Journal of Universal Computer Science, vol. 28, no. 1 (2022), 54-79
submitted: 16/7/2021, accepted: 6/12/2021, appeared: 28/1/2022 CC BY-ND 4.0

Middleware for the Internet of Things: a systematic literature review

Rodolfo Medeiros
(Universidade Federal Rural do Semi-Árido, Mossoró, RN, Brazil
https://orcid.org/0000-0001-9337-7357, rodolfo.felipe@ufersa.edu.br)

Sílvio Fernandes
(Universidade Federal Rural do Semi-Árido, Mossoró, RN, Brazil
https://orcid.org/0000-0001-7300-8423, silvio@ufersa.edu.br)

Paulo G. G. Queiroz
(Universidade Federal Rural do Semi-Árido, Mossoró, RN, Brazil
https://orcid.org/0000-0003-3993-0208, pgabriel@ufersa.edu.br)

Abstract: The Internet of Things (IoT) emerged to describe a network of connected things on a large scale to offer services to a large number of applications in different environments and domains. Middleware is software that seeks to facilitate the management and communication of all these things, providing the necessary functionalities to manage things, to discover, to compose services, and perform communication. For this reason, several proposals for middleware solutions for IoT have been developed. In this article, we conducted a systematic review of the literature to bring together middleware solutions for IoT, identifying the requirements and communication protocols used. In addition, we present some gaps and directions for future research in the development of IoT middleware.

Keywords: Internet of Things, Middleware, Communication Protocols, Middleware Requirements
Categories: A.1, C.2.4, D.2.11, D.2.12
DOI: 10.3897/jucs.71693

1 Introduction

The term Internet of Things (IoT) is credited to Kevin Ashton because in 1999 he began a presentation entitled “That ‘Internet of Things’ Thing” [Kramp et al., 2013]. However, this reality was also predicted by Mark Weiser, when he proposed the idea of an adaptive, intelligent world with the background completely omnipresent [Weiser, 1991]. The acronym ”IoT” refers to the Internet of Things. The term ”Internet” comes from the global network infrastructure with scalable, configurable resources based on communication protocols. The ”Things” are physical objects or devices, virtual objects (devices or information) which have an identity and attributes [Bandyopadhyay et al., 2011]. Therefore, IoT refers to objects and systems that may constitute communication and share information through the internet.

Sensors and actuators are fundamental components in the Internet of Things. Sensors are able to measure and respond to physical stimulus, similar to some structures found in the human body, such as the retina or the voice patterns. Also, are able to detect changes in their location, position, orientation, elevation, speed, or distance of movement. On the other hand, actuators are the response mechanisms that make actions and effects in the context that are inserted [Hassan, 2018]. Actuators are able to make changes in the chemical composition of an entity and respond to altered levels of heat, light, or sound detected by the sensors [Bunz and Meikle, 2017]. Therefore, the function of these sensors and actuators is to detect, record, act on changes and share information.

The goal of IoT is to achieve synergy between different Things through the Internet. This signifies that they must interoperate and communicate automatically to provide innovative services [Hassan et al., 2018]. Thus, standardization is necessary to ensure that IoT platforms allow different systems to successfully interoperate. Although, several application protocols have become available, amongst which are: CoAP, MQTT, AMQP, and HTTP/REST [Sztajnberg et al., 2018], each with its own characteristics and objectives. Moreover, several options are available on the physical layer for wireless network communication, such as Wi-Fi, LoRa, ZigBee, BLE. Any solution for IoT should be able to communicate with different protocols or be scalable to include other forms of communication. Thus, in order to communicate, for different devices that use different protocols and networks, an intermediate layer called middleware is needed. Middleware is an interface between the hardware layer and the application layer, responsible for interacting with devices and for managing information [Chaqfeh and Mohamed, 2012]. The role of middleware is to present a unified programming model for interacting with devices. Moreover, middleware for IoT is responsible for masking the problems of heterogeneity and distribution that exist between the IoT devices.

Based on the above, the following paper brings some contributions. First, we define a systematic literature review protocol for conducting this research and for future researches. In this way, we avoid any erroneous assumptions or biases, in addition, the protocol can be replicated and used as a basis for review IoT middleware. Second, we perform the systematic review, according to the protocol, that aims to address IoT middleware used in applications, their architectures, covered requirements, and communication protocols. Furthermore, we analyze the application layer protocols that each middleware is using. Finally, we present the main research gaps to conduct future work on developing middlewares for IoT.

The remainder of this paper is organized as follows: section two presents the related works; section three defines the SLR protocol; section four discusses the threats to work validation; section five presents the conduct of the review; in section six, the results are presented and discussed. Lastly, section seven presents the conclusions of this systematic literature review.

2 Related Work

The IoT has become an area of research with the possibility of developing in other areas to solve different problems. Thus, considerable research has been conducted on middleware to assist in the construction of applications in the IoT. The types of middleware may be different, but the vast majority have mechanisms to manage devices or collect data. However, technological advances in products bring new opportunities and challenges. Therefore, it is essential to have works that periodically collect the newest middleware and the recent advancements. In this section, we relate other works with each purpose similar to our, exploring the literature about middleware applied to IoT.

The works Rathod and Chowdhary, 2018, Razzaque et al., 2016, Ngu et al., 2017, Farahzadi et al., 2018, Vikash et al., 2020, and Raza et al., 2021 bring together a substantial amount of middleware and explain its characteristics. The papers address the requirements of middleware for IoT, of which interoperability requirement is the most cited. Razzaque et al., 2016 provide a more comprehensive overview as well as a high-level view of the challenges of middleware in the IoT. Raza et al., 2021 presents a work similar to that of Razzaque et al., 2016. Both articles focus on non-functional requirements. We note that although Raza article was published later, the author did not analyze nor included Razzaque work. Vikash et al., 2020 define the relationship between WSN and IoT, explaining the challenges of WSN middleware in IoT. However, Ngu et al., 2017 addressed an example of application in collecting data from an environment that has an easy connection. In addition, Farahzadi et al., 2018 addressed the use of middleware in the cloud of things. Table 1 compares a list of surveys, showing the key aspects that characterize each work.

Reference	Year	Citation	Middleware	Architecture	Communication Protocols	Requirements	Systematic Review
Razzaque et al., 2016	2016	430	✓	✓	✗	✓	✗
Ngu et al., 2017	2017	152	✓	✓	✗	✗	✗
Farahzadi et al., 2018	2018	52	✓	✓	✗	✓	✗
Rathod and Chowdhary, 2018	2018	1	✓	✓	✗	✓	✗
Vikash et al., 2020	2020	1	✓	✓	✗	✓	✗
Raza et al., 2021	2021	0	✓	✗	✗	✓	✗
our work	2021	-	✓	✓	✓	✓	✓

Table 1: Comparison among related works

Finally, the related works do not explain, in-depth, the application-layer protocols used by IoT middleware. Also, the related works do not define a systematic review protocol for researching those middleware. Our work is bringing a protocol to be replicated in new research to update the state-of-the-art on middleware for the internet of things. Therefore, we intend to show the current research scenario in this area, reducing bias by formalizing a systematic review.

This paper is intended as complementary to the surveys already available in the literature. Thus, we bring the latest middleware that is related and seeks to solve a problem in the field of IoT. Thus, we identify the requirements that these works assign in their definition of architecture. We analyze architecture based on definitions, validation, and problem-solving. Also, we identified the protocols that were used in the middleware validation process.

3 Systematic Literature Review Protocol

An informal review of the literature on middleware for IoT is subject to investigation bias, such as in the formulation of research questions, collection, evaluation, interpretation, and presentation of data, which may be influenced by the personal interests of the researchers involved, thereby ultimately leading to poor, untrustworthy results [Biolchini et al., 2005]. Informal reviews of the literature, which are conducted without an a priori planning, are characterized as being incomplete, not repeatable, unreliable, and for which the quality is dependent on the experience of the researchers [Snyder, 2019]. In contrast, a systematic literature review (SLR) is conducted through a process composed of a well-defined sequence of phases, and presents a fair assessment on a research topic, using a rigorous, reliable, and auditable review form [Biolchini et al., 2005].

The main objective of the systematic review protocol is to reduce errors that may occur while conducting a systematic literature review [Gough, 2017]. The protocol defined in this research follows the process defined by Biolchini et al., 2007 and uses some of the guidelines defined by Kitchenham, 2004. Our main reference focuses on a systematic literature review for Software Engineering because this systematic review focuses not only on IoT but also on the Software Engineering used to build the IoT applications for which the middleware was built. Thus, the protocol encompasses objectives, research questions, search strategies, study selection criteria, study selection process, and data extraction procedures, as presented in Figure 1.

