Self-Configuring IoT Service QoS Guarantee Using QBAIoT

Ahmad Khalil, Nader Mbarek, Olivier Togni

computers

MDPI

Article

Self-Configuring IoT Service QoS Guarantee Using QBAIoT

Ahmad Khalil *, Nader Mbarek and Olivier Togni

University of Bourgogne Franche-Comté, LE2I, 21000 Dijon, France; Nader.Mbarek@u-bourgogne.fr (N.M.); Olivier.Togni@u-bourgogne.fr (O.T.)

* Correspondence: Ahmad.Khalil@u-bourgogne.fr

Received: 17 October 2018; Accepted: 14 November 2018; Published: 17 November 2018

Abstract: Providing Internet of Things (IoT) environments with service level guarantee is a challenging task for improving IoT application usage experience. We specify in this paper an IoT architecture enabling an IoT Service Level Agreement (iSLA) achievement between an IoT Service Provider (IoT-SP) and an IoT Client (IoT-C). In order to guarantee the IoT applications' requirements, Quality of Service (QoS) mechanisms should be implemented within all the layers of the IoT architecture. Thus, we propose a specific mechanism for the lowest layer of our service level based IoT architecture (i.e., sensing layer). It is an adaptation of the IEEE 802.15.4 slotted CSMA/CA mechanism enabling to take into consideration the requirements of real-time IoT services. Our access method called QBAIoT (QoS based Access for IoT) extends IEEE 802.15.4 systems by creating a new contention access period for each specified traffic class in the iSLA. Furthermore, due to the huge number of IoT connected devices, self-configuring capability provisioning is necessary for limiting human intervention and total cost of ownership (TCO). Thus, we integrate a self-configuring capability to the QBAIoT access method by implementing the MAPE-K closed control loop within the IoT High Level Gateway (HL-Gw) of our proposed QoS based IoT architecture.

Keywords: IoT; QBAIoT; self-configuring; IEEE 802.15.4; Slotted CSMA/CA; e-health

1. Introduction

Internet of Things (IoT) is nowadays evident in our everyday life. We estimate the economic impact of the IoT to be between $3.9 and $11 trillion dollars by 2025 [1]. Indeed, billions of connected objects exist with an average of two devices per human on earth resulting in creating the IoT environment [2]. The future growth of IoT will lead to an important usage of technology in our daily life. Thus, this usage of technology aims to improve the human's quality of life by facilitating everyday tasks. To ensure better user experience and improve the usage of IoT applications, a certain service level corresponding to specific IoT services is needed to motivate the users to use the IoT applications. The expected service level should consider not only QoS guarantee, measured through different performance parameters (i.e., delay, packet delivery ratio, etc.), but also security and users' privacy issues within different layers of the IoT architecture [3,4]. We consider in this paper the QoS part of a service level and we specify a QoS mechanism enabling the guarantee of a service level prior to the IoT service subscription. In this context, the IEEE 802.15.4 standard is used as a base for different communication technologies in the IoT environment and specifically in the e-health application domain. Thus, we specify a novel QoS based wireless access method for IoT environments called QBAIoT. Our method is an enhancement of the slotted Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) technique, used by the beacon-enabled mode of the IEEE 802.15.4 standard. In this paper, we extend our previous work [5]. The objective of the proposed method is to guarantee QoS parameters corresponding to the requirements of a specific IoT service such as e-health. These requirements are

specified into a contract called IoT SLA (iSLA) negotiated between the IoT Service Provider (IoT-SP) and the IoT Client (IoT-C). In this research work, we aim to present the design details and performance evaluation of our proposed QBAIoT access method and its usage in an IoT architecture in order to satisfy the requirements of an e-health service according to a proposed iSLA. In addition, we present an enhancement of QBAIoT considering a self-configuring capability for minimizing human intervention and responding to the IoT environment changes. The reminder of the paper is organized as follows. We present in Section 2 the state of the art concerning the IoT environment, the self-management in IoT and the important characteristics of the IEEE 802.15.4 technology. Section 3 describes the QoS motivations in the IoT as well as our service level based IoT architecture along with the proposed iSLA and its establishment. Then, we specify in the same section our proposed method enabling QoS based access for IoT environments and the corresponding self-configuring capability. Section 4 presents a detailed performance evaluation of our novel self-configuring access method while we compare it to the standard IEEE 802.15.4 access method as well as another QoS based access method called SDA (Service Differentiated and Adaptive) CSMA/CA. Finally, we conclude the paper in Section 5 and present future works.

2. State of the Art

2.1. Internet of Things

IoT is a new paradigm in our everyday life. It is created through the interconnection of heterogeneous systems communicating by using different communication technologies. In what follows, we present a general overview of the IoT environment.

2.1.1. Definitions

Different standardization organizations had presented the definition, architecture, recommendations, and application domains of the IoT. The International Telecommunication Union–Telecommunication Standardization Sector (ITU-T) presents a definition of the IoT environment and its applications through the SG20 study group. According to the ITU-T Y.2060 document, IoT is a ubiquitous network available anywhere, at any time and to anyone [6]. It is a global infrastructure allowing to offer advanced services by interconnecting objects through various communication technologies [7]. In addition, the International Organization for Standardization/International Electrotechnical Commission (ISO/IEC) proposes a definition and specifies the vocabulary used in the IoT environment in [8]. According to this organization, the IoT is a network of physical objects that collect and transmit data. It is an infrastructure of interconnected objects, humans, and information resources that can process the information reported and react accordingly [8,9]. Moreover, the Internet Engineering Task Force (IETF), defines the general idea of the IoT as the fact of connecting objects to provide contextual services, and this through different technologies to achieve a service accessible from anywhere and at any time [7].

2.1.2. IoT Related Technologies and Application Domains

External resources for processing and storage of IoT big data (generated by objects) such as cloud and fog or edge computing are used in IoT environments to enhance reliability and efficiency of IoT service provision [10]. Fog or edge computing decentralize the computing capacities and distribute the operations on network extremities to reduce costs and end-to-end delay of service provision, whereas cloud computing centralizes these capacities on powerful servers [10,11]. In this context, the IoT environment integrates different application domains like e-health, smart cities, smart building, vehicular networks, industry, etc. [8]. We focus in this research on e-health services characteristics in order to evaluate our proposed QoS based access technique. An e-health service combines public health and ICT (information and communication technologies) for providing humankind with novel health services [12]. These services are varied and can be simple as a remote patient's health management or

complex as a remote performed operation. In such an environment, connected objects can be sensors deployed on the patients for remote monitoring of their vital signals and storing the corresponding data in the cloud for further analyses. Furthermore, objects can be connected robots managed by distant surgeons for performing operations remotely.

2.1.3. IoT Communication Technologies

Several communication technologies enable IoT objects interconnection. A specific communication technology is used in conformance with the IoT objects characteristics (limited energy consumption and CPU utilization) and applications requirements. IoT communication technologies correspond to an adaptation of an existing technology or a new specifically specified technology. We present in Figure 1 a non-exhaustive classification of IoT communication technologies into wireless cellular technologies (LTE, 4 G, NB-IoT, 5 G, etc.) [13] and wireless non-cellular technologies (IEEE802.15.4 [14], LoRaWAN [15], ZigBee [16], 6LoWPAN [17], etc.).

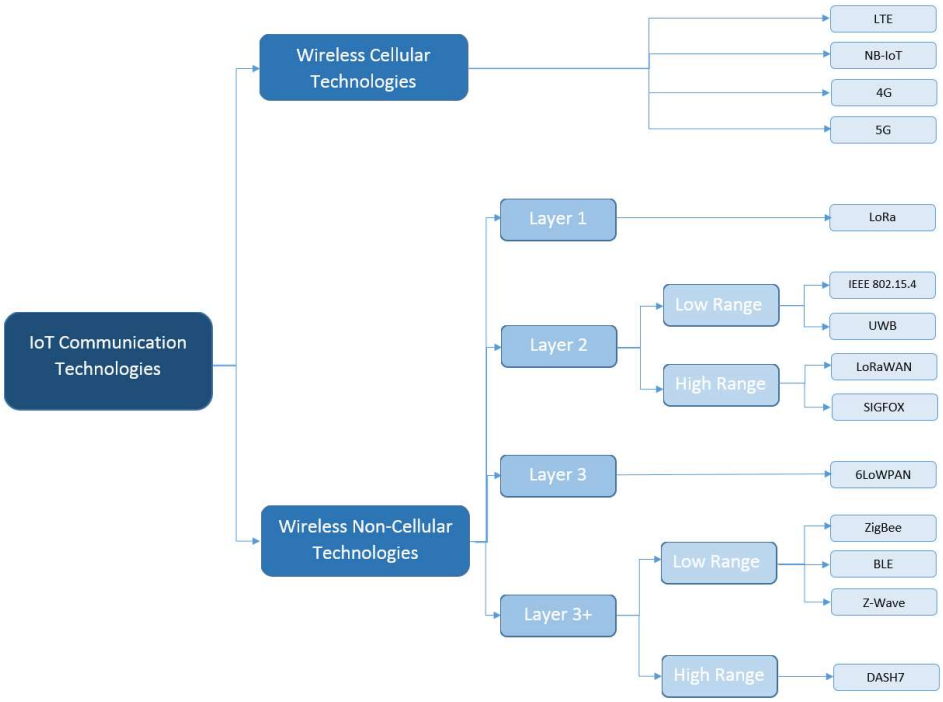

Figure 1. IoT communication technologies.

In the e-health application domain, different research works use the ZigBee communication technology to interconnect different IoT objects. In [18], a wireless healthcare monitoring system based on ZigBee is specified in order to provide real-time information concerning the health condition of patients. Furthermore, the authors in [19] developed a continuous patients monitoring system based on ZigBee protocol. In fact, ZigBee, 6LoWPAN and other communication technologies use IEEE 802.15.4 as a basis for their lower layers (physical and medium access control layer). The expanding number of connected devices in the IoT makes the IPv6 usage a necessity. Consequently, 6LowPAN [17], based on IEEE 802.15.4 for the physical (PHY) and medium access control (MAC) layers, will be greatly used in the IoT environment for services requiring low data rate and low range coverage. Thus, several IoT real-time services can use IEEE 802.15.4 based communication technologies, like patient vital signal monitoring and emergency situations monitoring (i.e., fire alarms, intrusion detection alarms, etc.). In addition, other non-real-time services, such as the environment monitoring services (humidity monitoring, temperature monitoring, etc.), are also candidates for the usage of IEEE 802.15.4 based communication technologies. Therefore, we describe in Section 2.2 the most important characteristics of this standard that we adapt to design a QoS based access method called QBAIoT.

2.1.4. Self-Management in the IoT Environment

The human and operational cost for managing the IoT infrastructure is very important due to the interconnection of billions of smart devices and the heterogeneity of used technologies. In this context, the autonomic computing paradigm characterized by the self-management concept allows to cope with the increasing complexity and operational cost of the IoT infrastructure management. The importance of considering self-management capability within the IoT environment had been highlighted by different organizations such as the ITU-T in their Y.2066 document presenting the common requirements of the IoT [20]. ITU-T describes the importance of self-management within IoT environments provided by self-configuring, self-healing, self-optimizing, and self-protecting functions.

Most of the research works integrating the Autonomic Computing paradigm are based on the IBM MAPE-K closed control loop [21] that specifies the self-management concept and describes the transition from a manual to an autonomic system. The latter is based on two main components, the autonomic manager (AM) and the managed element (ME). MEs are the controlled elements that undergo the decision made by the AM based on the knowledge base.

MAPE-K closed control loop, illustrated by Figure 2, stands for monitor, analyze, plan, execute-knowledge [21]. Each function has a specific role in the closed control loop process. The monitor function collects information from the MEs, through the sensors interfaces. Monitored data are considered by the analyze function to predict future states of the system. Therefore, the plan function aims to schedule actions according to policy information and rules provided by the knowledge base. At the end, the execute function controls the execution of the provided plan and dispatches the recommended actions on the effectors of the MEs. The knowledge base stores the policy information and the specified rules corresponding to different scenarios that can occur within the autonomic system environment [21]. Sensors are used to communicate sensed information by the ME to the AM, whereas, effectors are used by the AM to execute orders or communicate information to the ME.

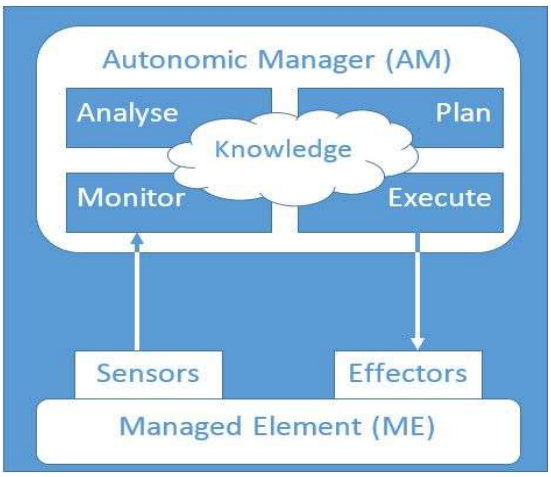

Figure 2. MAPE-K closed control loop.

