An MQTT Brokers Distribution Based on Mist Computing for Real-Time IoT Communications

Fatma Hmissi (fatma.hmissi@ensi-uma.tn)

National School of Computer Science (ENSI) University of Manouba https://orcid.org/0000-0001-5216-5577

Sofiane Ouni

National School of Computer Science (ENSI) University of Manouba

Research Article

Keywords: Internet of Things, Time-sensitive Application, MQTT, Mist Computing, Broker Distribution

Posted Date: July 13th, 2021

DOI: https://doi.org/10.21203/rs.3.rs-695717/v1

Wireless Personal Communications manuscript No.
(will be inserted by the editor)

An MQTT Brokers Distribution Based on Mist Computing for Real-Time IoT Communications

Fatma Hmissi * · Sofiane Ouni *

Received: date / Accepted: date

Abstract As we consider the number of IoT time-sensitive applications , the transfer of data to a remote data center and server such as Cloud, Fog, and Edge becomes inefficient since the deadline constraint is not satisfied. Thus, ensuring that the IoT time-sensitive applications meet their timing constraints is a challenge. Mist Computing is closer to IoT devices, presenting the lowest communication delay but less computational resource than the Cloud, Fog, and Edge. Seeing several IoT devices use MQTT protocol to access the data due to its lightness and flexibility, we propose an architecture for IoT time-sensitive applications based on MQTT protocol and integrating Mist Computing. We focus on distributing the MQTT brokers over Mist nodes to satisfy the deadline constraints with the consideration of the limited resource of Mist nodes. Hence, we propose an approach for the selection of the appropriate MQTT Mist broker. We have also proposed MQTT communication model that provides the M/M/1 based analysis for delay computing and energy conception. The experiment results show that our proposal is very effective for time-sensitive applications and also maximize the lifetime of IoT systems since it minimizes the cumulative energy of the system. Compared to MQTT Edge broker distribution, our solution provides the lesser delay of communication between IoT devices.

Keywords Internet of Things · Time-sensitive Application · MQTT · Mist Computing · Broker Distribution

* Cristal Laboratory, National School of Computer Science (ENSI) University of Manouba, Tunisia

F. Hmissi
E-mail: fatma.hmissi@ensi-uma.tn

S. Ouni
E-mail: soufiane.ouni@insat.rnu.tn

1 Introduction

The Internet of Things, shorted as IoT, is a trending topic of our daily life, industrial and researcher life. As technology continues to develop, the IoT sweep the world. The main aim of the IoT is to allow communication between physical devices without the human-been intervention by connecting them to the Internet. And for such communication, every single physical device is being integrated with sensors and actuators.

To permit communication in IoT systems, many messaging protocols have been proposed to deal with limited capabilities of the physical devices i.e CoAP, MQTT. And according to recent research, the use of the broker-based IoT middle-ware increase swiftly due to its adapted properties with the requirements of IoT networks namely the MQTT protocol. The MQTT messaging protocol allows connectivity in IoT and M2M applications over TCP/IP networks using the Publisher-Subscribers structure. IoT communications using MQTT are asynchronous. Here, every client continues to do his tasks without blocking the CPU cycles just for waiting. In the MQTT structure, a client can be either a sender of information, named the MQTT publisher, or a receiver of the required information, called the MQTT subscriber, at the same time. The MQTT clients exchange messages through a central server known as the MQTT broker. The MQTT is a suitable communication protocol for IoT systems since it has many characteristics that deal with IoT requirements. Generally, the MQTT is a lightweight protocol used in environments with low-bandwidth and low-power. Despite the progress of using the MQTT protocol, however, it does not allow real-time communications. Within real-time IoT applications, communication requires data delivery constraints. Generally, captured data delivery delays in real-time IoT applications are bounded. Assuredly IoT communication in real-time applications is problematic.

As the IoT devices have limited capabilities like low-power, limited memory, low communication bandwidth, an increasing amount of Computing paradigm has been proposed to provide several services such as Cloud Computing (CC), Fog Computing (FC), Edge Computing (EC), and Mist Computing (MC). The Mist Computing paradigm is the center of interest in this study. In Mist Computing, IoT devices analyze captured data locally instead-to transfer them to the Cloud. Such devices are known as analysis servers, named Mist nodes. The real-time analytic of data make the model of Mist Computing suitable for real-time applications.

Consequently, the fundamental key to the success of real-time IoT application is the adaptation of the MQTT protocol according to the Mist Computing paradigm. Intending to provide real-time communication, such adaptation refers to MQTT Broker Distribution in Mist Computing . Hence, the location of the MQTT broker is of the utmost importance because all IoT devices escorted by the same topic must know and be connected to the broker that answers within real-time constraints. As the number of devices connected to the broker enlarge, more and more broker transfer latency get larger owing to resources lessening e.g CPU, Power. Suitably, a mechanism is required to select and set MQTT Brokers for IoT devices. Thus, this work aims to propose an MQTT middle-ware based on Mist Computing, named MQTT-MBD, to allow real-time communication in IoT time-sensitive applications, and propose a

mechanism to discover the suitable broker for the publisher and subscribers related to the same topic.

The remainder of this paper is organized as follows. Section 2 gives a general overview of the messaging protocol used in this study which is MQTT. Section 3 surveys some of the recent works that use distributed MQTT middle-wares based on Edge Computing in IoT systems. Section 4 introduce and describe the architecture of distributed MQTT brokers based on Mist Computing. Section 5 introduce our proposed MQTT-MBD approach by studying it's principle and operation process needed to connect to the suitable MQTT broker. Section 6 details the operation of the MQTT broker selection and provides the algorithm for Mist Broker selection. Sections 7 and 8 introduce the models used to analyze the performances of the proposed approach, in section 7 we analysis the end-to-end response time ,and in section 8 we provide the energy and CPU models. Section 9 concerns the approach implementation and the performance evaluation, in this section we show the effectiveness of the MQTT-MBD approach and the MQTT broker selection process through experiments. Finally, section 10 concludes the paper.

2 MQTT Protocol

The main focus of this work is the IoT messaging protocol. In this section, a general overview of the MQTT protocol is given.

Message Queuing Telemetry Transport is widely known by the acronym MQTT, is an IoT messaging protocol. MQTT is a lightweight and flexible messaging protocol based on the Publish-Subscribe communication pattern. Mainly designed for constrained devices to minimize devices resources requirements and network bandwidth. As figure1 shows, four main terminology appear in the process of message exchange of MQTT. Let's present these terminology to understand the message exchange process of MQTT.

- MQTT Publisher is an MQTT client that acts as a sender which send a message to an MQTT broker to a particular topic, MQTT publishers are sensors like humidity sensors.
- MQTT Broker is a server that receives all data from the MQTT publisher and then routes the published data to the appropriate destination subscriber.
- Topic is considered as message subject that allows publisher and subscriber to exchange data with a defined semantic.
- MQTT Subscriber is an MQTT client that acts as a receiver which subscribe to receive a published data on a particular topic from the MQTT broker, MQTT subscribers are entities which consume data such as smartphone, iPad or other IoT device.

MQTT offer three level of **Q**uality **o**f **S**ervice (formed by QoS) for the process of the message exchange between publishers and MQTT broker or between subscribers and MQTT broker. The three level of QoS are 0, 1 and 2. The vocabulary QoS represent the message delivering assurance. The publisher or the subscriber are charged to choose the QoS level. To learn more about the MQTT QoS, let's dig just a bit

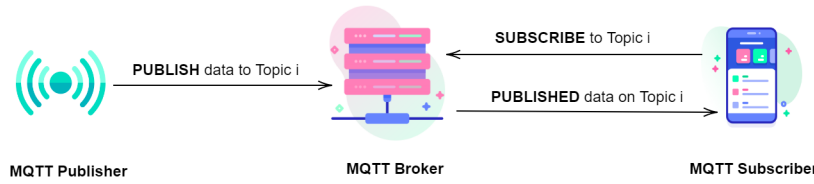

Fig. 1 Message exchange process in MQTT

deeper into the message exchange process between the publisher and MQTT broker. As shown in Figure 2 :

- QoS=0: the publisher sends message to Broker only once with QoS=0, in response it does not wait for acknowlEdgement (ACK) from broker. In this QoS, there is no guarantee that the broker receive the data sent by the publisher, and if the data sent by the producer is not received by broker, it is lost as there are no re-transmissions in this QoS.
- QoS=1: publisher sends the data to Broker at least once with QoS=1. To avoid the problem of the loss of data (in level zero), the publisher waits for ACK (PUBACK) from broker. If the ACK is not received after a predefined time interval (Time Out), data is re-transmitted even if the broker receives the data sent by the publisher.
- QoS=2: Publisher sends the data to Broker exactly once with QoS=2. Publisher sends data to broker and wait for Publish Receive (PUBREC) message back. As soon as the PUBREC is received, it discards the reference to published data and Publish Released (PUBREL) to the broker. As soon as the PUBREL is received, it complet the message exchange and Publish Complet (PUBCOM) to the Sender. When both publisher and broker perform their tasks, it ensures successful message delivery.

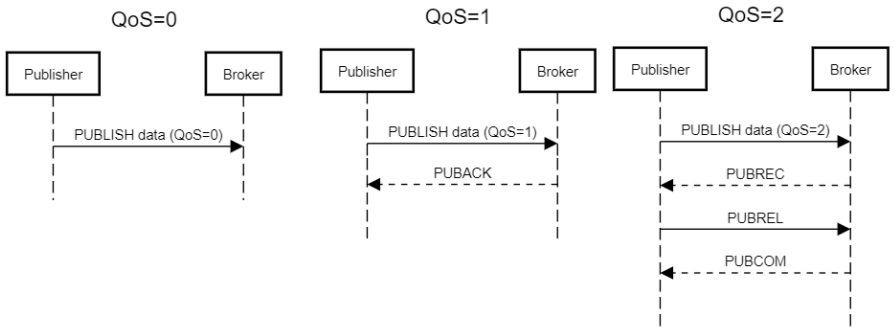

Fig. 2 Message exchange process between publisher and broker with QoS=0,1 and 2.

The delivery of MQTT topics is important in the MQTT protocol. That's why MQTT has 3 delivery modes of message. Therefore, it is necessary to understand the principle of each delivery mode :

- Default: The general principle of this mode is to filter the messages published by topics, then check the correspondence between the topics of the filtered messages and those of the registered customers. If a match is found the broker forwards the published message to the subscribed clients. For this mode of delivery the broker does not keep the messages in memory he serves the message immediately at the time of his arrival to the server.
- Retain: In this mode, all subscribers do not wait to receive a topic, each time they demand a value of the topic they will be served immediately. MQTT broker always retains the last published event on a specific topic.
- Persistent: Here, every single subscribers open a persistent session with the broker to save all the published data on a topic. The use of this mode guarantees the delivery of undelivered messages when the subscriber reconnects after disconnections.