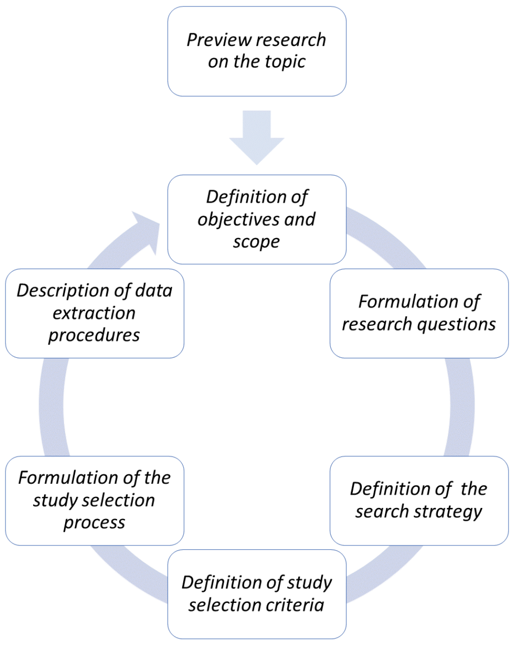
Preview research on the topic
Definition of objectives and scope
Formulation of research questions
Definition of the search strategy
Definition of study selection criteria
Formulation of the study selection process
Description of data extraction procedures

Figure 1: Planning stages of the systematic literature review protocol

3.1 Objectives and Scope

The main objective of this systematic review is to identify middleware and middleware architectures designed to assist the development of applications for the Internet of Things. It is also important to identify the communication requirements and protocols used by these middleware. Therefore, the following secondary objectives have been defined:

- **Objective 1:** to identify the middleware used in applications for the Internet of Things.

- **Objective 2:** to understand the architecture of these middleware
- **Objective 3:** to identify the middleware requirements used.
- **Objective 4:** to identify the communication protocols used by the middleware.
- **Objective 5:** to compare the identified middleware in relation to architecture, requirements and communication protocols.

At the end of the research, and the expected result is to obtain a state-of-the-art definition for the Internet of Things in relation to existing middleware. In accordance with the objectives of the SLR, the following items were selected as specifics for the scope of the review:

Scope Specificities: these scope specificities, presented below, aims at defining the research questions semantics, in other words defining these questions range. Intervention describes what is going to be observed in the context of the systematic review; Control is the initial data set that we possessed before we initiate the systematic review protocol and we use it to evaluate the search string, by verifying if the search performed found, at least, the papers we already knew; Population refers to the group that will be observed by the intervention; Outcomes aim to list the results of this systematic review; and, Application defines the roles, professional types, or application areas that will benefit from outcomes.

- Intervention: work that describes middleware in the context of device (Thing) interconnection.
- Control:
  - Georgi, N; Corvol, A.; Le Bouquin Jeannès, R. Middleware Architecture for Health Sensors Interoperability. IEEE Access, volume 6, 2018.
  - Jeon, S., Jung, I. MinT: Middleware for Cooperative Interaction of Things. Sensors, volume 17, issue 6, 2017.
  - Lalanda, P.; Morand, D.; Chollet, S.Autonomic Mediation Middleware for Smart Manufacturing. IEEE Internet Computing, volume 21, 2017.
- Population: research works or practical applications in the field of IoT.
- Outcomes: to provide information, such as architecture, requirements, and communication protocols about designed and implemented IoT middleware.
- Application: to developer and apply knowledge about IoT middleware by industry, researchers, teachers, software engineers, and freelance professionals interested.

3.2 Research Questions

In order to satisfy the objectives of the systematic literature review, some research questions were formulated. Hence, a primary question and some secondary questions were defined:

- **Primary question (PQ)**: which middleware were built for or used in IoT applications?

- **Secondary question (SQ1)**: which architecture were used to build these middleware?
- **Secondary question (SQ2)**: which requirements were met by these middleware?
- **Secondary question (SQ3)**: which communication protocols do these middleware support?

3.3 Search Strategy

In order to perform an exhaustive search for primary studies, the search strategy consisted of an online search in the five major digital libraries that present a strong emphasis on computer science. In addition, it was necessary to define the keywords and their synonyms to construct the search string to be applied in each selected platform. Lastly, we defined the parameters and specificities for executing the search in each repository.

- **Digital Libraries:** IEEE Xplore Digital Library, ACM Digital Library, ScienceDirect, SCOPUS and Web of Science

Before building a standard search string, we defined the keywords related to the research questions. In order to broaden the search scope, we defined synonyms for each keyword as presented in Table 2.

Keyword	Synonyms
Internet of Things	IoT, Future Internet, Web of Things, WoT and Industry 4.0
Middleware	Middle Software

Table 2: Keywords and synonyms related to the research questions

With the keywords defined, the standard search string was created using the logical operators. Synonyms, abbreviations, and plurals of keywords were also included:

- **Standard Search String**: "("internet of thing" OR "internet of things" OR "iot" OR "web of thing" OR "web of things" OR "wot" OR "future internet" OR "industry 4.0") AND ("middleware" OR "middle software")"

The search string was validated and refined through a pilot search conducted at the IEEE Xplore Digital Library and was adapted for the selected platforms. The selected digital libraries have different characteristics including syntax. For this, the defined standard string was adapted as presented in Table 3.

Due to certain characteristics, the difficulties and observations for each platform have been documented and presented below.

- **IEEE Xplore Digital Library**: the search was conducted using the advanced search page and command search option. We restricted the search to titles and abstracts of publications, by selecting the option " Metadata Only" in the advanced search.
- **ACM Digital Library**: we restricted the search to publication titles and abstracts, by adding the words "recordAbstract:" and "cmdlTitle:" before the search terms. Also, observe that the string was entered in the special "show query syntax" field of the platform advanced search page.

Library	**Search String**
IEEE	(“internet of thing” OR “internet of things” OR “iot” OR “web of thing” OR “web of things” OR “wot” OR “future internet” OR “industry 4.0”) AND (“middleware” OR “middle software”)
ACM	(recordAbstract:(“internet of thing” “internet of things” “iot” “web of thing” “web of things” “wot” “future internet” “industry 4.0”) AND recordAbstract:(middleware “middle software”)) OR (acmdlTitle:(“internet of thing” “internet of things” “iot” “web of thing” “web of things” “wot” “future internet” “industry 4.0”) AND acmdlTitle:(middleware “middle software”))
Science Direct	(“internet of thing” OR “internet of things” OR “iot” OR “web of thing” OR “web of things” OR “wot” OR “future internet” OR “industry 4.0”) AND (“middleware” OR “middle software”)
SCOPUS	TITLE-ABS-KEY((“internet of thing” OR “internet of things” OR “iot” OR “web of thing” OR “web of things” OR “wot” OR “future internet” OR “industry 4.0”) AND (“middleware” OR “middle software”)) AND (LIMIT-TO(SUBJAREA,”COMP”))
Web Of Science	(TS=((“internet of thing” OR “internet of things” OR “iot” OR “web of thing” OR “web of things” OR “wot” OR “future internet” OR “industry 4.0”) AND (“middleware” OR “middle software”))) OR (TI=((“internet of thing” OR “internet of things” OR “iot” OR “web of thing” OR “web of things” OR “wot” OR “future internet” OR “industry 4.0”) AND (“middleware” OR “middle software”))) AND SU=(“Computer Science”)

Table 3: Standard search string adapted for each digital library

– **ScienceDirect**: we restricted the search by titles, abstracts, and keywords adding the string in the “Title, abstract or author-specified keywords” on the advanced search page.

– **SCOPUS**: we restricted the search by titles, abstracts and keywords adding the term “TITLE-ABS-KEY” at the beginning of the search string. Also, we adding a term specifying the search area for Computer Science (LIMIT-TO (SUBJAREA, “COMP”)).

– **Web Of Science**: we restricted the search using the “TS” tag for topics and “IT” tag for titles. Also, we adding the “SU” tag to restrict the search by the Computer Science area.

3.4 Study Selection Criteria

The located papers were assessed by selection criteria defined in two categories: inclusion and exclusion. The inclusion criteria aimed to define which papers were directly related to the research questions. The exclusion criteria aimed to exclude papers unrelated to the research. For the inclusion of a paper in each phase of analysis, it is enough that it meets at least one inclusion criteria. Consequently, papers that meet at least one exclusion criteria were discarded.

3.4.1 Inclusion Criteria

The primary studies included needed to meet at least one of the following four inclusion criteria:

– Inclusion Criteria (IC1): papers that present and apply middleware for the internet of things.

– Inclusion Criteria (IC2): papers that present proposals for middleware requirements for the internet of things.

– Inclusion Criteria (IC3): papers that present communication protocols for middleware applied for the internet of things.

