may require additional development and testing efforts to resolve them.

3. JSON-based protocols

JSON-based protocols are very popular due to their ease of use and small data sizes. JSON (JavaScript Object Notation) is a lightweight and flexible data format that can be used in a variety of programming languages.

In general, JSON-based protocols are currently preferred due to their ease of use and small data sizes. They can be used in a variety of applications and are very useful for developing APIs. However, it is important to choose the right protocol based on the specific needs of the application and the desired level of performance.

The following subsections outline the main advantages and disadvantages of the major JSON-based protocols, JSON-RPC and GraphQL,

3.1. JSON-RPC

JSON-RPC is a client-server protocol that uses the JSON format to make procedure calls to a server [8][9]. This protocol is easy to use and does not require much configuration. Another advantage of JSON-RPC is that it can be used in a variety of programming languages, including web, mobile, and desktop [10].

The advantages of the JSON-RPC protocol are:

• Ease of use: JSON-RPC is easy to use and can be implemented in a variety of programming languages.

• Small transmitted data sizes: JSON is a compact and easily parsed data format, which makes data transmission efficient and fast.

• Does not require much configuration: The JSON-RPC protocol does not require much configuration and can be used quickly and easily.

• Compatibility: JSON-RPC can be used in a variety of programming languages and is compatible with many platforms and technologies.

The disadvantages of the JSON-RPC protocol are:

• Lack of security: The JSON-RPC protocol lacks built-in security functionality, which can be a problem for applications that transmit sensitive data.

• Data structure limitations: JSON-RPC works best with relatively simple data, and more complex data structure can be difficult to transmit and manage.

• Lack of standardization: There are several different implementations of the JSON-RPC protocol, which can lead to compatibility issues between different applications and platforms that use the protocol.

3.2. GraphQL

GraphQL is another popular JSON-based protocol [11][12]. This is a query protocol that allows the client to define exactly what data it needs and receive only that data [13][14]. GraphQL can be used in a variety of programming languages and is particularly used to create APIs [15][16][17]. Another benefit of GraphQL is that it allows developers to avoid data overload and improve application performance by reducing the number of server calls [18][22][19][23].

Advantages of the GraphQL protocol include:

• Flexibility and query power: GraphQL offers great flexibility in terms of querying data, allowing the client to ask for only the data it needs without having to receive all the data about a resource.

• Performance: Because GraphQL can be used to request only the necessary data, this protocol can reduce bandwidth and improve application performance.

• Automatic documentation: In GraphQL, documentation is automatically generated based on the defined schema. This makes it easier for developers who want to use the schema because they don't have to manually generate the documentation.

• Compatibility with multiple platforms and programming languages: GraphQL can be used with a variety of programming languages and platforms, making it easy to implement in a wide range of scenarios.

• Scalability: GraphQL can be used to improve application scalability by allowing data to be distributed across multiple servers and processed separately.

Disadvantages of the GraphQL protocol include:

• Learning curve: For developers who have not used GraphQL before, it may be necessary to learn a new set of concepts and syntax.

• Implementation complexity: GraphQL implementation can be more complex than other protocols such as REST.

• Security: Due to the great flexibility offered by GraphQL, it is important to take security measures to protect sensitive data.

• Dependency management: Because of the flexibility offered by GraphQL, it can be difficult to manage dependencies between different resources and ensure proper interconnectivity between them.

4. HTTP-based protocols

HTTP-based protocols are very popular and used worldwide because HTTP is a standard protocol used for communication between servers and clients on the Internet.

In general, HTTP-based protocols are preferred due to the popularity and standardization of this protocol. They are easy to use and can be implemented in a variety of programming languages and platforms. However, it is important to choose the right protocol based on the specific needs of the application and the desired level of performance.

Two of the most popular HTTP-based protocols are REST and HTTP POST. The following subsections outlines their advantages and disadvantages.

4.1. REST

REST (Representational State Transfer) is a protocol that uses standard HTTP methods such as GET, POST, PUT, and DELETE to perform CRUD (Create, Read, Update, and Delete) operations on resources [20][23][21]. REST is an easy-to-use protocol that can be implemented in a variety of programming languages and platforms [22][23][24][25][26]. Another advantage of REST is that it is flexible and can be used to build APIs for different types of applications, such as web or mobile applications [27][28].

The advantages of the REST protocol include:

• Scalability: REST is highly scalable and can handle a large number of requests without affecting system performance.