Different research works focused on the integration of the self-management capabilities in the IoT environment. Indeed, the OpenIoT project [22], a FP7 funded European research project, specified an OpenIoT autonomic service control loop and an OpenIoT self-management framework in order to implement the autonomic computing concept in the IoT environment. The specified control loop is based on sensors and effectors interfaces as well as different control functions. The authors in [23] insist on the idea that loT devices should not require any technical configuration. According to their vision, an IoT device should be simply plugged in, switched on, and everything else should be automatically configured. The authors used semantic data in order to describe the capabilities of the objects, and the IoT environment in general. In this research work, devices deduct their configurations by finding other devices with similar states through semantic data processing.

2.2. IEEE 802.15.4

After presenting the IoT concepts, related technologies, communication technologies and the importance of the autonomic computing paradigm adoption within IoT environments, we describe in the following the IEEE 802.15.4 standard as it is the basis of our proposed access method.

2.2.1. IEEE 802.15.4 Specification

IEEE 802.15.4 is a low-rate wireless personal area network (LR-WPAN) standard defining the physical PHY and MAC layers. PHY and MAC layers specify essential parameters such as data rate, control functions, data management format, and usage of slotted CSMA/CA access method. It provides also other management features like access to medium, synchronization, etc. [14]. Standards and technologies based on IEEE 802.15.4 use its PHY and MAC layers and add their specificities through higher layers.

IEEE 802.15.4 specifies six different access methods in its last version [14]: the unslotted CSMA/CA in non-beacon enabled PAN, the slotted CSMA/CA in beacon enabled PAN, time slotted channel hopping clear channel assessment (TSCH CCA) in non-shared slots, time slotted channel hopping CSMA/CA (TSCH CSMA/CA) in shared slots, CSMA/CA with priority channel access (PCA) and low-energy critical infrastructure monitoring (LECIM) ALOHA with PCA. In what follows, we briefly describe these different access methods except the slotted CSMA/CA beacon-enabled mode that is described meticulously, as it is the basis of our contribution.

In the unslotted CSMA/CA in non-beacon enabled PAN, the access method corresponds to the CSMA/CA traditional method. The device tests the availability of the channel before trying to send the data. Whereas, in the slotted CSMA/CA beacon-enabled mode, a specific virtual frame structure (called 'superframe') is used. The superframe consists of an active part known as the superframe duration (SD) divided into 16 time-periods known as slots and an optional inactive period. The slots form a contention access period (CAP) and an optional contention free period (CFP). During the CAP, nodes should compete to gain access to the shared channel, whereas, during the CFP, objects are allocated guaranteed time slots (GTS). At each beacon interval (BI) corresponding to the end of a superframe, the coordinator sends a beacon frame to all its corresponding nodes in the WPAN. The beacon allows the coordinator to identify its WPAN nodes, to synchronize the nodes and to communicate different superframe configuration values like the beacon order (BO) and the superframe order (SO). BO and SO are used to compute the BI and the SD according to Equations (1) and (2) respectively. Base superframe duration (BSFD) is the minimum duration of a superframe, corresponding to a SO value of 0. It is fixed to 960 symbols of 4 bits each. Furthermore, BO and SO should respect the inequality $0 \leq \mathrm{SO} \leq \mathrm{BO} \leq 14$ [14].

$$\mathrm{BI} = \mathrm{BSFD} \times 2^{\mathrm{BO}}, \tag{1}$$

$$\mathrm{SD} = \mathrm{BSFD} \times 2^{\mathrm{SO}}. \tag{2}$$

The TSCH CCA mode uses enhanced beacon frames that contains the TSCH synchronization information, the channel hopping information, the TSCH timeslot information, the TSCH slotframe and the link information. Collision avoidance using CCA is ineffective in TSCH PAN. In fact, the devices operating in TSCH mode use channel hopping, which eliminates the usage of backoff periods. When a TSCH device has a packet to transmit, it shall wait for a link to the destination device. If TSCH CCA was set to ON, the MAC layer requests the PHY layer to perform a CCA at the designated time in the timeslot without back off delays in order to send the data. As for the TSCH CSMA/CA mode, it is used in case of shared links. The shared links are assigned to multiple devices that can lead to collisions and retransmissions. To reduce the probability of multiple retransmissions for the same packet, a retransmission backoff algorithm is used in this mode. Furthermore, the CSMA/CA with PCA mode is used before transmitting the critical event message during the CAP. In this mode, the MAC sublayer ensures that the remaining CSMA/CA operations can be undertaken and the transmission can be

achieved before the end of the CAP. The PCA corresponds to the back off algorithm to be in use in this case. Finally, the last mode known as the LECIM ALOHA PCA is used when the CCA mode is set to four. Different modes for the CCA are available (1 to 4), each has its own way to determine the availability of the channel. The CCA mode 4 reports always an idle channel and is used in low duty cycle applications. It implies the usage of a modified version of the PCA backoff algorithm [14].

The latest version of IEEE 802.15.4 uses a special extension called deterministic and synchronous multi-channel extension (DSME) that extends the usage of GTS to all network devices by using multiple channels for slot allocation in dense networks. This extension increases the complexity of slots coordination and complicates the process of avoiding inconsistencies in slots schedules [14].

2.2.2. IEEE 802.15.4 Slotted CSMA/CA

The slotted CSMA/CA algorithm is used by every node to transmit a new packet (command or data packets) in the beacon enabled mode. It is executed only during the CAP (See Figure 3) period [14].

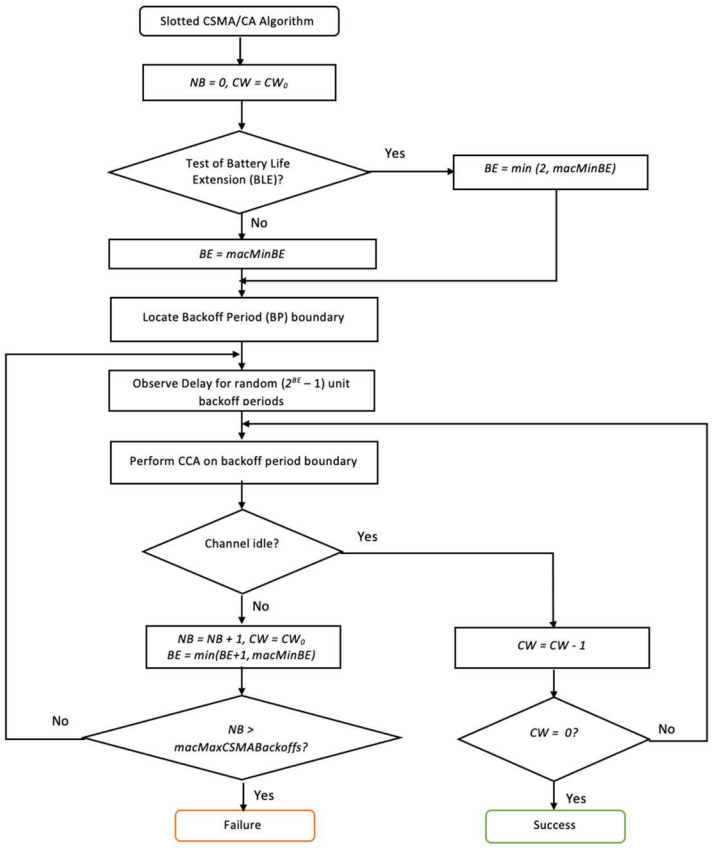

Figure 3. Slotted CSMA/CA algorithm.

Three variables are specified within the slotted CSMA/CA algorithm:

- Backoff Exponent (BE): it computes the backoff delay observed by the nodes before performing the CCA test on the shared medium. The value of BE is randomly chosen between 0 and $(2^{BE} - 1)$.
- Contention Window (CW): it indicates the number of backoff periods during which the channel must be idle before a node can access the channel to send its data. It has a default value of 2.
- Number of Backoffs (NB): it corresponds to the number of backoff executed before channel access. It is initialized to 0 and compared to a maximum value, macMaxCSMABackoffs, which is by default equal to 5. If the value of *NB* is greater than the maximum value, a failure occurs.

First, NB is initialized to 0 and CW is initialized to the CW_0 value corresponding to 2 by default. After the initialization, the algorithm tests the Boolean value of the battery life extension (BLE) variable.

If the latter is fixed to true, BE is set to the minimum between 2 and macMinBE (by default fixed to 3). Whereas, if BLE value is fixed to false, BE is initialized to 2. After fixing the values of the different variables, a node implementing the CSMA/CA algorithm waits for the backoff delay and performs the first CCA test. Two possible channel states exist:

- The channel is busy and consequently CW is reinitialized to CW_0, if it has been changed. Furthermore, a busy channel results in incrementing NB and BE. BE must not exceed the value of aMaxBE fixed by default to 5. When a node reaches the macMaxCSMABackoffs value, the CSMA/CA algorithm reports a failure to the higher layer. Otherwise, if NB has not reached the macMaxCSMABackoffs value, the backoff operation restarts and the CCA is performed again.
- The channel is sensed idle and CW is greater than 0: the CCA is repeated and CW decremented. Otherwise, the node attempts to transmit if the remaining time in the current CAP is sufficient to transmit the frame and receive the acknowledgement. If not, the process is deferred to the next CAP of the next superframe.

3. QoS Guarantee within Internet of Things Environment

In order to provide IoT objects with QoS guarantee according to a negotiated service level between the IoT-SP and the IoT-C, we have to deploy various QoS mechanisms at different layers of the IoT architecture. We describe in the following the QoS challenges and motivations in the IoT environment. Then, we present our service level based IoT architecture as well as our QoS adaptation of the slotted CSMA/CA access method used in IEEE 802.15.4.

3.1. QoS Motivation and Challenges in IoT

The exponential increasing number of IoT connected objects leads to a greater amount of data created in the IoT environment. In addition, the generated data has different and even contradictory characteristics that we have to cope with by providing QoS guarantee within the IoT environment, which is a challenging task. QoS models such as IntServ and DiffServ are difficult to be used within the IoT environement due to the constraints in terms of memory and computing capacity of IoT objects but also due to the large scale of the IoT and the random deployment of objects. Therefore, new and well-adapted QoS models must be proposed to provide IoT services performance guarantees [24]. Various international organizations consider that QoS integration in IoT environments is very important and necessary. The ITU-T describes the importance of integrating QoS into the IoT environment through their Y.2066 document [21]. They focus on the importance of service priority in order to satisfy specific requirements of different IoT services during congestion periods that are frequent due to the huge amount of generated data in the IoT. Indeed, during congestion periods, IoT traffic performance will be affected and especially the performance of QoS constrained data streams. In order to avoid performance degradation when delivering critical data, an effective and optimized management of the available resources is necessary to guarantee a certain service level in the IoT environment. This service level guarantee is possible thanks to the specification of QoS mechanisms in different layers of the IoT architecture in order to satisfy different QoS requirements and parameters such as delay, jitter, bandwidth, and packet loss ratio for various traffic types [25]. Furthermore, these QoS mechanisms must take into consideration different components of the IoT environment (sensing, gateway, network, and cloud) in order to achieve an end-to-end service level guarantee for IoT services.

Different research works developed approaches to integrate the QoS in different layers of the IoT architecture. OpenIoT [22], a FP7 funded European project, had presented different QoS parameters for the IoT environment. Each parameter has its own set of metrics: energy consumption associated to the lifetime of the sensor layer, data volume produced by the sensors, trustworthiness correlated to the quality of sensor, etc. Traditional metrics had been also used by OpenIoT such as the jitter, the delay, etc. All these metrics are used to define the service level expected by the client and to specify the requirements of each IoT service. In addition, this European research project proposed a QoS manager that keeps tracking of the defined metrics for guaranteeing the service level. Furthermore,

different research works proposed the integration of QoS mechanisms within the lower layer of the IoT architecture. F-Interop Platform [26] is a H2020 European funded research project that focuses on developing online interoperability and performance test tools for the IoT environment. Remote online testing software was proposed to measure the QoS, the scalability and the energy in such environment. The collection of statistics in F-Interop testing tools can be ensured by using a passive or an active method thanks to software defined networking (SDN) and network functions virtualization (NFV) techniques. QUASIMODO [27], a French national research project, proposed different methods and algorithms for providing an end-to-end self-adaptive QoS support for real-time applications and energy efficiency in wireless sensor networks (WSN) and IoT systems. Different issues are studied for guaranteeing the QoS. Thus, QUASIMODO presented a routing protocol for multi-hop routing under multi-constraints (energy, link reliability, delay). It is based on the operator calculus algebra and it was successfully implemented on Contiki/TelosB platform. Furthermore, several academic research projects focused on the guarantee of service level in the IoT environment. In [28], the authors presented a three-layer architecture for the IoT environment (perception layer, network layer, and application and service layer). The architecture includes also a cross-layer QoS management facility as well as QoS brokers for supporting end-to-end QoS operations. In fact, the broker is responsible for resolving QoS requirements received from the upper layer and translating them to QoS requests on the local layer. The QoS requirements are a set of performance parameters to be respected. In this work, the authors had grouped the IoT applications into different types, and assigned a QoS level to each application type. Indeed, control applications need a guaranteed service, query applications need a guaranteed or a differentiated service, real-time monitoring applications need a differentiated service, while the non-real-time monitoring applications need best effort service. The research work conducted in [29], specifies different queues and a scheduler within an object in order to ensure a specific priority while processing QoS constrained flows. Moreover, some research works tried to adapt the slotted CSMA/CA algorithm to ensure a QoS guarantee. The research conducted in [30] had adapted the IEEE 802.15.4 standard and specifically the slotted CSMA/CA algorithm by integrating a QoS mechanism. This approach is called SDA-CSMA/CA and is based on the specification of different values of CSMA/CA variables (contention window, minimal and maximal BE) enabling different priorities according to traffic types. Thus, three priority levels are considered while using SDA-CSMA/CA. In this context, smaller CWs and BE intervals (minimal BE and maximal BE) are assigned to high-priority traffics. Consequently, important messages have higher chances for channel access with shorter backoff time.