– Inclusion Criteria (IC4): papers that present a systematic review on middleware for the internet of things.

3.4.2 Exclusion Criteria

Studies were excluded based on the following exclusion criteria: EC1, EC2, EC3, and EC4 are opposed to the inclusion criteria, in other words, are the criteria that exclude articles that are not about middleware, architecture, requirements, communication protocols, or related works; we defined EC5 and EC7 because searching for short papers is resource-intensive; and, the information in short papers may not be as dependable as full papers, especially because they usually address preliminary results; EC8 is because this work is interesting and intends to gather the latest middleware and finding gaps that are still open in middleware for the internet of things; and, EC10 is because we are looking for the most important works about middleware for IoT. The other criteria are introduced to refine the search for complete papers.

– Exclusion Criteria (EC1): papers that present middleware, but are not applicable in the context of the internet of things.

– Exclusion Criteria (EC2): papers that present requirements, but are not applicable for middleware in the context of the internet of things.

– Exclusion Criteria (EC3): papers that present communication protocols, but are not for middleware in the context of the internet of things.

– Exclusion Criteria (EC4): papers that present a systematic review, but do not cover middleware for the internet of things.

– Exclusion Criteria (EC5): incomplete, unavailable, or duplicated studies.

– Exclusion Criteria (EC6): papers written in languages other than English.

– Exclusion Criteria (EC7): papers with less than five pages.

– Exclusion Criteria (EC8): papers published before 2016.

– Exclusion Criteria (EC9): old versions of the papers.

– Exclusion Criteria (EC10): impact factor less than 0.9.

3.4.3 Quality Criteria

Selected primary studies were assessed based on the quality criteria described below. We defined quality criteria to measure the quality of the works found on middleware for the internet of things, in case anyone interested wants to go deeper into a given article. We observe that there is a trade-off between objectivity and subjectivity in quality assessment, but a balance was sought. The result of the assessment is presented in results and discussions (Section 6).

– Quality Criteria (QC1): work with an impact factor between 0.9 and 1.9 (1 point), between 2.0 and 2.9 (2 points), between 3.0 and 5.0 (3 points), and greater than 5.0 (5 points).

– Quality Criteria (QC2): middleware applied in building applications to solve real problems (5 points).

– Quality Criteria (QC3): middleware applied in building applications to assist precision agriculture or zootechnics (5 points).

– Quality Criteria (QC4): work that provides the code developed (3 point).

3.5 Study Selection Procedures

The study selection process was divided into four phases: In the first phase, searches were performed using the standard string adapted for each digital library defined in the search strategy, the returned papers were cataloged and duplicate papers were removed; in the second phase, the abstracts of the papers cataloged in the previous phase were read carefully and measured based on the selection criteria specified in the study selection strategy; in the third phase, the selected texts with more than four pages were read in full and re-assessed according to the inclusion and exclusion criteria; and lastly, the papers were considered either valid or irrelevant to the research objectives. The phases of study selection are described in chronological order.

– Phase 1 - Paper Collection: conducting searches in the digital libraries with the defined string, cataloging and removing duplicate papers.

– Phase 2 - Initial Selection: reading titles and abstracts of papers with an assessment based on the selection criteria.

– Phase 3 - Final Selection: reading full papers with more than four pages approved in the previous phase and reapplying the selection criteria.

– Phase 4 - Selection Validation: validation or invalidation based on the research objectives.

It is important to note that in the process we carried out, there one master's degree student evaluating the works. During the process, two researchers assessed the articles whose abstracts generated doubts. In the last stage, the assessments were reviewed by the researchers.

Fields	Description
Paper Title	Title of the paper studied
Year	Publication year the paper
Publication Title	Name of the conference, journal, or book in which the paper was published
Impact Factor	Identification of the impact factor of the source
Pages Number	Number of pages in the paper
Paper Language	Language in which the paper was written
Middleware	Description of the presented middleware
Architecture	Description of the architecture used
Requirements	Description of the functional and non-functional middleware requirements
Communication Protocols	Description of the applied communication protocols
Advantages	Description of the advantages of the middleware
Limitations	Description of the middleware limitations
Validation	Description of the middleware validation process

Table 4: Attributes of the data extraction form and their descriptions

3.6 Data Extraction and Presentation of Results

For the data extraction procedure, a data extraction form was defined in order to assist in the extraction and for including information on the works read in full and validated according to the selection criteria. The form contained the necessary fields to answer the research questions. A data extraction form was completed for each paper, including the title, year of publication, source of publication, impact factor, middleware, architecture, requirements found, application protocols used, advantages, limitations, and validation process. Table 4 presents an outline of the form.

The extracted data are presented below by means of a synthesis of the papers located, by presenting comparative tables and using graphs to summarize the outcomes as well as to answer the research questions. The conduction of this SLR was presented and illustrated using graphs and tables. Lastly, the keywords of the primary studies selected were applied to VOSViewer[1] [Eck and Waltman, 2010] software in order to build a network between the keywords. Thus, it is possible to understand the choice of keywords defined for this review.

4 Threats to Work Validation

Our goal is to research all the most recent middleware for the internet of things that achieve our inclusion criteria. We are trying to execute the research as widely as possible without losing focus, to reduce the gap between general middleware purpose and middleware for the internet of things. However, some relevant issues can threaten the results of this work, as listed below:

[1] https://www.vosviewer.com/

- Only papers indexed by the listed databases were collected. Therefore, if the paper of a specific conference has not been indexed in those databases, that paper was not included. To mitigate that threat we verified that the select database's index the most important journals and conferences related to computer science and IoT.
- Middleware that does not have published scientific work has not been included. Although we recognize that there red are some middleware available on the market, we understand that we could not have access to all the necessary information about that middleware. Also, we follow a well-defined research protocol to cover all of the most complete papers published to show the current research scenario on middleware for the internet of things.
- Short papers were not included in the results. Searching for short papers is resource-intensive; and, the information in short papers may not be as dependable as full papers, especially because they usually address preliminary results. Our goal is the latest middleware that has been validated in IoT applications.
- Articles published before 2016 were not included in the results. However, the related works already address the older middleware. We are interested in updating the current state of the middleware for IoT and finding gaps that are still open in middleware for the internet of things.

5 Conduction of Systematic Literature Review

This section presents the execution of the protocol defined in the previous section. Hence, the following items are presented and detailed: the search results, the paper selection process; and the results obtained with the SLR.

5.1 Paper Collection

Initially, searches were conducted in the selected databases with the strings adapted for each one of them. Figure 2 presents the number of papers collected per library, resulting in a total of 2516 papers. Duplicate papers were removed, thereby totaling 1450 individual papers during the first phase for the period between 2016 and April/2021.

5.2 Initial Selection

During the initial selection, papers were selected by applying the inclusion and exclusion criteria, from reading the titles and abstracts of the cataloged papers. A total of 1450 abstracts resulting from the search in the digital library were read. Figure 2 present the numbers of papers included and excluded for each platform. In total, 579 papers were selected.

5.3 Final Selection

In the last stage of the selection, the selected papers included in the previous phase were read in completion and the inclusion and exclusion criteria were reapplied, so as to select the most appropriate papers. Figure 2 presents the total of papers included and those excluded by the digital library exclusion criteria.

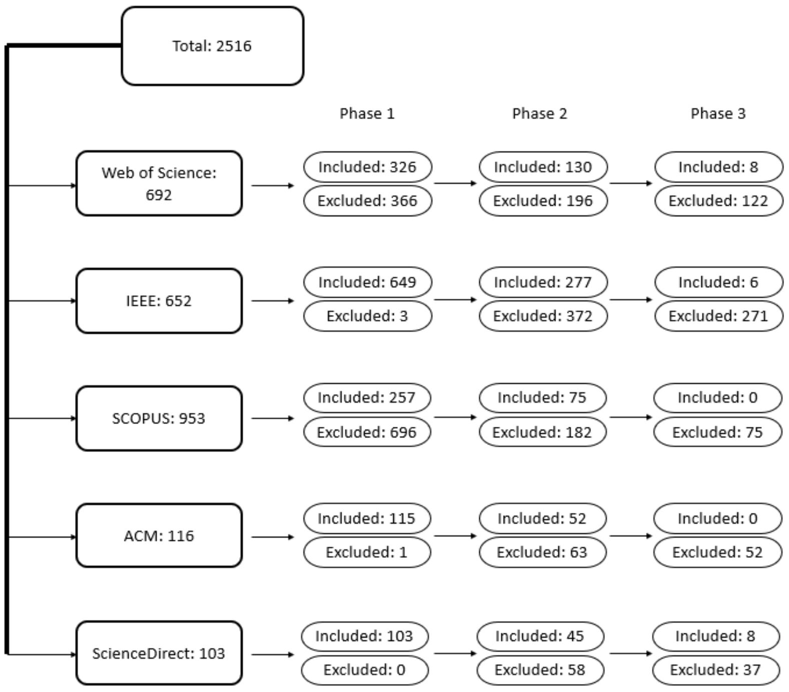

Figure 2: The inclusion and exclusion result in the SRL.

