A Comparative Qualitative and Quantitative Analysis of the Performance of Security Options for Message Protocols: Fog Computing Scenario

Wesley R Bezerra\textsuperscript{a,*,1} (Phd Candidate), Fernando L Koch\textsuperscript{a,*} (Visiting Researcher) and Carlos B. Westphall\textsuperscript{a,**} (Advisor)

\textsuperscript{a}UFSC - Federal University of Santa Catarina, Campus Universitário - Trindade, Florianópolis/SC - Brazil, 88040-380

ARTICLE INFO

Keywords:
message protocol
authentication
identity management
iot
lpwan

ABSTRACT

We analyze the utilization of publish-subscribe protocols in IoT and Fog Computing and challenges around security configuration, performance, and qualitative characteristics. Such problems with security configuration lead to significant disruptions and high operation costs. Yet, these issues can be prevented by selecting the appropriate transmission technology for each configuration, considering the variations in sizing, installation, sensor profile, distribution, security, networking, and locality. This work aims to present a comparative qualitative and quantitative analysis around diverse configurations, focusing on Smart Agriculture's scenario and specifically the case of fish-farming. As result, we applied a data generation workbench to create datasets of relevant research data and compared the results in terms of performance, resource utilization, security, and resilience. Also, we provide a qualitative analysis of use case scenarios for the quantitative data produced. As a contribution, this robust analysis provides a blueprint to decision support for Fog Computing engineers analyzing the best protocol to apply in various configurations.

1. Introduction

The utilization of publish/subscribe protocols \cite{1, 2} (such as AMQP, MQTT, STOMP, and CoAP) are increasingly popular in scenarios which involve distributed time-sensitive processing applications. The usage of these protocols is also existent in scenarios such as Smart Agriculture, Smart Building, and Healthcare among others. Their configurations typically depend on fast communications between nodes, one-to-many aspects of signaling, event management, state synchronization, and bulk transfers in a net-centric distributed environment.

However, there are significant security and vulnerability issues due to missing configuration and installation issues. For instance, poor practices including the use of standard passwords, generic configurations, and lack of appropriate security mechanisms are standard in these configurations \cite{3, 4, 5}. In general, more sophisticated mechanisms require larger computing capacity (such as processing, memory, and power). Engineers must measure the trade-offs between performance, security, and expected device cost while designing secure IoT devices and deploying Fog Computing configurations.

Consequently, in order to better understand the relationship between the project variables impacted by security options, it is of fundamental importance to fully evaluate the existing security options for the message protocols. As message protocols are the necessary support for transmission, a deepened understanding of their characteristics, both qualitatively and quantitatively, is necessary for the selection of adequate security for applications and research in the area.

Even though there are significant and thorough studies on the analysis of message protocols \cite{6, 7, 8} and advances in the area of identification \cite{9, 10, 11}, an update is necessary in order to take into consideration recent developments in this area. Such consideration of the analysis demands a robust bibliographic base to support the qualitative component of the analysis and experimentation using methods which take into account the particularities of each implemented protocol.

\textsuperscript{*}Corresponding author

\textsuperscript{**}Principal corresponding author

\texttt{wesleybez@gmail.com} (W.R. Bezerra); \texttt{fkoch@acm.org} (F.L. Koch); \texttt{westphal@inf.ufsc.br} (C.B. Westphall)

ORCID(s): 0002-6098-7172 (W.R. Bezerra); 0001-7136-3253 (F.L. Koch); 0002-5391-7942 (C.B. Westphall)

WB Bezerra et al.: Preprint submitted to Elsevier
A comparative qualitative and quantitative analysis is being introduced to support choices in security mechanisms in publish/subscribe (pub/sub) protocols. Our experimentation scenario was designed around diverse configurations of IoT in Smart Agriculture, focusing on the case of Fish Farming, ranging from few sensors in a local configuration to a distributed scenario and a configuration with many sensors. This Fish Farming scenario was selected due to its relevance to the local productive arrangement (in the State of Santa Catarina) and promoting the availability of research and technology for professionals, such as researchers, developers, small businesses and small producers, working in their respective area. This study provides the following contributions to the field:

• a qualitative and quantitative assessment of protocols in terms of performance, resource utilization, and security;

• analysis of diverse use case scenarios in the fish-farming application domain and applying IoT to monitor and compare the experiments’ performance with each scenario.

This study also consists of a second section with a theoretical review of the theme and related work presentations. Following this, in the third section, is an experiment presented in order to compare the pub/sub message protocols and analysis on the proposed scenarios’ point of view. In the fourth section, the results obtained are commented on and analyzed. Lastly, the conclusion and future work lists the objectives achieved in the work and proposes future work to be carried out on the theme.

2. Background and Related Work: A Qualitative Analysis

Unlike the internet, the IoT has its layered organization model, see Figure 1. The model has five layers instead of four, as in the web. Due to the IoT devices’ peculiarities, the two lower layers (physical and link) are distinct, which does not happen in the web model. The transport takes place basically over TCP and UDP. However, the data flows as messages through a message protocol implemented in the application layer.

The message protocols can be implemented through middleware, such as stand-alone services and clients, or implemented through both servers and customized clients. In the first scenario, we can use an abstraction layer to access data and devices, as in the example of DDS, which provides middleware for real-time data-centric messages. In the second scenario, some steps require the installation, configuration, and access to an existing server, such as a broker or server, according to the chosen architecture. The latter case would involve the incorporation of the server or client into a customized application without installing third-party software, using only the libraries which implement the protocol.

In regards to architecture, a message protocol can be classified as either: (i) client/server or (ii) publish/subscribe (pub/sub). Client/server protocols (i) require the information consumer to access the server and check its value, checking if this value has changed since the last request. In the pub/sub protocols, the client does not need to connect and check new messages. In this case, customers subscribe to the service, and if there is a change in the data, the protocol will inform them. It shows that is possible to have this same signature type in some protocols that implement the observer design pattern even in the client/server. Both pub/sub and protocols with observers allow loosely coupled parts of the system.

A typical pub/sub system is provided via the broker and organized around the concept of publisher and subscriber objects; publisher/subscriber design pattern. These objects establish logical data streams typically identified by a topic – a string such as “boiler_1_temp”. Communication occurs between the publishers producing topic “issues” and subscribers who registered an interest in that topic.

2.1. IoT Messaging Protocols

One dilemma with IoT involves the bottleneck generated by the communication between the sensors and the cloud. In a system that uses CC, the sensor can communicate directly with the cloud or with an intermediary Gateway (GW), which acts as a bridge between sensors and the internet[12, 13, 14]. GW’s responsibility is to centralize data and provide a connectivity capability, which is unreachable for some devices due to project budget constraints or other restrictions that limit connectivity[15, 16].

This scenario emerges due to the generalization of both CC and FC. In the case of Fog Computing, the GW must communicate with the FN (not directly with the CC), this being the main difference. In this case, there are two communication segments: sensor for gateway (n1) and gateway for Fog (n2). In the first segment (n1), the sensor would
Figure 1: IoT × Web × OSI models - in the center is positioned the OSI reference model which serves as the basis for comparison to the Web model on the left, which is the most simplified of the three, and to the IoT model, which is on the right-side of the figure. It’s noteworthy that the IoT model shares the four lower layers with the OSI reference.

have a wireless link with less bandwidth capacity (LPWAN\(^1\)). Such characteristics are essential for the evaluation and the experiments’ results. In the second case (n2), the link may be a wired network, WiFi, or any other type of protocol accessible by more modest devices and would have more bandwidth capability.

Message protocols are classified in different ways: URI\(^2\) or topics\(^3\); the QoS in each protocol, security, etc. Some relevant protocols are MQTT, CoAP, DDS, AMQP, XMPP, STOMP, and HTTP. The protocols selection depends on their notoriety for IoT systems[21, 6].

