Future Generation Computer Systems 97 (2019) 145–152

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

A proposal for bridging application layer protocols to HTTP on IoT solutions

Mauro A.A. da Cruz [a,b], Joel J.P.C. Rodrigues [a,c,d,e,*], Pascal Lorenz [b], Petar Solic [f], Jalal Al-Muhtadi [d], Victor Hugo C. Albuquerque [g]

[a] National Institute of Telecommunications (Inatel), Santa Rita do Sapucaí-MG, Brazil
[b] University of Haute Alsace, Colmar, France
[c] Instituto de Telecomunicações, Portugal
[d] College of Computer and Information Sciences (CCIS), King Saud University (KSU), Riyadh 12372, Saudi Arabia
[e] Federal University of Piauí (UFPI), Teresina-PI, Brazil
[f] University of Split, Split, Croatia
[g] University of Fortaleza (UNIFOR), Fortaleza-CE, Brazil

HIGHLIGHTS

- MiddleBridge converts MQTT, CoAP, Websockets, and DDS messages into HTTP.
- The solution reduces the amount of time that a device spends transmitting.
- MiddleBridge receives a smaller message, restructures it and forwards to a middleware.
- Packet size produced by an IoT device was 17 times smaller than a direct HTTP request.

ARTICLE INFO

Article history:
Received 23 September 2018
Received in revised form 20 December 2018
Accepted 5 February 2019
Available online 22 February 2019

Keywords:
Application layer
Bridge
Gateway
Internet of Things
MQTT
CoAP
DDS
WebSockets

ABSTRACT

In the Internet of Things (IoT), data is handled and stored by software known as middleware (located on a server). IoT devices send such data through an application layer protocol that may be different from those supported by the middleware. This paper proposes an application layer gateway, called MiddleBridge, that translates Constrained Application Layer Protocol (CoAP), Message Queuing, Queuing Telemetry Transport Protocol (MQTT), Data Distribution Service (DDS), and Websockets messages into HTTP. MiddleBridge can be deployed on any computer with Java virtual machine because all servers are embedded in its code, enabling IoT gadgets to transmit data to any REST endpoint seamlessly. With the proposed approach, devices can send a smaller message to an intermediary (MiddleBridge), which restructures it and forwards to a middleware, reducing the time that a device spends transmitting. The created graphical user interface allows users to configure messages conversion and forwarding in runtime. The efficiency of such approach is evaluated through the packet size and response times considering the data sent to Orion context broker (a Fiware project). Results show that packet size that is sent by an IoT device through MiddleBridge is 17 times smaller than sending a straight HTTP request to the server and significantly reduces the transmission time.

1. Introduction

The concept of connecting any object to the Internet is as a new paradigm known as the Internet of Things (IoT) [1]. These connected objects, called "things", connect to the Internet to enhance their functionalities, turning regular objects into smart objects. Its economic impact could range from 2.7 to 6.2 trillion USD (United States Dollars) by 2025 [2], and by 2020 there will be approximately 30 billion Internet-connected objects [3]. The IoT can be used to improve every life aspect, bringing into existence many scenarios, such as smart homes, self-driving cars, smart cities, and smart agriculture [4]. IoT can be applied in all the areas because any object regardless of type or purpose can be connected, which includes virtual objects, inanimate objects, and even living organisms [5]. IoT devices introduce various challenges,

* Correspondence to: National Institute of Telecommunications (Inatel), Av. João de Camargo, 510 - Centro, Santa Rita do Sapucaí-MG, 37540-000, Brazil.
E-mail addresses: maurocruzter@gmail.com (M.A.A. da Cruz), joeljr@ieee.org (J.J.P.C. Rodrigues), lorenz@ieee.org (P. Lorenz), psolic@fesb.hr (P. Solic), jalal@ccis.edu.sa (J. Al-Muhtadi), victor.albuquerque@unifor.br (V.H.C. Albuquerque).

https://doi.org/10.1016/j.future.2019.02.009
0167-739X/

mainly because most present small size and resources-constrains (memory, battery, processing, disk space) [6]. These challenges include security, scalability, standardization, connectivity, user trust, communication, and much more [5].

IoT environments are composed of countless objects, produced by a variety of brands. In the current status of IoT, most IoT brands are incompatible among them [7,8]. This restriction is inconvenient for the users because it can determine which brands will be acquired by users without losing functionalities [8]. When purchasing home appliances, such as a television or a microwave, users should know that compatibility will not be an issue. Furthermore, it is unlikely that one brand will produce every type of IoT devices, and users will be limited to sets of brands or will purchase incompatible equipment without using all their functionalities if this issue is not addressed.

The heterogeneity and compatibility among IoT devices may be easily solved through a worldwide standard [9,10]. However, standards are difficult to enforce [9] taking cellphone chargers, as an example. In 2009, the European Union announced an agreement with ten (10) of the top cell phone manufacturers to adopt micro-USB as the connector to charge smartphones and tablets [11]. Such a standard would ensure fewer cables in landfills and was expected to be enforced by 2011. Unfortunately, some of the companies that agreed to such a treaty are yet to adhere to it. In IoT, an alternative for providing interoperability support may be performing communication among nodes through an IoT middleware. An IoT middleware is a software that not only allows incompatible devices and applications to communicate but also stores the collected data for posterior treatment and analysis [10,12]. A middleware is deployed on a Web server, and data is transmitted from the device to the server through an application layer protocol. The most popular Internet application layer protocol is the HTTP (Hypertext Transfer Protocol). However, it consumes too many resources [13], and alternative application layer protocols are being used for IoT communications. Despite not being the "best" IoT protocol, it is used in many IoT scenarios.

Unfortunately, most IoT middleware only supports two application layer protocols (HTTP and generally MQTT). If one of the gadgets purchased by the user is not compatible with application layer protocols supported by the middleware, such gadget will not achieve its potential regarding functionalities. In such cases, instead of sending a message straight to the middleware, there could be an intermediary device that translates the data into an application protocol supported by the middleware, and the intermediate process would be seamless to senders and receivers. The paper identifies the practical impact of such solution and proposes MiddleBridge, an application layer gateway for IoT protocols that "translates" messages sent by Message Queuing Telemetry Transport Protocol (MQTT), Constrained Application Protocol (CoAP), Data Distribution Service (DDS), and WebSockets to HTTP. The other solutions available in the literature merely send a message with the same payload as an HTTP request, and the reduction in the packet size only occurs because a lighter protocol is being used. In other way, MiddleBridge also reduces the payload and it is easily configurable through a graphical user interface. The utility of such solution is demonstrated through a use case where data is transmitted to Orion context broker (from Fiware project). Then, the main contributions of this paper are the following:

- Review the related literature on IoT application layer protocols, as well as gateway solutions for such protocols;
- Proposal and creation of an IoT gateway for application layer protocols called MiddleBridge;
- Performance evaluation, demonstration, and validation of the proposed solution through a real deployment using Orion context broker (from the Fiware project).