3.2. IoT Service Level Based Architecture

We propose a service level-based architecture (see Figure 4) while considering an e-health service scenario enabling an IoT-SP to deliver to an IoT-C an IoT service in conformance with a global iSLA. The proposed architecture, compliant with the ITU-T reference model, is based on four layers (sensing, gateway, network, and cloud) as well as different entities that each have a specific role:

- The IoT-C requests a service from the IoT Service Provider and can use an IoT objects infrastructure provided by the IoT-SP or its own infrastructure. The IoT-C is not necessarily the end user of the IoT service. Indeed, it can be the organism in charge of the end users.
- The IoT-SP offers IoT services for different IoT-Cs. It has its own cloud infrastructure or subscribes specific SLAs with different cloud service providers (CSPs). The cloud infrastructure stores the huge amount of data created, provides computation capabilities and hosts different IoT applications. In addition, the IoT-SP subscribes SLAs with different network service providers (NSPs) to interconnect the IoT objects infrastructure with the corresponding cloud infrastructure.

The IoT service is divided into two parts: an application part and an objects infrastructure part. The objects infrastructure is managed by two categories of gateways in order to retrieve information or execute orders. The first category of gateways is called high level gateways (HL-Gws). They ensure minimal

bandwidth consumption while transmitting data to the cloud using the data aggregation techniques and offer fog computing capacities for non-delay tolerant data processing in order to minimize the corresponding end-to-end delay. The second category consists in low level gateways (LL-Gws) that are used to ensure not only data classification for processing differentiation but also management of objects' clusters. On the other hand, the application part of the IoT service is usually hosted on the cloud infrastructure and can be provided as a software as a service (SaaS) by the CSP or developed by the IoT-SP while using a platform as a service (PaaS) or an infrastructure as a service (IaaS).

A SLA concluded between the IoT-SP and the IoT-C allows defining the expected service characteristics. In the context of our IoT architecture, the SLA should include clients' expectations relative to all specified layers and components: devices, network, data and cloud. Thus, our proposed global IoT-SLA (iSLA) specified between the IoT-SP and the IoT-C is based on different sub-SLAs concluded with the CSP (cSLA: cloud SLA) as well as the NSP (nSLA: network SLA). The *cSLA* includes performance parameters according to different types of available cloud services (IaaS, PaaS, SaaS). As for the nSLA, it is based on traditional QoS network parameters such as bandwidth, latency, jitter, packet loss ratio, and availability. In addition, an internal SLA, called gSLA (gSLA: gateway SLA), is specified on the HL-Gw and includes the different characteristics of the IoT service processing within the gateways. We presented in our previous work [31] a detailed description of these service level agreements.

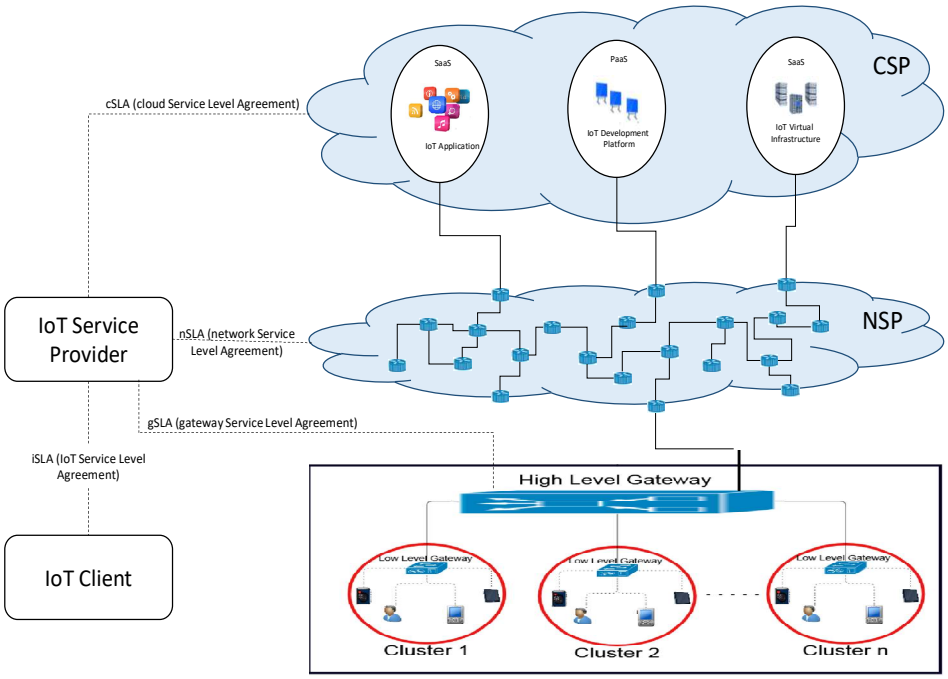

Figure 4. IoT Service level based global architecture for an E-health application.

3.3. IoT Service Level Agreement

The IoT-SP uses different cSLAs and nSLAs concluded with various NSPs and CSPs, in order to achieve an iSLA with the IoT-C. In the cSLAs and nSLAs, the different information and QoS guarantees regarding the IoT service on the corresponding IoT layer are agreed on. For example, in the cSLA, the number of users accessing the IoT application is indicated in order to allow the CSP to provide these users with the QoS guarantee corresponding to the cloud layer (i.e., response time, storage, etc.). Whereas, in the nSLA, the maximum bandwidth utilization is specified in order to guarantee the delay in the network layer. The different information specified in the iSLA sub agreements (i.e., cSLAs, nSLAs, gSLAs) are used to specify the service characteristics to be respected (i.e., number of users, numbers of connected objects, packets generation intervals, etc.) in order to benefit from the desired

QoS. We specify the iSLA as a service level agreement specific to IoT environments and we use it in our proposed IoT architecture. The specified iSLA includes multiple QoS parameters corresponding to the technical part of the iSLA. The iSLA global XML schema is presented in Figure 5.

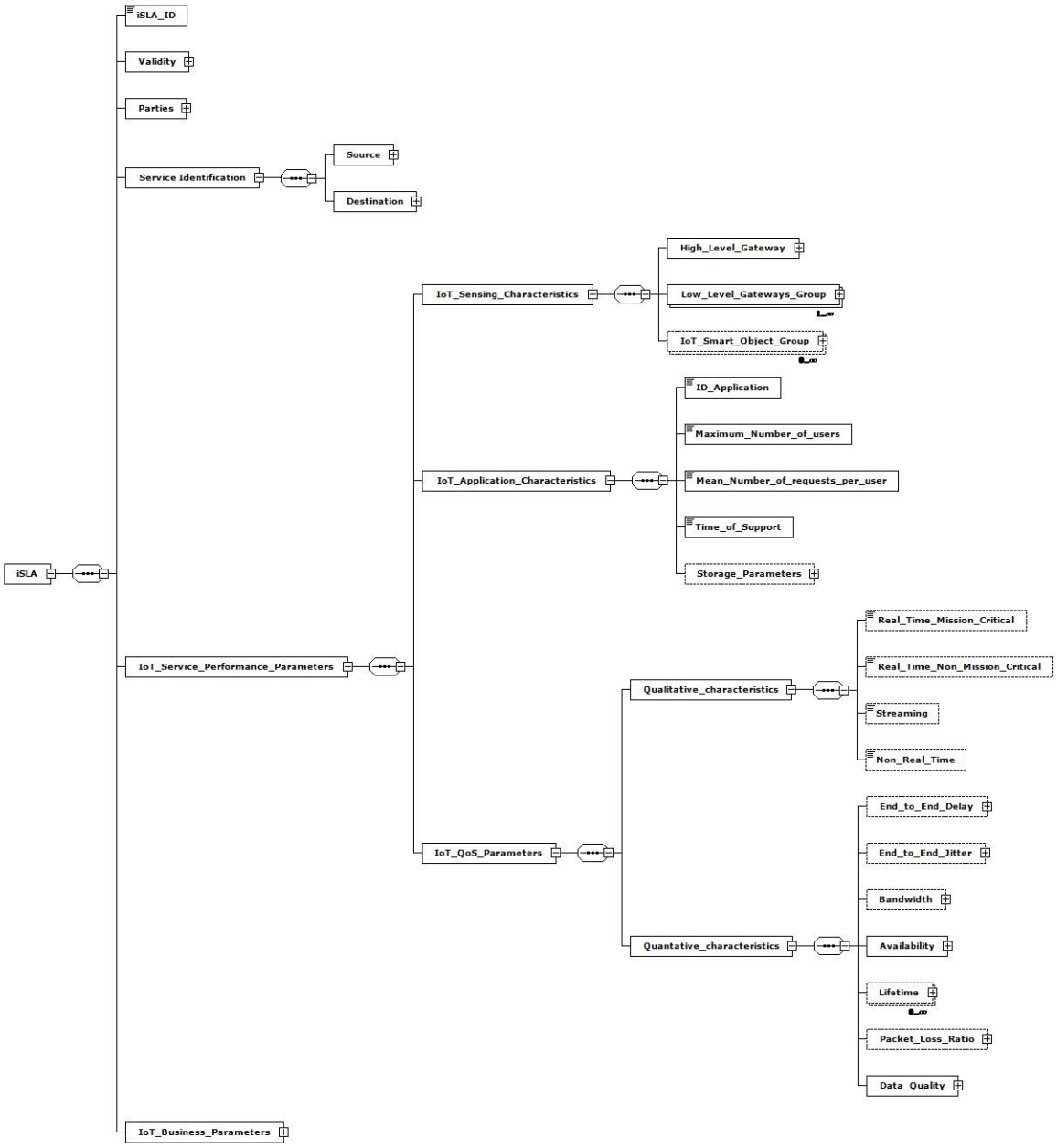

Figure 5. iSLA XML global schema.

The iSLA identifies the IoT service by a list of client IP addresses for the source side. As for the destination side, it uses an application ID, an application Interface IP along with the port number as well as the ID of the corresponding HL-Gw and LL-Gws. Furthermore, the iSLA contains an IoT service performance parameters attribute describing the IoT Sensing Characteristics as well as IoT application characteristics and IoT QoS parameters. Thus, the IoT sensing characteristics part of the iSLA includes all the characteristics of the gateways (HL-Gw, LL-Gw) and the IoT smart objects group. Then, the IoT application characteristics part of the iSLA includes parameters related to the cSLA such as the application ID, the maximum number of users, the mean number of requests per second per user, the time of support and the storage parameters. Finally, the IoT QoS parameters part of the iSLA is

classified into two different categories; the qualitative parameters and the quantitative parameters. The qualitative characteristics define the application type by specifying the QoS class of the data generated (i.e., RTMC: real-time mission-critical, RTNMC: real-time non-mission-critical, Streaming, and NRT: non-real-time). The quantitative QoS parameters include the end-to-end delay, availability, lifetime, and data quality (standard deviation, sensing frequency, and data error ratio).

Financial elements are essential in SLAs. Therefore, the iSLA contains an IoT business parameters attribute. It includes an IoT service cost part as well as IoT violations and penalties part where we specify thresholds and the monitoring interval time for each IoT QoS parameter.

As a usage case, we consider an e-health service belonging to the real-time mission-critical class and we describe how to establish the corresponding iSLA within our IoT architecture in order to satisfy the requirements of such application. In this context, we present a message sequence chart that illustrates the interactions that occur between the different components of our architecture (see Figure 6).