No paper indexed by the ACM and SCOPUS platforms were selected in this phase of the SLR. However, different paper platforms may index the same newspapers and conferences. For this reason, the same number of papers may be obtained by other platforms that index the same academic dissemination vehicles.

The validation of the papers was the last phase of executing the SLR. In this phase, the selected papers were validated, during the execution stages, by two senior researchers.

In Table 5, the selected studies may be qualified as follows: the ID of work, the author, the points assigned by the quality criteria and the total points. It may be noted that the selected papers were validated, and the list of papers remained unchanged.

6 Results and Discussions

6.1 PQ - which middleware were built for or used in IoT applications?

There was a total of 22 selected papers that presented middleware for the Internet of Things. We identify the papers by the authors and assign identifiers (Table 5). In addition, we evaluate the works with the quality criteria, as defined in Subsection 3.4.3. Two middleware were applied in agriculture. Furthermore, only three works provide their source code publicly. No work applied to agriculture is open source. Table 6 presents a summary of the selected middleware. The description is organized as follows: the name of the middleware, application of the middleware or area that the middleware proposes to solve, the architecture of the middleware, the requirements, the communication protocols,

ID	Author	QC1	QC2	QC3	QC4	Total
1	da Cruz et al., 2020	5	5	0	3	13
2	Trilles et al., 2020	3	5	5	0	13
3	Zhao et al., 2018	3	5	5	0	13
4	Antonić et al., 2016	5	5	0	0	10
5	Xie et al., 2021	5	5	0	0	10
6	Esposte et al., 2019	5	0	0	3	8
7	da Cruz et al., 2019	5	0	0	3	8
8	Georgi et al., 2018	3	5	0	0	8
9	Santos de Souza et al., 2019	3	5	0	0	8
10	Gomes et al., 2017	3	5	0	0	8
11	Alvisi et al., 2019	3	5	0	0	8
12	Alvaro et al., 2019	3	5	0	0	8
13	Lalanda et al., 2017	2	5	0	0	7
14	Bouloukakis et al., 2019	5	0	0	0	5
15	Benayache et al., 2019	5	0	0	0	5
16	Mohamed et al., 2017	3	0	0	0	3
17	Jeon and Jung, 2017	3	0	0	0	3
18	Lilis and Kayal, 2018	3	0	0	0	3
19	Dipsis and Stathis, 2020	3	0	0	0	3
20	Mesmoudi et al., 2018	2	0	0	0	2
21	Merlino et al., 2019	2	0	0	0	2
22	Shokrollahi and Shams, 2017	1	0	0	0	1

Table 5: Ranking of works selected based on quality criteria

and the physical layer used. In our research aims to list the most current middleware. There are middleware defined before 2016 that have already been presented, analyzed, or commented on related works.

While some authors mentioned that they were using some technologies, it was not possible to detect this use in their applications. The words indicated with the symbol "*" are the technologies that were not used in the applications for the "protocols" and "physical layer" columns.

Moreover, many of the works have not been applied in real environments. Another gap is related to the fact of the "interoperability" requirement since many authors mentioned that they developed middleware based on this requirement. However, it is clear that a reasonable number of protocols have not been assigned to solve the problem of device heterogeneity. Additionally, it may be observed that the physical layer has not been specified nor validated in real applications.

The keywords for each paper were inserted in VOSViewer to generate the network visualization graph (Figure 3). It is possible to identify a great correlation between IoT and middleware. We believe this correlation is strong due to the need for an infrastructure that facilitates the construction of more complex IoT applications. The proposed middlewares represent this infrastructure and are used in the construction of IoT applications, mainly for validation purposes. We note that these applications used for validation are responsible for other weaker correlations such as IoT and smart farming, smart cities, and biomedical

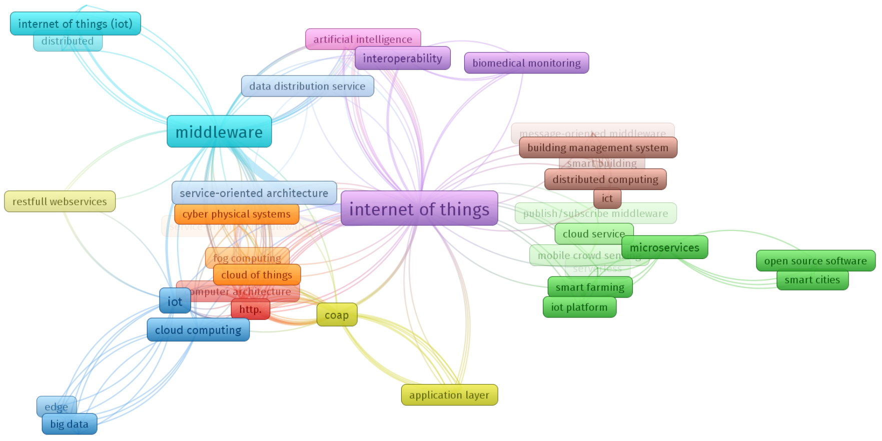

Figure 3: Network of keywords of selected works

monitoring. The selected works are discussed below.

In the work presented by Antonić et al., 2016, the authors presented the CUPUS middleware. The middleware was developed to acquire data from mobile device sensors in a flexible manner, with low energy consumption and real-time processing of Big Data streams. This came about in the interest of detecting and collecting data from mobile sensors to observe phenomena over a large geographical area. The architecture consists of two components: the cloud broker and the mobile broker. The middleware uses TCP/IP and Google Cloud Messaging (GCM) to send data to the cloud broker. The mobile broker communicates with devices via Bluetooth. The middleware is used in an application to check urban air quality.

The In.iot was presented by da Cruz et al., 2020 and is a middleware platform for IoT that presents an approach for IoT devices to connect and share data. In.iot uses microservices architecture and its first deployment was written in Java. The solution was assessed and demonstrated in comparison with some open-source alternatives, which considered the response times, the percentage of error requests, and packet size. Furthermore, the solution uses topic restrictions to increment security features for MQTT brokers. Currently supports MQTT, CoAP, and HTTP as application-layer protocols. In.iot was used in a smart energy meter application.

The MSOAH-IoT was presented by Mesmoudi et al., 2018. The middleware is based on a service-oriented architecture to deal with the heterogeneity issues. The middleware was developed using Java technology and uses the HTTP/REST protocol to communicate with devices. The solution was validated in three scenarios simulated: in the first one, the middleware was installed on a Raspberry Pi as a gateway to collect data from devices connected via Wi-Fi; in the second case, the installed middleware handle data coming from devices connected through Wi-Fi and Bluetooth; in the third one, was used a personal computer working as a gateway.

The CoIoT was presented by Santos de Souza et al., 2019. This middleware to assists in the monitoring of seed tests and was developed in partnership between UPEL, UFRS, and EMBRAPA. The middleware provides support for ubiquitous applications and mechanisms for managing IoT devices. It was developed by EXEHDA to add to the treatment of actuator and sensor devices, in addition to using the infrastructure resources

ID	Middle-ware	Application	Architec-ture	Requirements	Protocols	Physical Layer
1	In.iot	Smart Energy	Cloud	Interoperability, Scalability, Reliability	MQTT, CoAP, HTTP	-
2	SEnviro Connect	Smart Farming	Edge	Interoperability, Scalability, Reliability	HTTP, MQTT	2G/3G
3	-	Device Management	Cloud	Interoperability, Reliability	HTTP/REST, MQTT, COAP	WiFi, BLE*, ZigBee*, Z-Wave*, M-Bus*, CANBus*
4	CUPUS	Mobile Crowd Sensing	Cloud	Flexibility	HTTP/GCM	BLE
5	-	Sewage Treatment Stations	Edge	Interoperability	MQTT, HTTP	Modbus, RS485, RS422, RS232
6	InterSCity	Smart City	Cloud	Interoperability, Flexibility, Scalability, Reliability	HTTP/REST, MQTT*	-
7	Middle-Brigde	Different Aplication Protocols	Edge	Interoperability	MQTT, CoAP, DDS, WebSocket, HTTP/REST	-
8	-	e-Health	Edge	Interoperability, Reliability	-	WiFi, Bluetooth
9	CoIoT	Smart Lab	Edge	Context Management, Interoperability	HTTP/REST	WiFi
10	M-Hub/CDDL	Context Quality	Cloud	Context Management	MQTT	Bluetooth
11	SWaMM	Smart Watering	Cloud	Interoperability	HTTP/REST, MQTT	WiFi, M-Bus, LoRa
12	Thinger.io	Data Integration	Cloud	Interoperability, Connectivity	HTTP/REST, MQTT*, CoAP*	WiFi, SigFox*, LoRa*
13	Cilia	Smart Manufacturing	Cloud	Flexibility	HTTP/GCM	ModBus
14	DeXMS	Different Aplication Protocols	Edge	Interoperability	MQTT, CoAP, WebSocket, HTTP/REST	-