This paper is a revised and extended version of the contribution presented in [14]. The previous study only supported MQTT to HTTP translation, and the MQTT broker (the server) must be separately installed. The current contribution supports more protocols (namely, CoAP, MQTT, Websockets, and DDS), and their corresponding servers are now embedded in MiddleBridge. As consequence, the performance evaluation considers all the new supported protocols.

The remainder of the paper is organized as follows. Section 2 provides valuable background on the topic, discussing application layer protocols for IoT, and the concept of an application layer gateway. Section 3 elaborates on the software functionalities, while Section 4 demonstrates the efficiency of the proposed solution by sending data to Orion context broker (which is part of the Fiware project). Section 5 concludes the paper, suggesting future challenges and open issues.

2. Background and related work

The Internet of Things transforms regular objects into smart gadgets, which improve the users' quality of life. It faces challenges that are mostly related to the objects few computational capacities. As an alternative for such challenges, several aspects of the current Internet have a lightweight alternative for each layer of the TCP/IP architecture [15]. At the application layer, alternative protocols to the HTTP protocol are the following: Message Queueing Transfer Protocol (MQTT), Constrained Application Protocol (CoAP), Data Distribution Service (DDS), Extensible Messaging and Presence Protocol (XMPP), and WebSockets [16]. Nevertheless, much more were proposed for IoT. This section addresses the most popular IoT application layer protocols and the most relevant application layer gateways that inspired the solution proposed in this study.

2.1. IoT application layer protocols

An application layer protocol determines how clients operate in different systems and exchange messages among them. Several application layer protocols were proposed for IoT, being the most notable MQTT and CoAP. The MQTT protocol was standardized in 2013 by the Organization for the Advancement of Structured Information Standards (OASIS). MQTT is based on a publish/subscribe model of communication and uses TCP (Transmission Control Protocol) at the transport layer [17]. CoAP is specified in RFC 7252 and is based on a REST architectural style, with a client–server architecture. It is generally described as a lightweight HTTP and uses UDP (User Datagram Protocol) at the transport layer [17], which means it is a stateless protocol. Other relevant solutions include DDS, XMPP, and Websockets. DDS was standardized by the object management group (OMG). It is based on a publish/subscribe model, follows a peer-to-peer architecture and is mostly used for reliable and efficient real-time communications [18].

In contrast to other publish/subscribe protocols, the DDS architecture is "brokerless". The downside of DDS comes from the fact that its specification does not determine which protocol should be used at the transport layer, stating that different implementations must provide "vendor-specific bridges" to interoperate [19]. XMPP was initially called Jabber, and it is specified at the RFCs 6120 and 6121. It uses a client/server architecture and was developed as an instant messaging protocol that allows users to chat on the Web in real time [20]. Most XMPP implementations are attached to a graphical chat client (useless for most IoT devices). Therefore, its usage in IoT is reduced. Furthermore, the protocol development seems stagnated, and Google ceased support for a standard. Websockets were standardized in 2011 by the IETF (Internet Engineering Task Force) through the RFC 6455 and uses TCP at the transport

layer. Websockets are useful for IoT because they provide full-duplex communication through a single TCP connection [21].

Regarding possible deployment of the above-described protocols, XMPP and DDS are lackluster. As previously mentioned, XMPP active development seems to be stagnated, and DDSs has few implementations. In contrast, MQTT popularity is increasing, mainly because the community is very active in online forums. Also, various deployments of the protocol are available for numerous platforms and the protocol is easy to use and understand. CoAP has many implementations, but the community is not so active as MQTT. It is important to note that despite its inefficiency in IoT scenarios, REST stills very popular, mainly because it is well documented and various examples are available online. Concerning efficiency, protocols that use UDP are more sensitive to packet loss, and TCP produces more overhead as a tradeoff for the increased reliability. In [22], the performance of MQTT, CoAP, and DDS (TCP version) was compared and revealed that DDS consumes at least two times more bandwidth than MQTT, followed by CoAP. Also, under degraded network conditions, MQTT experiences higher latency, followed by DDS and CoAP. In [23], the performance of MQTT, CoAP, and Websockets was compared under normal network conditions (no packet loss) revealing that MQTT produces more overhead, followed by Websockets and CoAP. Concerning latency, CoAP performed better, followed by Websockets and MQTT. Overall, the conclusions of both studies are consistent and allow this paper to infer that DDS produces larger packets, followed by MQTT, Websockets, and CoAP. Regarding latency, CoAP performs better, and MQTT presents the worst performance. There is not enough data to determine if WebSockets performs better than DDS regarding latency. Table 1 summarizes the comparison for the Websockets, DDS, CoAP, and MQTT protocols.

2.2. Application layer gateways

Connected objects do not add much value by themselves and remotely accessing objects functionalities does not entirely justify the expected trillion USD impact. However, data collected by such devices can reveal users behavior which, when analyzed, is extremely valuable because advertisers can target the groups that are more likely to purchase a service. Targeted advertisement represents a significant source of income for Facebook and Google [24], for example. The data collected in IoT environments is stored in an IoT middleware, which is a software that stores the collected data and also allows incompatible devices and applications to communicate [25,26]. An IoT middleware is generally deployed in a robust server, and it is also mentioned to as an IoT platform, IoT middleware platform, or simply middleware. These terms can be used interchangeably in the remainder of this paper and the related literature. Devices communicate with a middleware through an application layer protocol and objects that are not compatible with the protocols supported by the middleware cannot be used to its full potential. An easy solution to overlap this issue consists of sending the data to an intermediary entity that translates the message into a protocol that is supported by a middleware. The software solution responsible for such adaptation is called Application Layer Bridge (ALB), Application Layer Gateway (ALG), or simply Bridge; the three terms can be used interchangeably in the literature. ALBs are similar to real-life translators, in which devices request the ALB to transmit a message in the desired protocol. A bridge operation is illustrated in Fig. 1.