3 Related Works

In recent years, the need for distributed paradigm has arisen for modern IoT systems since the increase of the problems caused by an amount of connected IoT devices in the Cloud Computing Paradigm. Although these paradigms serve the same purpose, they do not all have the same communication function. Therefore, it is required to change IoT middle-wares communication according to specific requirements. In this sense, several related works are engaged in research to optimize existing MQTT middle-wares. Wherefore, this section present and discuss the most recent research focusing on Edge Computing architectures.

Rausch et al. [1] have proposed a distributed communication middleware of Edge Computing application based on MQTT messaging protocol. The principal purpose of the proposed middleware is to connect multiple clients to the available Brokers with consideration of dynamic broker network topology and variation in the availability of resources. To this end, the authors have proposed a distributed brokers. Publishers and subscribers are connected indirectly to the available brokers through gateways deployed either for local or remote networks. In addition, they suggested using an external control agent to verify the status of each broker. Broker status such as broker failure and availability increased latency between gateway and broker influence and change MQTT client-broker connections established via a gateway.

To solve problems caused by massive communication in IoT architecture based on Cloud Computing, Park et al. in [2] encouraged the usage of the Edge Computing paradigm. Given the advantages of the MQTT protocol, the authors have recommended adapting and enhance the MQTT middle-ware to ensure data transmission, the authors recommended. To do this, they suggest a static MQTT broker network topology based on tree hierarchies. Besides, they began to propound the idea of creating per-topic multi-cast groups as an enhancement to the MQTT, and that by the use of Software-defined networking (SDN) technology. The authors proposed to deploy

an SDN controller to construct the multi-cast tree according to the gathering information about clients. The SDN controller uses the master broker to collect clients' information from the different slave brokers (Edge brokers).

The contributors [3] have proposed a distributed resource discovery architecture for M2M communications based on MQTT protocol. IoT devices usually used to collect data can look up other devices and their services in local networks or the Internet. For this purpose, the framework is made up of two sniffer types in LAN: Local Sniffer and Internet Sniffer. Every LAN contains at least one Local Sniffer and exactly one Internet Sniffer. MQTT-RD sniffer used for resource discovery is associated with MQTT broker to communicate with other components. CloudMQTT is MQTT brokers which are hosted on the public Cloud, those brokers are required to be used with Internet sniffing communications on the Internet. Every Sniffer contains an information file copy named Resource Collection about existing resources on the network. As everyone knows the resources network topology is dynamic for this purpose sniffers should update their resources collection files regularly. Information about MQTT devices and sniffers should appear in the RC files for this purpose every resource needs to register in a Sniffer. To register in a Sniffer, the resource should know his IP address. To know the IP address of the appropriate Sniffer resources connects to the same multi-cast IP address when joining the network. All resources connected to that address can receive and respond to the ping message of new joining resources. Through this process, every process can know the IP address of the sniffer to which it will join.

The study on distributed MQTT middle-wares leads us to present a comparative table of research works discussed above. Table 1 presents the classification metrics supported for each approach. The classification metrics are:

- Distributed and decentralized: Describe the type of architecture of the middleware. Distributed middle-ware architecture eliminate centralisation completely where nodes are not co-located on the network but they are geography distributed. Since goals of IoT application cannot be achieved using a single node this architecture is used to divide treatment over different nodes. In a decentralized architecture all the nodes of IoT system are equal and located in the same physical location. Decentralized middle-ware architecture avoid the biggest problem of single point of failure with centralized network. The central network owner are multiple point of contact for information sharing. Information constantly copied to multiple central owners.
- Real-time : Describe whether the studied middleware enable real-time communication. Time-sensitive IoT application need to operate in real-time and should meet their deadlines. Continuously, data flow about events come from IoT sensors and applications. Therefore, it is necessary to produce timely responses to that events. This may involve reducing latency.
- Resource discovery and management: Describe whether the studied middle-ware offer a mechanism for resource discovery and management. The resource discovery is the search of topics by each device.
- Mobility support : Describe whether the studied middle-ware support devices mobility. A significant number of devices in IoT environment is unstable, they

changes place continuously. As a result, they should be reconfigure to succeed join to IoT environment. According to the context of IoT environment some studied middle-wares take mobility into account.
- Dynamic configuration : Describe whether the studied middle-ware provide dynamic configuration. IoT environment are continuously monitor to determine their changes and dynamically reconfigure the system accordingly in order to minimise data loss, communication delays and reconfiguration costs.
- Zero configuration : New devices are automatically integrated into large IoT environments.
- Massive communication : As the amount of IoT devices connected to a node increase causing massive connection to that point, therefore causing massive communication to a single node.
- Transparently connection : Clients should be unaware of the broker architecture neither their location, IP address and name and should be transparently connected to brokers.

This study allows us to notice that:

- The use of a distributed MQTT broker architecture as [3,2,2] is gainful for modern real-time IoT scenario since this allows to avoid dependence problem between the different component on the network.
- The common idea in the existing contributions [1] consist to provide MQTT middle-ware for IoT applications which optimize the delay. However, study undertaken has revealed that the traffic routing by selecting the closest brokers to optimize the delay of communication and congestion not enough to obtain effective real-time real-time IoT system. Therefore, to obtain an effective real-time real-time IoT system a mechanisms for stable broker selection is needed.
- Papers [1,3] offer mechanism for resource discovery and that allows the detection unpredictable resource availability.
- Concerning devices mobility, [2] approach don't support it. Proposal of a middleware like EMMA [1] and DM-MQTT [3] that support mobility of devices is necessary.
- The use of an MQTT middle-ware that gives more stable network meaning dynamic and zero configuration is suitable, as EMMA [1] and MQTT-RD [3] made. This is why DM-MQTT [2] can't be considered sufficient for IIoT and real-time applications. To provide a stable network connection between MQTT devices and brokers should be transparent.
- The amount of data gathered increases exponentially according the growth of the number of connected devices. Authors in [2] take into consideration massive communications when proposing their middle-wares.

4 Architecture Overview

Cloud Computing was the ideal paradigm for real-time IoT applications. It soon became clear that Cloud Computing has many limits that affect negatively real-time IoT application performance related to communication delay. The trend of solving

Table 1 Main requirements of the proposed approach: ✗indicates that the requirement is not met, ✓indicates that the requirement is supported

Requirement	EMMA [1]	DM-MQTT [2]	MQTT-RD [3]
Distributed	✓	✓	✓
Real-time	✗	✗	✗
Delay optimization	✓	✓	✗
Resource discovery	✓	✗	✓
Resource management	✗	✗	✓
Mobility support	✓	✗	✓
Dynamic configuration	✓	✗	✓
Zero configuration	✓	✗	✓
Massive communication	✗	✓	✗
Transparently connection	✓	✗	✓

this problem has been in the direction of performing calculations as possible close to the device. To this end, many researchers tried to outline a new paradigm.

In this regard, many research studies suggest the use of Fog Computing. Studies [5,12,9,11,35] show how Fog Computing benefit real-time IoT applications. In [8, 13,10] a general frameworks for industry applications were introduced. Papers [4,6, 7] use the Fog Computing for specific real-time applications. In [25] the authors have suggested a framework for Fog resource management where they have an interest in resource provisioning, they provided a delay-sensitive utilization of available Fog-based computational resources. Fog is defined as mini-Clouds, these instances are deployed near to the IoT devices [14]. The main merits of the use of Fog Computing are generally related to minimizing bandwidth, latency, to support devices' mobility and improve security level. But over time the massive communication traffic in local networks degrades the Computing resources i.e CPU, RAM which degrade the latency. So it is necessary to have another complementary paradigm of Fog Computing. To solve this problem a paradigm has been proposed that aims perform calculations much closer to IoT devices. Through this paradigm, researchers sought to have low latency but with more limited Computing resources. This paradigm is called Mist Computing.

Mist Computing is located at the Edge of the network. In paper [28], authors have introduced an algorithm to decide the best infrastructure Cloud, Fog, or Mist based on restriction requirements such as latency. Authors in [29] have suggested the use of Mist Computing to secure the architecture of smart and connected health. Papers [30,31,32,33,34] the Mist Computing paradigm is presented.

We can conclude that these three paradigms are complementary. So to benefit from the advantages of each paradigm it is very important to combine them. Also, it should be noted that the use of heavy communication protocols i.e HTTP introduces code complexity and message size overload. Therefore, to propose an architecture combining the infrastructures of the Cloud, Fog, and Mist Computing the use of the lightweight and super-fast protocols [31] not only preferable but also necessary and especially the protocols based on the Publish-Subscribe communication models. In

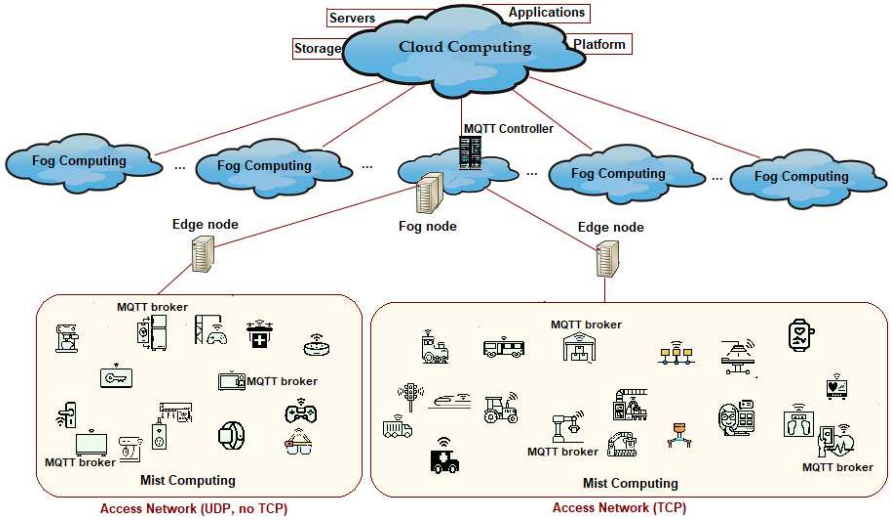

Fig. 3 Overview of Mist Computing Architecture

Figure 3 describe our proposed distributed architecture for real-time IoT network based on Cloud, Fog and Mist Computing.

Based on the reference architecture of the Cloud, Fog, and Mist network, we suggest a distributed architecture to provide real-time communication in time-sensitive IoT applications. Our proposed architecture provides TCP and datagram connections like UDP, sequentially we suggest the use of MQTT and MQTT-SN messaging protocols. Every single access network (TCP or UDP) is controlled by a single Edge node. The latter plays the role of an MQTT broker, MQTT-SN gateway in access network based on UDP, or a normal Gateway in access network based on TCP. Every single Mist in the access network is controlled by a multiple Mist node. Each Mist node acts as an MQTT-SN broker for neighboring IoT devices. It can receive requests from multiple IoT devices of the same Mist . The offered architecture consists of five components: Cloud server, Fog servers, Edge nodes, Mist nodes, and IoT devices. Next, we describe each component independently of the others.