• Interoperability: REST can be used in a variety of programming languages and platforms, making it a very versatile protocol.

• Ease of use: REST is easy to understand and use, thanks to the use of standard HTTP methods such as GET, POST, PUT and DELETE.

• Flexibility: REST can be used in a variety of scenarios, from creating simple APIs to creating complex applications.

• Cacheable: REST allows users to store resources locally to reduce load time.

• Security: REST provides a variety of security options, including authentication and authorization.

Disadvantages of the REST protocol include:

• Complexity: REST can be complex in certain scenarios, especially when it comes to building complex applications.

• Lack of a standard: REST does not have a defined standard and its implementation may vary from application to application.

• Limited performance: In the case of requests with a lot of data, the performance of the REST protocol can be limited.

• Lack of functionality: REST focuses more on resources rather than actions, which means it is not suitable for all scenarios.

• Vulnerability to attacks: REST can be vulnerable to some attacks, such as Cross-Site Request Forgery (CSRF) or Cross-Site Scripting (XSS) attacks.

4.2. HTTP POST

HTTP POST is another HTTP-based protocol that is used to send data from the client to the server. This protocol is mainly used to send data in an HTML form or to send data via an XMLHttpRequest object in web applications. It is a simple protocol and can be used in a variety of scenarios [25][29][30].

Benefits:

• The HTTP POST (form data) protocol is easy to use and does not require much technical knowledge to implement.

• This protocol is supported by most web servers and can be used in a variety of scenarios.

• It allows sending a large amount of data because the data is sent through the body of the HTTP request.

Disadvantages:

• The HTTP POST (form data) protocol is not ideal for sending sensitive data because the data is transmitted in clear text and can be intercepted by an attacker.

• This protocol is not optimal for sending large files as it may be blocked by servers that have file size limits.

• Using the HTTP POST (form data) protocol can be slower than other protocols because the entire body of the HTTP request must be sent to the server.

5. Protocols for real-time communication

Real-time communication protocols are essential in many modern applications that require two-way communication between client and server in real time. Such apps include multiplayer games, chat apps, monitoring apps and more.

In general, protocols for real-time communication are essential in many modern applications and must be considered when designing an application that requires two-way real-time communication [31][29][30]. The following subsections describe Websockets and MQTT, two protocols which are preferred by software developers whenever real-time communication is considered for an application.

5.1. WebSockets

WebSockets is one of the most popular protocols for real-time communication. It allows a persistent connection between the client and the server, which means that the client and the server can communicate in real time over this connection. WebSockets is a bandwidth-efficient protocol and can be used in a variety of scenarios, including multiplayer games, chat applications, and monitoring applications [32][29][30][33].

Advantages of the WebSockets protocol:

• Two-way real-time communication: WebSockets enable real-time two-way communication between client and server, thus enabling the creation of complex applications such as multiplayer games or real-time collaboration applications.

• Bandwidth efficient: The WebSockets protocol is bandwidth efficient because it allows persistent communication between client and server without the need to send HTTP headers with each request and response.

• Improved security: WebSockets allows the use of SSL/TLS encrypted connections, thus enabling secure communication between client and server.

Disadvantages of the WebSockets protocol:

• Firewalls: WebSockets can be blocked by firewalls, which can prevent users from accessing applications that use this protocol.

• Not optimized for transferring large files: The WebSockets protocol is not optimized for transferring large files because they must be broken into smaller pieces and sent as messages.

• Complexity: The WebSockets protocol can be more complex than other protocols because it requires a special implementation on the server to enable real-time two-way communication.

5.2. MQTT

MQTT is another popular protocol for real-time communication. It is mainly used to send messages between IoT (Internet of Things) devices. MQTT is a very bandwidth-efficient protocol because it uses a publish-subscribe model, meaning that messages are only sent to clients that subscribe to those messages. This protocol can be used in a variety of scenarios, including monitoring and controlling IoT devices [32][33][29][30][33].

The advantages of the MQTT protocol include:

• Power efficiency: MQTT was designed for use in IoT devices with limited power resources. This protocol consumes little power and can be used on batteries for long periods of time.

• Bandwidth efficiency: MQTT uses a publish/subscribe model, which means that only devices that are interested in certain messages receive those messages. This makes MQTT a very bandwidth efficient protocol.

• Scalability: MQTT is designed to be highly scalable. This protocol can be used in systems with thousands or even millions of connected devices.

