# MO2MD: Message-Oriented Middleware for Dynamic Management of IoT Devices

Imen Ben Ida
Electronic systems and communications networks laboratory (SERCOM),
Polytechnic School of Tunisia, Carthage University
Tunis, Tunisia
Email: Imen.benida@gmail.com

Takoua Abdellatif
Electronic systems and communications networks laboratory (SERCOM),
Polytechnic School of Tunisia, Carthage University
Tunis, Tunisia
Email: takoua.abdellatif@ept.rnu.tn

Abderrazek Jemai
Electronic systems and communications networks laboratory (SERCOM),
Polytechnic School of Tunisia, INSAT, Carthage University
Tunis, Tunisia
Email: Abderrazek.Jemai@insat.rnu.tn

*Abstract*— **Middleware are fundamental components for Internet of Things (IoT) solutions. They provide general and specific abstractions through which smart devices and their related applications can be easily interconnected. However, due to the wide variety of software and hardware technologies of IoT solutions, the management of the connected devices is still a challenging task, especially for Cloud-Edge based solutions. In this paper, we take advantage of message-oriented computing and Web technologies to propose a solution for devices management without having to worry about the underlying infrastructures or implementation details. In particular, we describe a Web-based middleware that enables configurability and manageability of connected devices in a dynamic manner at the Cloud level.**

***Keywords-Message-Oriented Middleware; IoT; Cloud comuting; Edge computing; Devices Management.***

## I. INTRODUCTION

IoT devices, also known as smart objects, are projected to grow exponentially both in terms of quantity as well as variety [1]. Some examples include wireless body sensors, smart vehicles, and surveillance cameras. By connecting devices in an Internet-like structure, a variety of data can be collected, which is of great benefit in industry and daily life. To support such data streams, the Cloud infrastructure provides several services namely, Software as a Service (SaaS), Platform as a Service (PaaS) and Infrastructure as a Service (IaaS) for massive-scale and complex data computing. It takes advantage of virtualized resources, parallel processing and data service integration [1].

However, due to the explosive growth of connected lightweight devices, Cloud computing is facing increasing challenges, especially in managing and reconfiguring IoT devices. The challenge of flexible devices configuration from the Cloud layer is due to several domain requirements, such as context-awareness and delay-sensitive control [2]. In addition, designing and implementing an appropriate mechanism that can dynamically manage resources across the Cloud-IoT spectrum is a challenging task to resolve due to the highly dynamic behaviors of the devices [1].

As response to this challenge, the Edge computing paradigm enables the Cloud resources to move closer to the devices network by offering localized devices configuration. The Edge layer is exploited by reinforcing edges, such as gateways, with sufficient processing power, intelligence, and communication capabilities to integrate efficiently the Cloud layer with the sensors layer and to provide efficient configuration. It provides not only local data processing and data storage, but also it ensures local management of the IoT devices [3].

In this paper, we explore the Edge computing paradigm to implement a distributed solution for a dynamic management of IoT devices. In particular, we propose a Message-Oriented Middleware for Dynamic Management of IoT Devices (MO2MD). Middleware are widely used in distributed systems and they are considered as fundamental tools that provide general and specific abstractions for the design and the implementation of smart environment applications [4]. A middleware can ease the development process by integrating heterogeneous computing and communications devices and by supporting the interoperability within the diverse applications and services [5].

More specifically, our proposed middleware MO2MD offers a flexible configuration of IoT devices to support a dynamic management of the Edge layer. The remainder of this paper is organized as follows. In Section 2, we describe the message-oriented approach for IoT middleware and the challenges of IoT devices. Some related works are highlighted in Section 3. Our proposed solution is detailed in Section 4. Section 5 presents concluding remarks and future work.

## II. BACKGROUND

### A. *Message-oriented Middleware for Internet of Things*

A middleware is a software component that allows programming IoT solutions with a higher level of abstraction. It provides a simpler interface to ensure appropriate abstractions and mechanisms for dealing with the heterogeneity of IoT devices. It simplifies the development and execution of distributed applications and hides their complexity. In particular, middleware removes the programmer from the complex aspects of IoT, such as the handling of wireless communications, power management and hardware programming.

In event-based middleware, components, applications, and all the other participants interact through events. Each event has a type, as well as a set of typed parameters whose specific values describe the change to the producer's state [5]. Events are propagated from the sending application components (producers) to the receiving application components (consumers).

Message-Oriented Middleware (MOM) is a type of event-based middleware. In this model, the communication is based on messages. Generally, messages carry publisher and subscriber addresses and they are delivered by a particular subset of participants, whereas events are broadcast to all participants.