– IoT devices: they are devices equipped with actuators and sensors to collect data from the environment where they are deployed. Any IoT device may be a Mist node. The Mist nodes are the heart of architecture based on Mist Computing and they have located at the level of the device. They are complex physical elements in other words they are the most powerful IoT devices of the device layer i.e cameras, 3D scanners. Complex physical elements are equipped with microcontrollers that can perform certain communication and data management tasks, it is also necessary to note that their processing capabilities are limited [31]. In access networks based on the TCP transport protocol, we have associated each Mist node with an MQTT broker, in this case, each single Mist node is an intermediary in the interactions between IoT devices. Now in the access networks

based on the UDP transport protocol we have associated a Mist node with both an MQTT broker and an MQTT-SN gateway.
- Edge node: They are used to connect several networks i.e connect local networks and the Internet network. In our case, they connect IoT devices to the Fog servers. Also, They provide communication services and data management. Therefore, we have associated its nodes with the MQTT servers.
- Fog Node: is the heart of architecture based on Fog Computing and they have located between the Cloud and IoT devices. They may be physical elements i.e computer, gateway, or virtual i.e virtual machine. The main services provided by Fog nodes are data management and communication. For the communication service, each Fog node has been associated with one or several MQTT brokers. These MQTT brokers are intermediaries between IoT devices and users.
- MQTT controller : is a physical or virtual element that is responsible for first deciding which Cloud Fog or Mist Computing infrastructure is the best to provide Computing resources and communication service based on latency requirements that are defied by MQTT devices. Secondly to control the network and reconfigure it according to the criteria of CPU, energy, and latency.
- Cloud server, in this architecture, is used to group companies that have common service interests and implement an MQTT server to gather common services from Fog.

5 Proposed MQTT-MBD Approach

Despite the advantages of the MQTT messaging protocol it still unsuitable for time-sensitive applications since it does not provide real-time communication between IoT devices. And one of the solutions is to get close as possible the servers to the IoT devices. For that, Mist Computing was used. To enjoy the benefits of the MQTT protocol and Mist Computing model, it's evident that the adaptation of the MQTT protocol and Mist Computing is one of the solutions to ensure real-time communication. Such adaptation consists to creates an MQTT middle-ware based on Mist Computing to ensure that time-sensitive applications meet their deadlines. In this regard, we suggest a new approach for time-sensitive systems. The main objective of the proposed approach consists of distributing MQTT brokers in IoT architectures based on Mist Computing. To identify our approach we were set a name for it. The proposed approach is called as MQTT-MBD. The acronym MQTT-MBD is for MQTT-Broker Distribution based on Mist Computing.

In the rest of this section, we detail and present the principle and the operation process of the MQTT-MBD proposed approach.

5.1 MQTT-MBD Principle

In IoT communications based on MQTT protocol all IoT devices (publisher and subscribers) exchange messages through an MQTT broker. All publishers sent captured data to the MQTT broker. Then, the MQTT broker transfers the analyzed results to

the desired subscribers. To recapitulate, published data are routed from the publishers to subscribers by MQTT brokers. In traditional IoT Cloud architecture, MQTT brokers are located on the network and centralized to Cloud nodes. Here, MQTT brokers serve IoT devices from different locations. Such a strategy raises the delay of communication due to several issues. Some of the issues associated with the increase of the communication delay are:

- MQTT broker failures cripple the time-sensitive applications and prevent communication between IoT devices. This makes sense cuz there is no other MQTT broker to process published data and subscription requests.
- The growth of the IoT devices number increases the processing delay resulting inappropriate deadlines of the time-sensitive applications.
- Since publisher and subscribers with the same topic bellow to the same access network, sending data to the Cloud for analysis and returning results to IoT devices is an unnecessary delay. This increases the latency and affects negatively the performance of real-time applications.

Even though, recent research suggests distributing multiple brokers to Fog and Edge nodes to improve the problems of centralizing MQTT brokers, this type of distribution does not ensure that IoT time-sensitive applications meet their deadlines.

Hence, the goal of our approach is to allow real-time communications by distributing MQTT brokers to Mist nodes. The principle of the MQTT-MBD is described as follows. First, we propose to divide treatment over multiple MQTT brokers geography distributed on Mist nodes. Second, we suggest getting close the task of data analysis as possible to IoT devices to respect deadlines of real-time applications.

Since multiple MQTT brokers are distributed in IoT networks based on Mist Computing a mechanism to allocate the appropriate MQTT broker that provides real-time communication between IoT devices that produce data and IoT devices that consume data is needed.

Next, we describe the MQTT-MBD process enabling MQTT brokers allocation but above all, we present the operation process to connect MQTT clients with a suitable broker.

5.2 Operation Process

In this work, we aim to improve the quality of communication between MQTT publishers and subscribers for time-sensitive applications. The improvement here is to minimize the communication delay and ensure that the time-sensitive applications meet their deadlines. In the context of the Mist environment, many MQTT brokers are hosted in Cloud, Fog, Edge, and Mist nodes to ensure the messages exchange process between the MQTT publishers and subscribers. In some case, publishers and subscribers need to be connected to the appropriate MQTT broker that improves the delay of communication and ensure deadlines. Thereby we set an assembly of the process enabling to connect the MQTT clients to the appropriate MQTT brokers with the attention of allowing real-time message exchange. As shown in Figure 4, this assembly is made of four processes. All these processes are initiated and controlled by

the MQTT management controller located in appear network level at the Fog layer (as defined in figure 3).

In the registration process, publishers and subscribers are registered to obtain the IP address of the appropriate MQTT brokers by providing their IP address, topics of interest, and the deadlines values to the MQTT controller. To register, the publishers and subscribers send PINGREQ (ping request [1]) packet to the MQTT controller containing the needed information. Next, the MQTT controller analyses the received PINGREQ packet to arrange for the publisher and subscribers to have the same topic in one group. Once the registration is done the publisher and subscribers keep waiting until the MQTT controller sends a PINGRES (ping response [1]) messaging containing the IP address of the MQTT broker.

The second process is known as MQTT brokers selection. It aims to allocate the appropriate MQTT broker to each group of publisher and subscribers. This process is based on 3 criteria for the MQTT brokers selection. These criteria are:

- The MQTT broker should verify the real-time constraint.
- The MQTT broker should have enough energy to perform his operation
- The MQTT broker should have sufficient Computing capabilities (CPU) to perform and sustain the required computations.

After the allocation of the appropriate MQTT broker, we go to the assignment process. The assignment is the process that informs the MQTT publishers and subscribers of the suitable MQTT broker through a third party which is the Edge nodes. To do that, the MQTT controller sends the IP address of every selected MQTT broker to an adequate group of MQTT clients. For every group, the MQTT controller sends a PINGRES message containing the IP address of the suitable broker to the Edge nodes. The Edge nodes analyze the received message to retrieve the IP address of the destination and so on of the broker, then it broadcast the broker's address to their destination in PINGRES message.

Now we detail the Connection process. After the IP address of the MQTT broker reaches the registered nodes, the connection between the subscriber or publisher and the target MQTT selected broker is established. This step aims to open the connection by the MQTT broker with publishers and subscribers. Here the typical scenario of connection is applied, publishers and subscribers sent a CONNACT message to the broker and the broker responds by CONNACK message.

6 MQTT Broker Selection

To recall, we suggested distributing the MQTT brokers over different Mist nodes to satisfy the deadlines constraints of some time-sensitive applications. In the suggested approach Cloud, Fog, Edge, and Mist Computing complement and cooperate to satisfy the IoT applications' needs. Consequently, a large number of MQTT brokers are available in different locations. These locations are Cloud, Fog, Edge, and Mist nodes. The challenge here is to select the MQTT brokers that satisfy the deadline constraint of real-time applications and by the way select the shortlist path to the broker to reduce energy consumption.Therefore, we need a process to select suitable MQTT

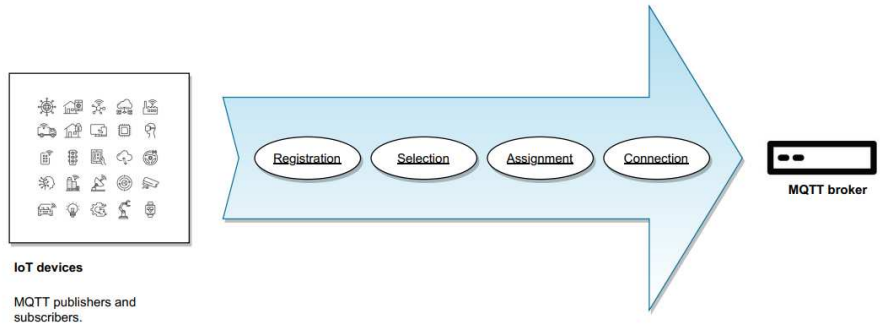

Fig. 4 Operation process to connect the MQTT publishers and subscribers to the appropriates MQTT brokers.

brokers for the groups of MQTT clients. The selection process picks and allocates the appropriate MQTT broker for a group of MQTT clients. MQTT controller uses this module to speedily lock up for an IP address of the most potential MQTT broker for a group of MQTT clients.

In the rest of this section, we detail the process of MQTT brokers selection. Next, we present an algorithm for the Mist broker selection.

6.1 Process Description

The MQTT broker for a group of MQTT clients may be one of the MQTT brokers hosted in Cloud, Fog, Edge, and Mist nodes. And built on the deadline constraint, we can decide such or such MQTT broker is suitable for a group of MQTT clients. Based on two hypotheses, the MQTT controller follows five steps to perform the MQTT broker selection process. The first hypothesis that all MQTT clients of the same group belong to the same access network. The second one is that Cloud, Edge, and Fog nodes are powerful nodes that have enough energy and computational capabilities to process any coming task.

Figure 5 describes the different steps of the MQTT broker selection process in sequential order. The figure clearly shows that the MQTT broker selection process follows five steps. The steps of the MQTT broker selection are:

Step 1. For every group of MQTT client the MQTT controller have to verify whether the real-time constraint is satisfied by the Cloud brokers, if yes the selection process for this group ends otherwise prove whether Fog brokers ensure that time-sensitive application meets their deadlines.

Step 2. For all IoT devices of the same group MQTT controllers have to verify whether the real-time constraint is satisfied by the Fog brokers, if yes the MQTT controller ends the selection process for this group otherwise prove whether Edge brokers meet the real-time restriction.

Step 3. Now, the MQTT controller verifies for all groups not associated with a broker if the Edge brokers satisfy the deadline constraint, if yes the selection process for this group ends otherwise lockup for a Mist broker that meets the real-time restriction.

Step 4. After making sure that neither Cloud, Fog nor Edge brokers answer to real-time restriction it has become necessary to build a list of eligible brokers within the same Network. To select the most potential Mist broker which answers to real-time restriction for all MQTT clients of the same group, the MQTT controller creates an eligible broker list by selecting Mist brokers having enough Energy and CPU.

Step 5. Once, MQTT controller accomplishes the fourth step, it's time to select the Mist broker which proves real-time restriction from the Mist eligible brokers list.