An application layer gateway is a software that intermediates the connection between an application server and the Internet. ALGs operation is straightforward since they receive a message through a protocol (protocol A) and forward the message in another protocol (protocol B). They are useful in IoT scenarios because most devices support a single application protocol (i.e., MQTT),

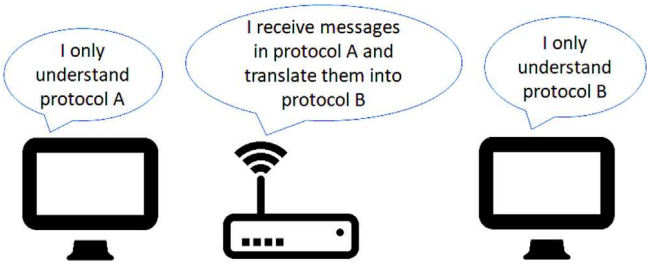

Fig. 1. Illustration of a bridge operation where a gateway receives messages through a protocol A and translates them to a protocol B.

and the server that stores the data might support a different protocol (i.e., HTTP). ALGs forward the received messages to the server, enabling a seamless communication for users (the device and the platform do not notice the intermediary). These software allow incompatible devices to communicate with the middleware without changing the core code neither in the middleware nor on the devices. In theory, ALGs can even be used to bypass restrictions imposed by a firewall or a network administrator where specific protocol traffic is blocked inside a network (i.e., BitTorrent).

All the ALGs follow a similar principle and users can opt for every device to have a dedicated bridge (direct bridge) or share a single bridge for every device (indirect bridge) [27]. Regarding performance, a direct bridge is advantageous because only one device makes use of the bridge and there is no concurrency for its resources. However, dedicating a gateway for every device is costly and increases the solution complexity. The concept of ALBs is not new in information and communication technologies as well as in IoT. Few ALGs have been proposed in the literature, such as Ponte [28] and Gothings [29], both supporting MQTT, CoAP, and HTTP. Ponte is an Eclipse project and an evolution of the QEST broker [30]. Furthermore, Gothings and Ponte do not provide any functionalities beyond the "message translation", merely sending a message with the same payload as an HTTP request in a different protocol. The proposed solution benefits from a smaller packet that is sent by IoT protocols and, unlike other bridges in the literature, it does not receive full JSON or XML messages from the devices. Instead, MiddleBridge accepts a packet containing a smaller payload, reconstructs it, and forwards them to an IoT platform, which results in less stress for constrained devices. Additionally, the proposed solution provides a graphical user interface (GUI) which facilitates its deployment in IoT solutions that are managed by the average user (other bridges are configured through the command line).

3. Bridging from IoT protocols to HTTP

MiddleBridge translates CoAP, Websockets, MQTT, and DDS requests into HTTP. It allows constrained devices to send less data when communicating with IoT middleware. The graphical user interface is simple and allows users to configure the conversion and forwarding of messages during runtime. MiddleBridge runs on any device with Java installed; this means that it can be deployed on a Raspberry Pi, personal computer, or even on the server that hosts the Middleware (in advanced cases where the user has access to the server). The solution includes embedded servers written in Java (for MQTT it is called a broker). The used MQTT broker is Moquette [31], RTI (Real-Time Innovations) connect DDS is used for DDS [32], the CoAP server is Californium [33], and the WebSockets deployment is TootallNate [6]. In all these protocols, the device sends data to MiddleBridge, which then forwards it to the Middleware chosen by the user (seamless to the sender and receiver). In a case of the MQTT broker, the proposed software subscribes to a topic specified by the user and then forwards the

Table 1
Summarized comparison among Websockets, DDS, CoAP, and MQTT protocols.

Comparison criteria	Comparison results (Best to worse)
Less bandwidth consumption [22]	CoAP; MQTT; DDS
Less latency under degraded network conditions [22]	CoAP; DDS; MQTT
Less overhead with no packet loss [23]	CoAP; Websockets; MQTT
Latency with no packet loss [23]	CoAP; Websockets; MQTT
Less Packet size (inferred from [22] and [23])	CoAP; Websockets; MQTT; DDS
Less latency (inferred from [22] and [23])	CoAP; MQTT; DDS /Websockets[a]

[a]There is not enough data to determine whether WebSockets performs better than DDS regarding latency.

received data to the HTTP server. Furthermore, MiddleBridge is licensed under GNU GPL (General Public License) v3, which allows any third party to modify and distribute versions of the solution if they are open-source. The project is available at [34], and contains documentation explaining its usage. Fig. 2 shows the MQTT GUI that was configured to send data to Orion context broker.

Most bridges follow a similar architecture because their operation method is simple and straightforward. MiddleBridge does not deviate from that basic architecture, and the only difference is the inclusion of a graphical user interface. Its architecture includes the following three elements: (*i*) message translator, (*ii*) protocol plugins, and (*iii*) graphical user interface. **Message translator** is the entity responsible for receiving, modifying, and forwarding messages to the desired protocol. **Protocol plugins** are the implementation of specific application protocols themselves while the communication among the different protocols must go through the message translator. The **graphical user interface** allows users to configure aspects related to the conversion and messages forwarding on runtime. On the device side, it must support at least one of the protocols supported by MiddleBridge. A minor modification is mandatory on the devices because they will send data to the proposed bridge instead of their default server. An intermediate understanding of the protocols that will be used for the requests is mandatory to apply the appropriate modifications to the IoT object. MiddleBridge documentation explains the usage of each protocol and it is available at [34]. If the gadget vendor does not provide means to make such modifications, the bridging operation is not possible.

When devices send MQTT, CoAP, DDS, and Websocket messages, they will produce a packet with less overhead than a regular HTTP message. This means that if a device sends a payload A that contains a JSON message (i.e., {“name”:“NIA”}), it will always produce a packet size of at least 14 Bytes (1 Byte per character and the brackets also count) plus overhead. MiddleBridge takes such principle further and, instead of expecting a full JSON or XML message from the device in the example {“name”:“NIA”}, it merely expects a message containing “NIA”, which means that the device sends a packet with 3 Bytes from the payload plus overhead. The proposed software receives the message, converts it to an HTTP request, and forwards it to an IoT platform. Devices save time when sending data to MiddleBridge because, at the same time, it is forwarding a message to the IoT platform and ends the connection with the IoT device. For the IoT platform, MiddleBridge is an IoT device and, for the IoT device, MiddleBridge is the IoT platform. This occurs because the process is seamless for the sender (device) and receiver (IoT platform). Fig. 3 presents a flowchart of the MiddleBridge algorithm.