## *B. Challenges of IoT Devices Management*

We present in the following paragraphs the main challenges of implementing middleware components to manage IoT devices [2][3][5]-[7]:

1) Heterogeneity:

Device heterogeneity emerges from different characteristics, such as differences in capacity, features, and application requirements. Added to that, the emergence of new protocols as an important lightweight support for the IoT devices communication requires appropriate communication mechanisms for the delivery, support and management of the different types of resources.

2) Dynamic behaviors:

In an IoT environment, thousands of devices may interact with each other even in one local place (e.g., in a building, supermarket, and hospital), which is much larger scale than most conventional networking systems. The interactions among a large number of devices will produce an enormous number of events. This may cause problems, such as event congestion and reduced event processing capability. Consequently, any predefined and fixed set resource management policies will be rendered useless for a dynamic management of devices.

3) Scalability:

Scalability is a significant challenge for current IoT platforms, where lightweight IoT devices can hardly extend their functionalities by adding new hardware modules. For example, it is very difficult to integrate a temperature detection service on an IoT device by simply attaching a temperature sensor. Added to that, other components, such as resource discovery and data analytics of IoT solutions need to be scalable to achieve system-wide scalability.

4) Resource Constrains:

With smaller, more compact sensors, the available battery power is always limited. The IoT systems must be designed to manage limited power by designing efficient processes and capabilities of the sensors. Mechanisms to ensure efficient power consumption are necessary for IoT-based services.

## III. RELATED WORK

Numerous middleware proposals have been put forward for IoT-based applications. They provide abstractions through which connected devices and their related applications can be easily built up and managed.

In database-oriented middleware, the devices are considered as virtual objects in a relational database, so that an application can perform queries using a syntax in SQL language, allowing complex queries to be performed. For example, Global Sensor Net-works (GSN) [8] is an IoT middleware that aims to provide flexible management of heterogeneous IoT devices. It enables developers to specify XML-based deployment descriptors to deploy a sensor. An implementation of the wrapper in Java is required in order to add a new type of sensor to the middleware.

In event-driven middleware, components, applications, and other participants interact through events. In [9], the authors present an event-driven user-centric middleware for monitoring and managing energy consumption in public buildings and spaces. The proposed middleware allows the integration of heterogeneous technologies in order to enable a hardware independent interoperability between them.

UbiSOAP (Service-Oriented Middleware for Ubiquitous Networking) is a service-oriented middleware [10] that provides complete integration of the network with Web Services. The architectural resources layer has the necessary functions, including a unified abstraction for simple services (sensors, actuators, processors or software components) to help integrate applications and services with resources. A service support component facilitates the discovery and dynamic composition of resources (eg services). Dynamic composition and instantiation of new services are facilitated by semantic models and descriptions of sensors, actuators and processing elements.

In [11], the authors present a QoS aware publish/subscribe middleware for Edge computing called EMMA (edge-enabled publish–subscribe middleware). They show that EMMA can provide low-latency communication for devices in close proximity, while allowing message dissemination to different locations at minimal overhead costs. Gateways allow existing publish/subscribe client infrastructure to transparently connect to the system.

Another message-oriented middleware used for communication between the system services of a proposed framework is presented in [12]. The authors analyze the driver real-time data using a distributed system architecture and send alert messages in a timely manner. According to their experimental results, the middleware design increases the speed and stability of information transmission.

Other types of middleware are used in a distributed environment, such as Transaction-Oriented Middleware (TOM), which is used to ensure the correctness of transaction operations and Object-Oriented/Component Middleware (OOCM), which is based on object-oriented programming models requests [13].

Considering the characteristics of middleware introduced above and the goal of a dynamic management of IoT devices, it is possible to argue the suitability of message-oriented middleware for a flexible configuration. The interactions of database-driven and object-oriented models are synchronous, which limits the scalability to large volumes of data. Not being designed for concurrent event management, these usually do not attain the same level of performance as systems designed for the event-based interaction paradigm.

## IV. MO2MD PRESENTATION

Our proposed message-oriented middleware for Dynamic Management of IoT Devices, named MO2MD, extends the capabilities of messages-oriented middleware and provides high flexibility for adding new configurations of devices at the Edge layer. The proposed middleware considers all connected devices, such as sensors and actuators, as data providers. We focus on the challenge of dynamic behavior and scalability.