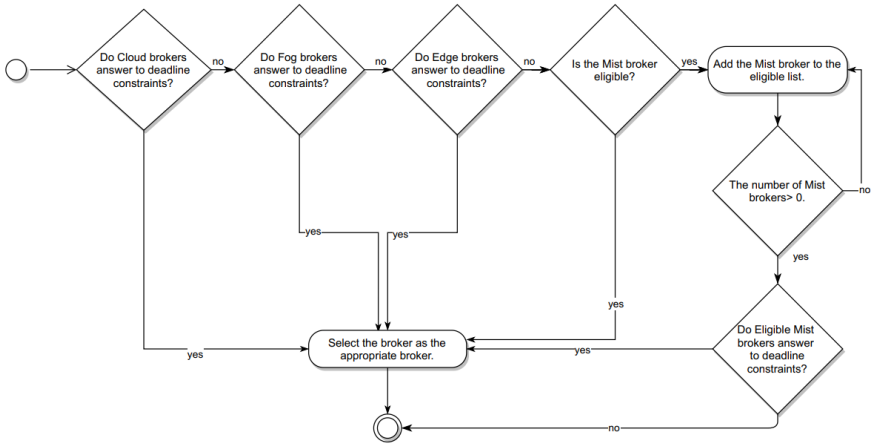

Fig. 5 Flowchart of the MQTT broker selection process for all IoT devices of the same group.

6.2 MQTT Mist Broker Selection

Fog, Edge, and Mist Computing are closer to IoT devices than Cloud Computing. And Mist Computing is the most closer one since it provides the lowest delay of communication. But it suffers from limited resources, and it has fewer resources than Fog, Edge, and Cloud. Hence in the Mist environment, the MQTT controller selects the Mist node where the MQTT broker is hosted, having the highest resources and satisfying the deadline constraint for a group of MQTT clients. To do that, the MQTT controller verifies the feasibility of three conditions to select the MQTT broker. We present these conditions from the more to the least priority as follows:

1. Condition 1 : The energy consumed per broker (E_j) must be smaller or equal to the energy threshold (E_{th}), ($E_j \leqslant E_{th}$).

2. Condition 2 : CPU utilization needed per broker (ρ_{b_h}) must be smaller than the CPU utilization needed per broker's node (γ), ($\rho_{b_h} < \gamma$).
3. Condition 3 : In order to meet the required data delivery time restriction, the sensed data must be delivered to the subscriber ($R_{R_r \to b_h}$) before minimum deadline constraint of the MQTT subscribers of the same group d_{s_i}. Otherwise, the process no longer serves any purpose ($R_{R_r \to b_h} \leqslant Min(d_{s_1}, ..., d_{s_{|S_r|}})$).

Where :

- b_h denotes the broker selected.
- R_r represents the a category of all IoT devices with the same topic r.
- E_j is the energy consumed per broker on node j.
- E_{th} is the maximum energy threshold per broker.
- ρ_{b_k} denotes the CPU rate needed per broker.
- γ denotes CPU rate needed per node.
- $R_{R_r \to b_h}$ represents the end-to-end response time between R_r and b_h.
- d_{s_i} is the threshold of the recommended communication delay between $p_r \in R_r$ and $s_i \in S_r$.
- $S_r = \{s_1, s_2, ...s_n\}$ is the set of IoT devices subscribed to topic r, where $|S_r|$ is the total number of subscriber with the same topic r.
- p_r denotes the publisher to topic r.

Thence, The step of selecting a Mist broker for all devices with the same topic must go through two steps, examine the eligibility of Mist brokers to be processed and choose the eligible Mist brokers simultaneously by examining real-time constraints. In the beginning, as shown in Algorithm 1 the MQTT controller starts by proving the resources of the broker. Here, the main goal is to build an eligible broker list from the available broker set. So, it is necessary to check the eligibility of all Mist brokers existing in the same Network. Thus, we have set the first two conditions discussed above. The Mist broker must be added to the eligible list only if he meets the first two conditions. Once, we have built the set of eligible Mist brokers it is time to choose the selected broker of all IoT devices with the same topic. Any selected Mist broker must satisfy the real-time constraints.

For the algorithm Mist Broker Selection (MBS), we define these symbols :

- B is the set of the available Mist brokers in the same network, and $|B|$ is the number of available Mist brokers.
- B_e represents all eligible Mist broker set, and $|B_e|$ is the number of eligible Mist brokers.
- R is the list of all IoT devices category.
- b_h denotes the broker selected.

Algorithm 1 Mist Broker Selection

Input B, R
Output b_h

```
/* Build the eligible broker (B_e) set from B                                  */
B_e := {}
for k = 1 to B do
    b_k := Current Broker (B)
    if Condition 1 then
        if Condition 2 then
            B_e := B_e ∪ b_k

/* Select the appropriate MQTT broker from B_e according Feasibility Condition 3
   */
for r = 1 to R do
    R_r := Topic r's Group (R)
    p_r := Current r's Topic Publisher (R_r)
    S_r := Current r's Topic Subscribers (R_r)
    k := 1
    while R_{R_r→b_k} > MIN(d_{s_1}, ..., d_{s_{|S_r|}}) do
        b_k := Current Eligible Broker (B_e)

        /* Calculate the End-to-End response time                              */
        R_{b_k} := Apply Equation 14 or 20
        R_{b_k→p_r} :=Apply Equation 3
        for i = 1 to S_r do
            R_{b_k→s_i} := Apply Formula 4
            R_{p_r→s_i} := R_{b_k→s_i} + R_{b_k→p_r} + R_{b_k}
        R_{R_r→b_k} := MAX(R_{p_r→s_1}, R_{p_r→s_2}, ..., R_{p_r→s_n})
        k := k + 1
    b_h := b_k
    SendToEdgeNode(b_h, R_r)
```

6.3 Broker Failures

MQTT brokers play a crucial role in real-time IoT systems based on publisher-subscriber communication models. Any failure of these entities affects the network negatively. Therefore, it is important to study and solve the malfunction and change of any MQTT broker.

This phase consists of reconfiguring the network according to the states of the MQTT brokers. A key step in reconfiguring the network is to migrate MQTT publishers and subscribers from one broker to another. Considering the three main purposes that case the network re-configuration, namely:

- MQTT clients can lose the connection with a broker due to several issues. Hence we need an alternative plan. This plan is to migrate the MQTT clients of the same group to a second MQTT broker satisfying the real-time constraint.
- The second purpose that may lead us to reconfigure the network is the lack of energy of the node where hosted the selected broker. This lack refers to the fact that the energy needed to perform the MQTT broker tasks became much bigger than the threshold ($E_j > E_{th}$). Because of this lack, we propose to migrate the MQTT clients to another MQTT broker that satisfies the deadline constraint.

– The lessness of resources of the MQTT broker increases the processing delay of the tasks. Increasing this delay enlarges the communication delay between the MQTT publisher and subscribers. In such a case, the end-to-end response time between b_k and R_r became much bigger than the minimal deadline of the subscribers set ($R_{R_r,b_k} > Min(d_{s_1},...,d_{s_{|S_r|}})$). How to solve this issue is a challenge. Here, plan B is to migrate all the IoT devices of the same group to another MQTT broker that has the capabilities to follow his missions.

As the CPU is one of the brokers' computational resources, we need to control it. Once the CPU rate of the selected broker is much bigger than the CPU rate the node needs ($\rho_{b_k} > \gamma$), we suggest applying the workload balancing across multiple MQTT brokers. Workload balancing is a technical word. Workload balancing is the process that divides the tasks on different nodes to ensure the effectiveness of the systems.

Periodically, Fog, Edge, and Mist nodes send the current state of its energy and CPU. Based on its values, the MQTT controller resets the conditions of feasibility. In case the non-feasibility test is checked the MQTT controller re-select an MQTT broker based on the restriction requirement. If the MQTT controller does not receive the status of the broker in its desired period, it sends it a ping message to test its connectivity after a timeout if no response is received the MQTT controller restarts the selected broker selection step.

7 End-to-End Delay Analysis

As we are focusing on real-time IoT applications, the response time analysis is needed to verify the timing constrains especially deadlines.

Related to the MQTT communication model based on publisher and subscriber, we define the end-to-end delay or response time (noted: $R^{b_k}_{p_r \to s_i}$) as the elapsed delay from a publisher (p_r) to a subscriber (s_i) passing through a broker (b_k). This delay is computed as the sum of delays of publishing a message to the broker (noted: $R_{b_k \to p_r}$), the broker treatment (R_{bk})and the subscriber receiving the message. Thus, the following equation of the end-to-end response time:

$$R^{b_k}_{p_r \to s_i} = R_{b_k \to p_r} + R_{b_k \to s_i} + R_{b_k} \tag{1}$$

Where:

1. $R_{b_k \to p_r}$ is the delay induced by the transmission network between b_k and p_r.
2. $R_{b_k \to s_i}$ is the delay induced by the transmission network between b_k and s_i.
3. R_{b_k} is the average time within a broker.

In practical cases, we not only working with one subscriber and publisher for a message topic in the broker. We assume a set of publishers (noted: P), a set of subscribers (noted: S) related to each group of nodes that work on the same message topic. The global end-to-end response time will be associated with all these possible links between publisher nodes and subscriber nodes. The response time will change

according to the location of these nodes. We can compute the worst-case end-to-end response time of nodes related to a topic and a broker:

$$R_{R_r \to b_k} = Max_{p_i \in P}(R^{b_k}_{p_i \to b_k}) + R_{b_k} + Max_{s_i \in S}(R^{b_k}_{b_k \to s_i}) \quad (2)$$

In the following sections, we will develop the details of computing the response times from publisher to broker and from broker to subscriber. They depend on the communication delay according to MQTT protocol and multi-hop communication.

7.1 Communication Delay

The architecture of the proposed MQTT-MBD approach well handles TCP/IP and UDP/IP wireless access networks to connect things. And to ensure the reliability of message transmission both MQTT and MQTT-SN support three levels of QoS. Hence, the transmission delay between publisher client, broker server, and subscriber client depends on both the QoS and the messaging protocol. Thus, an MQTT client must be connected to the broker before the transmission of data, therefore we must add the TCP connections to the broker [24] within the communication delay. For example for QoS=0, the communication delay between p_i and b_k is given by equation 3.

$$R_{p_i \to b_k} = R_{p_i \to b_k}(CONNECT) + R_{p_i \to b_k}(PUB) \quad (3)$$

And the transmission delay between s_i and b_k in TCP/IP access network is given by formula 4.

$$R_{b_k \to s_i} = R_{s_i \to b_k}(CONNECT) + R_{s_i \to b_k}(SUB) + R_{b_k \to s_i}(PUB) \quad (4)$$

To be able to determine the communication delay, we need to compute the delays spend at each link from a source node to a destination node. As we are using multi-hop networks, intermediate nodes will have the role of routing messages from the publisher to the broker and then from the broker to the subscriber. The next sections will deal with the one hop delay to then determine the multi-hop delay.

7.1.1 One-hop Response Time Analysis

In this section, we will determine the response time related to one direct link from a node to its neighbor node. This depends on the medium access sub-layer (MAC: Medium Access Control) that gives the manner to get access to the network and then to directly send data.