The **MQTT** protocol (Message Queueing Telemetry Transport) is a lightweight protocol for sensing data traffic (telemetry), often used in SCADA systems (Supervisory Control and Data Acquisition), having been created in 1999 [22]. Because that telemetry protocol is well known and validated, both in academia and industry, it is the protocol on which there are more studies and articles published. This protocol has three levels of QoS; also can have security on the SSL/TLS communication channel; on some servers, authentication based on login/password [23] is implemented; uses the publish/subscriber architecture; the client publishes data in topics; is a binary protocol; uses a centralized broker and travel using TCP.

**CoAP** is a protocol that has gained notoriety due to being even lighter than the MQTT protocol. The Constrained Application Protocol is lightweight for machine-to-machine (M2M) communication by the IETF CoRE Working Group [6]. As CoAP does not use TCP for transport, but rather UDP, its performance on low-quality networks (where there is a lot of packet forwarding) is superior to MQTT. Despite using a protocol such as UDP, it can be reliable by using the QoS\(^4\) options provided by CoAP. This protocol has two QoS options; implements channel security using DTLS; uses a client-server architecture, but with the option of observer; uses URI to publish resources and access them; it is a binary protocol; uses a centralized server and is transported using UDP.

**DDS** (Data Distribution System) is a real-time interoperability standard [26], developed in 2004 by OMG. Unlike other protocols, this one does not have a central component, such as a server or a broker, decentralized protocol. DDS is said to be a data-centric protocol [26], but in fact, it is a MOM (Message Oriented Middleware) [27] protocol. Its comparability to CORBA is inevitable, as both were developed by OMG and use IDL [28]. Such OMG protocol offers more than 20 QoS options, such as standard configurations; channel security with DTLS for SSL/TLS; allows authentication; uses the decentralized architecture publisher/subscriber; it is a binary protocol; it has no central component,

\(^1\)Low Power Wide Area Network - which can provide long-distance connectivity[17] and excellent energy conservation capacity[18]
\(^2\)Uniform Resource Identifier[19]
\(^3\)Topics links to a key that represents a registered quantity[1, 20]; in some protocols topics, can be nested and have an hierarchy
\(^4\)Quality of Service[24, 25]
it uses a middleware, and its transport can happen both through UDP and also TCP.

**AMQP** (Advanced Message Queuing Protocol) is a message exchange protocol originating from the financial market, created by JPMorgan in 2003 [6], by John O’Hara. The protocol architecture is of publisher/subscriber or client/server, with exchanges and queues for distributing messages among subscribers. It also has different exchange types, such as fan-out, topics, and headings. The protocol offers two types of QoS (unsettle format and settle format); allows the use of TLS/SSL; authenticates using SASL\(^5\); has the central element of broker/server; is a binary protocol, and when it comes to transport, it uses TCP[30].

**XMPP** (eXtensible Message and Presence Protocol) is a protocol that gained ground in IoT, mainly due to its security features [7]. To be utilized in IoT, it is necessary to use the extension XEP060 (pubsub), which allows the use of the protocol with the architecture publisher/subscriber [31]. One critical drawback is that the protocol does not implement QoS, making it less attractive than other options. The protocol does not have QoS; channel security uses SSL/TLS; authentication happens using SASL; the architecture is publisher/subscriber; it has a central server; its protocol is textual and based on markings; and regarding transport, it uses TCP.

**STOMP** (Simple/Streaming Text Oriented Message Protocol) is a textual protocol based on frames modeled on HTTP [32]. STOMP is proposed to be an easy-to-use protocol, for implementation on the server-side and the client-side, does not implement QoS options; it has TLS/SSL channel security and user/password authentication; its architecture is publisher/subscriber; it has a central process of broker[33]; is textual protocol, and transports using TCP.

**HTTP** (HyperText Transfer Protocol) is a protocol for the web[34], proposed by Tim Bernes-Lee and standardized in 1997 [6]. Its main benefit lies in the fact that it is a widely used protocol for web systems and sites’ construction, thus, it is well known by developers [7]. However, because it is a text-based protocol and uses TCP transport, it does not perform well on networks and devices with energy and transmission restrictions. The protocol does not implement QoS; has channel security with SSL/TLS and various authentication mechanisms[35]; its architecture is client/server; has the central component of the server; its protocol is textual, and uses TCP for transport.

There is a great deal of complex in deciding which technologies to adopt for IoT projects. IoT systems bring variable security, robustness, and data delivery capabilities with various security options for message protocols. The choice of the most appropriate protocol must consider the solution’s characteristics for implementation.

### 2.2. Identity Management

The management of digital identities (Identity Management - IdM) is a fundamental factor for computational systems [36]. Such management allows customization to be made according to the users’ preferences and information associated with the identity, or even propose improvements in the experience and privacy protection, among other aspects. This study focuses on security aspects related to identity management, such as access control, identification, privacy, and other issues associated with privacy.

Identity violations can occur due to several factors [36], such as identity cloning for criminal purposes. Correspondingly, the exceptional implementation of identity management is also a fundamental factor for systems security. In regards to what the systems must ensure, do not exclude those using machine-to-machine communication, taking as an example those using the Internet of Things.

Within the scope of IoT, identity management transpires in different ways. Such management identifies the user who is accessing the collected data, or information processed from such data, and in the permission to use a “thing” which is seen as a resource, or even in regards to the performance of the “thing” itself. "It also identifies a data producer/consumer subject and actuator in the physical world, through some specific data configuration.

Specific cases related to sensors, loss of productivity, property damage, or access violations are all examples of the incorrect use of identity management[37, 38]. For instance, equipment that sends information about strawberry greenhouse humidity in an unsafe manner can suffer a violation. An attacker may intercept and change the moisture measurement values, possibly deactivating the irrigation system causing the strawberries to decay productivity. Another example was evident in the worm, Stuxnet [39, 40], which changed the reference for the rotation of uranium centrifuges in Iran and caused significant damage to them. Therefore, failure to implement adequate identity management in IoT is a critical risk factor.

A brief overview of current studies is necessary in the area of identity, which can describe the current scenario and security trends which may impact future advances in this study. Even implementations, enabled by the analysis done here, should look ahead in the horizon to view the impacts arising from these trends expressed through the following studies.

---

\(^5\)Simple Authentication and Security Layer[29]
A Comparative Qualitative and Quantitative Analysis of the Performance of Security Options for Message Protocols

Table 1
Related work overview. From the left, as the first column, there are the works, followed by the column of protocols addressed

| Authors                  | Protocols                                      |
|--------------------------|------------------------------------------------|
| Naik[6]                  | MQTT, AMQP, CoAP and HTTP                      |
| Dizdarevic et al[7]      | MQTT, CoAP, AMQP, XMPP, DDS and HTTP           |
| Luzurianaga et al[48]    | MQTT and AMQP                                  |
| Fernandes et al [49]     | AMQP and HTTP                                   |
| Yokotani e Sasaki[21, 50]| HTTP and MQTT                                  |
| Zorkany et al[8]         | CoAP and MQTT                                   |
| Sueda, Sato e Hasuike [51]| HTTP, MQTT and WebSockets                      |
| Hazra et al.[52]         | MQTT, CoAP, AMQP, XMPP, DDS and HTTP           |
| Babun et al.[53]         | MQTT, CoAP, AMQP, XMPP, DDS, HTTP, and ZeroMQ  |
| Donta et al. [54]        | MQTT, CoAP, AMQP, XMPP, DDS, HTTP, and others  |
| Aboubakar, Kellil e Roux[55]| MQTT, CoAP, AMQP, XMPP, DDS, and others        |
| Wytrębowicz, Cabaj e Krawiec [56]| MQTT, CoAP, AMQP, XMPP, DDS, and others        |

The studies are grouped into three parts:

- identity-based privacy preserving[9, 41, 42];
- mutual and multi factor authentication[10, 43, 44];
- lightweight XOR and hash based authentication[45, 46, 47];

In the work attributed to Ji et al.[9] an important contribution can be found within the scope of identity-based integrity, which is contextualized in the scope of cloud computing and presents a provable data possession (PDP) protocol which serves to guarantee the integrity of user data. Still in regards to the same topic, Li et al. [41] present a complementary study for the verification of data integrity without either the need to download it or great complexity of managing multiple certificates. Lastly, the work by Chang et al. [42] can be commented on, which brings the relevance of using a lightweight approach to data integrity in the cloud with the support of the Proof of Retrievability” (PoR).