To reduce the packet size that is sent by the IoT device, users specify the variables that will be sent (through the GUI). Therefore, instead of sending “Brightness:X”, the device sends “X”, a similar concept is applied to the headers, where the name and value of the header are specified. If the header value is specific to each sender, users can specify the header name and leave the value in blank. In the brightness example, the device sends 11 Bytes less on the payload. For each additional character on the payload, the message size is increased by 1 Byte (blank spaces also count [25]). When other bridges are used, the message contains the same payload as the HTTP request regardless of the used protocol. The reconstruction process is illustrated in Fig. 4 and it is seamless to both sender and receiver.

Building this type of solution can be difficult because each supported protocol demands its server. Therefore, similar solutions may be difficult to run on computers with less than 256 MB of RAM (Random Access Memory). Moreover, almost every server of the supported protocols must be modified to process the received messages and forward them to the platform. The only exception to this rule are publish/subscribe protocols such as MQTT, in which software can subscribe to every message the server receives and forward it. However, finding an open-source deployment of the protocol server in a desired programming language can be challenging.

4. Orion context broker use case

Orion context broker [35] is a middleware platform developed by Fiware and follows a publish/subscribe deployment of the NGSI-9 and NGSI-10 Open RESTful API specifications. The data can only be sent to Orion through REST (Representational State Transfer). Therefore, Orion is an excellent candidate to evaluate MiddleBridge, namely because devices that support other IoT protocols such as CoAP, Websockets or MQTT cannot send data directly to Orion. The performance assessment of the solution regarding response times and packet size will be evaluated through experiments performed in a real wired LAN, and MiddleBridge was deployed in a Raspberry Pi 3. The Orion Context Broker was deployed on a Dell Precision 5820. The packets were generated through Apache JMeter, which was deployed on another Dell Precision 5820 machine. Since Jmeter does not natively support MQTT, Websockets, or CoAP. Therefore, the following community plugins were used [36,37], and [38] for MQTT, Websockets, and CoAP, respectively. Although MiddleBridge supports the DDS protocol, it will not be evaluated in this paper because there are no JMeter plugins for the DDS protocol.

Jmeter was also used to determine the packet size as well as the response times. The headers and body sent in an HTTP request are shown in Fig. 2. Instead of sending a payload identical to it, the device sends a message through one of the supported protocols with the payload like “Luminaire,Luminaire1,99”. Then, the bridge application reconstructs the rest of the message before sending the HTTP request, eliminating blank spaces beforehand. Since the goal of MiddleBridge is to reduce the stress in an IoT device, only the message sent by the IoT device to MiddleBridge was accounted. When MiddleBridge receives the message, the previous communication with the device is closed. Therefore, the IoT device that sends the message is not aware of the remaining process (MiddleBridge reconstructing the message and forwarding to an IoT platform). If the entire process is considered, the total network consumption for the packet to be delivered to the IoT platform will always be higher than a direct HTTP request (the same principle applies to response time). The hardware used in the experiments are shown in Fig. 5.

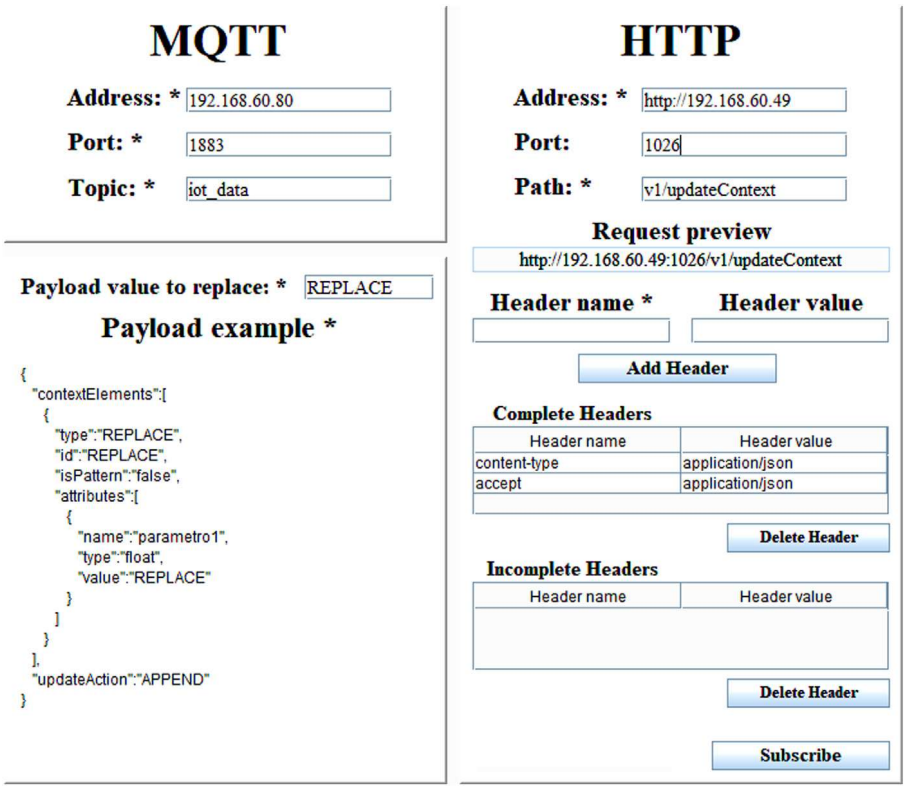

Fig. 2. MiddleBridge' graphical user interface configured to send data to the Orion context broker through the MQTT protocol.