### *A. Architecture*

The proposed architecture is depicted in Figure 1. The implementation of MO2MD is achieved through a distributed architecture which integrates global Cloud services with local services in different Edge nodes. All participating Edge nodes communicate with the Cloud layer to exchange devices data and to receive configuration data. The following paragraphs describe the principal components of the proposed architecture:

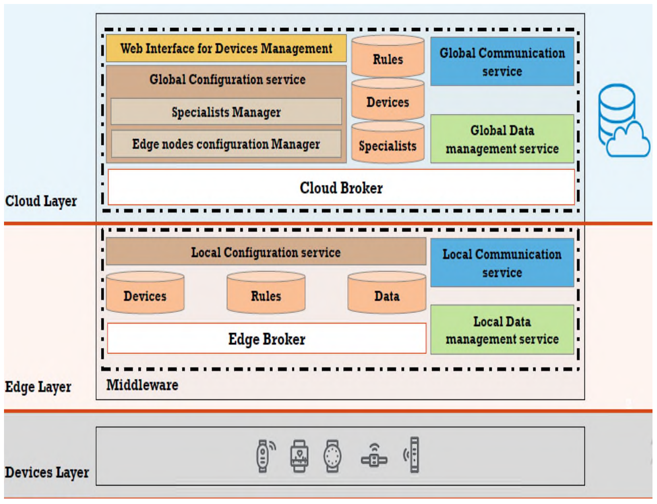
Web Interface for Devices Management
Rules
Global Communication service
Global Configuration service
Devices
Specialists Manager
Edge nodes configuration Manager
Specialists
Global Data management service
Cloud Broker
Cloud Layer
Local Configuration service
Local Communication service
Devices
Rules
Data
Edge Broker
Local Data management service
Edge Layer
Middleware
Devices Layer

Figure 1. MO2MD architecture.

#### *1) Cloud Layer*

The Cloud layer is composed of a cluster of servers with massive computing and storage resources. This layer communicates with three types of participants:

- *Admins*

They are the middleware users who control the different IoT devices. They are responsible of adding or modifying the middleware configuration.

- *Specialists*

They are responsible for interventions in case of devices malfunction. They are considered as data subscribers and receive notifications about the devices status depending on their subscriptions.

- *Gateways at the Edge layer*

They are local gateways, in which local services process the collected data from different devices and apply the configuration requests received from the Cloud layer.

A Configuration Web application is the entry point of the Cloud layer services. It allows the middleware admins to view everything as a single large system that provides basic and advanced services. The Web application provides management tools to control the whole IoT platform and to request new configurations that will be processed by a global configuration service and published to the corresponding Edge Node. The exchange of messages between the Cloud layer and the Edge layer is ensured using the publish/subscribe pattern [14]. A global broker handles the requests of configuration from the admins and publishes the submitted configurations as messages to the corresponding Edge node.

#### *2) Edge Layer*

The Edge layer reinforces the devices layer with storage, processing and communication capabilities. It is the intermediate layer between the IoT devices and the Cloud services. A local broker service in each Edge node regularly updates the global broker about the availability and status of their devices and it receives the data provided by IoT devices. The Edge layer aims to provide low latency responses.

#### *3) Distributed Services for dynamic management of IoT devices*

##### *a) Data management service*

Data are the key of efficient devices management. In the proposed middleware, we consider 3 types of exchanged data, which are presented in Table 1.

TABLE I. DATA TYPES

| Sensed data | Configuration data | Notifications |
|---|---|---|
| Data collected by the devices. | The configuration parameters requested by the admins. | Notifications of the devices dysfunction. |

The proposed middleware provides data management services, such as data acquisition, data processing and data storage.

The configuration data are the parameters and the rules which must be taken in consideration for data management at the Edge layer. In case of non-respect of configured rules, notification messages are sent to the concerned specialists.

##### *b) Configuration service*

The core component of the middleware is the global configuration service which is composed of two major units. The first unit is the specialist's manager, which allows the admin to add, update or delete the corresponding specialists of each type of device. The registered specialists will be notified in case of an abnormal behavior of any device at the Edge layer. The second one is the Edge nodes manager, which is responsible for processing the configuration requests of the admins. We choose as configuration requests:

- The interval of saving data in the local data base of each device.
- The priority of each device.
- The corresponding specialist to notify in case of device disfunction.
- The interval of sending data to the Cloud layer.
- Specific rules for each device.

All the requests may be introduced through the Web application, depending on the decision of the admins. This Web-based configuration enables a dynamic management of the device's behaviors. For example, in the case of smart buildings, the buildings' owner can change the interval of saving temperature data depending on the building location. Another example, in a smart hospital, a doctor can modify the priority of each medical device depending on the patient situation.