Another advance in the field of identity management involves the utilization of multi-factor mutual authentication. Research undertaken by Loffi et al. [10] present an authentication option utilizing certificates and a nonce as process factors. On the other hand, studies such as Kalaria et al. [43] use mutual authentication with elliptic curves to provide robust yet lightweight authentication. Lastly, it is worth mentioning research put forward by De Smet et al. [44] which implements mutual authentication with physical unclonable function (PUF). This approach uses unique device characteristics in conjunction with mutual authentication to ensure secure identification of IoT devices.

Finally, it is worth mentioning some studies which propose the provision of a lighter authentication option, such as Fotouhi et al. [45] which uses a set of XOR and hash functions to provide lightweight authentication within the scope of e-health. Another study utilizing the same approach is found in Cheng, Lee and Hsu [46] which integrates this approach with the use of biometrics in a mobile device for user authentication. Lastly, the research of Dhillon and Kalra[47] which integrates mutual authentication, XOR and hash to propose a robust, lightweight multi-factor authentication.

As can be seen, some areas of research have current significance. The preservation of identity-based privacy in a cloud computing environment has been benefited from advances in the field. Mutual authentication has established itself as an approach which provides robust and reliable security for IoT systems. All things considered, the search for different factors such as PUT, XOR functions and hash and nonces; has increased security and robustness in the authentication process.

2.3. Related Work

In literature, many performance analyses can be found of message protocols in terms of performance, resource utilization, security and resilience.

In Naik[6], a comparison was drawn among four of the most popular message protocols. The MQTT, AMQP, CoAP, and HTTP protocols were all presented. The author made a qualitative comparison and directly associated it
A Comparative Qualitative and Quantitative Analysis of the Performance of Security Options for Message Protocols

to a bibliographic survey on each work’s characteristics. Despite being a complete work on the subject, there was no quantitative analysis on data generated in each protocols’ use. It is also worth noting that some protocols with less visibility, but significant relevance, such as DDS and XMPP, were not addressed.

The authors Dizdarevic et al. [7] put forward a comparative analysis based on literature regarding the MQTT, CoAP, AMQP, XMPP, DDS, and HTTP protocols. Similar to the previous work, the comparison restricted qualitative data extracted from the bibliography. There was no evaluation of quantitative data or experimentation proposed by the authors. The context used for evaluation was Fog Computing.

Akin to the others, Luzurianaga et al. [48] proposed the assessment of the MQTT and AMQP protocols on mobile and unstable wireless networks. Such work involved a practical experiment using WiFi routers to simulate the transfer between device networks. The protocols evaluated were pub/sub, and there was no need to implement servers. The experiment analyzed jitter and packet loss.

Fernandes et al. [49] compared the AMQP and HTTP protocols’ performance using RESTful web services. In the study, RabbitMQ was used as a broker, supporting both protocols. Metrics, such as the average number of messages sent, the number of messages stored in the database, and the broker were listed and analyzed in the study.

Yokotani and Sasaki [21] presented a comparison between HTTP and MQTT. They set forth data about the overhead by topic to send data in both protocols and evaluate the payload’s size. The requirements for use on a server and overhead on extended MQTT topics were also evaluated. In a second study [50], the same authors submitted more information about some metrics and formulas and presented the data.

Zorkany et al. [8] compared the CoAP and MQTT protocols for IoMT (Internet of Medical Things). As claimed by the authors, these protocols played an essential role in the context of IoT technologies. The work chose a UDP option (CoAP) and a TCP option (MQTT) for performance comparison. The metrics used for the evaluation were the speed of sending the message, the average size of the message, and the average delay in sending messages. As per architecture, a centralized server to receive and store data was proposed. They selected the WANEM simulator to evaluate the protocols.

Sueda, Sato, and Hasuike [51] compared MQTT, HTTP, and Websockets. This work evaluated the size of packets in each option, QoS options, and the impact of data transmission on performance and overhead. It is worth noting that Websocket is a messaging protocol with the option of sending its data through a WebSocket. One of the conclusions reached was that the importance of planning the topic name’s size, which could influence overhead.

Research done by Hazra et al. [52] presents a survey on technologies and platforms that provide interoperability between systems, such as message protocols. In their work, the authors analyze the platform regarding the adoption of message protocols, organizations related to them, among others. Despite the widened scope, this is an important and relevant work.

Another contribution emerged from a survey in a study by Babun et al. [53] which presents an extensive survey of different messaging protocols, platforms and network protocols utilized in IoT and their security characteristics. It is also worth mentioning the evaluation of the ZeroMQ protocol, even though it is less popular.

Continuing with surveys, Donta et al. [54] present an evaluation of IoT application layer protocols within the scope of Machine Learning. In this study, the authors present an evaluation of message protocols, tools which support them as servers and development libraries, and put forward a comparative list of other surveys which focus on the same scope.

In their review [55], Aboubakar, Kellil and Roux provide an analysis of existing solutions for network management in restricted devices together with a taxonomy of these solutions. Their research presents an overview of platforms, message protocols, and management protocols. Furthermore, this study presents an analysis of the characteristics of network management frameworks based on SDN6, Semantics, and Machine Learning.

In Wytrebowicz, Cabaj and Krawiec [56] a pragmatic analysis of messaging protocols is presented. Analysis is done in a documentary way for the general characteristics of the protocols. The following is worth noting: the authors differentiate MQTT from MQTT-SN in their analysis, present different protocols such as WAMP, OPC UA, LwM2M, and Weave for analysis together with the most popular ones. However, even though it is well utilized, the research does not provide an analysis of HTTP;

From the 2 table, four relevant aspects of the related works can be noted: most of the works contain only a documentary analysis; experimental works analyze protocols in pairs; the experimental works do not bring a deeper document review (Systematic Literature Review); and there is no more consistent discussion between real case (use case), literature and experimentation.

Of the total number of works analyzed, seven (approximately 63%) are evaluations based on documents and without
experimentation. Even though they are significant works to provide documentary support to several types of research, these studies could benefit from an experiment that allows analyzing programming aspects, and execution of source code written for the analyzed protocols. Thus, these works fail to provide some important information for more practical profiles that seek this type of analysis, such as developers, undergraduate students, and system designers. Thus, such works would benefit from experimental complementation.

On the other hand, four (25%) works bring an experimental analysis of message protocols. Within these works, everyone performs a peer-to-peer analysis of protocols in search of a solution to some specific aspect of their research, leaving behind a deeper review of the literature. Since these works are experimenting with protocols in pairs, a broader view may be crucial for a better choice of technology. This is even more important in situations where researchers and graduate students need more information to support their research. Finally, experimental work could have benefited from a more thorough review of the literature on protocols.

None of the related works combines documental analysis (qualitative aspects of the protocols) with experimental analysis (quantitative aspects) for the evaluation of the protocols. This type of evaluation allows a more mature view of the circumstances of use of the protocols and serves as a guide for the implementation, modeling, and readjustment of the experiment scripts for the need for other future similar research. In this way, a theory-practice correlation brings benefits to a more mature vision and guides future work.

As can be seen in Table 2, this work is the only one that brings a joint analysis (practical and theoretical) of the listed protocols. As for the theoretical (qualitative) evaluation, the seven protocols listed in their safety characteristics and functionalities were analyzed. For the practical evaluation, three protocols were analyzed in terms of time, memory, and network performance. In addition, there is a correlation between qualitative and quantitative results and usage situations, thus promoting a more mature evaluation of results according to usage situations. It is thus concluded that within the proposed scope, this work presents itself as a more complete option in its analysis of the protocols.

### 3. Quantitative Analysis - Description of the Use Case, Experimentation with Protocols and Analysis of Partial Results

This section discusses (i) details of the modeling of the scenarios used to analyze the results and characteristics of the protocols, (ii) algorithms and components architecture used for evaluation, (iii) the configuration of the experiment conducted, and details of the tools adopted in its implementation, (iv) analysis and comparison of the results obtained in the experiment, and (v) analysis of the protocols based on the scenario’s vision and the threat model created.