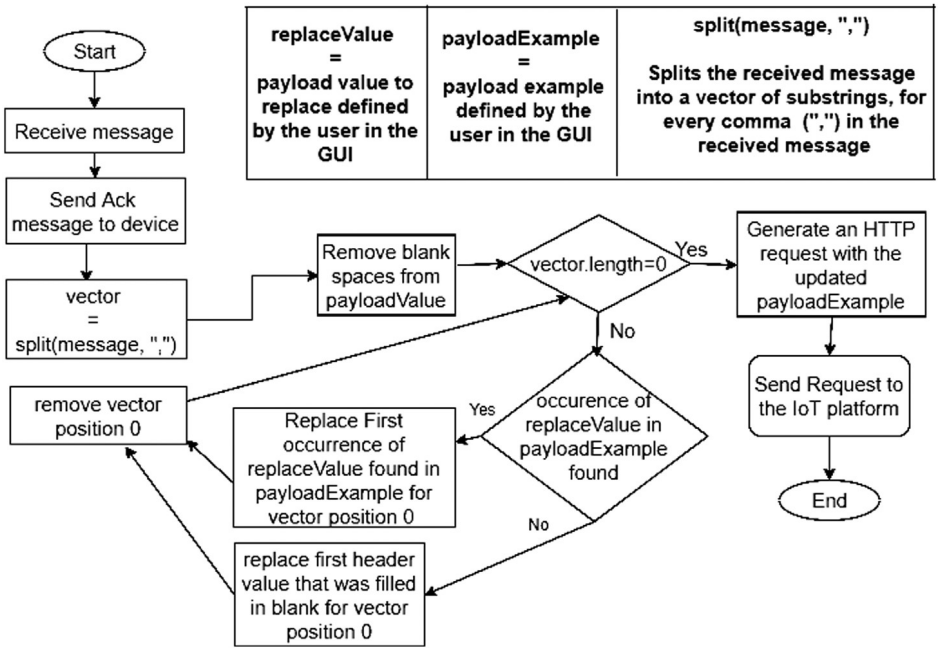

Fig. 3. Flowchart explaining the MiddleBridge algorithm.

4.1. Packet size

The packet sizes are important for IoT devices because, in theory, they imply shorter transmission times. When communicating over the Internet, the receiver confirms the reception through an ACK (acknowledgment) or NACK (negative acknowledgment). Therefore, the size of the response message will be included in the experiments. The packets were generated through Apache Jmeter. Fig. 6 presents the packet size analysis for a single request with MQTT, CoAP, WebSockets, and direct HTTP communication with the server.

It is observed that a direct HTTP request to Orion demands 401 sent Bytes and 595 received Bytes, and it is significantly less efficient than sending a request to MiddleBridge. Furthermore, there are no significant differences among the other three protocols. MQTT sends 23 and receives 20 Bytes, CoAP sends 23 and receive 8, while WebSockets send 29 and receive 2. Overall, using MiddleBridge, the sent packet size is 17 times smaller than a direct

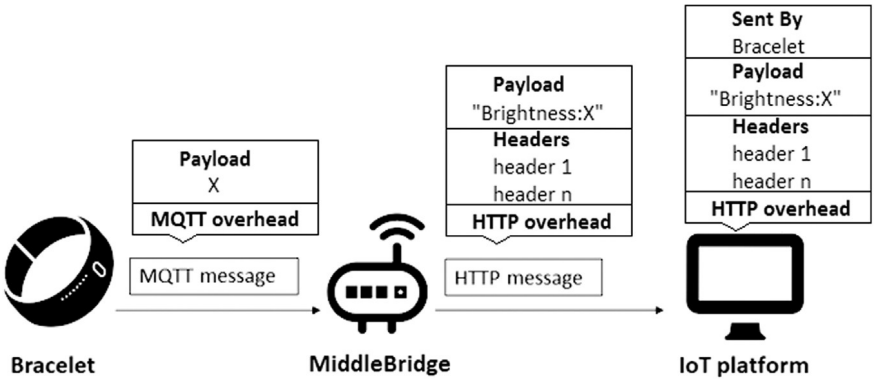

Fig. 4. Illustration of a bracelet (i.e., an object) sending a small message to MiddleBridge that is reconstructed and sent to the IoT platform.

Fig. 5. Photo of the hardware used in the experiments: (a) Dell Precision 5820 that hosts Orion Context Broker (b) Raspberry Pi 3 that hosts MiddleBridge; (c) Dell Precision 5820 that hosts Apache Jmeter, used to generate the CoAP, MQTT, Websockets, and direct HTTP requests.

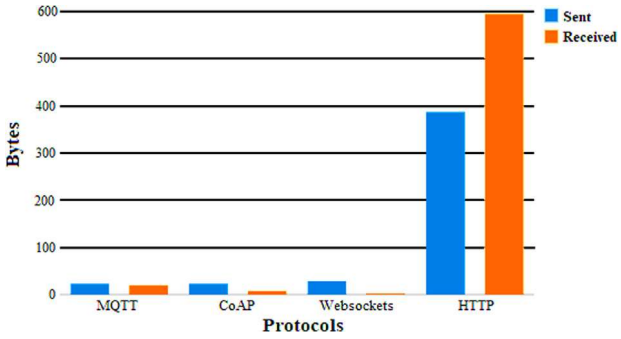

Fig. 6. Analysis of the packet size (in bytes) of a single request with MQTT, CoAP, Websockets, and a direct HTTP Request to Orion Context Broker.

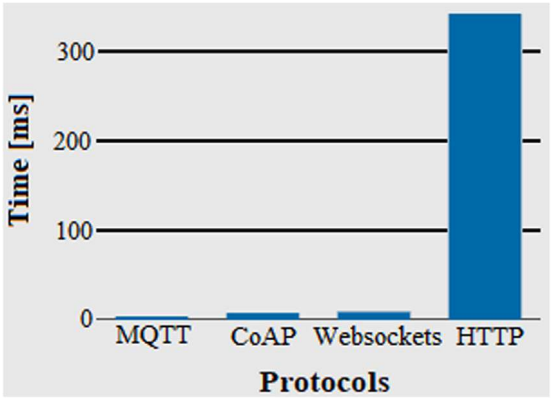

Fig. 7. Analysis of the average response time (in milliseconds) with MQTT, CoAP, and WebSockets, and direct HTTP communication with a server.

HTTP request and produces an ACK message 29, 74, and 297 times smaller with MQTT, CoAP, and Websockets, respectively.

4.2. Response times

Response times can be crucial in some IoT scenarios, such as healthcare, especially when the server is receiving several messages. When analyzing the response times, the round-trip time (RTT) to MiddleBridge for the different protocols will be evaluated. To verify the RTT, data were sent using a scenario with 100 concurrent users generated through Apache Jmeter. Fig. 7 presents the analysis of the average response time with MQTT, CoAP, WebSockets, and direct HTTP communication with a server. All the requests were successfully delivered.