15	MSM	Device Management	Cloud	Interoperability, Privacy, Security	MQTT, HTTP, CoAP, WebSocket, DDS	-
16	SmartCityAware	Smart City	Cloud	Flexibility	HTTP/REST	WiFi
17	MinT	Device Management	Edge	Interoperability, Reliability	MQTT, CoAP, HTTP/REST	WiFi, Bluetooth, BLE, ZigBee
18	-	Smart Building	Edge	Interoperability	Socket	Wi-Fi, Ethernet
19	eVATAR+	Smart Home	Edge	Interoperability	HTTP/REST	-
20	MSOAH-IoT	Smart Home	Edge	Interoperability	HTTP/REST	WiFi, Bluetooth, 6LowPAN*, ZigBee*
21	-	Smart Environment	Cloud	Flexibility	HTTP/REST, AMQP	WiFi
22	RDS	Device Management	Edge	Interoperability, Scalability	DDS	-

Table 6: Summary of selected middleware

of the ubiquitous environment. The middleware uses the HTTP/REST protocol.

The Thinger.io was presented by Alvaro et al., 2019. The middleware provides a cloud service to connect devices to the internet for data integration. The middleware is exposed at work as open source, but in the repository, the source code is not available, and only a few APIs for cloud access are available. The middleware states that it is hardware-independent, however, it does not provide access by different hardware (LoRa, for example). It uses the protocols HTTP/REST, MQTT and CoAP. The middleware is validated in a meteorological application using WiFi and Arduino.

The MiddleBridge was presented by da Cruz et al., 2019. This is a gateway that offers a solution to mediate communication between devices that do not understand the same protocol. Understanding that all middleware must support other protocols, the gateway serves as a bridge, transforming MQTT messages, CoAP, DDS, and WebSocket for HTTP/REST and forwarding to applications.

Lilis and Kayal, 2018 presented a middleware without brokers to serve as an interface and to interconnect the digital and the physical for systems in smart buildings using ZeroMQ distributed messaging library. It is oriented towards messages between publish/subscribe with queues and service discovery. The nodes communicate in a peer-to-peer manner over computer networks: Wi-Fi and Ethernet. The solution was deployed on embedded hardware for validating the middleware functionalities and energy assessment.

The DeXMS was presented by Bouloukakis et al., 2019. This is a middleware built to serve as a bridge abstracting the communication between different protocols, which may be a direct bridge from one protocol to another or using two mediators, to go from an initial protocol to a final protocol, going through another protocol between them. It uses the communication protocols: MQTT, CoAP, HTTP/REST, and WebSocket. For this, DeXMS creates a protocol, called: DeX, which encapsulates each mentioned protocol.

The M-Hub/CDDL was presented by Gomes et al., 2017. This middleware is for managing context data (acquisition, processing, and distribution) abstracting the complexity of the context, and providing the appropriate service. It is divided into two main layers: MHUB and CDDL, and uses the MQTT communication protocol. It is applied to a case study involving the development of a mobile application for remote monitoring of patients.

The eVATAR+ was presented by Dipsis and Stathis, 2020 and is a middleware that seeks to incorporate AI resources into existing sensor-actuator networks or IoT infrastructures making the offered settings services smarter. eVATAR+ uses Java server technologies enhanced by mediator functionality providing interoperability and maintainability support. The solution is a monolithic architecture software that was divided into three assignments: controller layer, business layer, and persistence layer. Currently offer the HTTP / REST API as an application-layer protocol to communicate with devices that offer this same communication interface. eVATAR+ was used in a smart home scenario with Google Nest and Jade Agent.

The InterSCity was presented by Esposte et al., 2019 and is an open source middleware platform developed and maintained by the INCT/USP institution. The project was developed to support the challenges related to the Smart Cities software infrastructure, with a focus on High-Performance Networks and Distributed Computing, Software Engineering, Data Analysis, and mathematical modeling. The platform offers and makes available all web-based services, such as service management, heterogeneous IoT resources and supports the discovery of city resources based on context data, to store and process data and intermediate action commands. The middleware uses the HTTP/REST and MQTT protocols.

The MinT was presented by Jeon and Jung, 2017. This is a distributed middleware focused on the cooperative interaction of things that provides all the basic resources to operate a distributed IoT device. It aims at energy efficiency and responding quickly to requests, allowing an efficient interaction between IoT devices, taking into account power consumption, network delays, and reliability. It uses the MQTT and CoAP protocols.

The MSM was presented by Benayache et al., 2019 and is a middleware inspired by the architecture of Artificial Neural Networks and microservices, obtaining a dynamic interaction between services and supporting some communication protocols: MQTT, HTTP, CoAP, WebSocket, DDS. MSM was written in Python language programming. The solution was compared with other middleware solutions in terms of different qualitative and quantitative metrics: intelligence, service registration, service discovery, and latency.

The SWaMM was presented by Alvisi et al., 2019 and is an interoperable, wireless middleware, based on the Edge Computing paradigm for preprocessing and gaining efficiency in time. Its main focus is to interact with several types of smart water consumption meters, operating with some communication protocols. The platform uses the HTTP/REST protocol in the API and MQTT between the gateway and the data manager.

The SEnviro Connect was presented by Trilles et al., 2020 and is an IoT cloud platform to encompass the physical and cloud scope to managing IoT devices and analyzing the data produced. The solution uses microservices architecture divided into four parts: persistence, analytics, brokers, and API. The layers use some open source services and private services. The platform used the RabbitMQ broker for communication using MQTT. Furthermore, uses Micro Mu for analytics, InfluxDB for persistence, Kapacitor for alerts, and Firebase for auth, storage, and messaging. SEnviro Connect was used in a smart farming application using IoT Devices called SEnviro Nodes.

Georgi et al., 2018 presented middleware in the form of a software library, capable of communicating with a set of sensors in order to access any data. This middleware is intended for use by third-party developers, healthcare applications, or providers to retrieve patient data, allowing the user to ensure interoperability with a maximum number of sensors applied to health with minimal effort. The middleware was developed for Android to communicate with the devices using Wi-Fi and Bluetooth physical layers.

Merlino et al., 2019 presented a middleware platform extending the current functionality of the existing platforms OpenStack and Stack4Things, through which resource containers at the Edge, Fog, and Cloud levels may be discovered, combined, and provisioned for end-users and applications to facilitating and orchestrating the download processes. The middleware uses the HTTP/REST and AMQP communication protocols. The solution was evaluated with a case study on an intelligent surveillance system.

Zhao et al., 2018 presented a service-oriented middleware using HTTP/REST, MQTT and CoAP compatible with oneM2M. Variable and McSugar techniques were created to use heterogeneous device resources and to perform uniform interactions between applications. These techniques also provide the basis for cutting-edge IoT research, such as semantic reasoning and Fog Computing. The middleware was designed to facilitate the creation of new applications and tests.

The Cilia was presented by Lalanda et al., 2017 and was designed for smart factories. The architecture is based on services. It provides dynamics and resource autonomy, allowing great flexibility in runtime with minimal human intervention. It was applied in a real case study that involved the supervision and maintenance of pumping stations.

The SmarCityAware was presented by Mohamed et al., 2017. This was a service-oriented middleware platform designed specifically to use CoT and Fog Computing in order to support the development and operation of smart city applications. The main objective of SmartCityWare is to provide a virtual environment used to develop and deploy smart city applications. SmartCityWare consists of a set of services and a multi-agent runtime environment. The middleware uses the HTTT/REST protocol.

The RDS was presented by Shokrollahi and Shams, 2017 and is a middleware based on an approach derived from service-oriented architecture, called Rich Services. RDS is an approach that exploits DDS middleware to publish/subscribe, centralize data, function in real-time, and loosely coupled communication between devices in order to enable a scalable and dynamic integration of heterogeneous devices in IoT.

Xie et al., 2021 presented a multilayer IoT middleware for providing flexible IoT device access. The proposed solution is based on an IoT knowledge graph, which aims at reducing the heterogeneity among IoT devices. IoT knowledge graph consists of three parts: an IoT base ontology; an IoT domain ontology; and, an IoT instance. The middleware was divided into two middleware L1 and L2. The L1 IoT middleware only stores the root node and the static attributes of a device while the dynamic data of the device are stored in the L2 IoT middleware. Supports MQTT and HTTP as application layer protocols for communication between L1 and L2 structures. Middleware was used in remote monitoring of rural sewage treatment stations.