Through these steps, a mature vision can be shown to choose each situation’s message protocols. We believe that using the proposed scenarios intend to bring the analyzed scenario closer to a generic scenario that can serve as a blueprint for equivalent IoT systems. As such, the appropriately reduced costs and development time of products and services for the fish-farming system with IoT became achievable.

### Table 2
Related works and message protocols analyses - (d) documents analyses, (e) experimental analyses, e (n) not applicable

| Publication                          | MQTT | STOMP | AMQP | DDS | XMPP | HTTP | CoAP |
|--------------------------------------|------|-------|------|-----|------|------|------|
| Naik[6]                               | d    | n     | d    | d   | n    | n    |      |
| Dizdarevic et al. [7]                 | d    | n     | d    | d   | d    | d    |      |
| Luzuriaga et al. [48]                 | n    | e     | n    | n   | n    | n    |      |
| Fernandes et al. [49]                 | n    | e     | n    | n   | e    | n    |      |
| Yokotani and Sasaki [21]              | e    | n     | n    | n   | e    | n    |      |
| Zorkany et al. [8]                    | e    | n     | n    | n   | n    | e    |      |
| Sueda, Sato, and Hasuike [51]         | e    | n     | n    | n   | e    | n    |      |
| Hazra et al. [52]                     | d    | d     | d    | d   | d    | d    |      |
| Babun et al. [53]                     | d    | d     | d    | d   | d    | d    |      |
|onta et al. [54]                       | d    | d     | d    | d   | d    | d    |      |
|Aboubakar, Kellil e Roux [55]          | d    | d     | d    | d   | d    | d    |      |
|Wytrębowicz, Cabaj e Krawiec [56]      | d    | d     | d    | d   | d    | n    |      |
|This work                             | d+e  | d+e   | d+e  | d   | d    | d    | d    |
3.1. Use Case Modeling

This paper puts forth three generic scenarios covering most of the existing fish-farm main plants. These scenarios were constructed varying the energy supply type, connectivity between the parts (Sensor-Gateway-Fog Node, with the Gateway not existing in case one), the distance between the parts of the system and network reach, the density of sensors implanted in the area, distribution, number of sensors, characteristics of the installation, wired powered and battery-operated sensors, the resilience of the communication network, locality of the data processing unity, among others. Using the suggested characteristics, a designer could analyze which scenario best suits their reality.

Figure 2: Descriptive Scheme of the Different Scenarios Proposed for Fish-farm Analysis - the scenarios are described from the simplest (with the greatest amount of available computing resources), on the left, to the most complex (due to restricted access to computing resources) on the right.

In the first scenario, a small fish-farm plant was modeled. This plant is a common form of fish-farming [57, 58], especially in situations which aim to improve small rural communities’ income or access food in vulnerable locations. This type of production does not require much financial investment and can supplement small producers’ income. Some projects [59, 60] are open technologies and allow their multiplication to reduce starvation.

This scenario has a low environmental impact, as there is no need for large spaces for fish tanks, Figure 2-(a). The tanks can be built in small warehouses or in the backyard of homes, with no need for deforestation. Another important consideration is that there is no way to elope and come into contact with the local fauna and consequently cause ecological imbalances.

For this plant, two components are required: the sensor and the fog node. The sensor consists of a power source connected to the energy grid, a local network link (i.e., Ethernet), and I/O bus for the sensors, and a monitoring program. The fog node must be composed of at least one energy source connected to the grid, a local network link, a program for receiving and processing data, and data visualization and reports for the end-user (Farmer 1).

The tanks are small and placed nearby, Figure 3. The sensors are positioned in the tanks, have access to a power outlet and an Ethernet network cable or a WiFi connection; there is no requirement for data traffic on the internet. The data is transmitted to a fog node located in the same building or location. For more information, follow the project git repository\(^6\).

Figure 2-(b) represents the second scenario: interconnected production distributed in multiple farms. Small fish producers commonly have their farming integrated with other forms of agriculture. Fish tanks are used as a water source for irrigating crops planted on the property. This integrated use is a way to enhance the utilization of the existing property’s resources, allowing small farmers to diversify their production. This possibility leads to an increase in the producer’s inflow of resources, guaranteeing a better subsistence condition [61].

This type of production provides a significantly greater environmental impact among others listed here. The production of irrigation tanks often involves altering watercourses, creating dikes/ponds [62], or partial diversion of springs or small streams. This altering is a reality on small farms throughout Brazil, especially where sprinkler irrigation is used.

The components of this arrangement include a sensor as well as a fog node. The sensor must be supplied with power directly from the grid or a battery (depending on the characteristics of the monitoring location), the network link is no longer local to become an internet link (ADSL, GRPS, among others), a bus node I/O for sensors, and a program which manages the data collection and publication cycle. The fog node must be connected to the energy source.

\(^6\)https://github.com/wesleybez/iot_fishfarm_experiment
Multiple sensors are allocated in various sites, running on heterogeneous configurations. The sites communicate over an open network (internet), eventually upon long-range protocols, such as a Low Power Wide Area Network (LPWAN), to an external central processing unit.

Figure 2-(c) represents the third scenario: a large production. In this production model, it is possible to integrate fish-farming with nature creating a low impact. Such a case can take advantage of existing electric production dams or use production tanks that take advantage of water flow in the river.

In this scenario, the existing natural structures are used, such as dams, rivers, and streams. There is no alteration of watercourses by the fish-farm plant project using existent structures and running water to fatten the fry. Because the water is flowing, waste management and food become factors with less impact. However, it is necessary to guarantee water quality through constant monitoring, thus avoiding external contamination of the plant. Cases like red tide can contaminate specimens [63], not allowing them to be consumed.

Unlike the others, this plant has three components: sensor, gateway, and a fog node. A battery must power the sensor, have a 6LoWPAN link for communication with the gateway at a short distance, have an I/O bus for analog grid, have an internet link, a program to process data, a data storage component, and also a front-end for information consumption by the end-user (Farmer 2), Figure 4.
A Comparative Qualitative and Quantitative Analysis of the Performance of Security Options for Message Protocols

Figure 5: Farm3 Components Architecture - sensor components are seen on the left, FN components in the center, and lastly, further to the right is the person who will use the arrangement. The software components are represented in green, the network components in blue, the power supply in red, the communication bus in gray, and finally, the user interface in purple.

sensors, and a data acquisition program and publication. A battery also powers the gateway which has a 6LoWPAN link (i.e., BLE) for communication with the sensor, an LPWAN link (i.e., LoRAWAN) for communication with the fog node, a program for data buffer and publication in the fog node. The fog node has a grid power supply, an LPWAN link with the gateway, a data processing program, a data storage component, and lastly, a front-end for data visualization by the end-user (Farmer 3), Figure 5.

This configuration is characterized by a vast number of sensors in one site, in a heterogeneous configuration. Sensors are mainly local-powered, batteries or harvested\(^7\) from the environment, in order to supply energy. Devices communicate through an LPWAN with potential problems in resilience and security, transmitting to a central processing unit. This remote site is a complex scenario, characterized by diverse configurations, potentially unreliable communications, large data volumes, and unsafe networks. The data transmission via LPWAN and the batteries’ use is a challenge for security solutions.

Correspondingly, the most common fish-farm methods in scenarios are presented this way. According to the aforementioned fish-farm architectures and models, most situations in which this animal protein is farmed can be generalized. It will be possible to assess the impact of using different messaging protocols and distinct security options on IoT systems’ performance through such scenarios.

3.2. Experiment Components Model

There are two main algorithms in the experiment: the evaluation algorithm and the data collection algorithm. It was observed that even if it is described linearly and in the imperative paradigm, it occurs parallel through different execution lines. The loop represents what simultaneously happens in each thread.