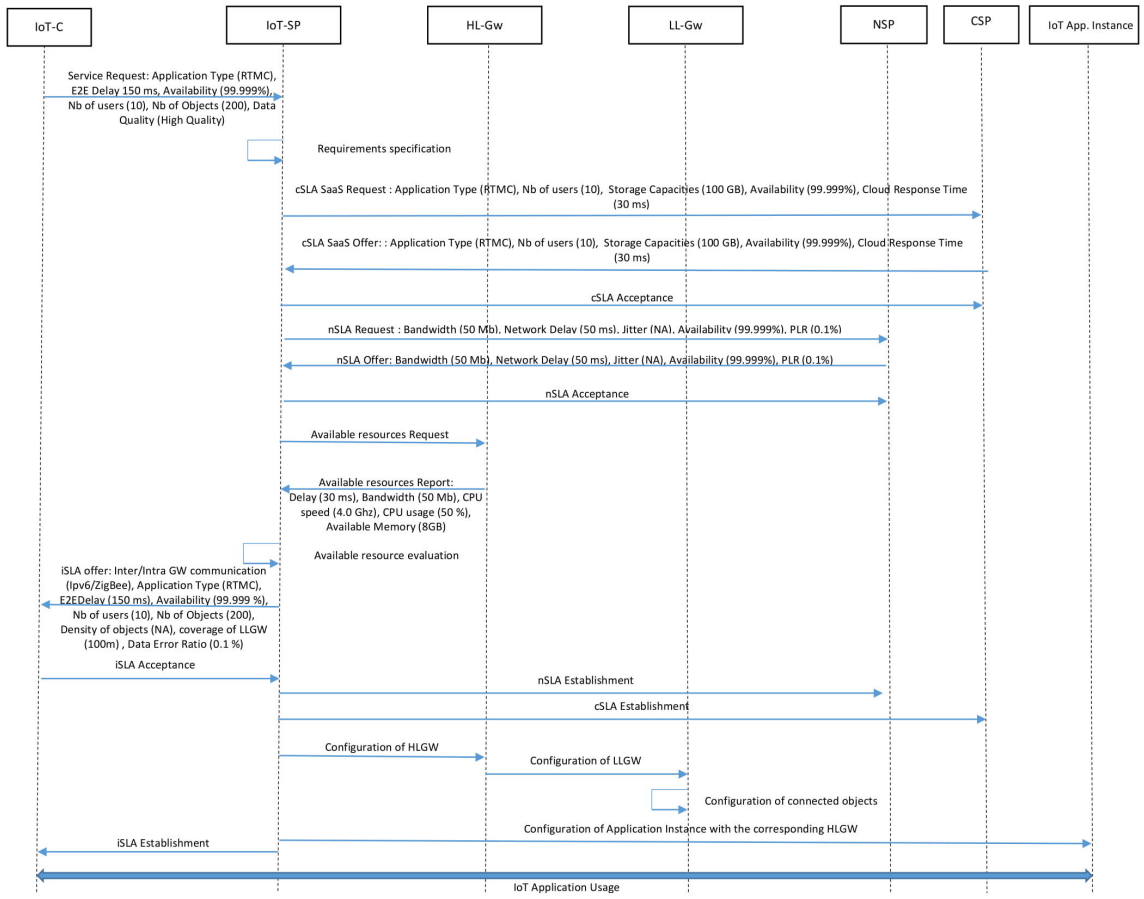

Figure 6. IoT components interaction for e-health application iSLA establishment.

The e-health application is a real-time mission-critical application that requires a limited end-to-end delay (150 ms) with high Availability (99.999%) and a high data quality. The end-to-end delay declines into IoT application response time (cloud), the IoT sensing response time (objects) and the network delay. As for the IoT service availability, QBAIoT does not take into consideration radio noise issues. Thus, QBAIoT could provide high availability only if no noise issues exist in the sensing environment. As shown in the MSC (see Figure 6), the IoT-C initiates the iSLA establishment by sending a request concerning the desired IoT service with its requirements. Then, the IoT-SP will classify the requirements into three types; cloud requirements, network requirements and gateways/sensing requirements. After this classification step, the IoT-SP chooses among the best offers available from

the different CSPs and NSPs in order to serve the IoT-C. The IoT-SP initiates a cSLA request to the corresponding CSP and waits for his proposal. If the proposed cSLA meets the specified requirements, a cSLA Acceptance will be sent by the IoT-SP. Similarly, an nSLA request is initiated by the IoT-SP. The corresponding NSP should send its proposal and the IoT-SP accepts it if the requirements are satisfied. Afterwards, the IoT-SP evaluates the available resources on the corresponding HL-Gw to be used by the IoT-C requested service. A report about gateway resources' availability is collected by the IoT-SP. Based on this report, the IoT-SP concludes an internal SLA called gateway SLA (gSLA) with the existing HL-Gw or with a new implemented one. Thus, the IoT-SP has the capability to propose an iSLA to the IoT-C based on the cSLA, nSLA, and gSLA. If this offer is accepted by the IoT-C, the IoT-SP concludes the cSLA and nSLA with the CSP and NSP, configures the HL-Gw, LL-Gw, nodes and the application instance. Finally, the e-health IoT service is available for usage by the IoT-C users.

3.4. QBAIoT: QoS Based Access for IoT

In order to guarantee the negotiated values of the QoS parameters specified in the iSLA of our e-health usage case and illustrated by the MSC of Figure 6, we propose in this research work a novel QoS mechanism concerning the sensing layer of our IoT architecture. Indeed, we specify a QoS based wireless access method called QBAIoT enabling a QoS based communication between the IoT objects and the low-level gateway of our IoT architecture. The proposed QoS mechanism is an enhancement of the IEEE 802.15.4 slotted CSMA/CA process in order to allow a traffic differentiation while satisfying the QoS requirements of each traffic type (for example: minimal delay for delay sensitive applications, etc.).

Our QBAIoT method is based on the IEEE 802.15.4 standard while adapting its superframe structure to satisfy the requirements of different traffic types available within the IoT environment. Indeed, we can take into consideration up to four QoS classes that we define in our iSLA qualitative characteristics (RTMC, RTNMC, Streaming, and NRT). Real-time traffics (RTMC and RTNMC) are highly sensitive to delay, whereas streaming traffic is highly sensitive to jitter variation and the NRT traffic is a best-effort QoS class. To adapt the structure of the superframe to these QoS classes, we specify different CAPs within the standard IEEE 802.15.4 superframe. Each CAP is specific for one QoS class and it is called a QoS CAP. Our adapted superframe structure contains up to four QoS CAPs (see Figure 7). The number of QoS CAPs configured in the adapted superframe within our low-level gateway (LL-Gw) depends on the number of QoS classes specified in the iSLAs that concern these LL-Gws. Furthermore, we do not use neither a CFP period nor an inactive part in our adapted superframe. The removal of the inactive part results in minimizing the delay for real-time buffered data.

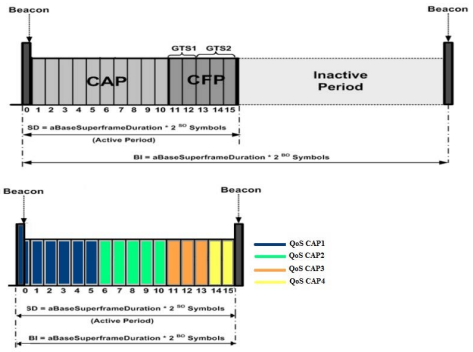

Figure 7. IEEE 802.15.4 and QBAIoT superframe structure.

During each QoS CAP, only objects belonging to the corresponding QoS class can compete to gain access to the shared medium. Each QoS CAP is allocated a number of slots. The slots configuration and the fixed values of BO and SO depends on the number of available QoS classes and the number of real-time QoS classes. If only one QoS class exists, the standard IEEE 802.15.4 superframe structure will be used in our architecture but without CFP and inactive periods, as a single QoS CAP exists.

In the case of multiple QoS classes, BO and SO will be initialized to a value equal to 2 if there is at least one real-time QoS class and to a value equal to 3 if there is no real-time QoS classes. The value of BO and SO allows computing the superframe duration, consequently, the slot duration and the delay that the objects have to wait to be in their corresponding QoS CAP. Our QBAIoT access method is implemented on the LL-Gw (acting as a coordinator) as well as the IoT objects. Therefore, we describe in the following the design details of the proposed wireless access method not only for the IoT gateway but also for the IoT objects.

3.4.1. IoT Gateway QoS Based Access Method

The IoT LL-Gw of our architecture adopts the proposed QBAIoT mechanism as a QoS based access method enabling QoS guarantee for IoT objects according to an achieved iSLA. We specify in Figure 8 the finite state machine that illustrates the QBAIoT process within the LL-Gw.

In state S0, the LL-Gw tests if there is a new configuration to be sent to the objects of its cluster. This new configuration is specified by the HL-Gw thanks to a QBAIoT self-configuring capability (cf. Section 3.5). Thus, if a new configuration exists, the LL-Gw updates the former beacon and sends it to the IoT nodes while reaching State S1. If there is no new configuration the beacon is sent directly to the IoT nodes at the beacon interval time and the LL-Gw switches to state S1. In State S1, the LL-Gw receives data from the different nodes according to their QoS CAPs. The LL-Gw counts the number of received packets during each QoS CAP enabling a self-optimizing capability that is out of the scope of this paper. At the end of each superframe, the number of received packet during the different QoS CAPs is communicated to the HL-Gw and the LL-Gw switches to state S0.

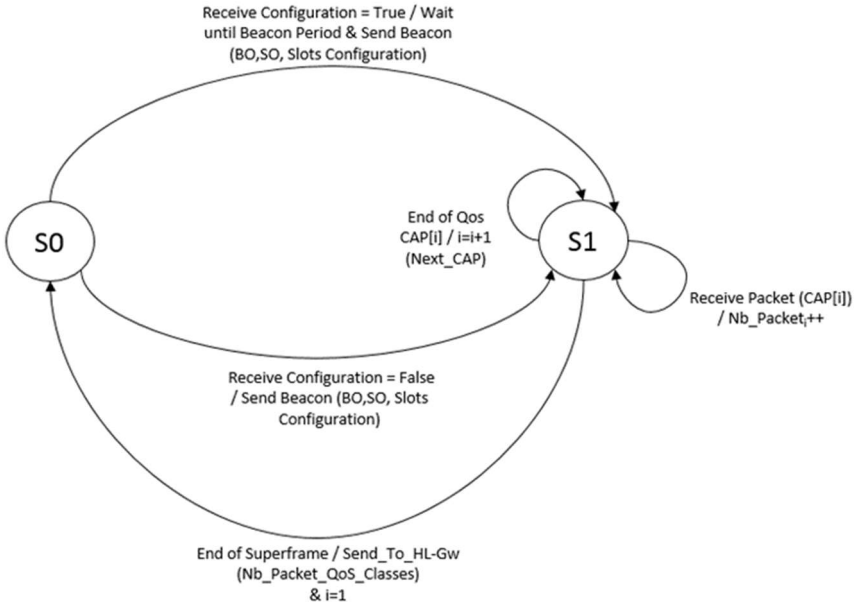

Figure 8. Finite state machine of a gateway QBAIoT process.

The QBAIoT process within the LL-Gw is illustrated in Figure 9. In this context, the LL-Gw specifies the beacon configuration according to the received values. Then, the beacon is sent to the IoT nodes if there is at least one existing traffic in the IoT environment. The existence of traffic in IoT environment can be deducted from the existence of gSLAs on the HL-Gw for the corresponding LL-Gw. If there is no traffic in the IoT environment, there is no need to send the beacons to the nodes. After sending the beacon, the LL-Gw should receive data from different nodes depending on the corresponding QoS CAPs of each node. In case of reception of new configurations resulting from added, removed or modified iSLAs, the LL-Gw updates its configuration values and sends a new beacon to the IoT nodes.

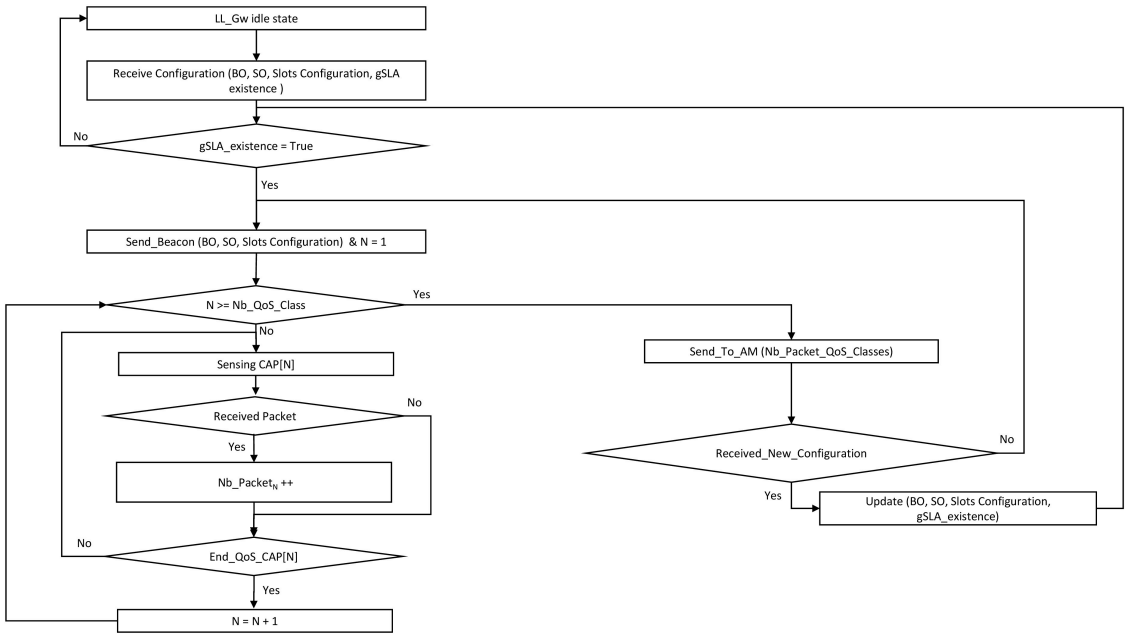

Figure 9. Algorithm schema of gateway QBAIoT process.