In IoT access networks, the most used wireless networks are Wireless Sensor Networks (WSN), Personal Area Networks (PAN) as Zigbee, Z-Wave, and also Wi-Fi 6 for Low-Power IoT. All these networks are based on contention access that uses the CSMA/CA as a MAC protocol to ensure transmission with minimal interference and collisions. To gain access for transmission, the CSMA/CA uses a contention window of size W, where every single node can choose a random instance from W to transmit data. The use of CSMA/CA reduces collisions or interference by minimizing the probability of having another concurrent access. In the case of a collision

caused by simultaneous transmissions, there will apply a back-off of random choice before access to re-transmit data, this operation prevents having the collision for another time. The collision probability is related to the waiting contention window of the back-off (W) and the number of neighboring nodes (N) in the same transmission range (in which these nodes can be in interference in their transmissions). Referring to research studies [19,20,22] the probability that a node transmits into a random slot in the contention window is $1/W$.Thus, the probability that a node correctly transmits data without interference with another is $(1-\frac{1}{N})^{N-1}$. Therefore, the collision probability is given by formula 5.

$$P_{collision} = 1-(1-\frac{1}{W})^{N-1} \tag{5}$$

In addition to collisions, there is the problem of transmission errors that are common in wireless networks. The probability of error on the transmission of a packet is given by the equation 6 where P_{size} indicates the packet size and BER is for Bit Error Rate that may be around 10^{-4} [20].

$$P_{error} = BER * P_{size} \tag{6}$$

Considering the two main purposes that cause the non-delivery of a packet, namely errors due to collisions and on transmissions. The probability of non-delivery of a package β is determined by the equality 7.

$$\beta = P_{collision} + P_{error} \tag{7}$$

On every single transmission, a positive acknowledgment message is sent. In case of no positive acknowledgment message is received, nodes will re-transmit the packet, as a result, the delay of transmission packet increases. Based on the research paper [21], the necessary average number of re-transmission until reaching the delivery of the packet is specified by equation 8.

$$\widetilde{Nt} = \frac{\beta}{1-\beta} \tag{8}$$

Based on the number of re-transmission, we calculate the average of successful delivery transmission time as shown in equation 9.

$$\widetilde{C} = \widetilde{Nt} * T_{macAckWaitDuration} + T_{Trans} + T_{Ack} \tag{9}$$

Where $T_{macAckWaitDuration}$ is the maximum waiting time on acknowledgment, T_{Trans} is the packet transmission time which depends on the wireless network transmission rate and T_{Ack} is the acknowledgment time.

7.1.2 Multi-hop Response Time Analysis

Now, we will generalize the delay from the one hop to multi-hops by taking into account the path where the data packets go through intermediate nodes that forward these packets to reach the destination node.

To represent the transmission treatment of each node we used the M/M/1 queue. The main parameters of the used queue are the service rate μ_i and arrival rate λ_i. μ_i is given according to equation 9 on the average delivery time of a single-hop packet. λ_i is calculated in function of the packets generated and forwarded by this node.

Communication between a node and the broker goes through a path of intermediate nodes which will route the communications. A node i has a generated traffic with a rate of packets noted: g_i to be sent, but it can also forward the packets with a rate of forwarded packets noted: f_i of the others nodes [27] which have chosen it as an intermediate node in their communication paths noted $path(j)$. The values of the arrival rates and the service rates, in this case, are estimated as follow:

$$\lambda_i = g_i + f_i, \mu_i = 1/\widetilde{C_i} \tag{10}$$

Referring to the principle of estimating arrival rates of the paper [23], we can consider that the forwarded packet rate f_i passing through a node i is the sum of the packet rates generated g_j through the j nodes and passing through their routing through the node i.

$$f_i = \sum_{\forall j / i \in path(j)} g_j \tag{11}$$

In the case of the M/M/1 queuing model the waiting time in the node is given by $W_i = \frac{1}{\mu_i - \lambda_i}$ where:

$$W_i = \frac{1}{1/\widetilde{C_i} - (g_i + \sum_{\forall j / i \in path(j)} g_j)} \tag{12}$$

Finally, we conclude the multi-hop response time for communication along a path from a source (i) to a destination (k) as the sum of the waiting times in each intermediate node of the path as shown in equation 13.

$$R_{i \to k} = \sum_{j \in path(i \to k)} W_j \tag{13}$$

7.2 Delay within Broker Model

Throughout this section, we adopt this mathematical notation related to M/M/1 queue for the broker:

- q_{in} for inbound queue.
- $q_{s_i,out}$ for outbound queue.
- μ_{in} represents the service rate.
- λ_{in} represents the arrival rate.
- $\lambda_{p_i,r}$ is the rate of publication of message of topic r by publisher p_i.
- λ_{in} is arrival rate of published events or messages.

– μ_{in} denotes the service rate of the published events or messages.

For energy conception prevention, the IoT nodes use the sleeping mode to preserve their energy. Every node makes the plan of periods for sleeping as inactive period (noted off) and other periods to be active for transmission (noted on). In this case we use the on/off queue model [15, 16, 18, 17].

– T_{on} is the active period.
– T_{off} is the inactive period.
– $\lambda_{s_i,out}$ denotes the subscriber notification rate.
– $\lambda_{s_i,off}$ represents arrival rate of virtual event.
– λ_{s_i} represents arrival rate of normal event.

7.2.1 Broker Model in Default Delivery

According to the work process of the MQTT broker in default delivery mode, the queuing model can be used to analyze his performance as shown in figure 6. The waiting time in the MQTT broker (b^k) depends only on the access queue, the access queue is an inbound queue that queues the published events. Every inbound queue is an M/M/1 queue characterised by the two parameters : the service rate (μ_{in}) and the arrival rate (λ_{in}). As we consider multiple publishers of the same topic going to single broker, the arrival rate will be the sum of all arrival publishers ($\lambda_{p_1,r}, ..., \lambda_{p_i,r}$) as shown in the figure 6.

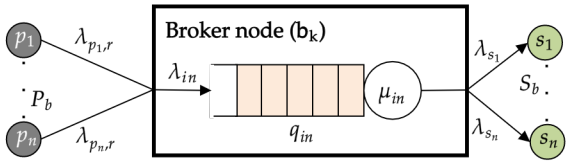

Fig. 6 The MQTT broker model in default delivery mode.

Hence, according to the response time of the M/M/1 queue, the time that an event remains in the broker before it is served to subscribers can be calculated according to the service rate and the arrival rate. Thus, the end-to-end response time in a specific MQTT broker can be calculated as follow:

$$R_{b_k} = \frac{1}{\mu_{in} - \lambda_{in}} \tag{14}$$

7.2.2 Broker Model in Persistent Session

In default mode, The MQTT broker drops the message in the queue whenever the publisher is disconnected. However, the persistent session model is made in the MQTT broker to maintain the message in the queue independently of the session connection

and disconnection. This mode is interesting in many IoT applications where publishers and subscribers are connected and disconnected in independent ways. This persistent mode is interesting in the case of energy preservation mode where we define periods for activity and other periods for inactivity or sleeping. Thus, we can retain messages in the broker queue independently of the alternation between disconnection in the sleep period and connection in active period.

Therefore, the process of sending and receiving data for a subscriber must be performed at the period of activation T and blocked at the period of sleep T_{off}. Frequently, this phenomenon is called the duty cycle. A duty cycle is the proportion of delay during which the sending and receiving process is on during each complete cycle. As shown in equation 15 giving T_{on} is the activity period and T_{on} is the sleep period then the cyclic ratio is the proportion of the active period over the total time of a cycle.

$$d_c = \frac{T_{on}}{T_{on} + T_{off}} \tag{15}$$

Building on recent works [15,16,18,17], we model the performance within an MQTT broker in persistent session mode basing on queue network theory. As shown in 7 each broker uses one inbound to receive and process published events and multiple outbound queues in parallel to retain and to transmit events to subscribers. Hence, two types of queues are used *1)* Queuing Centre *2)* ON/OFF queuing centre (As in figure 7). The On/OFF queuing is added to model the intermittent connection of nodes due to the sleeping mode of nodes especially subscribers.

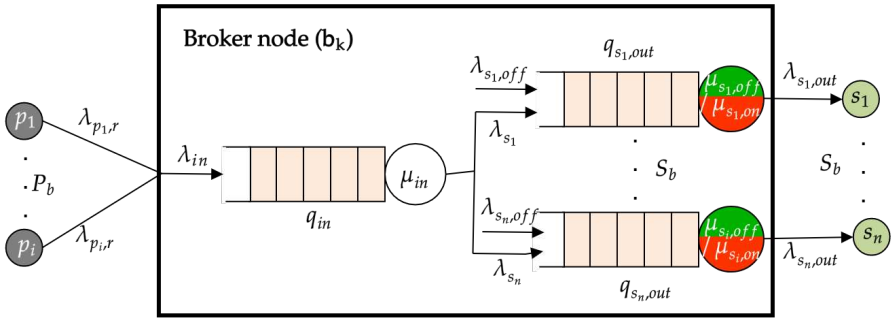

Fig. 7 The MQTT broker model in persistent delivery mode.

The receiving and processing service is maintained by one M/M/1 queue. Events line up in the queue in the order of their arrival then this queue center applies the first-come-first-served (FCFS) principle. Each publisher is based on Poisson Process for publishing events to a broker.

Each outbound queue depends on two periods of time subscriber's active period and inactive. Thus, it is necessary to calculate the parameters of this type of queue according to active and inactive periods. To evaluate the ON/OFF Queue Centre. We introduce two event classes normal events and virtual off events that follow the active/inactive timing of the subscriber. During an inactive period, only one case is

possible, normal events can arrive without one leaving. And during the active period, only two cases are possible, the first case is that a normal event can leave with a normal event can arrive, the second case is that a virtual off event arrives at the end of T_{on} and be processed along with T_{off}.

Thus, the arrival rate of the off event $\lambda_{s_i,off}$ is specified as [17]:

$$\lambda_{s_i,off} = \frac{1}{T_{on} + T_{off}} \tag{16}$$

For a given subscriber the average response time within a broker can be computed as follows:

$$R_{b_k} = R_{in} + R_{s_i,out} \tag{17}$$

Where R_{in} the response time according to the queue centre and $R_{s_i,out}$,out is the response time of the on/off queue.