**Algorithm 1**: Evaluation process for each message protocol

```python
Result: evaluation process algorithm
1 c = read_config();
2 f = generate_file(c);
3 while !total_instances do
4     connect(s);
5     d1 = collect_data();
6     publish(s, data);
7     d2 = collect_data();
8     disconnect(s);
9     d3 = collect_data;
10    save_data(d1,d2,d3);
11 end
12 close_file(f);
```

The evaluation algorithm 1 is parameterized through a configuration file which establishes the number of instances.

\(^7\)It can possibly occur with the combination of more than one modal, for example: solar and wind[64]
Initially, the above algorithm reads the configuration file (line 1) and creates the datasets files for generating the reports (line 2). From this point on, it loops until it has executed all instances (line 3), according to the configuration. In the loop, the algorithm connects to the broker (line 4), calls for data collection (line 5), publishes data (line 6), collects data again (line 7), disconnects from the broker (line 8), collects data after disconnection (line 9) and then saves the data line of that instance of execution in the specific datasets. When the loop ends, the dataset is closed (line 12).

Algorithm 2: Evaluation process for each message protocol

Result: data

```python
Function collect_data() : int,int is
    t = collect_time();
    m = collect_memory();
    return t,m;
end
```

The data collection algorithm 2 is more straightforward than the evaluation algorithm 1. The second algorithm collects two metrics: execution time (line 2) and memory usage (line 3). For memory measurements, this algorithm collects the amount of memory used at each moment it is called, counting the amount of memory used by the thread since the beginning of the process execution. As for the execution time, the algorithm counts the time spent for each step from connection to publication and disconnection.

As the generated files are separate for each day and selected configuration, it is possible to compare the distinct security configurations' spatial and temporal impact. This comparison is also possible for the different protocols used in the performance evaluation environment. Different datasets (12 datasets) are generated for this experiment.

3.3. Experiment Description

In this experiment, source codes were developed in Python. Such codes were built separately for the exact algorithm implementation for each of the three message protocols and in each of the two proposed test cases: a code without authentication and a code with authentication. Thus, six algorithms were produced to carry out this work’s experimentation.

By implementing a data-generation-workbench, the protocols were able to be approached fairly. Each implementation takes into account characteristics of the protocol in question, not approaching the experiment as a black box but as a series of equivalent implementations which can be modified to evaluate different configurations. A researcher or developer who wants to add modifications or business rules to the workbench source code can express their need in a much more realistic way.

The environment consists of four parts: (a) scripts, (b) simulation environment, (c) datasets, and (d) reports. The scripting subsystem is responsible for the source code that will be executed during the simulation. This subsystem also includes the basic specification of a monitoring script and the configuration file used to adjust the number of instances to be executed, among other details of the execution. The scripts are developed in Python 3.7 using the Eclipse development environment. Specific libraries were used for each message protocol.

In the simulation environment, the subsystem (b) is comprised of the software components responsible for executing the main processes, such as instantiating monitoring and reporting threads; monitoring threads are also allocated. The execution of threads occurs through the Python 3.7 threading API, and the generation of comparative graphics is done using GNUPlot.

The information storage is in charge of the dataset subsystem (c). This subsystem is responsible for the storage of data published by the monitoring instances. In this experiment, three types of information were stored: temporal information, memory consumption, and packages. The materialization of this storage took place through CSV files saved on the computer’s hard disk, where the environment was performed.

Lastly, there is the reporting subsystem (d). This part of the system is responsible for data processing and generating comparative reports, enabling more accurate analysis of the different security options simulated in the different reports. A subsystem, such as this, inputs the datasets generated by the monitoring threads and the scripts developed in the GNUPlot language. Datasets are also used as input for scripts which analyze data and generate reports with sample statistical data. The generation of statistical data takes place through the NumPy library.

Time measurements were taken along with the total of one hundred instances of the source code run in each execution. The script connects, authenticates\(^8\), publishes and disconnects from the broker with each execution. Mea-

\(^8\)Configurable through files where security options are described
measurements in the following were accounted for: connection time\(^9\), publication time, memory, and packet size.

In order to avoid measurement errors, or any variation which may happen and cause discrepancies in data measurement, one hundred instances were chosen. The possibility of utilizing averages would help to eliminate random errors resulting from this situation.

Test cases were performed for each of the protocols, and reports were created from the data generated, leading to infer the performance of the protocols\(^10\). Such security mechanisms are essential, as presented in all listed protocols.

### 3.4. Quantitative Analysis of Partial Results and Comparison

The experiment results can be separated into three dimensions: time evaluation, memory evaluation, and packet size evaluation. It is possible to evaluate a protocol’s performance quantitative data characteristics through the developed scripts.

Each dimension of the analysis was chosen to support IoT design decisions quantitatively. The measurement of time will allow analyzing quantitatively the amount of time necessary to execute a task, which can be translated to processor usage and energy consumption. Conversely, the memory dimension will explain the need to use this resource to adopt the correlated protocol. The analysis of the packages, on the other hand, brings the expectation of network consumption and energy resources associated with transmission. Such dimensions are of great importance for a good design or analysis of an IoT system.

#### 3.4.1. Time Evaluation

Processing time refers to the battery consumption time of the processor. For some devices, this time is crucial for design choices. Correspondingly, this study analyzes two moments in time: the time of connection to the server and the data publication time. The server’s connection involves some steps and possibly negotiation in some situations. In short, it is the most complex. On the other hand, the publication assumes that there was a previous connection to the server; that is, only the sending time of the data to be published is measured.

\(^9\)A critical observation pertains to the time values which were multiplied by 10,000 (ten thousand units) to improve the graphs’ visualization.

\(^10\)In the AMQP case, the option without authentication was made using RabbitMQ’s default login and password (guest/guest), so consider AMQP’s unauthenticated data as a reference only, not being used for comparison.
Table 3
Time averages (ms) generated during the simulation (x10). From the left, as the first column, there are the protocols, followed by the connection time without authentication, in the third, the time with authentication, and finally the data publication time.

| Protocol | NoAuth | Auth | Publishing |
|----------|--------|------|------------|
| AMQP     | 5.341  | 5.25679 | 0.01849 |
| MQTT     | 3.71779 | 5.99770 | 0.10060 |
| STOMP    | 1.33851 | 1.47076 | 0.07443 |

Such a distinction brings benefits to data analysis and facilitates its use in different situations. For applications which will need to connect once and send much data, the weight of the connection time is less relevant than the weight of publishing data. However, in some applications, it is necessary to connect successively to the server each time data is sent, and for such applications, the connection time is more relevant than the publication time. For this reason, the separation of these two moments in the measurement of time is more appropriate for a comprehensive analysis of the data.

![Figure 7: Connection Time Comparison](image)

As Figure 7 shows, both the STOMP protocol and the MQTT have a connection time much shorter than the AMQP. The execution time of AMQP is twenty times longer than the other protocols. However, this is expected, as this is a safer protocol than the other two, having been proposed for exchanging messages not only on IoT devices. The other protocols have superior performance, connecting in milliseconds, even a text-based protocol such as STOMP.

Nevertheless, in Figure 7-(a), it is evident that the best performance came from a protocol with a small header (with a minimum size of 2 bytes), the MQTT. This protocol, designed for telemetry, would be expected to have superior performance compared to the others. MQTT is also a binary protocol, unlike the textual protocol STOMP, with the second-best time performance.

If clients were authenticated, the worst performance would be with MQTT. This fact may be related to the tool with more features than the other brokers. However, since Mosquitto is the most used MQTT broker today, such a comparison is valid. A difference of a few milliseconds gives AMQP the second-best performance, see Table 3, getting the best performance with STOMP in this item.

When publishing data, the AMQP was executed the fastest, followed by the STOMP; see Table 3. Even though there was a difference of a few milliseconds, the STOMP protocol performed better on this item. The one with the lowest performance was the MQTT protocol, which did not have data sending in full-duplex mode unless using WebSockets; AMQP’s better performance in this function may be due to its full-duplex transmission capacity, absent in the other studied protocols.
The time analysis in the listed protocols enabled important details to be viewed, which are not easy to infer. Details such as the good performance of AMQP in the measured publication time are obviously not a priority, even with the knowledge that the protocol is capable of full-duplex transmission. The experimentation allowed the verification of some data brought by the literature and to quantify the differences between the protocols in a practical way.