3.4.2. QoS Class Based Access for IoT Objects

We specify in Figure 10 the Finite State Machine that illustrates the QBAIoT process within an IoT object (i.e., a node). In state S0, an IoT object receives the LL-Gw gateway beacon specifying the configuration of BO, SO, and QoS CAPs. Based on its QoS Class, the IoT object configures its Object_CAPStart and Object_CAPEnd and then switches to state S1. At state S1, if the IoT object determines that its QoS CAP has not started yet (i.e., Current slot < Object_CAPStart), then it will remain in state S1 and waits for its QoS CAP in the current superframe. On the other hand, if the node determines that its QoS CAP had ended (i.e., Current slot > Object_CAPEnd), then it will remain in state S1 and waits for its QoS CAP in the next superframe after receiving a new beacon. Finally, if the node determines that its QoS CAP is effective (i.e., Object_CAPStart $\leq$ Current slot $\leq$ Object_CAPEnd), then it switches to state S2. At S2, an IoT object evaluates the remaining time in its QoS CAP in order to send a packet. If there is sufficient time, the node executes the CCA. An idle channel reduces the CW value. If CW is greater than 0, the CCA should be executed again, whereas a CW equal to 0, enables the node to send its data to the LL-Gw and to get back to state S1. Then, it waits for the beacon or tries to send another data packet.

On the other hand, if the channel is not idle, NB is compared to macMaxBackoffs. A value of NB greater than macMaxBackoffs leads to a MLME (MAC sublayer management entity) status set to "Channel Access Failure" and the node gets back again to state S1 where it waits for the beacon or tries to send another data packet. If NB is lower than macMaxBackoffs, the NB value will be incremented and CCA should be executed again while the node remains at state S2.

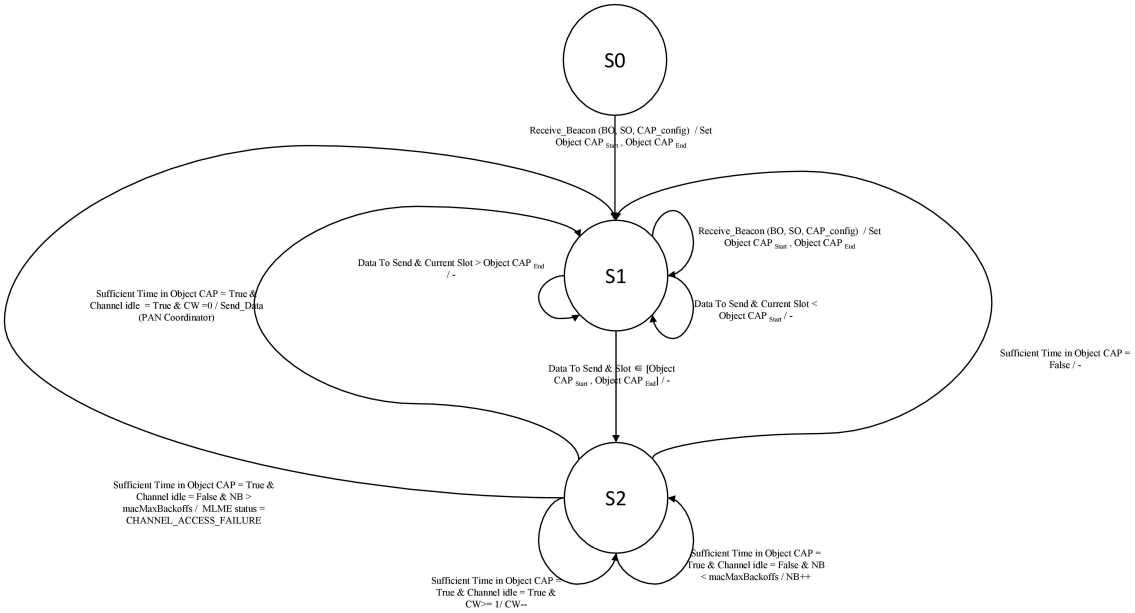

Figure 10. Finite state machine of an IoT object QBAIoT process.

We present in Figure 11 our adaptation of the slotted CSMA/CA process for IoT objects implementing QBAIoT. After receiving the beacon frame, the node (i.e., IoT object) synchronize the superframe duration with the coordinator (i.e., LL-Gw) by setting the values of BO, SO, its QoS CAP duration and the different values of CSMA/CA process (e.g., CW, NB, BE). Upon the creation of a data packet to be sent to the coordinator, the node has to check whether the current slot of the superframe is in its QoS CAP duration. If the number of the current slot of the superframe is greater than the number of the last slot of the QoS CAP duration, the node should wait the QoS CAP duration in the next superframe as its QoS CAP had ended. If the number of the current slot of the superframe is less than the number of the first slot of the QoS CAP duration, the node should wait until the start of its QoS CAP duration in the actual Superframe. Whereas, if the current slot is greater or equal than the first slot of the QoS CAP and less or equal than the last slot of the QoS CAP, the current slot is in the QoS CAP duration of the node and the CSMA/CA process is executed. When the current slot of the superframe is in the QoS CAP duration of the node, the remaining time in the QoS CAP is tested. If there is no sufficient time to send the data and receive the acknowledgement, the node should wait until the next QoS CAP in the next superframe. If there is sufficient time, the node determines the boundary of the next backoff period (each slot is divided into several backoff periods) as any operation should start at the boundary of a backoff period and cannot start during a backoff period. Then, the node observes a delay (random value between 0 and $(2^{\mathrm{BE}} - 1)$) before performing the first CCA test.

On one hand, if the channel is not idle, the NB value is incremented, CW value is reset to the initial value, and BE value allowing the computation of the delay is changed. If the NB value is greater than the maximum value, then a failure occurs. If NB value is greater than the maximum value, the process restarts from the beginning; the node check once again whether the current slot of the superframe is in its QoS CAP duration. In the other hand, if the first CCA indicated an idle channel, CW is decremented and another CCA is performed if the CW had not reached the 0 value yet. When the CW value is equal to 0, the node can send the data to the coordinator.

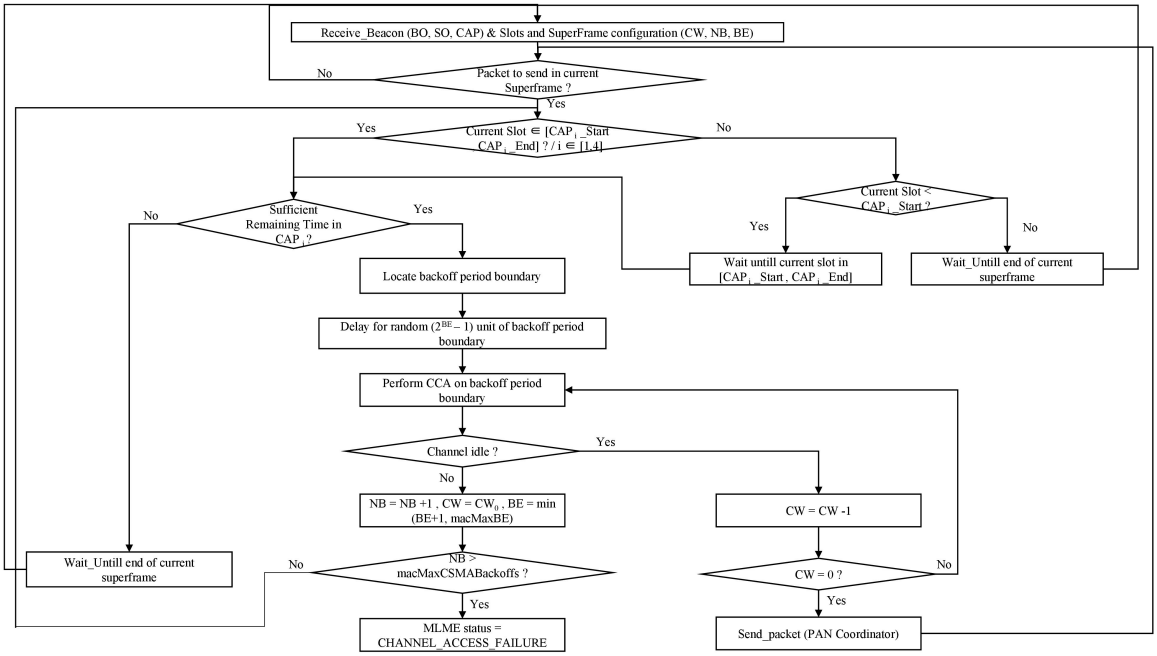

Figure 11. Algorithm schema of the adapted CSMA/CA process for QBAIoT nodes.

3.5. Self-Configuring QBAIoT

The existence of traffic on the network influences the chosen values for BO, SO and the slots configurations of the QoS CAPs as mentioned in Section 3.4. The characteristics of the underlying IoT environment in terms of traffics types' existence have an impact on the configurations of the different QoS CAPs. We specify in Table 1 different scenarios of traffic existence along with the corresponding slots configuration and BO/SO values. Table 1 represents a part of the knowledge base of our self-configuring QBAIoT method (see Figure 12). In this context, we consider that QoS CAP1 is allocated to the QoS class with the highest priority in the IoT infrastructure, whereas QoS CAP4 is allocated to the QoS class with the lowest priority if there are four QoS classes in the considered IoT environment. real-time class (RTC) includes the traffic of RTMC and RTNMC classes, whereas non-real-time class (NRTC) includes the traffic of streaming and NRT classes. The priority of the different available IoT traffics in descendant order is as follows: RTMC, RTNMC, Streaming, NRT. For example, let us consider the existence of two traffics belonging to two QoS classes: 1 RTC and 1 NRTC. According to Table 1, QoS CAP1 is allocated 12 slots and corresponds to the highest priority (i.e., RTC traffic) and QoS CAP2 is allocated four slots and corresponds to the NRTC traffic.

Table 1. Available configurations in the knowledge base.

Traffic Existence	BO/SO	Slot Duration	QoS CAP1	QoS CAP2	QoS CAP3	QoS CAP4
1 RTC or 1 NRTC	14	15.72 s	16	N/A	N/A	N/A
2 NRTC	3	7.68 ms	13	3	N/A	N/A
2 RTC	2	3.84 ms	9	7	N/A	N/A
1 RTC and 1 NRTC	2	3.84 ms	12	4	N/A	N/A
1 RTC and 2 NRTC	2	3.84 ms	8	5	3	N/A
2 RTC and 1 NRTC	2	3.84 ms	7	6	3	N/A
2 RTC and 2 NRTC	2	3.84 ms	6	5	3	2

Based on the different gSLAs stored on the HL-Gw, the number of QoS classes within a LL-Gw can be deducted. Therefore, the corresponding slots configuration of the Superframe is retrieved by the HL-Gw thanks to the knowledge base of QBAIoT that considers different available scenarios. The different available configurations in the knowledge base are the initial configurations to be set on the LL-Gw. These initial configurations could be adapted according to QoS CAPs usage by IoT objects

through a self-optimizing function. If the slots allocated to a specific QoS class are not used efficiently, the self-optimizing function allows reallocating unused slots to another QoS CAP in need of slots while serving the higher priority classes first. In case of changes occurrence concerning the existence of QoS classes (iSLAs removal or addition), an update of BO and SO values as well as slots configuration should be performed. The updated values of BO, SO, and slots configuration are communicated to the LL-Gw that sends an updated beacon at the BI to all IoT nodes. Considering the IoT environment changes in an autonomic manner through the detection of iSLAs evolution on the HL-Gw in order to adapt the beacon configurations corresponds to the QBAIoT self-configuring capability.

QBAIoT self-configuring process (i.e., MAPE-K control loop) is implemented on the HL-Gw and the LL-Gw of our proposed IoT architecture (see Figure 12). The HL-Gw corresponds to the AM (autonomic manager) that enables four functionalities (i.e., monitor, analyze, plan, and execute) along with the knowledge base. The LL-Gw contains an effector interface that receives the configurations sent by the HL-Gw in order to update beacon's information. The LL-Gw sensor interface is not used for the self-configuring capability but for a self-optimizing capability enabling slots usage optimization.

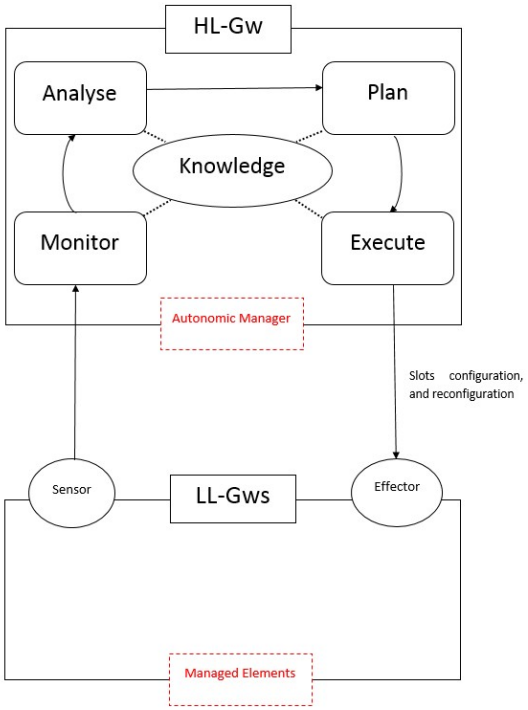

Figure 12. QBAIoT MAPE-K control loop implementation.