The configuration data, the specialist's subscriptions and the list of active devices are saved in both the Cloud and the Edge layer. Each configuration is considered as a rule to respect at the Edge layer. In Figure 2, we illustrate 2 different scenarios of data exchange between the Edge layer and the Cloud layer.

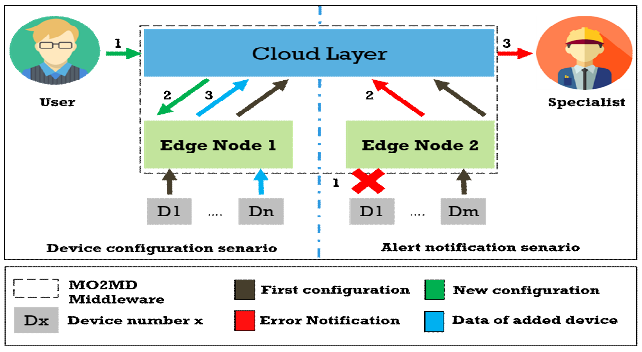

Figure 2. Scenarios of dynamic data management.

### c) *Communication service*

The communication service supports a publish-subscribe messaging service that ensures data-centric communication between the Cloud layer and the Edge nodes. It also supports the link between external databases and the Cloud layer.

With the publish-subscribe pattern, the local communication managers act as publishers on a given topic and send messages without the need to know about the existence of the receiving clients, who are the Specialists. At the Edge layer, the communication service ensures the data storage in a local database and the synchronization between the Cloud broker and the local services to send the collected data and any dysfunction detection.

## B. *Implementation and evaluation*

The implemented middleware is based on Node.js which allows multi-platform development and supports JavaScript-based webservers. The I/O architecture of Node.js is based on non-blocking asynchronous event-driven which makes it suitable for data-intensive and real-time applications in lightweight and efficient way. The Edge is a Raspberry Pi 3 which is a low-cost small-sized single board with 1 GB of Ram and 1.2 GHz processor [15]. The messages exchanged are ensured by MQ Telemetry Transport (MQTT) protocol [16]. The MQTT protocol is a lightweight application layer protocol designed for resource-constrained devices. It uses the publish-subscribe messaging system combined with the concept of topics to provide one-to-many message distribution. It supports a range of 10 to 100 messages per second.

We install the InfluxDB database, which is an open-source Time Series Database. At its core is a custom-built storage engine called the Time-Structured Merge (TSM) Tree, which is optimized for time-series data. InfluxDB provides support for mathematical and statistical functions across time ranges; also it is developed for custom monitoring, metrics collection and real-time analytics [17].

To prove the benefits of the dynamic configuration of the Edge layer, we consider three different scenarios of controlling 10 devices in one day. We suppose that the static configuration of the interval of data storage in the local database is a second. In each scenario, we change this interval for a certain number of devices. Table 2 shows the interval configuration for each scenario, as well as the percentage of gain in terms of memory compared to the static configuration.

TABLE II. CONFIGURATION SCENARIOS

| Scenario | Number of devices per time | | | Gain of memory |
|---|---|---|---|---|
| | *hour* | *minute* | *second* | |
| Fixed case | 2 | 1 | 7 | 29.83 % |
| Custom case 1 | 5 | 2 | 3 | 69.65 % |
| Custom case 2 | 2 | 7 | 1 | 88.82 % |

Figure 3 shows that the use of a single data processing strategy in a fixed and non-custom way can result in an unnecessary use of gateway memory which obviously affects performance and the time reaction e in emergency cases.

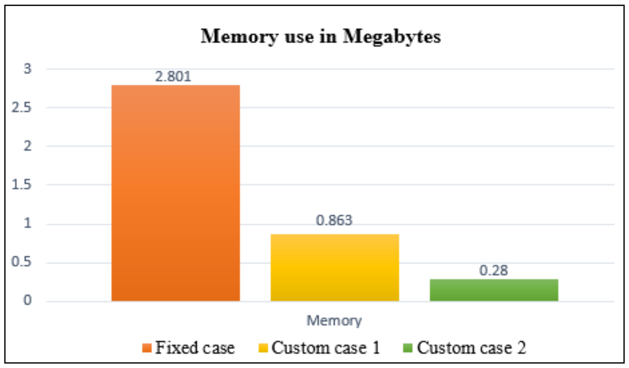

Figure 3. Scenarios of dynamic data management.

## V. CONCLUSION AND FUTURE WORK

Flexible configuration requires the development of a dynamic mechanism of devices management. In this paper, we describe a message-oriented middleware that offers distributed services for IoT devices management. The proposed architecture supports a flexible configuration of connected devices in different locations from a Web application. In particular, we show that our middleware gives the possibility to modify the data storage interval at the Edge layer in order to personalize the devices behavior and realize resources optimization.