3.4.2. Memory Consumption Evaluation

In this section, the scripts’ memory consumption will be evaluated. Security configuration related resource consumption will also be assessed for each message protocol selected in this study. Memory usage requirements may vary according to the security level established in the communication. This analysis can serve as a decisive factor for the device’s design.

Through comparison, it is possible to identify the best protocol regarding memory use. Due to specific memory usage requirements by devices and the amount of available memory, the memory consumption in each protocol and library must be analyzed more thoroughly.

Both situations will be analyzed for each protocol: authenticated (a) and non-authenticated (b). This analysis takes place considering the average memory consumption for each executed instance.

![Figure 8: Average Memory Consumption - Authenticated](image)

In the situation with authentication (a), Figure 8, the best performance was found with MQTT. With an average power consumption of close to 1MB (one Megabyte), the MQTT presented a suitable solution for more modest devices and low memory capacity. Secondly, there was STOMP; a protocol with a textual basis, processes and commands sent in text format, while still achieving performance suitable for second place (consumed less than 5MB on average). Finally, the AMQP was the most robust protocol in security, where it presented the highest memory consumption; almost 9MB on average.

The non-authenticated situation presented a divergent result, Figure 9. The lowest consumption was found in the STOMP protocol, with less than 1MB per instance. The MQTT protocol came in second place, consuming less than 3MB for communication without authentication. Finally, there was AMQP, again, with almost 8MB per instance. This robust protocol for security remained the least suitable for devices with memory restrictions.

It can be concluded that the AMQP protocol was the least suitable for situations with more modest hardware and RAM supply restrictions. This document presents several security options and settings for sending messages; including full-duplex messaging [65, 66].

As it was strongly indicated, the MQTT was chosen. Even though this protocol was in second place during the experiment without authentication, it showed a minor variation in the worst case. MQTT consumed less than 3MB
of memory per instance, in its worst case. STOMP consumed almost 5MB, a great difference in a restricted device design.

This memory evaluation makes it possible to obtain relevant data for choosing the message protocols during the design phase of a restricted IoT solution. Even if the experiment was not done in a dedicated computational environment, the data has the representativity and relevance of an experiment close to the real situation, disregarding network aspects.

### 3.4.3. Packet Evaluation

The evaluation of transmitted packets is necessary for a well-designed IoT system. Equally important as the other metrics adopted in this work, the package’s evaluation allows the designer to adjust resources and plan battery consumption, or even data package - in case of using tariff networks like NB-IoT\(^\text{11}\). Information such as the number of messages exchanged or the messages’ size can influence how long the network interface will be energized and what the battery consumption will be in scenarios without a battery. This dimension of the analysis is shown to be essential for success in the design of solutions.

With a greater amount of data in the data transmission of each protocol, it is possible to mitigate project risks. Depending on the scenario in which the solution will be implemented, the use of some protocols are not feasible. As exposed here, the greater level of detail of the different characteristics of the transmission allows the designer to analyze the time and memory in which a solution with authentication consumes, as well as quantify how many more packets it exchanges and the change in the average of packets sent.

The packet data analysis was conducted using five variables: largest packet, smallest packet, average packet size, total bytes transferred, and the quantities of messages exchanged. These variables represent quantitative data collected during the experimentation in which an instance was performed for each configuration of the protocols. The collection was made with Wireshark, the datasets were generated in CSV, and Gnuplot plotted the graphs. It is important to note that both AMQP and MQTT have filters implemented in Wireshark; in the case of STOMP, it was necessary to filter the packets by the server’s port number.

The largest packet generated among the protocols was AMQP, with the authentication option, which obtained 578 bytes. Followed by the STOMP protocol with 171 bytes, in both configurations. Finally, the smallest of the largest packets was MQTT - with 106 bytes. As such, it can be considered that the best performance in the worst measured case was the MQTT, which obtained a package more than five times lower than the worst case of AMQP.

---

\(^\text{11}\)Narrow Band - Internet of Things
Table 4
Packet Analysis by Protocol and Security Configuration. From the left, the first column shows the protocols and their security options, second the largest transmitted packet, followed by the smallest transmitted packet, fourth the average packet size, fifth the total transmitted during communication, and finally the number of transmitted packets. The unit of measure for the Major, Minor, Mean, and Total columns is bytes.

| Protocol(sec) | Major | Minor | Mean | Total | Amount |
|--------------|-------|-------|------|-------|--------|
| AMQP (auth)  | 578   | 74    | 116  | 2088  | 18     |
| AMQP (noauth)| 566   | 78    | 132  | 2239  | 17     |
| MQTT (auth)  | 106   | 70    | 90   | 718   | 8      |
| MQTT (noauth)| 93    | 70    | 81   | 648   | 8      |
| STOMP (auth) | 171   | 74    | 90   | 1429  | 16     |
| STOMP (noauth)| 171  | 74    | 88   | 1397  | 16     |

Table 5
Packet Analyses - Protocol Ranking

| measurement | 1st  | 2nd  | 3rd  |
|------------|------|------|------|
| major      | MQTT | STOMP| AMQP |
| minor      | MQTT | STOMP| AMQP |
| mean       | MQTT | STOMP| AMQP |
| total      | MQTT | STOMP| AMQP |
| amount     | MQTT | STOMP| AMQP |

The second variable was the smallest package, which remained with the MQTT. With only 70 bytes, the MQTT disconnect message was the smallest packet transmitted. In the second place was a tie between AMQP and STOMP, with 74 bytes in the smallest packet. It is evident that the smallest package is a metric which influences less in the project because it will not generate a cost increase. Thus, this variable identifies only the minimum battery time consumed for transmission in the best case.

In addition to these is the average message size metric, in which MQTT also achieved the best performance with 81 bytes. This was followed by the STOMP protocol with approximately 88 bytes, only 7 bytes more than the first. Finally, there was the AMQP which had 116 bytes, 140% greater than the first place and 131% greater than the STOMP, found in second place. As can be observed, the STOMP has its average performance very close to the MQTT, despite having a textual protocol.

With 648 bytes exchanged throughout the communication script, MQTT outperformed regarding the total bytes exchanged. In second was the STOMP with 1397 bytes (215.6% higher than the MQTT). Lastly, there was the AMQP with 2088 bytes, generating traffic 322.2% higher than the first place. This resource consumer demonstrates an excessive consumption of resources.

As the last metric, the number of messages exchanged by the monitoring script was used. With a smaller number of messages, there was the MQTT (only eight messages). Subsequently was the STOMP, with 16 messages (eight requests and eight responses). And lastly, the AMQP with 18 messages exchanged during the conversation.

In general, Figure 10 shows that MQTT had the best performance in all metrics chosen for packets, followed by STOMP, and finally AMQP, which systematically performed the worst in packages during the tests. The very positive result in the low number of messages exchanged by MQTT in each thread during the experiment were highlighted. As a negative highlight, many bytes were trafficked in total and the largest message sent; both are with the AMQP.

It can be concluded that more resources are used for packet traffic when using the AMQP protocol and that it performs 322% worse than MQTT during the standard communication process: connect, publish and disconnect. This negative assessment can impede the use of the mentioned protocol in several areas of the IoT. However, its low performance and high consumption of resources are associated with its robustness, greater message delivery capabilities, and a greater number of security options.
3.5. Quantitative Analysis Considerations

The objective of this study was to analyze quantitatively the impacts of each scenario’s security parameters by protocol to indicate the best security option and protocol for each of the chosen scenarios. By choosing the best business and safety ratio, the aim was to promote safety in the fish-farming environment and, as a result, avoid production losses.

Even though an experiment focused on the use case scenarios listed above, generalizing the results for different IoT applications. Depending on the IoT application area, the choice of one or another measured quantity may have a greater influence.