6.2 SQ1 - which architecture were used to build these middleware?

Two main architectures addressed in middleware have been identified. Table 7 presents the architecture and what the middleware is targeting with this architecture. In addition, each architecture is presented and discussed below.

Architecture	Middleware
Cloud-Based	In.iot, CUPUS, InterSCity, MSM, Zhao *et al.*, M-Hub/CDDL, Thinger.io, SWaMM, Cilia, SmartCityAware, Merlino *et al.*
Edge-Based	Xie *et al.*, eVATAR+, SEnviro Connect, MiddleBrigde, DeXMS, Lilis *et al.*, Georgi *et al.*, CoIoT, MinT, MSOAH-IoT, RDS

Table 7: Architectures addressed by middleware for IoT

6.2.1 Cloud-Based

Most of the middleware are based on the cloud, offering a Platform as a Service (PaaS) or a Service as a Service (SaaS). These solutions have some common features in this regard. Thus, the platforms offer mechanisms that cannot be offered on the devices due to the restricted nature of the resources that are present in the IoT. These limitations prevent offering complete services. Thus, the middleware addresses these difficulties by providing service provisioning, unlimited storage capacity, remote resource management, service discovery, data processing, and visualization. Cloud-based middleware primarily seeks to offer interoperability, scalability, and flexibility. There are monolithic architectures, SOA, and microservices architecture. A monolith middleware is still prevalent, possibly due to the ease of building the first version. However, for managing thousands of devices or for evolving forms of communication, this approach is not the best way.

6.2.2 Edge-Based

Edge-based middleware is strictly interested in interoperability between devices on the IoT. Therefore, they seek to offer management and communication solutions between devices as the minimum of the cloud. Thus, some of them seek only to carry out a translation between communication protocols to achieve an interaction between the devices. This approach facilitates the interoperability of devices. In addition, reliability is enhanced by the fact that it is in local communication, thereby avoiding problems related to data traffic with the cloud. Surely, some middleware are built for specific areas, such as industry or smart homes, providing user interfaces and environment-specific functionality. However, an architecture based on microservices could be a smart adoption for general purpose middleware. Thus, a hybrid approach could be facilitated with this type of architecture. Therefore, splitting the components between the edge and the cloud is a promising way and research gap.

6.3 SQ2 - which requirements were met by these middleware?

Seven main requirements addressed in middleware were cited. All the requirements and what the middleware are addressing for those requirements are displayed in Table 8. In addition, each requirement is presented and discussed below.

6.3.1 Interoperability

This requirement is a characteristic that provides the connection, services, and an exchange of information with different devices on the Internet of Things using and making available communication with various communication protocols. It should be noted

Requirement	**Middleware**
Interoperability	In.iot, Xie *et al.*, SEnviro Connect, eVATAR+, Thinger.io, MSOAH-IoT, CoIoT, MiddleBrigde, DeXMS, M-Hub/CDDL, InterSCity, MinT, MSM, SWaMM, Georgi *et al.*, RDS
Flexibility	CUPUS, InterSCity, Merlino *et al.*, Cilia, SmartCityAware
Scalability	In.iot, SEnviro Connect, InterSCity, RDS
Reliability	In.iot, SEnviro Connect, MinT, InterSCity, Georgi *et al.*, Zhao *et al.*
Connectivity	Thinger.io
Context Management	CoIoT, M-Hub/CDDL
Security and Privacy	MSM

Table 8: Requirements addressed by middleware for IoT

that almost all works sought to provide mechanisms to meet this requirement. Some middleware used more than one communication protocol on the application layer and also worked in different types on the network layer. However, it was not possible to find, on the selected primary studies, a multiprotocol middleware in which the authors describe how different devices using incompatible protocols can send, receive data, and control actions between them.

6.3.2 Flexibility

This requirement is inserted as a way of creating mechanisms that make middleware adaptable and extensible. It was introduced to address a problem related to the reuse of middleware in different environments. Thus, flexibility was used in defining the architecture to be used by these middleware in order to make the architectures adaptable. It may be noted that middleware has an architecture that works in the cloud and provides communication mechanisms with the infrastructure. Half of the middleware that mentions flexibility were applied to problems related to smart cities.

6.3.3 Scalability

In the definition proposed by Abbott and Fisher, 2015 there were three types of scalability as a scale cube: i) the scale of the X-axis that scales horizontally by distributing the total load over a certain number of nodes; ii) the scale of the Y-axis refers to the breakdown and distribution in services known as functional decomposition which is reflected in service-oriented architectures and microservices; and iii) the Z-axis scale refers to data partitioning, which is a way of distributing data amongst many nodes or blocks of data to improve the performance. Thus, the middleware that uses this requirement is directly related to the Y-axis. They seek to share their functionality, even when applying middleware in different environments.

6.3.4 Reliability

The reliability of a system is related to the average time between failures. In addition, this requirement also addresses the system's ability to recover after a failure and the time it takes to perform that recovery [Bajpai and Gorthi, 2012]. Middleware that uses this requirement seeks to justify using known, tested tools, libraries, and frameworks to avoid failures. Georgi et al., 2018 reinforced the need for reliable data generated by sensors in e-health environments. This concern is due to the risk of generating inaccurate data thereby causing wrong decision making.

6.3.5 Connectivity

In IoT different types are connected by different physical networks. Connectivity is related to this fact. Middleware must provide communication interfaces, together with interoperability to be able to manage group devices in the ecosystem across different networks (WiFi, Bluetooth, RFID, LoRa, ZigBee). Thus, some middleware cited and employ other forms of connection with the middleware, in addition to WiFi.

6.3.6 Context Management

Context management is managing the set of information and inferring the current state of an entity so as to make services available to users. Systems that use this information are considered context-sensitive [Abowd et al., 1999]. Thus, context recognition enables the storage of context information linked to the device data, allowing an easy, meaningful interpretation [Perera et al., 2014].

6.3.7 Security and Privacy

Security is an important requirement in any system. This brings great responsibility to an IoT middleware, in which it is in constant communication between devices and applications, especially when the system uses sensitive data issued by military, health or industrial domains. Thus, middleware needs to provide security mechanisms to prevent possible attacks on devices and data. The MSM middleware [Benayache et al., 2019] includes a security layer that monitors all other layers of the architecture. It was not possible to identify security mechanisms in the others. Thus, we realize that there is still a need for further work and a possibility of evolving so as to protect IoT middleware, devices, and private information.

6.4 SQ3 - which communication protocols do these middleware support?

Six main communication protocols were used. All communication protocols and the middleware that addresses these protocols are displayed in Table 9. The HTTP protocol was used by almost all middleware with different abstractions: REST or Google Cloud Messaging (GCM). In addition, the other communication protocol is presented and discussed below.

- CoAP: is an application layer protocol made by IETF [Shelby et al., 2014] for use with constrained nodes and constrained (low-power, lossy) networks. The protocol is designed for machine-to-machine (M2M) applications such as smart energy and building automation and UDP uses for transport. CoAP provides a request/response interaction model between application endpoints, supports built-in discovery of services and resources, and includes key concepts of the Web such as URIs and Internet media types. CoAP is designed to interface easily with HTTP for integration with the Web.

Protocol	**Middleware**
HTTP	In.iot, Xie *et al.*, SEnviro Connect, eVATAR+, CUPUS, Thinger.io, MSOAH-IoT, CoIoT, MiddleBrigde, Lilis *et al.*, DeXMS, InterSCity, MinT, MSM, SWaMM, Merlino *et al.*, Zhao *et al.*, Cilia, SmartCityAware
CoAP	In.iot, MiddleBrigde, DeXMS, MinT, MSM, Zhao *et al.*
MQTT	In.iot, Xie *et al.*, SEnviro Connect, Thinger.io, MiddleBrigde, DeXMS, M-Hub/CDDL, MinT, MSM, SWaMM, Zhao *et al.*
AMQP	Merlino *et al.*
DDS	MiddleBrigde, MSM, RDS
WebSocket	MiddleBrigde, DeXMS, MSM

Table 9: Communication protocols addressed by middleware for IoT

– MQTT: was created by Andy Stanford Clark of IBM and Arlen Nipper of Eurotech in 1999. The MQTT is a publish/subscribe model to provide improved communications with flexibility. Thus, the publishing devices generate the data and send that data to the broker. The broker forwards this data to interested entities. Subscribers may access this data through the broker. In addition, MQTT is a lightweight messaging protocol, designed for constrained devices and low-bandwidth, high-latency, or unreliable networks. Thus, the protocol is suitable for machine-to-machine communication. MQTT is based on TCP/IP protocol.