Building on the standard solutions and according to the response time in the $M/M/1$ queues, the average delay for s_i into q_{in} is given by:

$$R_{in} = \frac{1}{\mu_{in} - \lambda_{in}} \tag{18}$$

Referring to the research study [17], the principle of delay calculation for a given s_i within the $q_{s_i,out}$ server center of the broker b_k is defined as follow:

$$R_{s_i,out} = \frac{\frac{T^2_{s_i,off}}{T_{s_i,on}+T_{s_i,off}} + \frac{T_{s_i,on}+T_{s_i,off}}{T_{s_i,on}}}{1 - \lambda_{s_i} D_{s_i,on} \frac{T_{s_i,on}+T_{s_i,off}}{T_{s_i,on}}} \tag{19}$$

Consequently, form on equations 18 and 19 we use equation 17 to derive the global response time of message to be delivered to a given subscriber s_i using the broker b_k is:

$$R_{b_k} = \frac{1}{\mu_{in} - \lambda_{in}} + \frac{\frac{T^2_{s_i,off}}{T_{s_i,on}+T_{s_i,off}} + \frac{T_{s_i,on}+T_{s_i,off}}{T_{s_i,on}}}{1 - \lambda_{s_i} D_{s_i,on} \frac{T_{s_i,on}+T_{s_i,off}}{T_{s_i,on}}} \tag{20}$$

8 Resources Models

The resource constraints are selective criteria for the broker. Fewer resources will lead to change the broker node for the one that has enough resources to execute correctly the MQTT messaging services. In the following sections, the focus will be on the performance parameters: CPU and Energy.

8.1 CPU Model

Each node has a CPU that needs to perform its tasks. In this regard, we consider utilization of the CPU noted γ to every single node to meet its internal needs. A broker b_k receives several requests to register for message topics and several requests for publications. So, each broker is characterized by its CPU utilization rate noted ρ_{bk}, provided that the assigned tasks must not exceed the CPU resource of the node to which is integrated. Also, it should be noted that the total CPU utilization rate of a single node where a broker is integrated is less or equal to 1 ($CPU \leqslant 1$). The CPU monitoring rule of any broker is to compare his CPU utilization rate to a static threshold where the static threshold is the maximum node CPU utilization rate. Therefore, we can write the correlation between ρ_{bk} :broker's CPU utilization rate and $1-\frac{\gamma}{100}$: static threshold by the formula 21 ,on condition that $0 \leqslant \rho_{b_k} < 1$. If ρ_{b_k} is greater than $1-\frac{\gamma}{100}$, it means the processor is overloaded and the broker can't receive another request. In this case, we propose to balance the workload between several MQTT brokers. Here, the workload balancing appoints the process of dividing a group of tasks of an MQTT broker on a set of brokers. The main objective of the workload balancing is to make the global treatment more efficient in terms of end-to-end response time (Latency). Thus, the threshold of the broker CPU utilisation ρ_{b_k} is given by the following formula:

$$\rho_{b_k} < 1 - \frac{\gamma}{100} \tag{21}$$

As we have used M/M/1 queuing model for the broker services, we can deduce the broker utilization according to that model to then verify it according to the equation 21 as a CPU contain to be satisfied. The server utilization according to M/M/1 for the broker b_k represents can be calculated as :

$$\rho_{b_k} = \frac{\lambda_{in}}{\mu_{in}} \tag{22}$$

In the case of MQTT persistent mode and the related model, we have used two types of queues with brokers. Referring to Figure 23 , the broker CPU utilisation will refers to the two queues as following :

$$\rho_{b_k} = \frac{\lambda_{in}}{\mu_{in}} + \sum_{i=1}^{n} \frac{\lambda_{s_i}}{\mu_{s_i,on}} \tag{23}$$

8.2 Energy Consumption Model

An MQTT broker can be installed in an IoT device with energy constraints which is powered by a battery. When the energy of the device is insufficient then the broker will be unable to carry out his missions for his clients. Therefore, it is very important to know the energy consumed by the broker. Derived from the analytical energy consumption model discussed in articles [25,36,27] four main states must be taken :

- Reception: In this state, every single device is in its period of activity, it carries out the procedure of receiving data forwarded from other nodes in the form of packets.

- Transmission: Every single device carries out the procedure of transmitting forwarded and generated packets in the period of activity.
- Sleep: Every single device has an inactive period to conserve energy but this does not prevent the energy of the devices during these periods from degrading.
- Idle: In this situation, devices are in active period but they don't execute any procedure, they are just sniffing for the packets.

According to [19], to calculate the transmission energy consumption for a device i, we simply use:

- number of packets sent: To calculate it, we simply multiply the rate of packets forwarded (f_i) and generated (g_i) by the duration of the transmission period t. We assume that the number of packets sent is represented as follows:

$$N_p = t.(f_i + g_i) \tag{24}$$

- The energy expended for a single data packet and an acknowledgment packet, an acknowledgment packet is intended to validate the reception of the packets forwarded by other nodes. To calculate the energy expended for a single data packet, we simply multiply the transmission energy consumption per second (e_{tx}) and the data packet transmission delay ($TPkt$) as mathematically represented in formula 25. And to calculate the energy expended for a single acknowledgment packet, we simply multiply the transmission energy consumption per second (e_{tx}) and the acknowledgment packet transmission delay ($TAck$) that is formally introduced by 26.

$$E_p = e_{tx}.TPkt \tag{25}$$

$$E_{ack}e_{tx}.TAck \tag{26}$$

Hence, from 24, 25 and 26 we can define the energy consumption for a device i caused by transmission until t seconds as follows:

$$Etx = t.e_{tx}.(TPkt.(\widetilde{Nt}+1).(g_i+f_i)+TAck.f_i) \tag{27}$$

Likewise the energy consumption caused by transmission calculated in 27, we determine the energy consumption for a device i produced by reception until t seconds along these lines:

$$E_{rx} = t.e_{rx}.(T_{Pkt}.(\widetilde{Nt}+1).f_i + T_{Ack}.(g_i+f_i)) \tag{28}$$

Where e_{rx} is the energy consumption per second in reception state. An IoT device with no procedure to do may be in the sleep or idle state. To calculate the energy consumed by a device i in idle state simply multiply the duty cycle (d_c) by the energy consumption per second in idle state (e_{idle}) as represented in 29, where d_c represent the period of activation of the device i and it can be estimated as represented in formula 15.

$$E_{idle} = t.e_{idle}.d_c \tag{29}$$

As the same manner, we can calculate the energy consumed by a device i in sleep state during inactivity period which calculate as: $1 - DutyCycle_i$:

$$E_{sleep} = t.e_{idle}.(1-d_c) \tag{30}$$

Refer to equality 27, 28 and 29 the total energy consumption for a device i can be obtained in such a way:

$$E_i(t) = E_{rx_i}(t) + E_{tx_i}(t) + E_{sleep_i}(t) + E_{idle_i}(t) \quad (31)$$

By hypothesis, we considered that the MQTT brokers will be always active to maintain a continuous interaction with publishers and subscribers nodes. As a result, IoT devices selected to be brokers are implemented do not use sleep mode. For this, we can predict the energy consumed for the broker node by the following formula:

$$E_i(t) = E_{rx_i}(t) + E_{tx_i}(t) + E_{idle_i}(t) \quad (32)$$

9 Performance Evaluation

In the first place, to perform the performance of our approach we have implemented a simulation tool written in python 3.7 based on the discussed analytical performance models. We have used the Networx 2.5 package for the network topology construction and the matplotlib.pyplot package for the network visualization. In the second place, we have evaluated the proposed approach. The evaluation criteria are relayed to the main objective of the proposed approach. Our approach aims to select the potential MQTT home broker where the selected broker ensures the delivery of data within delay constraints. It is attached to the different metrics considered to evaluate the efficiency of the proposed approach: End-to-End response time and Energy.

9.1 End-to-end response time Evaluation

To assess the potential of the proposed approach in terms of end-to-end response time, our proposed approach MQTT-MBD and the presented related works approaches and the Extended-EMMA approach based on Mist Computing were simulated. The related works are: MQTT-RD, DM-MQTT, and EMMA build an MQTT broker on the Edge node. And for the Extended-EMMA, we proposed to apply the principle of the delay optimization suggested by the authors of EMMA on Mist nodes. This principle consists of providing the most closer broker to the subscribers or the publisher independently from each other, here an additional communication delay arises due to the topic bridging between brokers.

9.1.1 Simulation Environment

The simulation was conducted in a multi-hop wireless sensors network having a mesh topology with 201 nodes. In the basic simulation scenario, one node was deployed as publisher node, 3 nodes were deployed as MQTT broker nodes where Edge broker and two Mist brokers, 3 nodes are deployed as subscriber nodes while all the rest nodes were router, gateway, etc. Every single subscriber node remains in the ON and OFF states for exponentially distributed periods $T_{on} = 2s$, $T_{off} = 10s$. Each node in the network was connected to another node in the transmission range of 100 meters.

To enable the comparison of our proposed and related works approaches based on Edge Computing and the Extended-EMMA approach based on Mist Computing, we were observed different scenarios and calculated the metric end-to-end response time based on the analytical performances models. The main observed scenarios are :

- Increases the OFF states of the subscribers by increment the t_{off} period. The increment per t_{off} is $10s$.
- Increases the number of brokers deployed in the network. The increment per iteration is 1 brokers.
- Increases the number of nodes set up in the network. The increment per iteration is 50 nodes.
- Increases the range of connection between nodes. The increment per iteration is 10 meters.
- Increases the CPU utilization rate. To increase CPU utilization rate, we increased the arrival rate on the broker (λ_{in}).

9.1.2 Simulation Results

First, we varied the metrics associated with the network to compare the proposed MQTT-MBD approach to the related works and Extended-EMMA. The goal here is to prove the effectiveness of MQTT-MBD and learn the metrics that improve the proposed solution.

The curves 8 demonstrate the calculation of the simulation scenario where varying the number of the brokers. For our approach and Extended-EMMA approach, it is detected that more the number of brokers increases more the maximum end-to-end response time decrease, this is because the MQTT broker comes near to the subscribers and publishers. While the related works such as DM-MQTT have an end-to-end response time stable because the broker for this approach is static. Judged against the works of literature, MQTT-MBD is more efficient since more Mist brokers are distributed closer to the publishers and subscribers. Hence the tasks are processed much closer to MQTT clients. As a result, the transmission delay becomes negligible against the delay within an MQTT broker. And judged against Extended-EMMA, we conclude that the communication delay of the Extended-EMMA approach gets close to the communication delay of MQTT-MBD. Finally, we conclude that the number of Mist brokers is an important factor for the effectiveness of MQTT-MBD, as the number of brokers increases the delay of communication decrease.

The carving 9 shows the results of the simulation scenario where varying the period of inactivation of the subscribers T_{off}. We noticed that more the T_{off} increases the PUBLISH message needs much time to achieve the subscriber. Our proposed approach improves the Extended-EMMA approach, whatever the T_{off} value the end-to-end delay of the literature approach is greater than ours. The increase of the delay of inactivation of the subscribers increases the communication delay. Here the communication delay expands because of the time within an MQTT broker increase. From this simulation, we summarized that this metric does not affect the efficacity of the MQTT-MBD, nor the related works and Extended-EMMA since it has the same impact overall the approach.