4. Analyses Quantitative Results of the Experiment by Correlation with Qualitative Characteristics and Uses Cases

The results listed in this section were analyzed from each scenario’s point of view. This analysis brings the argument closer to a real situation and makes it possible for the IoT system designer, or even the small fish farmer, to decide. The choice of the technology used is an essential factor because it is considered that farmers will not have enough funds to improve or exchange equipment and related solutions in production situations for food security. It is crucial to deliver the most suitable project for this producer within the limited budget. This fact places responsibility on the designer, who must be aware of the maximum number of variables involved in the decision.

More resources are available in the first scenario, and it can be more costly to use these resources. In this case, security mechanisms can be implemented which consume more energy, processing, and data transmission resources. As the plant has a reliable energy supply, the devices used may have more computational power supporting more robust encryption, authentication, privacy algorithms, and auditing. Security breaches in this environment would be prevented or corrected with greater ease. Attacks can be more easily mitigated. However, it is important to be aware of existing vulnerabilities. Device configuration problems, weak security mechanisms, or MITM can also be problematic in this scenario.

Amidst this ample perspective, the AMQP protocol is the most suitable. It can impart security settings suitable for large indoor production, lend full-duplex communication, and other features; which make it the most robust and costly protocol among those selected. Even with the worst time performance in the evaluation, processing speed is not a problem within the scope of food production. In a more time-sensitive scenario like Industry 4.0, this could pose challenges. However, its robustness would still compensate for the slowness. The sensors along with the gateway used can be more expensive than excessive memory consumption. Regarding packet evaluation, even with the worst
A Comparative Qualitative and Quantitative Analysis of the Performance of Security Options for Message Protocols

Table 6
Protocol selection by scenario.

| Scenario | Protocol       | Pros         |
|----------|----------------|--------------|
| farm 1   | AMQP           | reliability  |
| farm 2   | STOMP or MQTT  | fewer resources |
| farm 3   | MQTT           | lightweight  |

performance by far, in a wired network and with little packet retransmission, this would not be a real issue.

AMQP brings a large variety of tools to a more resource-rich environment. It is possible to achieve faster communication when publishing data due to its full-duplex transmission capability. Since the farm 1 network has a robust and reliable link, it is possible to keep the connection open for longer and avoid the cost of opening a new connection with the broker. According to our experiments, opening the connection is one of the most expensive processes; maintaining an open connection between the sensor and the broker would save resources and time. Another capability provided involves the use of specific AMQP configurations, such as configurable exchanges and vhosts.

The second scenario uses primarily the wireless network, which can potentially present security problems. This wireless scenario is a more complex one, characterized by external links of potentially unreliable communications, stable configurations, and unsafe networks. The possibility of using sensors with batteries is a greater concern with energy consumption also in this scenario. The positioning of sensors in an open environment makes them vulnerable to climatic conditions and attacks by animals or people with malicious intent.

Correspondingly, Man-in-the-middle (MITM) and Denial-of-Service (DoS) attacks are more likely, leading to problems around accessing information from distant sites and reliability. The immediate consequence of such attacks would be latency in accessing production data and lack of configuration control, leading to an impact on production performance. The protocols’ restricted data transmission capacity must be considered when evaluating the security mechanisms.

In a wireless network scenario with the possibility of security breaches, authentication is needed. As a result, the analysis should focus on protocol measurements with authentication. In this perspective, STOMP was the protocol which had the best time performance in the experiment; ranked second for memory consumption and second for packet analysis. Thus, it was concluded that both STOMP and MQTT are suitable for the scenario; MQTT being the slightly lighter option.

Lastly is the third scenario, which uses LPWAN and more resource-constrained devices. It is important to note that some LPWAN protocols are very restricted in terms of the number of packets trafficked (the utilization of Sigfox protocol[15] 12). Thus, security options such as secure channels with SSL, mutual authentication and other security solutions are unviable. In this challenging scenario, a suitable solution may be the difference between security use in the IoT system.

Security remains a critical factor as a successful attack brings some potential threats: it makes some sensors unavailable, inserts erroneous information into the system, and changes references. Physical attacks can be common once there is no protection of the sensors within a production plant. Another problem regards interference with data transmission, which can happen either actively or passively. Playfully, malicious equipment can superimpose the signal transmitted by the sensor, preventing it from reporting its measurements. Passively, a natural barrier or even a storm can block signal transmission. Finally, it is important to reinforce the need to mitigate sleep deprivation attacks; in the context of sparse networks of battery-powered wireless sensors, massive energy consumption due to sleep deprivation can make several sensors unavailable, leading to system degradation.

In this last scenario, the most crucial factor is saving resources, which leads us to the MQTT protocol. In a scenario with few features, authentication must be light or non-existent, to maximize the sensors’ battery life. Since it uses the least number of messages during communication (only eight messages), MQTT is the protocol which saves bandwidth and transmission time the most. It also performs satisfactorily in script processing time and has the best memory consumption performance. As a result, among the protocols analyzed, the MQTT is the most suitable for scenario 3.

The results reflect that there is no absolute champion protocol; no silver bullet. Each scenario requires different challenges that can be archived differently by each protocol. Likewise, each drawback in the analysis of the protocols is more relevant in some scenarios than in others; it may not even have any relevance in some situations. Thus, with

12with limitations of 140 message downloads per day and four uploads, combined with a 12-byte payload[67, 68]
this study, we can analyze the situations, characteristics, and consequences of choosing each protocol in a closer way to the real one according to its use. This study analyzed a scenario within the scope of fish-farming. Although, it still can be expanded to any IoT area, respecting the necessary adjustments to analyze the data through new scenarios.

5. Conclusion

By means of the proof-of-concept analysis promoted in this study, it was determined that there is no single comprehensive solution, and each scenario proposes unique challenges in which different protocols and configurations could be utilized. It was argued that it is possible to analyze, in a closer way, the situations, characteristics, and consequences of choosing each protocol to the real one according to its use.

In addition, the main related studies were identified, along with their experiments, which were brought forth as the message protocols of greatest interest on the academy’s part. Conclusively, the importance of the these related work and literature review in the creation of a robust bibliographic portfolio, which supported the qualitative analysis of the evaluated protocols, can be affirmed.

More precisely, in the experimentation, it was possible to analyze the protocols through three dimensions: time, memory, and packets. Each dimension was detailed with different specific variables collected and analyzed through graphs. Notice that the possibility of customizing the data-generation-workbench adds to the possibility of evaluating the scenario of each researcher, with their own network peculiarities, business rules and others. No less important was the availability of source code for the experimentation and the data-generation workbench to allow use in new future research by third parties. As such, the analysis enabled researchers and developers to support their design decisions in this exposed data.

Finally, three scenarios were brought up for analysis. In each scenario, the environment was modeled based on its characteristics such as energy availability, network availability, location, etc. The components of the system and their technological characteristics were also modeled.

Through this overview, it was possible to correlate the quantitative aspects of the experiment, the qualitative aspects presented by the literature, and analyze the scenarios based on experimental data and documents. As a result, the protocol selection for each delimited scenario was listed and argued as to how each fit in to better meet each situation’s requirements.

In future work, the following is proposed: the expansion of the analysis of the protocols to more standard protocols, the use of a network simulator for analysis of the network in each evaluation instance (protocol, security configuration), and lastly, the incorporation of the channel security option (TLS/SSL) as an evaluation instance as well.