• Reliability: MQTT has a QoS (Quality of Service) mechanism that guarantees the delivery of messages under reliable conditions. This feature makes MQTT an ideal protocol for reliability-critical applications.

• Security: MQTT supports message authentication and encryption, which makes this protocol safe to use.

Disadvantages of the MQTT protocol include:

• Complexity: MQTT is a fairly complex protocol that can be difficult for inexperienced users to understand.

• Functionality limitations: MQTT is a simple protocol and does not have the same capabilities as other protocols such as HTTP. In certain scenarios, these limitations can be a problem.

• Latency: Because of the way the publish/subscribe model works, there is sometimes some delay between when a device publishes a message and when other subscribed devices receive that message.

6. Conclusions

In this article, we have examined several protocols used in software interfaces, analyzing their advantages and disadvantages. XML-based protocols are more complex, but can be used to connect different software systems. JSON-based protocols are easy to use and can be used to perform client-server and query interactions. HTTP-based protocols are very popular and easy to implement, and real-time communication protocols are especially used for chat applications, multiplayer games, and IoT devices.

In conclusion, the choice of a protocol largely depends on the specific requirements of the application and the ability to implement the protocol in a particular programming language or platform. In general, the best options are those that offer a combination of ease of use, efficiency, and scalability.

Acknowledgements: This work was supported by grant no. 383/390059/04.10.2021, project cod ID / Cod MySMIS: 120201: Innovative integrated maritime platform for real-time intervention through simulated disaster risk management assistance in coastal and port areas – PLATMARISC.

References

[1] Cerami, Ethan, "Web services essentials: distributed applications with XML-RPC, SOAP, UDDI & WSDL", " O'Reilly Media, Inc.", 2002

[2] Dissanaike, Suru; Wijkman, Pierre; Wijkman, Mitra, "Utilizing XML-RPC or SOAP on an embedded system", in 24th International Conference on Distributed Computing Systems Workshops, 2004. Proceedings., pp. 438-440, IEEE, 2004

[3] Bagci, Hakan; Kara, Ahmet, "A Lightweight and High Performance Remote Procedure Call Framework for Cross Platform Communication.", in ICSOFT-EA, pp. 117-124, 2016

[4] Hsu, I-Ching, "XML-based information fusion architecture based on cloud computing ecosystem", in Comput. Mater. Continua, vol. 61, no. 3, pp. 929-950, 2019

[5] Kiraly, Sandor; Szekely, Szilveszter, "Analysing RPC and Testing the Performance of Solutions", in Informatica, vol. 42, no. 4, 2018

[6] Királya, Sándor; Székelyb, Szilveszter; Királyc, Roland; Ballad, Tamás, "Some aspects of using RPC",

[7] Radhakrishna, Shraddha; Nachamai, M, "Performance inquisition of web services using SOAP UI and JMeter", in 2017 IEEE International Conference on Current Trends in Advanced Computing (ICCTAC), pp. 1-5, IEEE, 2017

[8] Dwi, YB; Estri, Shinta, "Multi-Tier Model with JSON-RPC in Telemedicine Devices Authentication and Authorization Protocol", in 2021 7th International Conference on Engineering, Applied Sciences and Technology (ICEAST),

[9] JSON-RPC Working Group, "Json-rpc",

[10] Veldkamp, LS; Olsthoorn, Mitchell; Panichella, A, "Grammar-Based Evolutionary Fuzzing for JSON-RPC APIs", in The 16th International Workshop on Search-Based and Fuzz Testing,

[11] Hartig, Olaf; Pérez, Jorge, "An initial analysis of Facebook's GraphQL language",

[12] Wieruch, Robin, "The Road to GraphQL", in Independently published,

[13] Hartig, Olaf; Pérez, Jorge, "Semantics and complexity of GraphQL", in Proceedings of the 2018 World Wide Web Conference,

[14] Bryant, Mike, "GraphQL for archival metadata: An overview of the EHRI GraphQL API", in 2017 IEEE International Conference on Big Data (Big Data),

[15] Quiña-Mera, Antonio; Fernandez, Pablo; García, José María; Ruiz-Cortés, Antonio, "GraphQL: A Systematic Mapping Study", in ACM Computing Surveys,