The HL-Gw monitors the changes of the IoT environment by considering the different gSLAs stored in the knowledge base. For each LL-Gw, the monitor function groups the corresponding gSLAs into different bundles. Each bundle is specific to a LL-Gw and includes all the gSLAs of the IoT services available through that LL-Gw. Consequently, the characteristics of the LL-Gw environment are available in the bundle. The analyze function counts the number of QoS classes (Nb_QoS_Classes) and counts the number of RTC (RT_Classes) in each bundle. The analyze function deducts also the number of NRTC (NRT_Classes). Then, the HL-Gw Plan function searches in the knowledge base the corresponding entry to the number of RTC and NRTC retrieved by the analyze function. Finally, the Execute function extracts the values from the knowledge base entry and sends them to the effector interface on the LL-Gw. The effector updates the configuration values to be sent in the beacon. This control loop is called at each time the IoT-SP makes changes concerning the IoT environment. If an iSLA is added, removed or modified, the self-configuring control loop is executed in order to adapt the IoT LL-Gw configuration according to IoT environment changes without human intervention.

Figure 13 shows the detailed self-configuring process deployed on the HL-Gw along with the different MAPE-K functions. The HL-Gw monitors the environment changes and updates the values of Nb_QoS_Classes and RT_Classes. When Nb_QoS_Classes is equal to zero, the IoT-SP has no gSLA concerning the LL-Gw. In this case, no QoS classes are available on the LL-Gw environment and there is no need for configurations. If Nb_QoS_Classes is equal to 1, a single QoS class exists within the LL-Gw environment, then BO and SO are set to 14, and the single QoS CAP that exists in the superframe is allocated the 16 slots. If there is more than a single QoS class, Nb_QoS_Classes is greater than 1 and RT_Classes is equal to 0 then there are two classes that are NRTC. Consequently, the BO and SO values are set to 3 and the Superframe includes two QoS CAPs. The first QoS CAP, corresponding to the streaming traffic, is allocated 13 slots. Whereas, the second QoS CAP, corresponding to the NRT traffic, is allocated 3 slots. If Nb_QoS_Classes is greater than 1 and RT_Classes is equal to or greater than 1 then a nested loop is to be browsed in order to allow the HL-Gw to choose the corresponding slots configuration for the considered IoT environment.

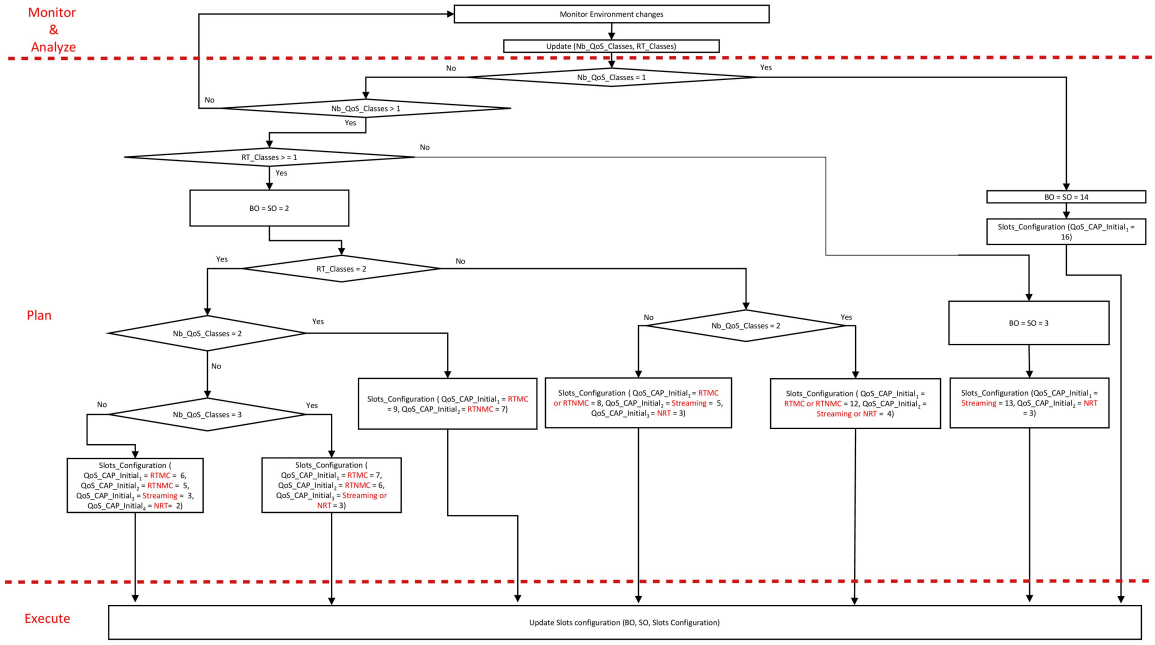

Figure 13. Self-configuring process on HL-Gw.

4. QBAIoT Performance Evaluation

We describe in the following the performance evaluation details of our proposed QoS based wireless access method as well as the additional self-configuring capability enhancement. We use OMNeT++ to implement and simulate our QBAIoT access method according to several scenarios in order to evaluate the performance of our proposal. To do so, we adapt an IEEE 802.15.4 model available for OMNeT++ [32]. The adaptation consists not only in removing the CFP and inactive parts of the standard superframe but also in creating different QoS CAPs within the same superframe. We conduct various simulation scenarios grouped into different set of simulations. The simulations common parameters for all sets are specified in Table 2.

Table 2. Common simulation parameters.

Parameter	Value
Simulation Time	100 s
Mac Payload Size	50 Bytes
Topology	Star
Number of coordinators per WPAN	1

To evaluate the performance of QBAIoT according to the specified scenarios in the different sets, we take into consideration the QoS parameters included in the iSLA specified in Section 3.3. First, the average delay refers to the average time experienced by a generated packet from the IoT object to be received by the LL-Gw. It is computed by dividing the sum of received packets delays by the number of received packets. Then, the packet delivery ratio (PDR) evaluates the reliability degree achieved by the QBAIoT based sensing layer in terms of successful transmissions. It is computed by dividing the number of received packet by the number of generated packets.

In the first set of simulations, we consider seven different scenarios described in Table 3 that shows the number of objects per QoS class in each scenario with the corresponding slots configuration. For all scenarios, the objects data generation interval time is fixed to 0.25 s resulting in 400 generated packets for each object during the simulation. Scenarios 1 to 4 take into consideration objects that generate the data at the same time, whereas Scenarios 5 to 7 take into consideration objects that do not generate the data at the same time. The first scenario corresponds to an IoT e-health service with three RTMC nodes resulting in allocating the 16 slots to the three nodes. The second scenario corresponds to an IoT e-health service requiring three RTMC nodes along with three RTNMC nodes. Based on the knowledge base, nine slots are allocated to RTMC nodes, and seven to RTNMC nodes. The third scenario corresponds to nine nodes (three RTMC, three RTNMC, and three streaming) resulting in 7/6/3 respectively for the slots configuration. The last scenario corresponds to the existence of 12 nodes (3 of each QoS class) resulting in a slots configuration of 6/5/3/2.

Table 3. Number of objects per QoS class in each scenario of the first set.

Scenario	RTMC Objects	RTNMC Objects	Streaming Objects	NRT Objects	Slots Configuration
1	3	0	0	0	16
2	3	3	0	0	9/7
3	3	3	3	0	7/6/3
4	3	3	3	3	6/5/3/2
5	4	4	4	4	6/5/3/2
6	5	5	5	5	6/5/3/2
7	6	6	6	6	6/5/3/2

Figures 14–17 show the performance evaluation comparison between our QBAIoT approach and the IEEE 802.15.4 standard according to Scenarios 1, 2, 3 and 4 correspondingly. Figure 14 presents the delay and PDR evaluation corresponding to scenario 1 with only one QoS class traffic (RTMC). We can observe that for one QoS class, QBAIoT acts like the traditional slotted CSMA/CA as one QoS class is allocated the totality of the slots. Indeed, the obtained results in terms of average delay and PDR are very close for the two approaches. In our approach, we can observe that we have a difference of 7 ms resulting from testing if the current slot is in the QoS CAP, which is not necessary in the traditional approach.

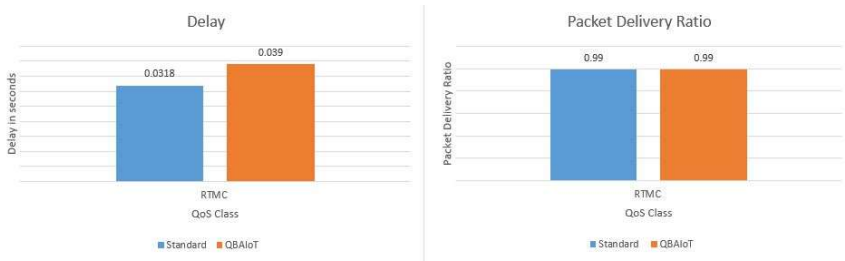

Figure 14. Delay and PDR evaluation for Scenario 1.

The obtained results in Figure 15 show that for 2 QoS classes, QBAIoT ensures a better delay for RTMC and an identical delay for RTNMC comparing to the standard IEEE 802.15.4 access method. Allocating two more slots for RTMC compared to RTNMC results in 10 ms better average delay for this

QoS class. For the PDR evaluation, QBAIoT ensures better values for RTMC (99%) and RTNMC (98%) than the standard approach (56% and 55%, respectively). Our approach enables these performance results in terms of PDR, as only objects of the corresponding QoS class can compete to access the channel during the slots of a QoS CAP. Thus, our class-based access will avoid collisions between different traffics generated by objects belonging to different QoS Classes.

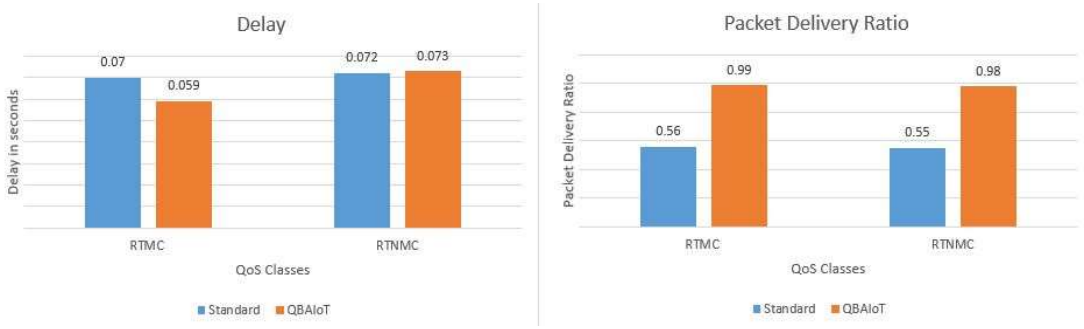

Figure 15. Delay and PDR evaluation for Scenario 2.

Figure 16 shows the evaluation performance corresponding to Scenario 3. The obtained results shows a better delay, while using QBAIoT, for RTMC and RTNMC traffic compared to the standard. With our QoS based access method, seven slots are allocated to the RTMC traffic and six for RTNMC traffic, resulting in better delays: 33 ms less than the standard for RTMC and 16 ms less as for RTNMC. Similarly, we obtain with our approach a better PDR for this scenario, as the number of collisions is lower. Indeed, the PDR for the RTMC traffic with QBAIoT is equal to 98.5% whereas it is equal to 26% with the IEEE 802.15.4.

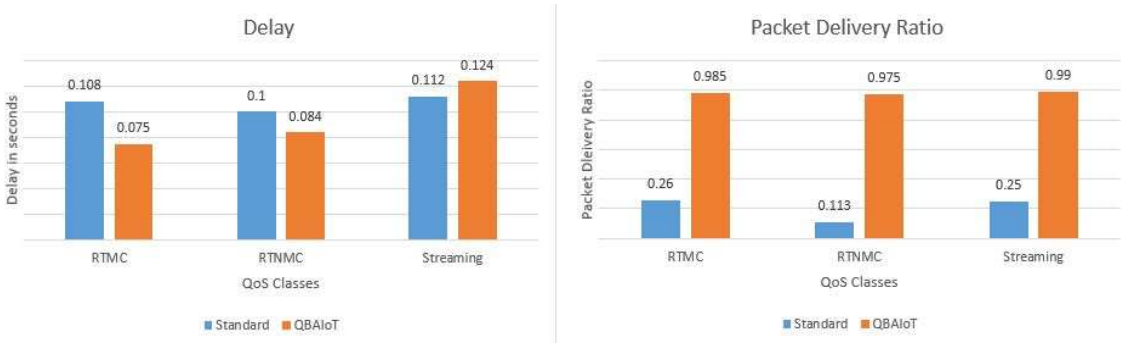

Figure 16. Delay and PDR evaluation for Scenario 3.