References

[1] P. T. Eugster, P. A. Felber, R. Guerraoui, A.-M. Kermarrec, The many faces of publish/subscribe, ACM computing surveys (CSUR) 35 (2) (2003) 114–131.
[2] V. Karagiannis, P. Chatzimisios, F. Vazquez-Gallego, J. Alonso-Zarate, A survey on application layer protocols for the internet of things, Transaction on IoT and Cloud computing 3 (1) (2015) 11–17.
[3] OWASP top 10 2018 internet of things, https://www.owasp.org/index.php/OWASP_Internet_of_Things_Project, accessed: 2019-12-03.
[4] S. Deep, X. Zheng, A. Jolfaei, D. Yu, P. Ostovari, A. Kashif Bashir, A survey of security and privacy issues in the internet of things from the layered context, Transactions on Emerging Telecommunications Technologies (2020) e3935.
[5] S. Yi, Z. Hao, Z. Qin, Q. Li, Fog computing: Platform and applications, in: 2015 Third IEEE Workshop on Hot Topics in Web Systems and Technologies (HotWeb), IEEE, 2015, pp. 73–78.
[6] N. Naik, Choice of effective messaging protocols for iot systems: Mqtt, coap, amqp and http, in: 2017 IEEE international systems engineering symposium (ISSE), IEEE, 2017, pp. 1–7.
[7] J. Dizdarević, F. Carpio, A. Jukan, X. Masip-Bruin, A survey of communication protocols for internet of things and related challenges of fog and cloud computing integration, ACM Computing Surveys (CSUR) 51 (6) (2019) 116.
[8] M. Zorkany, K. Fahmy, A. Yahya, Performance evaluation of iot messaging protocol implementation for e-health systems, Performance Evaluation 10 (11) (2019).
[9] Y. Ji, B. Shao, J. Chang, G. Bian, Flexible identity-based remote data integrity checking for cloud storage with privacy preserving property, Cluster Computing (2021) 1–13.
[10] L. Loffi, C. M. Westphall, L. D. Gründner, C. B. Westphall, Mutual authentication with multi-factor in iot-fog-cloud environment, Journal of Network and Computer Applications 176 (2021) 102932.
[11] J. Li, J. Jin, L. Lyu, D. Yuan, Y. Yang, L. Gao, C. Shen, A fast and scalable authentication scheme in iot for smart living, Future Generation Computer Systems 117 (2021) 125–137.
A Comparative Qualitative and Quantitative Analysis of the Performance of Security Options for Message Protocols

International Conference on Ubiquitous and Future Networks (ICUFN), IEEE, 2013, pp. 810–815.

T. Yokotani, Y. Sasaki, Transfer protocols of tiny data blocks in iot and their performance evaluation, in: 2016 IEEE 3rd World Forum on Internet of Things (WF-IoT), IEEE, 2016, pp. 54–57.

Y. Sueda, M. Sato, K. Hasuiche, Evaluation of message protocols for iot, in: 2019 IEEE International Conference on Big Data, Cloud Computing, Data Science & Engineering (BCD), IEEE, 2019, pp. 172–175.

A. Hazra, M. Adhikari, T. Amgoth, S. N. Srirama, A comprehensive survey on interoperability for iot: Taxonomy, standards, and future directions, ACM Computing Surveys (CSUR) 55 (1) (2021) 1–35.

L. Babun, K. Denney, Z. B. Celik, P. McDaniel, A. S. Ulugac, A survey on iot platforms: Communication, security, and privacy perspectives, Computer Networks 192 (2021) 108040.

P. K. Donata, S. N. Srirama, T. Amgoth, C. S. R. Annavarapu, Survey on recent advances in iot application layer protocols and machine learning scope for research directions, Digital Communications and Networks (2021).

M. Aboubakar, M. Kellil, P. Roux, A review of iot network management: Current status and perspectives, Journal of King Saud University-Computer and Information Sciences (2021).

J. Wytrybowicz, K. Cabaj, J. Krawiec, Messaging protocols for iot systems—a pragmatic comparison, Sensors 21 (20) (2021) 6904.

A. Akinwole, E. Faturoti, Biological performance of african catfish (clarias gariepinus) cultured in recirculating system in ibadan, Aquacultural engineering 36 (1) (2007) 18–23.

T. Hecht, et al., A review of on-farm feed management practices for north african catfish (clarias gariepinus) in sub-saharan africa, On-farm feeding and feed management in aquaculture (2013) 463–479.

J. M. Kapetsky, A strategic assessment of warm-water fish farming potential in Africa, no. 27, Food & Agriculture Org., 1994.

O. Olanye, S. Ashley-Dejo, E. Fako, N. Ikeweinwe, W. Alegbeleye, F. Asaolu, O. Adelaja, Assessment of socio-economic analysis of fish farming in oyo state, nigeria, Global Journal of Science Frontier Research Agriculture and Veterinary 13 (9) (2013) 45–55.

T. F. Ribeiro-Neto, A. H. G. da Silva, I. M. Guimãres, M. V. T. Gomes, Piscicultura familiar extensiva no baixo são francisco, estado de sergipe: brazil/extensive fish farming in lower são francisco river basin in sergipe state, brazil, Acta of Fisheries and Aquatic Resources 4 (1) (2016) 62–69.

M. Karim, D. C. Little, M. S. Kabir, M. J. Verdegem, T. Telfer, M. A. Wahab, Enhancing benefits from polycultures including tilapia (oreochromis niloticus) within integrated pond-dike systems: a participatory trial with households of varying socio-economic level in rural and peri-urban areas of bangladesh, Aquaculture 314 (1-4) (2011) 225–235.

L. d. S. Cardoso, Bloom of noctiluca scintillans (macartney) kofoid & swezy (dinophyceae) in southern brazil, Brazilian Journal of Oceanography 60 (2) (2012) 265–268.

S. Priya, D. J. Inman, Energy harvesting technologies, Vol. 21, Springer, 2009.

J. N. McAteer, M. I. Malik, Z. Baig, P. Hannay, Security vulnerabilities and cyber threat analysis of the amqp protocol for the internet of things (2017).

P. Rubio-Conde, D. Villarán-Molina, M. García-Valls, Measuring performance of middleware technologies for medical systems: Ice vs amqp, ACM SIGBED Review 14 (2) (2017) 8–14.

M. Carandell Widmer, D. Toma, J. d. Río Fernandez, K. Ganchev, J. Peudennier, Evaluation of sigfox lpwan technology for autonomous sensors in coastal applications, Instrumentation Viewpoint (20) (2018) 36–37.

K. Mekki, E. Bajic, F. Chaxel, F. Meyer, Overview of cellular lpwan technologies for iot deployment: Sigfox, lorawan, and nb-iot, in: 2018 ieee international conference on pervasive computing and communications workshops (percom workshops), IEEE, 2018, pp. 197–202.

Wesley dos Reis Bezerra is a Professor at the Federal Institute of Santa Catarina, Campus Rio do Sul, and PhD candidate at PPGCC, UFSC. He holds his Master in Engineering and Knowledge Management from EGC, UFSC. He holds a Bachelor of Information System from INE, UFSC, and a Bachelor in Business Administration from Estácio de Sá University.

Fernando Luiz Koch is a Visiting Researcher at PPGCC, UFSC, and Data & AI Solution Architect with IBM. He dedicated the past 25 years of his career in advancing Data & AI and Cloud Computing. He holds a PhD in Computer Sciences (2009) from the Utrecht University, with a thesis about the use of AI to improve Mobile Services; M.Sc. and B.Sc. in Computer Sciences from INE, UFSC. He published 6 co-edited books, 35+ patents and 70+ papers.

Carlos Becker Westphall is Full Professor at the Federal University of Santa Catarina since 1993, acting as leader of the Network and Management Group, founder and supervisor of the Network and Management Laboratory. PhD in Informatics, specializing in “Network and Service Management” at the Université de Toulouse (Paul Sabatier), France in 1991. Master in Computer Science in 1988 and Electrical Engineer in 1985, both from UFRGS. He is the author and / or co-author of more than 470 publications. In 2011 he received the “Awarded IARIA Fellow”. He is (and was) a member of the Editorial Board of more than a dozen journals. Serves (and has served) as a member of the organizing and / or program committee for hundreds of conferences. He has experience in Computer Science and Telecommunications, with emphasis on Administration and Management of Networks and Services, acting mainly on the following themes: security, autonomic computing, cloud computing and Internet of Things. He founded the LANOMS conference (Latin American Network Operations and Management Symposium). He also provided services: for IEEE acting on CNOM (Committee on Network Operation and Management); for IFIP acting in ”WG6.6 - Management of Networks and Distributed Systems”; to Elsevier as editor of COMNET (Computer Networks Journal); to Springer as senior editor at JNSM (Journal of Network and Systems Management) and to IARIA (International Academy, Research, and Industry Association) as Latin America - IARIA Liaison Board Chair.