– AMQP: was created by John O'Hara at JPMorgan Chase in 2003. AMQP is an application layer protocol with a peer-to-peer connection but widely used asynchronously with a focus on exchanging messages between publisher and subscriber through a broker [OASIS, 2021]. The publishers send the messages to the broker and the broker organizes them in queues for distribution to their subscribers. It provides three types of message delivery guarantees: delivery at most once, at least once, and exactly once. For this, the protocol uses TCP to reliably exchange messages. In addition, it uses authentication and/or encryption based on SASL and/or TLS. Therefore, any tool that can create and interpret messages in accordance with the AMQP data format can interoperate with any other compatible tool, regardless of the implementation language.

– DDS: is a data-centric protocol built by the Object Management Group (OMG). It integrates the components of the ecosystem. In addition, messages produced by these entities are sent via topics to a global storage location managed by DDS. Thus, it follows the published, subscribed pattern. Entities interested in the data may subscribe to the topics to receive the data. DDS works with UDP and TCP at the transport layer [OMG, 2021].

7 Conclusion

This work presents the state of the art of middleware for IoT, detailing the solutions published between the years 2016 to April/2021 in the main scientific dissemination vehicles. Additionally, we present some identified research gaps. These data were collected by performing a Systematic Literature Review whose research protocol was also detailed in this work. In relation to the identified IoT middleware, this work details their application layer protocols, architectures, and covered requirements. We note that other details related to IoT middleware such as applications or challenges were briefly commented on in the related works section where we presented surveys previously carried out, but which analyzed other aspects of IoT middleware. It is important to notice that our work is the only SLR about middlewares for IoT which included and analyzed previous surveys. By including previous related works, it becomes easier to visualize the evolution of the area and what research gaps have been addressed over the years.

Given the above, we analyzed twenty-two articles on IoT middleware. Some middleware works in the cloud and some in the edge. The requirements for each middleware were identified: interoperability, scalability, flexibility, context management, reliability, security and privacy. Most middleware uses a monolith or service-oriented architecture using HTTP as the communication protocol. A promising way is the use of a hybrid microservices approach with lighter protocols such as CoAP or MQTT. Furthermore, the method of validating some of the proposals was simulation. In some studies, however, it was not possible to identify how these simulations were conducted.

In addition to these results, we identified some research gaps related to the main requirements identified for IoT middleware as described below.

We observe that most works proposed to add interoperability to their solutions. However, there was a reduced capacity for communication between devices. These works have not added several application protocols for communication between middleware, devices, and applications. Thus, the ability to interoperate is compromised. Another point lacking in these proposals is in the physical layer since the works do not bring a range of options for communication between different networks. Some works only mention that they are using such networks, but it was not possible to identify any type of validation using the technology in a real application.

We also noted a lack of security and privacy requirements in the analyzed IoT middlewares. Only one paper addressed any concerns in this regard. Security and privacy requirements were one of the most neglected elements. Existing approaches tend to adapt security and privacy late. The question remains open with regard to the insertion of security and privacy in all stages of the ecosystem during the development of IoT middleware. Every approach presented strengths and weaknesses. No single approach will address all IoT problems, thus, there is a need for a hybrid approach. The middleware developed over the last five years has adopted the cloud as an integral part. Recent trends are changing to use fog and edge computing for small processing and to respond to latency-sensitive requests in the IoT.

Another requirement not addressed by any of the analyzed middleware is usability. Thus, mechanisms that facilitate the use of an IoT middleware by people with little knowledge about programming could be a good research topic or a possible requirement to be addressed for the evolution of IoT middleware.

Finally, we emphasize that future works can use the protocol of this systematic literature review in order to update the results and identify the evolution of the area over time. We also consider it important that other researchers be inspired by this review, in order to adapt their protocol to deepen the gaps observed in this work.

Acknowledgements

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001

References

[Abbott and Fisher, 2015] Abbott, M. & Fisher, M. (2015). *The Art of Scalability: Scalable Web Architecture, Processes, and Organizations for the Modern Enterprise*. Addison-Wesley Professional, 2nd edition.

[Abowd et al., 1999] Abowd, G. D., Dey, A. K., Brown, P. J., Davies, N., Smith, M., & Steggles, P. (1999). Towards a better understanding of context and context-awareness. In H.-W. Gellersen (Ed.), *Handheld and Ubiquitous Computing* (pp. 304–307). Berlin, Heidelberg: Springer Berlin Heidelberg https://doi.org/10.1007/3-540-48157-5_29.

[Alvaro et al., 2019] Alvaro, L., Miguel, P., & Molina, J. (2019). Thinger.io: An Open Source Platform for Deploying Data Fusion Applications in IoT Environments. *Sensors*, *19*(5), 1044, https://doi.org/10.3390/s19051044.

[Alvisi et al., 2019] Alvisi, S., et al. (2019). Wireless Middleware Solutions for Smart Water Metering. *Sensors*, *19*(8), 1853, https://doi.org/10.3390/s19081853.

[Antonić et al., 2016] Antonić, A., Marjanović, M., Pripužić, K., & Podnar Žarko, I. (2016). A mobile crowd sensing ecosystem enabled by cupus: Cloud-based publish/subscribe middleware for the internet of things. *Future Generation Computer Systems*, *56*, 607–622, https://doi.org/10.1016/j.future.2015.08.005.

[Bajpai and Gorthi, 2012] Bajpai, V. & Gorthi, R. P. (2012). On non-functional requirements: A survey. In *2012 IEEE Students' Conference on Electrical, Electronics and Computer Science* (pp. 1–4). https://doi.org/10.1109/SCEECS.2012.6184810.

[Bandyopadhyay et al., 2011] Bandyopadhyay, S., Sengupta, M., Maiti, S., & Dutta, S. (2011). Role of Middleware for Internet Of Things. *International Journal of Computer Science Engineering Survey (IJCSES)*, *2*(3), 94–105, https://doi.org/10.5121/ijcses.2011.2307.

[Benayache et al., 2019] Benayache, A., Bilami, A., Barkat, S., Lorenz, P., & Taleb, H. (2019). MsM: A microservice middleware for smart WSN-based IoT application. *Journal of Network and Computer Applications*, *144*(October 2018), 138–154, https://doi.org/10.1016/j.jnca.2019.06.015.

[Biolchini et al., 2005] Biolchini, J., Mian, P. G., Natali, A. C. C., & Travassos, G. H. (2005). *Systematic review in software engineering*. Technical Report RT-ES 679/05, Systems Engineering and Computer Science Department, UFRJ, RJ.

[Biolchini et al., 2007] Biolchini, J. C. d. A., Mian, P. G., Natali, A. C. C., Conte, T. U., & Travassos, G. H. (2007). Scientific Research Ontology to Support Systematic Review in Software Engineering. *Adv. Eng. Inform.*, *21*(2), 133–151, https://doi.org/10.1016/j.aei.2006.11.006.

[Bouloukakis et al., 2019] Bouloukakis, G., Georgantas, N., Ntumba, P., & Issarny, V. (2019). Automated synthesis of mediators for middleware-layer protocol interoperability in the IoT. *Future Generation Computer Systems*, *101*, 1271–1294, https://doi.org/10.1016/j.future.2019.05.064.

[Bunz and Meikle, 2017] Bunz, M. & Meikle, G. (2017). *The Internet of Things*. Medford, MA, USA: Polity Press.

[Chaqfeh and Mohamed, 2012] Chaqfeh, M. A. & Mohamed, N. (2012). Challenges in middleware solutions for the internet of things. *International Conference on Collaboration Technologies and Systems (CTS)*, https://doi.org/10.1109/CTS.2012.6261022.

[da Cruz et al., 2019] da Cruz, M. A., Rodrigues, J. J., Lorenz, P., Solic, P., Al-Muhtadi, J., & Albuquerque, V. H. C. (2019). A proposal for bridging application layer protocols to HTTP on IoT solutions. *Future Generation Computer Systems*, *97*(2019), 145–152, https://doi.org/10.1016/j.future.2019.02.009.

[da Cruz et al., 2020] da Cruz, M. A. A., Rodrigues, J. J. P. C., Lorenz, P., Korotaev, V., & de Albuquerque, V. H. C. (2020). In.IoT—A new Middleware for Internet of Things. *IEEE Internet of Things Journal*, https://doi.org/10.1109/JIOT.2020.3041699.

[Dipsis and Stathis, 2020] Dipsis, N. & Stathis, K. (2020). A RESTful middleware for AI controlled sensors, actuators and smart devices. *Journal of Ambient Intelligence and Humanized Computing*, *11*(7), 2963–2986, https://doi.org/10.1007/s12652-019-01439-3.