Future work includes the complete implementation of the proposed middleware, including its security and reliability guarantees, as well as automatic resource discovery to realize autonomous devices deployment at the Edge layer.

## REFERENCES

[1] H. El-Sayed et al., "Edge of Things: The Big Picture on the Integration of Edge, IoT and the Cloud in a Distributed Computing Environment," IEEE Access, vol. 6, pp. 1706-1717, 2018. doi: 10.1109/ACCESS.2017.2780087

[2] J. Ren, H. Guo, C. Xu and Y. Zhang, "Serving at the Edge: A Scalable IoT Architecture Based on Transparent Computing," IEEE Network, vol. 31, no. 5, pp. 96-105, 2017. doi: 10.1109/MNET.2017.1700030

[3] S. Shekhar and A. Gokhale, "Dynamic Resource Management Across Cloud-Edge Resources for Performance-Sensitive Applications," 2017 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID), Madrid, 2017, pp. 707-710.

[4] G. Fortino, A. Guerrieri, W. Russo and C. Savaglio "Middleware for Smart Objects and Smart Environments: Overview and Comparison," G. Fortino,P. Trunfio,eds. , Internet of Things Based on Smart Objects. Internet of Things (Technology, Communications and Computing), Springer, Cham, 2014, pp. 1-27.

[5] M. A. Razzaque, M. Milojevic-Jevric, A. Palade and S. Clarke, "Middleware for Internet of Things: A Survey," in IEEE Internet of Things Journal, vol. 3, no. 1, pp. 70-95, Feb. 2016. doi: 10.1109/JIOT.2015.2498900

[6] L. Alonso et al., "Middleware and communication technologies for structural health monitoring of critical infrastructures: A survey," Computer Standards & Interfaces, vol. 56, pp. 83-100, 2018, doi: 10.1016/j.csi.2017.09.007

[7] N. Naik, "Choice of effective messaging protocols for IoT systems: MQTT, CoAP, AMQP and HTTP," 2017 IEEE International Systems Engineering Symposium (ISSE), pp. 1-7, Vienna, 2017. doi: 10.1109/SysEng.2017.8088251.

[8] A. H. Ngu et al, " IoT Middleware: A Survey on Issues and Enabling Technologies," IEEE Internet of Things Journal, vol. 4, pp. 1-20, Feb. 2017, doi: 10.1109/JIOT.2016.2615180

[9] E. Patti et al., "Event-Driven User-Centric Middleware for Energy-Efficient Buildings and Public Spaces," IEEE Systems Journal, vol. 10, pp. 1137-1146, Sept. 2016, doi: 10.1109/JSYST.2014.2302750

[10] M. Caporuscio, P.G. Raverdy and V. Issarny, "Ubisoap: a service-oriented middleware for ubiquitous networking," IEEE Trans. Serv. Comput., vol. 5,pp. 86–98, 2012. https://doi.org/10.1109/TSC.2010.60

[11] X. Xu et al., "EAaaS: Edge Analytics as a Service," 2017 IEEE International Conference on Web Services (ICWS), Honolulu, HI, 2017, pp.9-356.

[12] P. Lai, C. Dow and Y. Chang. "Rapid-Response Framework for Defensive Driving Based on Internet of Vehicles Using Message-Oriented Middleware," IEEE Access, vol. 6,pp. 18548-18560, 2018, doi: 10.1109/ACCESS.2018.2808913

[13] M. Albano, L.L Ferreira, L. M. Pinho, and A. R. Alkhawaja, " Message-Oriented Middleware for smart grids," Computer Standards & Interfaces, vol.38, pp. 133-143, 2015. https://doi.org/10.1016/j.csi.2014.08.002

[14] A. Hakiri, P. Berthou, A. Gokhale and S. Abdellatif, "Publish/subscribe-enabled software defined networking for efficient and scalable IoT communications," IEEE Communications Magazine, vol. 53, no. 9, pp. 48-54, September 2015. doi: 10.1109/MCOM.2015.7263372

[15] J. Bermúdez-Ortega et al., "Remote Web-based Control Laboratory for Mobile Devices based on EJsS, Raspberry Pi and Node.js", IFAC-PapersOnLine,vol. 48, no. 29, pp. 158-163, 2015.

[16] A. Banks and R. Gupta, MQTT Version 3.1. 1. OASIS standard, vol. 29, 2014.

[17] C. Rudolf, "SQL, noSQL or newSQL–comparison and applicability for Smart Spaces," Network Architectures and Services, 2017.