It is observed that with 100 concurrent users, a direct HTTP request to Orion demands on average 343.265 ms and it is inefficient when compared to a request to MiddleBridge. Also, there are no significant differences among the other three protocols, with MQTT averaging 2.101 ms, CoAP 7.952 ms, and Websockets 9.081 ms. Overall, by sending their data to MiddleBridge, devices spend on average 163, 43, and 37 less time transmitting with MQTT, CoAP, and Websockets, respectively than a direct HTTP request. The Orion Context Broker was deployed on the local LAN and, in most IoT scenarios, the Middleware is located somewhere in the cloud, increasing latency of a direct HTTP request, further increasing the efficiency of MiddleBridge.

5. Conclusion and future work

The paper proposed MiddleBridge, an IoT application layer gateway that converts MQTT, CoAP, Websockets, and DDS messages into HTTP. MiddleBridge can be deployed on any computer that runs a Java virtual machine and allows IoT gadgets to communicate with a Middleware seamlessly. Unlike other application layer gateways, MiddleBridge can be configured through a graphical user interface and reduces the size of packets that are sent by an IoT device because a shorter message is sent to the bridge application that reconstructs it and forwards to the middleware. The proposed

solution is easy to deploy because all the servers of the supported protocols were embedded in the application. Moreover, the paper assessed and demonstrated the efficiency of such technique by evaluating the packet size and response times in a scenario where data is sent to Orion context broker (that only supports HTTP). The experiments were performed through Apache Jmeter and in the proposed scenario, packets that were transmitted to MiddleBridge were 17 times smaller than a direct HTTP request with any of the supported protocols. Also, with MiddleBridge, devices spent on average 163, 43, and 37 less time transmitting than with MQTT, CoAP, and Websockets, respectively. Furthermore, in most IoT scenarios, the IoT platform is located somewhere in the cloud, increasing latency, further improving the efficiency of MiddleBridge.

MiddleBridge is efficient because MQTT, CoAP, DDS, and Websocket messages produce less overhead than a regular HTTP message. Other bridges simply translate messages between the protocols and this is the only reduction in packet size performed by them. The proposed software takes advantage of such principle and, instead of expecting a full JSON or XML messages from the device (like other bridges available in the literature), it receives a message containing a smaller payload. Reconstructing the message and forwarding it to an IoT platform, which results in less stress for the constrained IoT device. For an IoT platform, MiddleBridge is seen like an IoT device and, for an IoT device, MiddleBridge is the IoT platform. If the entire process is considered the total network consumption for the packet to be delivered to an IoT platform will always be higher than a direct HTTP request. In this process, a device sends a message to MiddleBridge, which sends a response back to the device, then MiddleBridge processes and forwards the message to the IoT platform. The same principle is applied to response time.

Future work may include the evolution of the proposed solution to support bi-directional conversion among the multiple protocols as well as supporting other IoT protocols that might be proposed further, extending the flexibility of the solution. The translation latency for each protocol could also be analyzed as well as the impact the shorter transmissions and reduced packet size can represent for the power consumption in IoT devices. Also, a JMeter plugin that supports DDS should be developed to evaluate the efficiency and scalability of the DDS protocol in MiddleBridge. In the future, solutions like MiddleBridge can even optimize the packet size of requests made in the same protocol, especially in "heavy" protocols such as HTTP.

Acknowledgments

This work has been supported by FADCOM - Fundo de Apoio ao Desenvolvimento das Comunicações, presidential decree nº 264/10, November 26, 2010, Republic of Angola; National Funding from the FCT - Fundação para a Ciência e a Tecnologia through the UID/EEA/50008/2019 Project; by RNP, with resources from MCTIC, Grant No. 01250.075413/2018-04, under the Centro de Referência em Radiocomunicações - CRR project of the Instituto Nacional de Telecomunicações (Inatel), Brazil; by Finatel through the Inatel Smart Campus project; by Brazilian National Council for Research and Development (CNPq) via Grant No. 309335/2017-5; and by the International Scientific Partnership Program ISPP at King Saud University through ISPP #0129.