As shown in the figure 10, the increase in the range of connections has a positive effect on the delay of communication between MQTT clients for all the implemented approaches. This comes down to the reduction of the length of the path between publishers and subscribers. This reduction is justified by the fact that more paths between MQTT clients occur and the probability of having a short path is high. And the curves prove these conclusions. Despite that the MQTT broker and his location are static for the literary works, we conclude that the metric of the connectivity range helps the Edge-based approach to satisfy the real-time communication because we notice that the more the connectivity range increase the communication path is shorter and the communication delay decrease consequently. Also, we notice the same conclusions for MQTT-MBD and the Extended-EMMA approaches. But what about the effectiveness of the proposed MQTT-MBD in this scenario. The MQTT-MBD is the most effective for real-time communication since it granted less communication delay.

The figure 11 illustrates the same phenomenon as the figure 10 since enlarging the number of devices may increase the number of the shorted path. New devices create new links, consequently, a new link between the MQTT clients and the MQTT may occur. By analyzing the values of the curves of this figure we conclude that for all the implemented approaches the movement from the number of devices from 250 to 300 has a positive effect since the end-to-end delay decrease and this is justified by the fact new shorter links are created, but when the number of devices is higher than 350 we observe that the delay of communication fixed because the positions of the new devices do not enhance the length of the paths. This scenario shows that the proposed MQTT-MBD is the most suitable approach for the time-sensitive application since it provides the lowest delay of communication.

Now we varied the metrics related to MQTT brokers to show the effectiveness of the proposed MQTT-MBD. The goal here is to prove that our approach solves the problem of the resource lessness of the MQTT brokers. The figure 12 gathers in the same graph, the CPU-based end-to-end response time curves of the related works, Extended-EMMA, and the proposed approach. In this case, we show the effectiveness of our approach without applying the strategy of workload balancing. Here, our approach shows more efficient results than other approaches since it provides the least delay of communication. The increase of the utilization rate of the CPU harms the delay of communication, we observe from the curves that the delay of communication tends to infinity when the MQTT broker reaches his total Computing capabilities (utilization rate of the CPU equals 1). To solve this issue we suggest optimizing the delay of communication by dividing the task into multiple MQTT brokers to satisfy the resource constraints and reduce the communication delay consequentially.

To show the effectiveness of this strategy we proposed that for a group of MQTT clients the selected MQTT brokers need a utilization rate of CPU three times higher than his capabilities. As the figure 13 shows every time the delay of communication of the MQTT group comes to 12 s it decreases immediately. This is because we distribute the task in different brokers and share the overloaded resources between brokers to reduce the load on brokers for better response time. We conclude that every time the broker reaches a threshold of end-to-end delay the broker response time increase immediately so that we ensure that the MQTT clients meet their timing con-

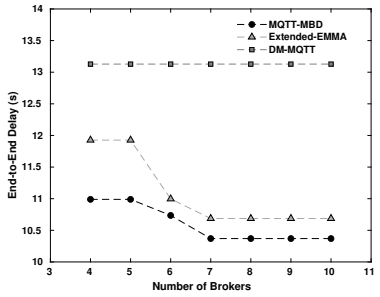

Fig. 8 End-to-End Response Time between Publisher and Subscribers by Varying the Number of Brokers.

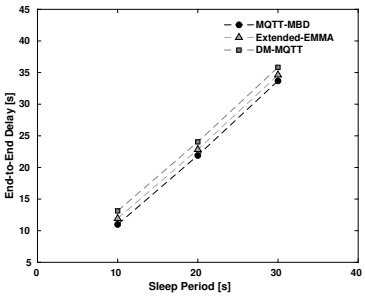

Fig. 9 End-to-end delay between publisher and subscribers by Varying the T_{off} Period

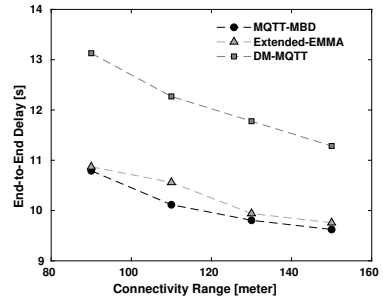

Fig. 10 End-to-End Response Time between Publisher and Subscribers by Varying the Connectivity Range.

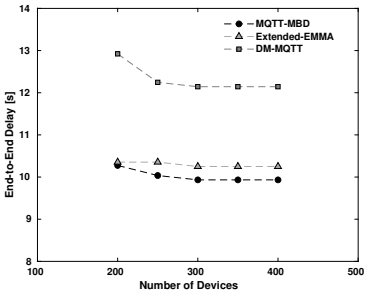

Fig. 11 End-to-end delay between Publisher and Subscribers by Varying the Number of Devices.

straints as deadlines. Since the related works and Extended-EMMA do not provide such a strategy. Our approach is the most effective for real-time applications.

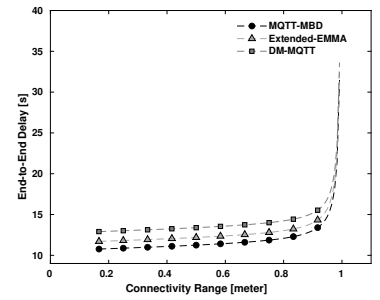

Fig. 12 End-to-end delay between publisher and subscriber by varying the CPU utilisation rate.

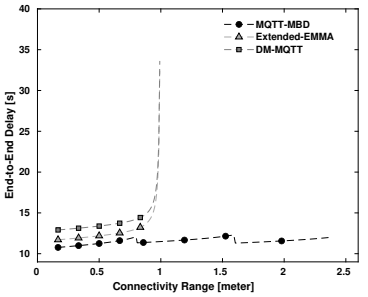

Fig. 13 End-to-end delay between publisher and subscriber by applying the strategy of workload balancing for MQTT-MBD.

9.2 Energy Analysis

9.2.1 Simulation Environment

To study the performance of the energy of our approach, we propose to simulate a multi-hop wireless sensors network having tree topology to control the number of jumps between MQTT clients and the broker. In this respect, a depth tree where the level number equals 5 has been simulated and each node of the tree has an average of 3 children with all leaves are located at the same depth. To calculate the energy consumption of devices we used the parameter values of MiCAz motes (IoT sensor node) summarized by [19]. Below table 2 which provides the values of energy conception that will be used.

Table 2 CC2420, MICAz motes parameters [19]

Parameters	Value	Description
Transmission data rate	250 Kbps	
Transmission range	100 m	
Current Draw	19,7 mA	Receiver mode
	17,4 mA	Transmission mode
	1 μA	Sleep mode
	10 μA	Idle mode
Battery	1600 mAh	2X AA, 1.2V, Ni-Mh

We analyze the effectiveness of the results in the following three cases:

- Compare the cumulative energy of our approach and related works by varying the time of the transmission, reception, and idle periods.
- Compare the cumulative energy of our approach and the related works based on Edge Computing by varying number generated packet.
- Increases the number of hops between MQTT clients and the broker.

9.2.2 Simulation Results

Figure 14 illustrates the cumulative energy vs. time of the related works and MQTT-MBD. The cumulative energy is the total energy needed to transmit a topic from the publisher to the subscriber. Through the presented curves we observe that the MQTT-MBD increases the lifetime of the network approximately two times higher than DM-MQTT since the number of hops between publisher subscribers in MQTT-MBD is much smaller than DM-MQTT. Also, as the time of the transmission, reception, and idle period increase we can observe that the cumulative energy increases at his turn. To conclude, we notice that MQTT-MBD is more efficient than the approach based on Edge Computing as the MQTT-MBD approach maximizes the lifetimes of devices while MQTT-RD runout the batteries of the devices quickly.

Now with the carving 15, we aim to study the effect of increasing the rate of the generated packet on the cumulative energy of the MQTT-MBD and DM-MQTT

approach. We notice that enlarging the generated packet rate increase the cumulative energy. From the obtained results we conclude that for the same generated packet rate our approach conserves the network energy two and a half times more than DM-MQTT. We conclude that the proposed is more effective for the cumulative energy of the network regardless of the rate of the generated packets. This is because the number of hops between publishers and subscribers in MQTT-MBD is much smaller than DM-MQTT.

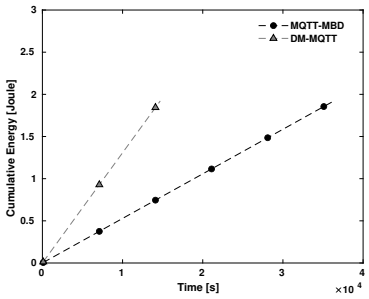

Fig. 14 Cumulative Energy between Publisher and Subscribers by Varying the transmission, reception Times

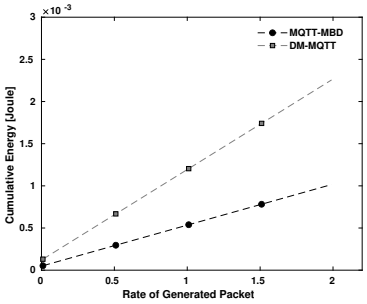

Fig. 15 Cumulative Energy between Publisher and Subscribers by Varying Generated Packet Rate

We conclude from the previous two simulations that the location of the MQTT brokers is a principal factor in the consumption of energy since the number of hops between the publisher and subscribers enlarges the cumulative energy of the network. Figure 16 illustrates the results of this experimentation. We observed from the results that the farther away the MQTT broker is from the MQTT clients, more the cumulative energy of the network increase. Also, the MQTT broker on Mist nodes is the foremost location for the access network to conserve energy since as we presented that it provides less cumulative energy consumption and it conserves the energy for much more time than the other position. Finally, we conclude that our MQTT-MBD is the most effective than Edge, Fog, and Cloud paradigm in terms of energy consumption since the Edge, Fog, and Cloud nodes further than Mist nodes. MQTT-MBD optimizes the resource utilization of the network by minimizing the cumulative energy consumption, consequently, it maximizes the lifetimes of the IoT applications.

10 Conclusion

In a real-time Internet of Things (real-time IoT) system, communications between publishers and subscribers with the same topic must occur in bounded time. Nowadays, the use of the Mist Computing paradigm is suitable for real-time communications between IoT devices. Since the MQTT protocol is the lightest and flexible protocol for IoT applications it requires a broker distribution under the Mist Computing paradigm for real-time IoT systems. Broker builds on Mist nodes that are constrained devices. Extremely, additional problem, limited resource of Mist broker can

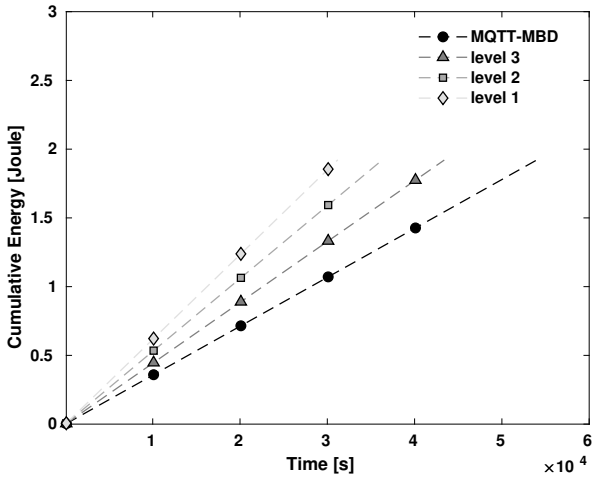

Fig. 16 Cumulative Energy between Publisher and Subscriber by Varying MQTT Broker Position

increase the delay of communication between IoT devices. In this work, we propose a distributed MQTT middle-ware under Mist Computing. The proposed MQTT-MBD builds MQTT brokers on Mist nodes. Seeing the Mist nodes have limited resources, we provide a mechanism to select the powerful Mist nodes satisfying the deadline constraint.