[Eck and Waltman, 2010] Eck, N. J. v. & Waltman, L. (2010). Software survey: VOSviewer, a computer program for bibliometric mapping. *84*(2), 523–538, https://doi.org/10.1007/s11192-009-0146-3.

[Esposte et al., 2019] Esposte, A. d. M. D., Santana, E. F., Kanashiro, L., Costa, F. M., Braghetto, K. R., Lago, N., & Kon, F. (2019). Design and evaluation of a scalable smart city software platform with large-scale simulations. *Future Generation Computer Systems*, *93*, 427–441, https://doi.org/10.1016/j.future.2018.10.026.

[Farahzadi et al., 2018] Farahzadi, A., Shams, P., Rezazadeh, J., & Farahbakhsh, R. (2018). Middleware technologies for cloud of things: a survey. *Digital Communications and Networks*, *4*(3), 176 – 188, https://doi.org/10.1016/j.dcan.2017.04.005.

[Georgi et al., 2018] Georgi, N., Corvol, A., & Jeannes, R. L. B. (2018). Middleware Architecture for Health Sensors Interoperability. *IEEE Access*, *6*, 26283–26291, https://doi.org/10.1109/ACCESS.2018.2835644.

[Gomes et al., 2017] Gomes, B., et al. (2017). A Middleware with Comprehensive Quality of Context Support for the Internet of Things Applications. *Sensors*, *17*(12), 2853, https://doi.org/10.3390/s17122853.

[Gough, 2017] Gough, D. (2017). *An introduction to systematic reviews*. Los Angeles: SAGE.

[Hassan, 2018] Hassan, Q. F. (2018). *Internet of Things A to Z: Technologies and Applications*. John Wiley Sons Ltd.

[Hassan et al., 2018] Hassan, Q. F., Khan, A. R., & Madani, S. A. (2018). *Internet of Things: Challenges, Advances, and Applications*. Boca Raton, FL, USA: Chapman Hall/CRC.

[Jeon and Jung, 2017] Jeon, S. & Jung, I. (2017). MinT: Middleware for Cooperative Interaction of Things. *Sensors*, *17*(6), 1452, https://doi.org/10.3390/s17061452.

[Kitchenham, 2004] Kitchenham, B. (2004). *Procedures for Performing Systematic Reviews*. Keele University. Technical Report TR/SE-0401, Department of Computer Science, Keele University, UK.

[Kramp et al., 2013] Kramp, T., van Kranenburg, R., & Lange, S. (2013). *Introduction to the Internet of Things*, (pp. 1–10). Springer Berlin Heidelberg: Berlin, Heidelberg https://doi.org/10.1007/978-3-642-40403-0_1.

[Lalanda et al., 2017] Lalanda, P., Morand, D., & Chollet, S. (2017). Autonomic Mediation Middleware for Smart Manufacturing. *IEEE Internet Computing*, *21*(1), 32–39, https://doi.org/10.1109/MIC.2017.18.

[Lilis and Kayal, 2018] Lilis, G. & Kayal, M. (2018). A secure and distributed message oriented middleware for smart building applications. *Automation in Construction*, *86*(November 2017), 163–175, https://doi.org/10.1016/j.autcon.2017.10.030.

[Merlino et al., 2019] Merlino, G., Dautov, R., Distefano, S., & Bruneo, D. (2019). Enabling Workload Engineering in Edge, Fog, and Cloud Computing through OpenStack-based Middleware. *ACM Transactions on Internet Technology*, *19*(2), https://doi.org/10.1145/3309705.

[Mesmoudi et al., 2018] Mesmoudi, Y., Lamnaour, M., El Khamlichi, Y., Tahiri, A., Touhafi, A., & Braeken, A. (2018). A middleware based on service oriented architecture for heterogeneity issues within the internet of things (msoah-iot). *Journal of King Saud University - Computer and Information Sciences*, https://doi.org/10.1016/j.jksuci.2018.11.011.

[Mohamed et al., 2017] Mohamed, N., Al-Jaroodi, J., Jawhar, I., Lazarova-Molnar, S., & Mahmoud, S. (2017). SmartCityWare: A Service-Oriented Middleware for Cloud and Fog Enabled Smart City Services. *IEEE Access*, *5*, 17576–17588, https://doi.org/10.1109/ACCESS.2017.2731382.

[Ngu et al., 2017] Ngu, A. H., Gutierrez, M., Metsis, V., Nepal, S., & Sheng, Q. Z. (2017). IoT Middleware: A Survey on Issues and Enabling Technologies. *IEEE Internet of Things Journal*, *4*(1), 1–20, https://doi.org/10.1109/JIOT.2016.2615180.

[OASIS, 2021] OASIS (2021). Amqp is an open internet (or “wire”) protocol standard for message-queuing communications. Acessed: 20/05/2021 https://www.amqp.org/product/overview.

[OMG, 2021] OMG (2021). What is DDS? Acessed: 20/05/2021 https://www.dds-foundation.org/what-is-dds-3/.

[Perera et al., 2014] Perera, C., Zaslavsky, A., Christen, P., & Georgakopoulos, D. (2014). Context Aware Computing for The Internet of Things: A Survey. *IEEE Communications Surveys & Tutorials*, *16*(1), 414–454, https://doi.org/10.1109/surv.2013.042313.00197.

[Rathod and Chowdhary, 2018] Rathod, D. & Chowdhary, G. (2018). Survey of Middleware for Internet of Things. In *2018 International Conference on Recent Trends in Advance Computing (ICRTAC)* (pp. 129–135). https://doi.org/10.1109/ICRTAC.2018.8679249.

[Raza et al., 2021] Raza, H. W., Kamal, M. A., Alam, M., & Su’ud, M. M. (2021). A Review Of Middleware Platforms In Internet Of Things: A Non – Functional Requirements Approach. https://doi.org/10.31645/18.

[Razzaque et al., 2016] Razzaque, M. A., Milojevic-Jevric, M., Palade, A., & Clarke, S. (2016). Middleware for Internet of Things: A Survey. *IEEE Internet of Things Journal*, *3*(1), 70–95, https://doi.org/10.1109/JIOT.2015.2498900.

[Santos de Souza et al., 2019] Santos de Souza, R., et al. (2019). Continuous monitoring seed testing equipaments using internet of things. *Computers and Electronics in Agriculture*, *158*, 122–132, https://doi.org/10.1016/j.compag.2019.01.024.

[Shelby et al., 2014] Shelby, Z., Hartke, K., & Bormann, C. (2014). The Constrained Application Protocol (CoAP). (7252), https://doi.org/10.17487/RFC7252.

[Shokrollahi and Shams, 2017] Shokrollahi, S. & Shams, F. (2017). Rich device-services (RDS): A service-oriented approach to the internet of things (IoT). *97*(2), 3183–3201, https://doi.org/10.1007/s11277-017-4669-2.

[Snyder, 2019] Snyder, H. (2019). Literature review as a research methodology: An overview and guidelines. *Journal of Business Research*, *104*, 333–339, https://doi.org/10.1016/j.jbusres.2019.07.039.

[Sztajnberg et al., 2018] Sztajnberg, A., Stutzel, M., & Macedo, R. (2018). Application protocols for the internet of things: concepts and practical aspects. In *Jornadas de Atualização em Informática*, volume 37 (pp. 99–148). Natal, RN: CSBC SBC.

[Trilles et al., 2020] Trilles, S., González-Pérez, A., & Huerta, J. (2020). An IoT Platform Based on Microservices and Serverless Paradigms for Smart Farming Purposes. *Sensors*, *20*(8), https://doi.org/10.3390/s20082418.

[Vikash et al., 2020] Vikash, Mishra, L., & Varma, S. (2020). Middleware Technologies for Smart Wireless Sensor Networks towards Internet of Things: A Comparative Review. *Wireless Personal Communications*, *116*(3), 1539–1574, https://doi.org/10.1007/s11277-020-07748-7.

[Weiser, 1991] Weiser, M. (1991). The computer for the 21st century. *Scientific American*, (pp. 94–104).

[Xie et al., 2021] Xie, C., Yu, B., Zeng, Z., Yang, Y., & Liu, Q. (2021). Multilayer Internet-of-Things Middleware Based on Knowledge Graph. *IEEE Internet of Things Journal*, *8*(4), 2635–2648, https://doi.org/10.1109/JIOT.2020.3019707.

[Zhao et al., 2018] Zhao, R., Wang, L., Zhang, X., Zhang, Y., Wang, L., & Peng, H. (2018). A OneM2M-Compliant Stacked Middleware Promoting IoT Research and Development. *IEEE Access*, *6*, 63546–63559, https://doi.org/10.1109/ACCESS.2018.2876197.