References

[1] M. Aazam, E.N. Huh, Fog computing and smart gateway based communication for cloud of things, in: 2014 International Conference on Future Internet of Things and Cloud, Barcelona, Spain, 2014, pp. 464–470.
[2] J. Manyika, M. Chui, J. Bughin, R. Dobbs, P. Bisson, Marrs, Disruptive Technologies: Advances that will Transform Life, Business, and the Global Economy, McKinsey Global Insitute, 2013, p. 163.
[3] D. Minoli, K. Sohraby, B. Occhiogrosso, IoT considerations, requirements, and architectures for smart buildings-energy optimization and next-generation building management systems, IEEE Internet Things J. 4 (1) (2017) 269–283.
[4] M. Alaa, A.A. Zaidan, B.B. Zaidan, M. Talal, M.L.M. Kiah, A review of smart home applications based on Internet of Things, J. Netw. Comput. Appl. 97 (1) (2017) 48–65.
[5] M. Nitti, V. Pilloni, G. Colistra, L. Atzori, The virtual object as a major element of the Internet of Things: A survey, IEEE Commun. Surv. Tutor. 18 (2) (2016) 1228–1240, 2nd Quart.
[6] TooTallNate/Java-WebSocket: A barebones WebSocket client and server implementation written in 100% Java. [Online]. Available: https://github.com/TooTallNate/Java-WebSocket. [Accessed: 20-Aug-2018].
[7] R. Nawaratne, D. Alahakoon, D. De Silva, P. Chhetri, N. Chilamkurti, Self-evolving intelligent algorithms for facilitating data interoperability in IoT environments, Future Gener. Comput. Syst. 86 (2018) 421–432.
[8] B. Farahani, F. Firouzi, V. Chang, M. Badaroglu, N. Constant, K. Mankodiya, Towards fog-driven IoT eHealth: Promises and challenges of IoT in medicine and healthcare, Future Gener. Comput. Syst. 78, part 2 (2018) 659–676.
[9] G. Aloi, et al., Enabling IoT interoperability through opportunistic smartphone-based mobile gateways, J. Netw. Comput. Appl. 81 (2017) 74–84.
[10] M.A.A. da Cruz, J.J.P.C. Rodrigues, J. Al-Muhtadi, V. Korotaev, V.H.C. Albuquerque, A reference model for internet of things middleware, IEEE Internet Things J. 81 (2) (2018) 871–883.
[11] T. Egyedi, S. Muto, Standards for ICT - A green strategy in a grey sector, in: 7th International Conference on Standardization and Innovation in Information Technology(SIIT 2011), Berlin, Germany, 2011, pp. 1–10.
[12] A. Farahzadi, P. Shams, J. Rezazadeh, R. Farahbakhsh, Middleware technologies for cloud of things-a survey, Digit. Commun. Netw. 4 (3) (2018) 176–188.
[13] A. Al-Fuqaha, A. Khreishah, M. Guizani, A. Rayes, M. Mohammadi, Toward better horizontal integration among IoT services, IEEE Commun. Mag. 53 (9) (2015) 72–79.
[14] M.A.A. Cruz, J.J.P.C. Rodrigues, E.S. Paradello, P. Lorenz, P. Solic, V.H.C. Albuquerque, A proposal for bridging the message queuing telemetry transport protocol to HTTP on IoT solutions, in: 2018 3rd International Conference on Smart and Sustainable Technologies (SpliTech), Split, Croatia, 2018, pp. 1–5.
[15] J. Zhou, Z. Cao, X. Dong, A.V. Vasilakos, Security and privacy for cloud-based IoT: Challenges, IEEE Commun. Mag. 55 (1) (2017) 26–33.
[16] Z. Meng, Z. Wu, C. Muvianto, J. Gray, A data-oriented M2M messaging mechanism for industrial IoT applications, IEEE Internet Things J. 4 (1) (2017) 236–246.
[17] C. Gomez, A. Arcia-Moret, J. Crowcroft, TCP in the internet of things: From ostracism to prominence, IEEE Internet Comput. 22 (1) (2018) 29–41.
[18] A. Hakiri, P. Berthou, A. Gokhale, S. Abdellatif, Publish/subscribe-enabled software defined networking for efficient and scalable IoT communications, IEEE Commun. Mag. 53 (9) (2015) 48–54.
[19] OMG, The Real-time Publish-Subscribe (RTPS) Wire Protocol DDS Interoperability Wire Protocol Specification v2.2, Sep. 2014.
[20] P. Anantharaman, M. Locasto, G.F. Ciocarlie, U. Lindqvist, Building hardened Internet-of-Things clients with language-theoretic security, in: 2017 IEEE Security and Privacy Workshops (SPW), San Jose, CA, USA, 2017, pp. 120–126.
[21] V. Pimentel, B.G. Nickerson, Communicating and displaying real-time data with websocket, IEEE Internet Comput. 16 (4) (2012) 45–53.
[22] Y. Chen, T. Kunz, Performance evaluation of IoT protocols under a constrained wireless access network, in: International Conference on Selected Topics in Mobile and Wireless Networking (MoWNeT 2016), Cairo, Egypt, 2016, pp. 1–7.
[23] S. Mijovic, E. Shehu, C. Buratti, Comparing application layer protocols for the Internet of Things via experimentation, in: IEEE 2nd International Forum on Research and Technologies for Society and Industry Leveraging a better tomorrow (RTSI 2016), Bologna, Italy, 2016, pp. 1–5.
[24] E. Ahmed, et al., The role of big data analytics in Internet of Things, Comput. Netw. 129, part 2 (2017) 459–471.
[25] M.A.A. da Cruz, J.J.P.C. Rodrigues, A.K. Sangaiah, J. Al-Muhtadi, V. Korotaev, Performance evaluation of IoT middleware, J. Netw. Comput. Appl. 109 (2018) 53–65.
[26] A.H.H. Ngu, M. Gutierrez, V. Metsis, S. Nepal, M.Z. Sheng, IoT Middleware: A survey on issues and enabling technologies, IEEE Internet Things J. 4 (1) (2017) 1–20.
[27] V. Issarny, A. Bennaceur, Y.-D. Bromberg, Middleware-layer connector synthesis: beyond state of the art in middleware interoperability, in: Formal Methods for Eternal Networked Software Systems: 11th International School on Formal Methods for the Design of Computer, Communication and Software Systems (SFM 2011), Bertinoro, Italy, June (2011) 13-18. Advanced Lectures, Springer Berlin Heidelberg, Berlin, Heidelberg, 2011.
[28] Eclipse Foundation, Ponte - Bringing Things to REST developers. [Online]. Available: https://www.eclipse.org/ponte/. [Accessed: 16-Mar-2018].
[29] W.L.D.a.M. Macêdo, T. Rocha, E.D. Moreno, Gothings an application-layer gateway architecture for the internet of things, in: 11th International Conference on Web Information Systems and Technologies (Webist 2015), Lisbon, Portugal, 2015, pp. 135–140.

[30] M. Collina, G.E. Corazza, A. Vanelli-Coralli, Introducing the QEST broker: Scaling the IoT by bridging MQTT and REST, in: 2012 IEEE 23rd International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC 2012), Sydney, NSW, Australia, 2012, pp. 36–41.
[31] A. La Marra, F. Martinelli, P. Mori, A. Rizos, A. Saracino, Improving MQTT by inclusion of usage control, in: Security, Privacy, and Anonymity in Computation, Communication, and Storage, Guangzhou, China, 2017, pp. 545–560.
[32] B. Al-Madani, H. Ali, Data Distribution Service (DDS) based implementation of Smart grid devices using ANSI C12.19 standard, Procedia Comput. Sci. 110 (2017) 394–401.
[33] M. Iglesias-Urkia, A. Orive, A. Urbieta, D. Casado-Mansilla, Analysis of CoAP implementations for industrial Internet of Things: a survey, Procedia Comput. Sci. 109 (2017) 188–195.
[34] GitHub - vivirodrigues/MiddleBridge: This project does the conversion of messages sent by an IoT protocol to HTTP messages. [Online]. Available: https://github.com/vivirodrigues/MiddleBridge. [Accessed: 20-Aug-2018].
[35] T. Zahariadis, et al., FIWARE Lab: Managing resources and services in a cloud federation supporting future internet applications, in: IEEE/ACM 7th Int. Conf. Util. Cloud Comput. (UCC 2014), London, UK, 2014, pp. 792–799.
[36] emqtt/mqtt-jmeter: MQTT JMeter Plugin. [Online]. Available: https://github.com/emqtt/mqtt-jmeter. [Accessed: 20-Aug-2018].
[37] pjtr / jmeter-websocket-samplers — Bitbucket. [Online]. Available: https://bitbucket.org/pjtr/jmeter-websocket-samplers/src/master/. [Accessed: 20-Aug-2018].
[38] Tanganelli/jmeter-coap: jmeter plugin to evaluate CoAP servers. [Online]. Available: https://github.com/Tanganelli/jmeter-coap. [Accessed: 20-Aug-2018].