The obtained performance results in terms of PDR and delay corresponding to Scenario 4, where the LL-Gw configures four QoS classes, are illustrated in Figure 17. In this scenario, we observe that QBAIoT enables better delays for RTMC (90 ms) and RTNMC (106 ms) traffics compared to the IEEE 802.15.4 standard access method (115 ms for RTMC and 123 ms for RTNMC). Moreover, we observe better PDR for all QoS classes (>96% for three classes) using our approach compared to the standard (<20% for all classes). Indeed, for the IEEE traditional approach, all QoS classes will be served similarly and in the same time, resulting in more collisions and approximatively four slots for each QoS class; whereas our QBAIoT access method enables different QoS CAPs to minimize collisions and configures RTMC with six slots and RTNMC with five slots in this scenario. Finally, we observe an important delay for non-real-time traffic due to the configuration of only two slots for this class, which is not sensitive to the QoS delay parameter. In the other hand, we obtain for NRT traffic a better PDR (26%) than the traditional approach (16%).

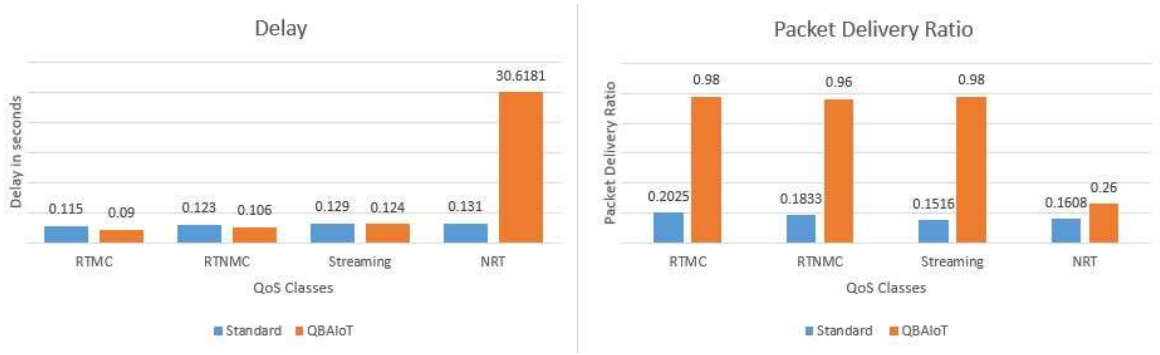

Figure 17. Delay and PDR evaluation for Scenario 4.

Figure 18 shows the average delay of RTMC and RTNMC packets for Scenarios 5 to 7. In these scenarios, we increment the number of objects per QoS CAP (i.e., per QoS class) while not sending data at the same time by the objects of the same class. This difference allows minimizing the number of possible collisions in each QoS CAP and therefore serving the packets faster. We can note that with four and five objects per QoS CAP (see Figure 18), QBAIoT, using a single coordinator, is capable of delivering RTMC and RTNMC packets while respecting the real-time traffic requirements. In Scenario 7, where six objects per QoS CAP exist, QBAIoT with a single coordinator is not sufficient to respect the requirements of real-time traffics. In fact, with six objects or more per QoS CAP in the context of these simulations, the number of objects competing during each QoS CAPs induces an additional delay before serving the data packets. Thanks to our proposed IoT architecture, we are capable of coping with this problem by implementing a new QBAIoT LL-Gw (i.e., coordinator) in order to extend the capacity of the IoT environment and respect the requirements of an increasing number of QoS classes corresponding to the subscribed iSLAs.

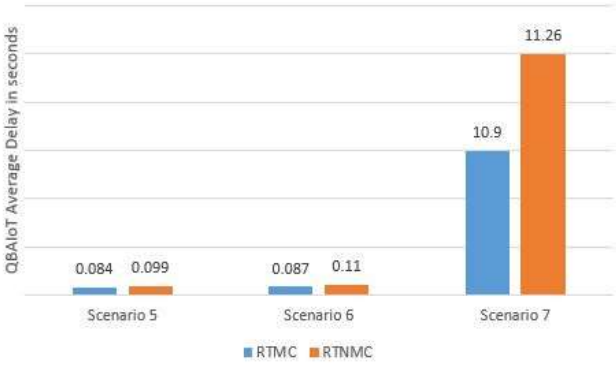

Figure 18. QBAIoT average delay for RTMC and RTNMC traffics.

The second set of simulations allows evaluating the self-configuring capability of our QoS based access method. Indeed, we evaluate the impact of providing different superframe slots configurations according to the characteristics of the IoT environment. Furthermore, self-configuring QBAIoT enables different beacon configurations for the same number of QoS classes according to the number of RTC classes.

For the second set of simulations, we use an IoT environment including three different QoS classes for the first two scenarios. Every QoS class includes three nodes with a data generation interval of 0.25 s per node. Different configurations are available in Table 1 depending on the number of RTC among the three QoS classes of the considered simulation environment. Consequently, two scenarios are presented. Scenario 1 corresponds to the existence of two RTC and one NRTC traffics, whereas Scenario 2 corresponds to the existence of one RTC and two NRTC traffics. In Figure 19, we compare the average delay of each traffic corresponding to these two scenarios while using QBAIoT and the

IEEE 802.15.4 slotted CSMA/CA. The obtained results show that QBAIoT offers better delays for QoS CAP1 and QoS CAP2 traffics, whereas, QoS CAP3 traffic (delay tolerant traffic: streaming or NRT traffic) experiences a more important delay. Indeed, in Scenario 1, 2 RTC traffics are available within the IoT environment. Therefore, the two traffics should observe a minimal delay while prioritizing QoS CAP1 traffic. The corresponding slots configuration offers an additional slot to QoS CAP1 and allows respecting the interactive characteristic of the two RTC real-time traffics (a delay of 69 ms for QoS CAP1 traffic and a delay of 70 ms delay for QoS CAP2). Scenario 2 includes a single RTC traffic. Consequently, the delay QoS parameter is not critical for QoS CAP2 and QoS CAP3. In this context, we provide QoS CAP1 with eight slots enabling to further minimize the corresponding delay. As a result, we obtain a delay of 58 ms for QoS CAP1 comparing to the 69 ms delay in Scenario 1. The obtained results show that QBAIoT performs better than the IEEE 802.15.4 standard thanks to specific configurations enabling achieving the requirements of different traffic types in various scenarios depending on the environment characteristics.

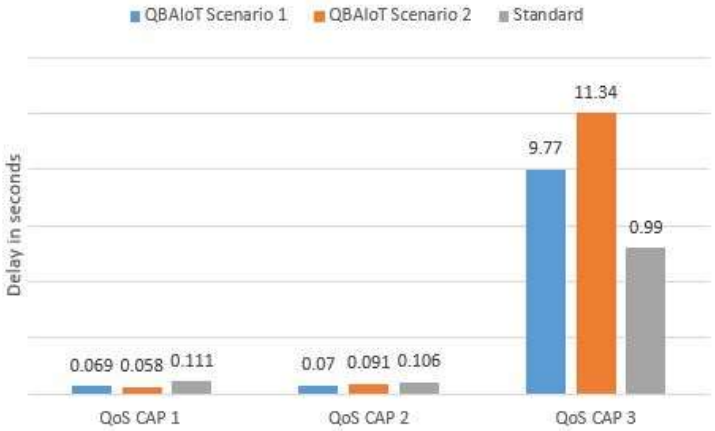

Figure 19. Delay comparison for QBAIoT and IEEE 802.15.4 with different configurations.

We consider a third scenario in the second set of simulations. In this context, the IoT environment contains four QoS classes initially. After a period of time, the IoT-SP deletes gSLAs from the HL-Gw due to the cancellation of two iSLAs and therefore, only two RTC traffics remain in the LL-Gw environment. Initially, every QoS class includes three nodes. After 15 s of the simulation beginning, the gSLA concerning the Streaming and NRT traffics are removed and their corresponding traffics are stopped.

We compare in Figure 20 the average delay while using or not QBAIoT self-configuring capability for the highly critical data (RTMC and RTNMC). In this scenario, without QBAIoT self-configuring capability, the HL-Gw is not able to change the initial configurations without human intervention. Consequently, the slots allocated to streaming and NRT traffics remain not used during every superframe after deleting the corresponding gSLAs. Thus, for each superframe, five slots are lost resulting in a usage of 68% (11/16 slots) of the superframe slots. The obtained results illustrated in Figure 20 show that, with a self-configuring QBAIoT enabling the usage of these lost slots due to environment changes, the delay of RTMC and RTNMC traffics are improved in a way to respond better to real-time traffic requirements. Furthermore, the QBAIoT self-configuring capability enhancement enables better superframe slots usage. Figure 21 shows the obtained results concerning the PDR performance parameter for RTMC and RTNMC traffics while using QBAIoT with and without self-configuring capability according to the same Scenario 3. We can notice that with the self-configuring capability, the slots of the streaming and NRT traffics are reallocated to the remaining QoS classes immediately. Consequently, the PDRs of RTMC and RTNMC traffics undergo a small improvement as more slots are allocated to their QoS classes.

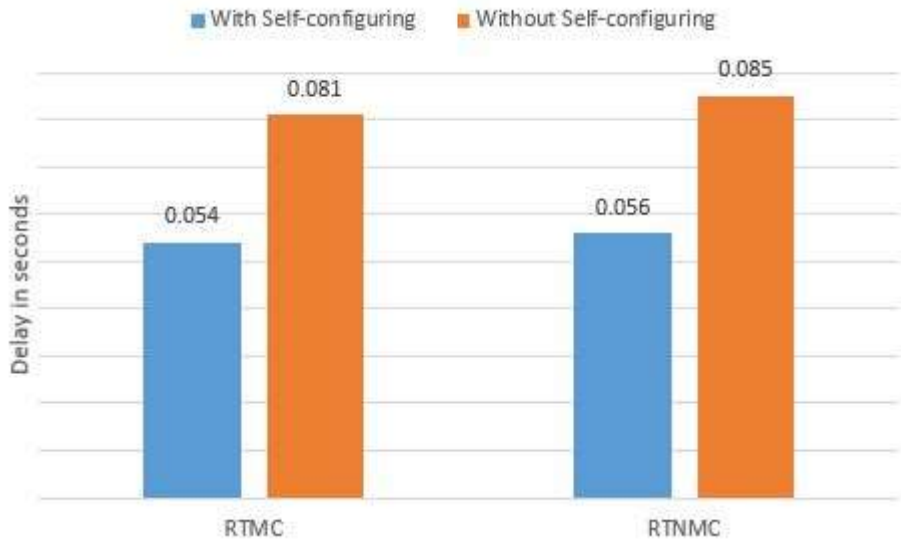

Figure 20. Delay evaluation using QBAIoT with and without self-configuring.

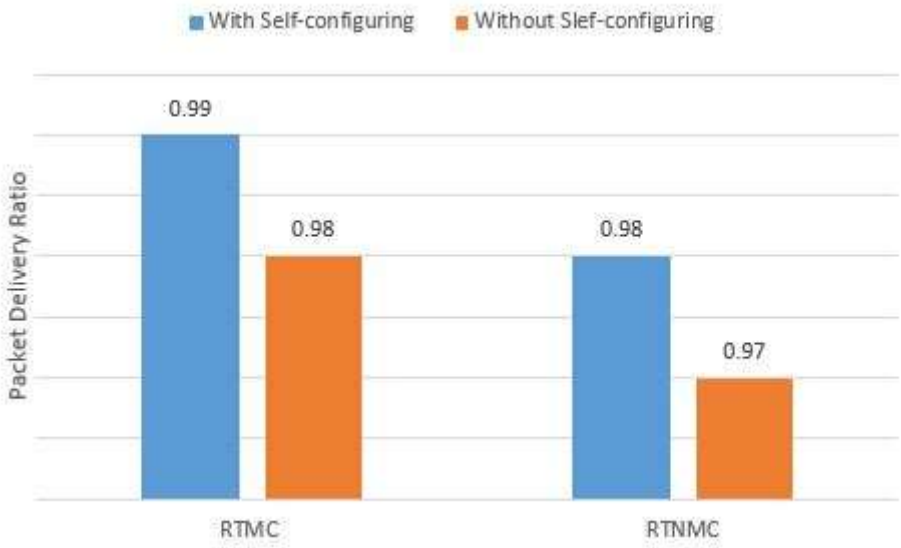

Figure 21. PDR evaluation using QBAIoT with and without self-configuring.

After comparing the performance of QBAIoT with the standard slotted CSMA/CA according to multiple scenarios and evaluating the importance of the self-configuring capability, it is crucial to compare the performance of our proposed method with other QoS based access methods such as SDA-CSMA/CA [30]. SDA-CSMA/CA and QBAIoT aims to offer a certain service level in an IEEE 802.15.4 environment using the slotted CSMA/CA process for accessing the channel. Each approach has its own parameters and is based on a certain configuration for BO/SO. For instance, SDA-CSMA/CA uses an SO equal to 7 and a BO equal to 8. The SDA-CSMA/CA approach does not depend on Superframe Duration, unlike QBAIoT where BO and SO are set to 2. For the comparison, we consider the same environment characteristics as for the simulations conducted for SDA-CSMA/CA. The latter is limited to three QoS classes, therefore we specify only RTMC, RTNMC, and streaming QoS classes when we compare QBAIoT to SDA-CSMA/CA. Concerning the slots configuration, we consider the 7/6/3 pattern. Finally, for the data generation interval, we use an interval equal to 0.25 s. The SDA-CSMA/CA and QBAIoT comparison is conducted while using two then three objects per QoS class. The results are shown in Figure 22.