We made simulations to prove the relevance of the proposed approach. We construct two types of networks to test the effectiveness of MQTT-MBD. As a result, the proposed approach provides real-time communication between IoT devices with the same topic and belonging to the same access network by allowing data to proceed in Mist nodes without connection to the Cloud, Fog, and edge nodes. In addition, MQTT-MBD maximizes the lifetime of the real-time IoT system by minimizing the cumulative energy of the system and the workload balancing between different brokers.

From the scenarios of the simulation discussed above, we conclude that our approach is more efficient than EMMA, DM-MQTT, and MQTT-RD approach in terms of end-to-end response time. The end-to-end response time of literature approaches based on edge Computing paradigm is so higher than ours, and this goes back to the topic bridging between brokers resulting a supplementary communication, the lack of the workload balancing technique, and the static position of the broker.

Declarations

– Funding: Not supported.
– Conflicts of interest/Competing interests: No Conflicts.

- Availability of data and material: The datasets generated during and/or analysed during the current study are available from the corresponding author on reasonable request.
- Code availability: The code is a custom code written in python 3.7. To write this code two package were used. The first one is Networx 2.5 that is used for the construction of the network topology. The second one is matplotlib.pyplot that is used for the network visualization.
- Authors' contributions:We have proposed MQTT-MBD as a new MQTT broker deployment based on Mist computing for IoT (Internet of Things). The MQTT-MBD is made to optimize real-time applications with constrained IoT devices. We have also presented MQTT communication modelling for performance and QoS evaluation.

References

1. T. Rausch, S. Nastic and S. Dustdar, EMMA: Distributed QoS-Aware MQTT Middleware for Edge Computing Applications, 2018 IEEE International Conference on Cloud Engineering (IC2E), pp. 191-197, , 2018, doi: 10.1109/IC2E.2018.00043.
2. Park, Jun-Hong Kim, Hyeong-Su Kim, Won-Tae. DM-MQTT: An efficient MQTT based on SDN multicast for massive IoT communications, Sensors, 18, 3071,2018, 10.3390/s18093071.
3. Pereira, Eliseu Pinto, Rui Reis, João Pedro Gonçalves, Gil. MQTT-RD: A MQTT based Resource Discovery for Machine to Machine Communication. 2019. 10.5220/0007716201150124.
4. Sharma, Pradip Chen, Mu-Yen Park, Jong. (2017). A Software Defined Fog Node Based Distributed Blockchain Cloud Architecture for IoT. IEEE Access. 6. 115-124. 2017. 10.1109/ACCESS.2017.2757955.
5. Basir, Rabeea Qaisar, Saad Ali, Mudassar Aldwairi, Monther Ashraf, Muhammad Mahmood, Aamir Gidlund, Mikael. Fog Computing Enabling Industrial Internet of Things: State-of-the-Art and Research Challenges. Sensors. 19. 4807.2019. 10.3390/s19214807.
6. Wang, Xiaojie Huang, Jun. Vehicular Fog Computing: Enabling Real-Time Traffic Management for Smart Cities. IEEE Wireless Communications. 26. 2019. 10.1109/MWC.2019.1700441.
7. Khattak, Hasan Ali Arshad, Hafsa Islam, Saif Ahmed, Ghufran Jabbar, Sohail Sharif, Abdullahi Khalid, Shehzad. Utilization and load balancing in fog servers for health applications. EURASIP Journal on Wireless Communications and Networking. 2019. 10.1186/s13638-019-1395-3.
8. A. Yousefpour, G. Ishigaki, R. Gour and J. P. Jue, On Reducing IoT Service Delay via Fog Offloading, IEEE Internet of Things Journal, vol. 5, no. 2, pp. 998-1010, April 2018, doi: 10.1109/JIOT.2017.2788802.
9. Matt, Christian. Fog Computing: Complementing Cloud Computing to Facilitate Industry 4.0. Business Information Systems Engineering. 60. 351-355. 2018. 10.1007/s12599-018-0540-6.
10. Verba, Nandor Chao, Kuo-Ming Lewandowski, Jacek Shah, Nazaraf James, Anne Tian, Feng. Modeling industry 4.0 based fog Computing environments for application analysis and deployment. Future Generation Computer Systems. 91.2018. 10.1016/j.future.2018.08.043.
11. Aazam, Mohammad Zeadally, Sherali Harras, Khaled. Deploying Fog Computing in Industrial Internet of Things and Industry 4.0. IEEE Transactions on Industrial Informatics. PP. 1-1.2018. 10.1109/TII.2018.2855198.
12. Gustavo Caiza, Morelva Saeteros, William Oñate, Marcelo V. Garcia, Fog Computing at industrial level, architecture, latency, energy, and security: A review, Heliyon, 6, 4, 2020.
13. Gs, Chalapathi Chamola, Vinay Vaish, Aabhaas Buyya, Rajkumar. Industrial Internet of Things (IIoT) Applications of Edge and Fog Computing: A Review and Future Directions.2019.
14. Taneja, Mohit Davy, Alan. Resource Aware Placement of Data Analytics Platform in Fog Computing. Procedia Computer Science. 97. 153-156.2016. 10.1016/j.procs.2016.08.295.
15. Gomes, Raphael Bouloukakis, Georgios Costa, Fabio Georgantas, Nikolaos Rocha, Ricardo. QoS-Aware Resource Allocation for Mobile IoT Pub/Sub Systems. 70-87. 2018. 10.1007/978-3-319-94370-1_6.

16. Bouloukakis, Georgios Moscholios, Ioannis Georgantas, Nikolaos. . Probabilistic Event Dropping for Intermittently Connected Subscribers Over Pub/Sub Systems. IEEE International Conference on Communications. 2019. 10.1109/ICC.2019.8761557.
17. Bouloukakis, Georgios Georgantas, Nikolaos Kattepur, Ajay Issarny, Valérie. Timeliness Evaluation of Intermittent Mobile Connectivity over Pub/Sub Systems. ACM/SPEC International Conference on Performance Engineering (ICPE).2017. 10.1145/3030207.3030220.
18. Bouloukakis, Georgios Kattepur, Ajay Georgantas, Nikolaos Issarny, Valérie. Queueing Network Modeling Patterns for Reliable and Unreliable Publish/Subscribe Protocols.Conference: the 15th EAI International Conference. 176-186.2018. 10.1145/3286978.3287002.
19. Ouni, Sofiane Ayoub, Zayneb Kamoun, Farouk. Auto-organization approach with adaptive frame periods for IEEE 802.15.4/zigbee forest fire detection system. Wireless Networks. 25.2019. 10.1007/s11276-018-01936-x.
20. Kumar, Anurag Altman, Eitan Miorandi, Daniele Goyal, Munish. New insights from a fixed point analysis of single cell IEEE 802.11 WLANs. IEEE/ACM Transactions on Networking. 15.2007. 10.1109/INFCOM.2005.1498438.
21. Bouabdallah, Fatma Bouabdallah, N. Boutaba, R.. Load-Balanced Routing Scheme for Energy-Efficient Wireless Sensor Networks. 1 - 6.2009. 10.1109/GLOCOM.2008.ECP.21.
22. Vu, Hai Sakurai, Taka. Collision probability in saturated IEEE 802.11 networks. 2010.
23. Qiu, Tie Xia, Feng Feng, Lin Wu, Guowei Jin, Bo. Queueing Theory-based Path Delay Analysis of Wireless Sensor Networks. Advances in Electrical and Computer Engineering. 11. 3-8.2011. 10.4316/aece.2011.02001.
24. Guha Roy, Deepsubhra Mahato, Bipasha De, Debashis Buyya, Rajkumar. Application-aware end-to-end delay and message loss estimation in Internet of Things (IoT) - MQTT-SN protocols. Future Generation Computer Systems. 89.2018. 10.1016/j.future.2018.06.040.
25. Skarlat, Olena Schulte, Stefan Borkowski, Michael Leitner, Philipp. Resource Provisioning for IoT Services in the Fog.2016 IEEE 9th International Conference on Service-Oriented Computing and Applications (SOCA).2016. 10.1109/SOCA.2016.10.
26. Jurdak, Raja Ruzzelli, Antonio O'Hare, Gregory. Radio Sleep Mode Optimization in Wireless Sensor Networks. Mobile Computing, IEEE Transactions on. 9. 955-968. 2010. 10.1109/TMC.2010.35
27. Ouni, Sofiane Ayoub, Zayneb. Cooperative Association/Re-association Approaches to Optimize Energy Consumption for Real-Time IEEE 802.15.4/ZigBee Wireless Sensor Networks. Wireless Personal Communications. 71. 2013. 10.1007/s11277-013-0996-0.
28. Vasconcelos, Danilo Andrade, Rossana Severino, V De, J Souza,. Cloud, Fog, or Mist in IoT? That Is the Question. ACM Transactions on Internet Technology. 19. 25.2019. 10.1145/3309709.
29. Barik, Dr. Rabindra and Dubey, Amaresh and Tripathi, Ankita and Pratik, Tanjappa and Sasane, Sapana and Lenka, Rakesh and Dubey, Harishchandra and Mankodiya, Kunal and Kumar, Vinay. Mist Data: Leveraging Mist Computing for Secure and Scalable Architecture for Smart and Connected Health. Procedia Computer Science. 125.647-653.2018.
30. Yogi, Manas and Chandrasekhar, K. and Kumar, G.. Mist Computing: Principles, Trends and Future Direction. International Journal of Computer Science and Engineering. 4.2017.
31. Galambos, Péter. Cloud, Fog, and Mist Computing: Advanced Robot Applications. IEEE Systems, Man, and Cybernetics Magazine. 6. 41-45.2020.
32. Iorga M, Feldman L, Barton R, Martin MJ, Goren N, Mahmoudi C. Fog Computing Conceptual Model. NIST Special Publication 500-325. Gaithersburg, MD: NIST. 2018.
33. Iorga M, Feldman L, Barton R, Martin MJ, Goren N, Mahmoudi C. The NIST Definition of Fog Computing. NIST Special Publication 500-325. Gaithersburg, MD: NIST. 6. 2018.
34. Galambos, Péter. Context- and Self-Awareness in Fog and Mist Computing. Optical Zeitgeist Laboratory. 2021.
35. OpenFog Consortium,Openfog reference architecture for fog Computing, 2017
36. Texas Instruments CC2420 radio transceiver, Single-Chip 2.4 GHz IEEE 802.15.4 Compliant and ZigBee™ Ready RF Transceiver, 2008.
37. MICAz Wireless Measurement system, Document Part Number: 6020-0060-04 Rev A, Crossbow Technology, Inc . 2010.