Mauro A.A. da Cruz received the master's degree in telecommunications from the National Institute of Telecommunications (Inatel), Santa Rita do Sapucaí, Brazil, and a five-year B.Sc. degree (licenciate) in informatics engineering from the Universidade Católica de Angola, Luanda, Angola. He is a Ph.D. student member of the IoT research group supervised by Joel J.P.C. Rodrigues. His current research interests include Internet of Things, middleware for IoT, and mobile computing.

Joel José P.C. Rodrigues [S'01, M'06, SM'06] is a professor and senior researcher at the National Institute of Telecommunications (Inatel), Brazil and senior researcher at the Instituto de Telecomunicações, Portugal. He received the Academic Title of Aggregated Professor in informatics engineering from UBI, the Habilitation in computer science and engineering from the University of Haute Alsace, France, a Ph.D. degree in informatics engineering and an M.Sc. degree from the UBI, and a five-year B.Sc. degree (licentiate) in informatics engineering from the University of Coimbra, Portugal. Prof. Joel is the leader of the Internet of Things Research Group (Inatel), Director for Conference Development - IEEE ComSoc Board of Governors, IEEE Distinguished Lecturer, the President of the scientific council at ParkUrbis – Covilhã Science and Technology Park, the Past-Chair of the IEEE ComSoc Technical Committee on eHealth, the Past- chair of the IEEE ComSoc Technical Committee on Communications Software, Steering Committee member of the IEEE Life Sciences Technical Community and Publications co-Chair, and Member Representative of the IEEE Communications Society on the IEEE Biometrics Council. He is the editor-in-chief of the International Journal on E-Health and Medical Communications, the editor-in-chief of the Journal of Multimedia Information Systems, and editorial board member of several high-reputed journals. He has been general chair and TPC Chair of many international conferences, including IEEE ICC, GLOBECOM, and HEALTHCOM. He has authored or coauthored over 650 papers in refereed international journals and conferences, 3 books, and 2 patents. He had been awarded several Outstanding Leadership and Outstanding Service Awards by IEEE Communications Society and several best papers awards. Prof. Rodrigues is a licensed professional engineer (as senior member), member of the Internet Society, and a senior member ACM and IEEE.

Pascal Lorenz (lorenz@ieee.org) received his M.Sc. (1990) and Ph.D. (1994) from the University of Nancy, France. Between 1990 and 1995 he was a research engineer at WorldFIP Europe and at Alcatel-Alsthom. He is a professor at the University of Haute-Alsace, France, since 1995. His research interests include QoS, wireless networks and high-speed networks. He is the author/co-author of 3 books, 3 patents and 200 international publications in refereed journals and conferences. He was Technical Editor of the IEEE Communications Magazine Editorial Board (2000–2006), IEEE Networks Magazine since 2015, IEEE Transactions on Vehicular Technology since 2017, Chair of IEEE ComSoc France (2014–2018), Financial chair of IEEE France (2017–2019), Chair of Vertical Issues in Communication Systems Technical Committee Cluster (2008-2009), Chair of the Communications Systems Integration and Modeling Technical Committee (2003–2009), Chair of the Communications Software Technical Committee (2008–2010) and Chair of the Technical Committee on Information Infrastructure and Networking (2016–2017). He has served as Co-Program Chair of IEEE WCNC'2012 and ICC'2004, Executive Vice-Chair of ICC'2017, Panel sessions co-chair for Globecom'16, tutorial chair of VTC'2013 Spring and WCNC'2010, track chair of PIMRC'2012 and WCNC'2014, symposium Co-Chair at Globecom 2007–2011, ICC 2008–2010, ICC'2014 and '2016. He has served as Co-Guest Editor for special issues of IEEE Communications Magazine, Networks Magazine, Wireless Communications Magazine, Telecommunications Systems and LNCS. He is associate Editor for International Journal of Communication Systems (IJCS-Wiley), Journal on Security and Communication Networks (SCN-Wiley) and International Journal of Business Data Communications and Networking, Journal of Network and Computer Applications (JNCA-Elsevier). He is senior member of the IEEE, IARIA fellow and member of many international program committees. He has organized many conferences, chaired several technical sessions and gave tutorials at major international conferences. He was IEEE ComSoc Distinguished Lecturer during 2013–2014.

Petar Solic (solic@fesb.hr) received his M.S. and Ph.D. degrees, both in computer science, from the University of Split in 2008 and 2014, respectively. He is currently employed at the Faculty of Electrical Engineering, Mechanical Engineering and Naval Architecture (FESB), University of Split, Croatia, as an assistant professor in the Department of Communication and Information Technologies. His research interests include information technologies, and RFID technology and its application. In 2016 he was awarded with *National prize for science*.

Jalal F. Al-Muhtadi, Ph.D., is the Director of the Center of Excellence in Information Assurance (CoEIA) at King Saud University. He is also an Assistant Professor at the department of Computer Science at King Saud University. Areas of expertise include cybersecurity, information assurance, privacy, and Internet of Things. He received his Ph.D. and M.S. degrees in Computer Science from the University of Illinois at Urbana-Champaign, USA. He has over 50 scientific publications in the areas of cybersecurity and the Internet of Things.

Victor Hugo C. de Albuquerque has a Ph.D. in Mechanical Engineering with emphasis on Materials from the Federal University of Paraíba (UFPB, 2010), an M.Sc. in Teleinformatics Engineering from the Federal University of Ceará (UFC, 2007), and he graduated in Mechatronics Technology at the Federal Center of Technological Education of Ceará (CEFETCE, 2006). He is currently Assistant VI Professor of the Graduate Program in Applied Informatics at the University of Fortaleza (UNIFOR). He has experience in Computer Systems, mainly in the research fields of: Applied Computing, Intelligent Systems, Visualization and Interaction, with specific interest in Pattern Recognition, Artificial Intelligence, Image Processing and Analysis, as well as Automation with respect to biological signal/image processing, image segmentation, biomedical circuits and human/brain–machine interaction, including Augmented and Virtual Reality Simulation Modeling for animals and humans. Additionally, he has research at the microstructural characterization field through the combination of non-destructive techniques with signal/image processing and analysis, and pattern recognition.