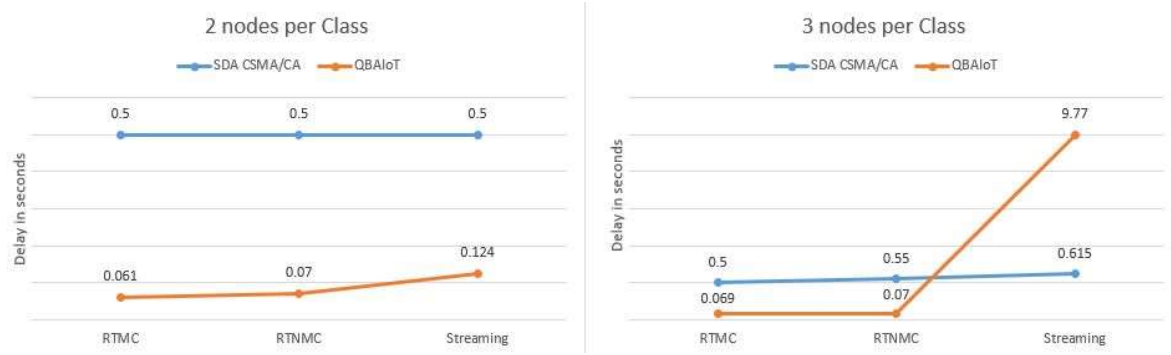

Figure 22. Delay evaluation comparison using QBAIoT and SDA CSMA/CA.

Figure 22 demonstrates that QBAIoT offers better delays for all QoS classes while considering two nodes per QoS class. Furthermore, when we consider three nodes per QoS class, the delay experienced by critical traffic (RTMC and RTNMC) is better while using QBAIoT compared to SDA-CSMA/CA. As for the streaming traffic, we observe a more important delay with QBAIoT due to the minimal slots allocation but without any QoS degradation because the delay QoS parameter is not critical for such traffic.

5. Conclusions

Recent research works in the field of the IoT are trying to optimize the services provided while satisfying a certain service level dependent on the class of the traffic. Therefore, IoT services and in particular e-health applications include different types of sensors and connected objects that create different types of data with different QoS requirements. Consequently, a prioritized processing for highly critical real-time data is necessary. Hence, we presented a QoS based access approach to minimize the delay for this type of traffic while giving better packet delivery ratio for all traffic types. Therefore, we proposed the QBAIoT access method as an enhancement of the IEEE 802.15.4 slotted CSMA/CA mechanism. In our proposal, four QoS classes are taken into consideration in order to configure the corresponding QoS CAPs within an adapted superframe structure. The QoS based access method is deployed within the sensing layer gateways and objects of our proposed IoT architecture in order to guarantee QoS requirements specified in the iSLAs. In addition, providing the IoT environment with a self-configuring capability allows minimizing the operational expenditure and improving the performance of our QoS based access method. Providing the sensing layer with a self-configuring capability allows responding better to environment changes. Through this research work, we presented the self-configuring process for our QoS based wireless access method. This self-management capability provides the LL-GW and IoT objects with specific configurations autonomously without human intervention, in order to adapt to existing traffic changes. We evaluated our self-configuring capability and compared our proposed access method to the IEEE 802.15.4 standard and the SDA-CSMA/CA approach. The conducted simulations showed that we obtain better results while using QBAIoT: reduced delay for real-time traffic as well as a greater PDR for all QoS classes. When incrementing the number of IoT objects in each QoS CAP, QBAIoT can use additional coordinators (i.e., LL-Gw) in order to extend the capacity of the IoT environment and respect the requirements of subscribed iSLAs.

As ongoing work, we are trying to minimize the overall energy consumption of the QBAIoT system due to the elimination of the inactive period. We are implementing an algorithm that classifies the different sensors based on different metrics. This classification allows choosing for each IoT application a set of sensors to be used while the others will be in a sleeping mode in order to minimize the energy consumption and to maximize the system lifetime.

Author Contributions: Conceptualization, A.K., N.M. and O.T.; Methodology, A.K., N.M. and O.T.; Validation, A.K., N.M. and O.T.; Writing—original draft, A.K., N.M. and O.T.

Funding: This research was funded by the Conseil Régional de Bourgogne Franche Comté through the "plan d'actions régional pour l'innovation (PARI)" and the European Union through the "PO FEDER-FSE Bourgogne 2014/2020 programs".

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Manyika, J.; Chui, M.; Bisson, P.; Woetzel, J.; Dobbs, R.; Bughin, J.; Aharon, D.; McKinsey & Company, McKinsey Global Institute. The Internet of Things Mapping the Value Beyond the Hype. Available online: https://www.mckinsey.com/~/media/McKinsey/Business%20Functions/McKinsey%20Digital/Our%20Insights/The%20Internet%20of%20Things%20The%20value%20of%20digitizing%20the%20physical%20world/The-Internet-of-things-Mapping-the-value-beyond-the-hype.ashx (accessed on 17 October 2018).
2. Nordrum, A.; IEEE Spectrum. Popular Internet of Things Forecast of 50 Billion Devices by 2020 is Outdated. Available online: https://spectrum.ieee.org/tech-talk/telecom/internet/popular-internet-of-things-forecast-of-50-billion-devices-by-2020-is-outdated (accessed on 17 October 2018).
3. Scarpato, N.; Pieroni, A.; Nunzio, L.D.; Fallucchi, F. E-health-IoT universe: A. review. *Int. J. Adv. Sci. Eng. Inf. Technol.* **2107**, *7*, 2328–2336. [CrossRef]
4. Guadagni, F.; Scarpato, N.; Patrizia, F.; D'Ottavi, G.; Boavida, F.; Roselli, M.; Garrisi, G.; Lisi, A. Personal and Sensitive Data in the e-Health-IoT Universe. In *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*; Springer: Berlin, Germany, 2016; Volume 170.
5. Khalil, A.; Mbarek, N.; Togni, O. IoT Service QoS Guarantee Using QBAIoT Wireless Access Method. In Proceedings of the 4th International Conference on Mobile, Secure and Programmable Networking (MSPN 2018), Paris, France, 18–20 June 2018; Springer LNCS: Berlin, Germany, 2018 (in press).
6. International Telecommunication Union (ITU-T). *Overview of the Internet of Things*; Y.2060; ITU-T: Geneva, Switzerland, 2012.
7. Minerva, R.; Biru, A.; Rotondi, D. *Towards a Definition of the Internet of Things (IoT)*; IEEE: Piscataway, NJ, USA, 2015.
8. International Organization for Standardization and International Electrotechnical Commission (ISO/IEC JTC 1). *Internet of Things (IoT) Preliminary Report 2014*; ISO/IEC: Geneva, Switzerland, 2015.
9. International Electrotechnical Commission. *IEC Role in the IoT*; IEC: Geneva, Switzerland, 2017.
10. Mell, P.; Grance, T. The NIST Definition of Cloud Computing; NIST; 2009. Available online: https://www.nist.gov/sites/default/files/documents/itl/cloud/cloud-def-v15.pdf. (accessed on 15 October 2018).
11. Banafa, A. Definition of Fog Computing; IBM; 2014. Available online: https://www.ibm.com/blogs/cloud-computing/2014/08/fog-computing (accessed on 17 March 2018).
12. International Telecommunication Union. Question 28/16—Multimedia Framework for E-Health Applications. Available online: http://www.itu.int/ITU-T/studygroups/com16/sg16-q28.html (accessed on 17 March 2018).
13. 4G Americas. Cellular Technologies Enabling the Internet of Things. 2015. Available online: http://www.5gamericas.org/files/6014/4683/4670/4G_Americas_Cellular_Technologies_Enabling_the_IoT_White_Paper_-_November_2015.pdf (accessed on 17 October 2018).
14. IEEE Standard for Local and Metropolitan Area Networks. *Low-Rate Wireless Personal Area Networks*; IEEE Computer Society: Washington, DC, USA, 2016.
15. Lora Alliance. A Technical Overview of LoRa® and LoRaWAN™. Available online: https://www.tuv.com/media/corporate/products_1/electronic_components_and_lasers/TUeV_Rheinland_Overview_LoRa_and_LoRaWANtmp.pdf (accessed on 17 October 2018).
16. Nath, S.; Aznabi, S.; Islam, N.; Faridi, A.; Qarony, W. Investigation and performance analysis of some implemented features of the ZigBee protocol and IEEE 802.15.4 mac specification. *Int. J. Online Eng.* **2017**, *13*. [CrossRef]
17. Thubert, P.; Bormann, C.; Toutain, L.; Cragie, R. *IPv6 over Low-Power Wireless Personal Area Network (6LoWPAN) Routing Header*; IETF RFC: Fremont, CA, USA, 2017.
18. AlSharqi, K.; Abdelbari, A.; Abou-Elnour, A.; Tarique, M. Zigbee based wearable remote healthcare monitoring system for elderly patients. *Int. J. Wirel. Mob. Netw.* **2014**, *6*, 53–67. [CrossRef]

19. Fernandez-Lopez, H.; Afonso, J.A.; Coreeia, J.H.; Simoes, R. ZigBee-based remote patient monitoring. *Stud. Health Technol. Inform.* **2012**, *177*. [CrossRef]
20. International Telecommunication Union (ITU-T). *Y.2066: Next Generation Networks—Frameworks and Functional Architecture Models*; ITU-T: Geneva, Switzerland, 2014.
21. Dobson, S.; Denazis, S.; Fernández, A.; Gaïti, D.; Gelenbe, E.; Massacci, F.; Nixon, P.; Saffre, F.; Schmidt, N.; Zambonelli, F. A Survey of autonomic communications. *ACM Trans. Auton. Adapt. Syst.* **2006**, *1*, 223–259. [CrossRef]
22. Serrano, M. OpenIoT Consortium, 2014. OpenIoT D.4.6 Quality of Service (QoS) for IoT Services, Project Number 287305. Available online: https://cordis.europa.eu/docs/projects/cnect/5/287305/080/deliverables/001-OpenIoTD46Draft.pdf (accessed on 17 October 2018).
23. Chatzigiannakis, I.; Hasemann, H.; Karnstedt, M.; Kleine, O.; Kroller, A.; Leggieri, M.; Pfisterer, D.; Romer, K.; Truong, C. True self-configuration for the IoT. In Proceedings of the 3th IEEE International Conference on the Internet of Things, Wuxi, China, 24–26 October 2012. [CrossRef]
24. Bai, D.P.; Rabara, A.; Jerald, V. Quality of service architecture for internet of things and cloud computing. *Int. J. Comput. Appl.* **2015**, *128*, 23–28. [CrossRef]
25. Gubbi, J.; Buyya, R.; Marusic, S.; Palaniswami, M. Internet of things (IoT): A vision, architectural elements, and future directions. *Future Gener. Comput. Syst.* **2013**, *29*, 1645–1660. [CrossRef]
26. Bröse, E.; Ngwu, C. F-Interop Consortium. Deliverable D3.2 Performance Test Tools 1st Iteration. Project Number 687884. Available online: https://www.f-interop.eu/images/deliverables/F-Interop_Deliverable_D3.2_v0.5.pdf (accessed on 17 October 2018).
27. The French National Research Agency Website, Quasimodo. Available online: http://www.agence-nationale-recherche.fr/Project-ANR-10-INTB-0206 (accessed on 31 October 2018).
28. Duan, R.; Chen, X.; Xing, T. A QoS architecture for IOT. In Proceedings of the International Conference on Internet of Things and Forth International Conference on Cyber, Physical and Social Computing, Dalian, China, 19–22 October 2011. [CrossRef]
29. Ezdiani, S.; Acharyya, I.; Sivakumar, S.; Al-Anbuky, A. An IoT environment for WSN adaptive QoS. In Proceedings of the 2015 IEEE International Conference on Data Science and Data Intensive Systems (DSDIS 2015), Sydney, Australia, 11–13 December 2015; ISBN 978-1-5090-0214-6. [CrossRef]
30. Xia, F.; Li, J.; Hao, R.; Kong, X.; Gao, R. Service differentiated and adaptive CSMA/CA over IEEE 802.15.4 for cyber-physical systems. *Sci. World J.* **2013**, *2013*. [CrossRef] [PubMed]
31. Khalil, A.; Mbarek, N.; Togni, O. Service level guarantee framework for IoT environments. In Proceedings of the First International Conference on Internet of Things and Machine Learning, Liverpool, UK, 17–18 October 2017; ISBN 978-1-4503-5243-7. [CrossRef]
32. Kirsche, M. IEEE 802.15.4-Standalone. Available online: https://github.com/michaelkirsche/IEEE802154INET-Standalone (accessed on 17 March 2018).