[16] Wittern, Erik; Cha, Alan; Davis, James C; Baudart, Guillaume; Mandel, Louis, "An empirical study of GraphQL schemas", in Service-Oriented Computing: 17th International Conference, ICSOC 2019, Toulouse, France, October 28–31, 2019, Proceedings 17,
[17] Mukhiya, Suresh Kumar; Rabbi, Fazle; Pun, Violet Ka I; Rutle, Adrian; Lamo, Yngve, "A GraphQL approach to healthcare information exchange with HL7 FHIR", in Procedia Computer Science,
[18] Taelman, Ruben; Vander Sande, Miel; Verborgh, Ruben, "GraphQL-LD: linked data querying with GraphQL", in ISWC2018, the 17th International Semantic Web Conference,
[19] Brito, Gleison; Mombach, Thais; Valente, Marco Tulio, "Migrating to GraphQL: A practical assessment", in 2019 IEEE 26th International Conference on Software Analysis, Evolution and Reengineering (SANER),
[20] Masse, Mark, "REST API design rulebook: designing consistent RESTful web service interfaces", O'Reilly Media, Inc., 2011
[21] Haupt, Florian; Leymann, Frank; Scherer, Anton; Vukojevic-Haupt, Karolina, "A framework for the structural analysis of REST APIs", in 2017 IEEE International Conference on Software Architecture (ICSA), pp. 55-58, IEEE, 2017
[22] Brito, Gleison; Valente, Marco Tulio, "REST vs GraphQL: A controlled experiment", in 2020 IEEE international conference on software architecture (ICSA),
[23] Seabra, Matheus; Nazário, Marcos Felipe; Pinto, Gustavo, "REST or GraphQL? A performance comparative study", in Proceedings of the XIII Brazilian Symposium on Software Components, Architectures, and Reuse,
[24] Miller, Mark A; Schwartz, Terri; Pickett, Brett E; He, Sherry; Klem, Edward B; Scheuermann, Richard H; Passarotti, Maria; Kaufman, Seth; O'Leary, Maureen A, "A RESTful API for access to phylogenetic tools via the CIPRES science gateway", in Evolutionary Bioinformatics, vol. 11, pp. EBO. S21501, SAGE Publications Sage UK: London, England, 2015
[25] Chen, Xianjun; Ji, Zhoupeng; Fan, Yu; Zhan, Yongsong, "Restful API architecture based on Laravel framework", in Journal of Physics: Conference Series, vol. 910, no. 1, pp. 12016, IOP Publishing, 2017
[26] Rodríguez, Carlos; et. al, "REST APIs: A large-scale analysis of compliance with principles and best practices", in Web Engineering: 16th International Conference, ICWE 2016, Lugano, Switzerland, June 6-9, 2016. Proceedings 16, pp. 21-39, Springer, 2016
[27] Sohan, SM; Maurer, Frank; Anslow, Craig; Robillard, Martin P, "A study of the effectiveness of usage examples in REST API documentation", in 2017 IEEE symposium on visual languages and human-centric computing (VL/HCC), pp. 53-61, IEEE, 2017
[28] Li, Li; Chou, Wu, "Design and describe REST API without violating REST: A Petri net based approach", in 2011 IEEE International Conference on Web Services, pp. 508-515, IEEE, 2011
[29] Kaur, Simran; Khanna, Vandana, "Implementation and Comparison of MQTT, WebSocket, and HTTP Protocols for Smart Room IoT Application in Node-RED", in IoT for Sustainable Smart Cities and Society, pp. 165-193, Springer, 2022
[30] Gemirter, Cavide Balkı; Şenturca, Çağatay; Baydere, Şebnem, "A comparative evaluation of AMQP, MQTT and HTTP protocols using real-time public smart city data", in 2021 6th International Conference on Computer Science and Engineering (UBMK), pp. 542-547, IEEE, 2021
[31] Gupta, Poonam, "A survey of application layer protocols for Internet of Things", in 2021 International Conference on Communication information and Computing Technology (ICCICT), pp. 1-6, IEEE, 2021
[32] Silva, Diego RC; Oliveira, Guilherme MB; Silva, Ivanovitch; Ferrari, Paolo; Sisinni, Emiliano, "Latency evaluation for MQTT and WebSocket Protocols: an Industry 4.0 perspective", in 2018 IEEE Symposium on Computers and Communications (ISCC), pp. 01233-01238, IEEE, 2018
[33] Werlinder, Marcus, "Comparing the scalability of MQTT and WebSocket communication protocols using Amazon Web Services", 2020