**EAI Endorsed Transactions**
**on Internet of Things**

Research Article **EAI.EU**

# Statistical Analysis of a Distributed Queuing Random Access Protocol in a Massive Communication Environment

Romeo Nibitanga[1], Elijah Mwangi[2] and Edward Ndung'u[3]

[1]Pan African University Institute of Basic Sciences, Technology and Innovation, Kenya
[2]University of Nairobi, Kenya
[3]Jomo Kenyatta University of Agriculture and Technology, Kenya

## Abstract

Most of the networks deployed for massive IoT communications use Aloha-based algorithms for channel access. However, those algorithms are known to be unstable and inefficient when the network size is high. Since recently, a Distributed Queuing (DQ) algorithm is being proposed as a solution to mitigate several of the Aloha issues in IoT networks. In this paper, a statistical performance analysis of the DQ algorithm without any prior consideration of any physical layer is presented. We evaluate the DQ algorithm in a massive communication environment and give the average values for these performance metrics: collision resolution time, access delay per sensor, channel throughput, number of attempts required by a sensor to complete the contention process, number of nodes contending per frame and the distribution of contention slots into idle, successful, and collided. The goal of this paper is to provide a statistical baseline performance evaluation of the DQ algorithm in general.

**Keywords:** Aloha, distributed queuing, collision resolution, massive Machine-to-Machine (M2M) communications, random access protocol, tree splitting

Received on 14 September 2020, accepted on 13 October 2020, published on 16 October 2020

doi: 10.4108/eai.16-10-2020.166663

*Corresponding author. Email: nibitanga.romeo@students.jkuat.ac.ke

## 1. Introduction

It is predicted that 22.3 billion devices will be connected to the Internet of Things (IoT) by 2024 [1]. With the years, that number will grow as more applications are developed, and new services are introduced. To address the massive number of connections that are going to be required to handle these communications, several new technologies have emerged. Currently, Long Term Evolution for Machine (LTE-M), Narrowband Internet of Things (NB-IoT), LoRa/LoRaWAN, and Sigfox are among the proposed solutions for the IoT massive connectivity scenarios. At the media access control layer, all these networks use Aloha or its variants for channel access.

The choice of Aloha-based algorithms is motivated by its easy implementation in low power, inexpensive and small size devices that are intended to dominate those networks. However, as many studies [2]–[5] pointed out, the Aloha-based algorithms are inefficient in terms of energy consumption, throughput, and access delay when the network size is high. Thus, there is a need for new access schemes to address the scalability issue of those networks with a potentially massive number of connections per session.

Recently, as an alternative to the Aloha-based access schemes, several works [2]–[16] proposed a Distributed Queuing (DQ) algorithm as a potential candidate to many of the challenges posed by the massive Machine-to-Machine (M2M) connectivity. In [6], it is predicted as one of the possible solutions to the majority of Media Access Control (MAC) issues in the IoT. In general, the DQ protocol is a tree-splitting based algorithm which tries to resolve an initial contention using different types of virtual queues before trying another one [17]. A DQ frame is divided into three parts: the contention period with a given number of contention slots, the contention-free period with one or multiple data slots, and the feedback period with different parts.

Most of the studies reviewed have been found to put an emphasis on the evaluation of these four performance metrics: the access delay [3]–[5], [9], [10], [12]–[16], the throughput [2]–[6], [15], [16], the energy consumption [2], [3], [5], [8], [14], and the number of attempts per node [3], [4], [10], [11]. Besides, it is a fact that the algorithm outperforms the Aloha-based schemes in terms of those performance metrics. It has also been proven that the algorithm can be implemented in the existing networks and systems [7].

However, most of the results presented in published works are bound to a specific technology because the evaluation of the DQ algorithm is made under the consideration of a given technology at the physical layer. Therefore, they cannot apply to other systems. Moreover, only these performance metrics are of interest in all the works: the access delay, the throughput and the energy consumption. Although very useful in the evaluation of the DQ algorithm, those metrics do not provide any quantitative information about the collision resolution process.

In this paper, we perform a statistical transient analysis of the DQ algorithm in a massive M2M communication environment. By transient analysis, we mean that the DQ algorithm is evaluated from the beginning of the contention process, with a given number of sensors, up to the moment when all the sensors have left the algorithm queues. In contrast to other works, the evaluation is made without any prior consideration of any physical layer technology or any given number of contention slots. Moreover, in addition to the existing ones, we introduce new performance metrics needed to improve the overall performance of the algorithm. The goal of this work is to provide a statistical baseline performance evaluation of the DQ algorithm.

Briefly, the contribution of this paper is two-fold: (i) we provide an extended evaluation of the DQ algorithm without any prior consideration of any physical layer and (ii) we introduce new performance metrics in the analysis of the DQ algorithm.

The rest of this paper is organized as follows. In section II, a literature review of works related to the DQ algorithm in IoT networks and systems is presented. The description of the DQ algorithm is given in Section IV. We present the system model used to evaluate the DQ algorithm and the parameters used for numerical simulations in section V. Further, the simulation results for each of the performance metrics considered and their discussion are presented in section VI. Finally, section VII is dedicated to the conclusion and future work.

## 2. Related Works

In this section, we present a literature review on works related to the DQ algorithm in IoT environments.

### 2.1. Background

In [17], the DQ algorithm was first introduced by Xu and Campbell as the Distributed Queuing Random Access Protocol (DQRAP). The protocol was intended to be used in cable TV systems for the transmission of digital data. In [17], the results showed that the algorithm could achieve a performance level near those of an M/D/1 queuing model in terms of throughput and delay.

Before the advent of the IoT, the algorithm was evaluated under various scenarios. Prioritized Distributed Queuing Random Access Protocol (PDQRAP) [18] and Extended Distributed Queuing Random Access Protocol (XDQRAP) [19] were proposed as variations of the DQRAP for priority-based traffic and to support variable packet lengths respectively. Also, there have been proposals for the adaption of the algorithm to wireless area networks [20], third-generation cellular networks [21], body area networks [22], mobile *ad hoc* networks [23] and cooperative networks [24].

Recently, the DQ algorithm has interested several IoT researchers [2]–[16]. The protocol is seen as a solution to many M2M communication challenges present in both unlicensed and licensed networks.

### 2.2. DQ in data collection systems

The energy aspect of the algorithm was the first to be evaluated. In [8], F. Vazquez et al. analyzed and compared the DQ algorithm to other protocols like the Contention Tree Algorithm (CTA) and the Frame Slotted Aloha (FSA) in terms of energy consumption in M2M area networks. The authors observed that the DQ algorithm could reduce energy consumption by 35% and 80% compared to CTA and FSA, respectively, for networks with 802.15.4 devices.

Additionally, the same researchers designed two protocols based on the DQ algorithm: these were the Low Power Distributed Queuing (LPDQ) [7], and the Energy Harvesting Distributed queuing (EHDQ) [2]. The first protocol was developed for low power wireless networks with bursty traffic, whereas the second was designed for data collection networks with devices equipped with energy harvester. Both protocols were evaluated for Radio Frequency Identification (RFID) networks. Apart from being energy efficient, the results also showed that the two algorithms could achieve better performance in terms of success probability (99%) and time efficiency (EHDQ only) compared to Aloha-based algorithms.

## 2.3. DQ in LTE networks

Laya et al. in [3] proposed the DQ algorithm as an alternative to the Aloha-based access mechanism used in LTE networks for M2M communications. Their results showed that applying the DQ algorithm in the media access layer of the LTE networks would achieve, under specific network configuration, a reduction of 85% and 40% with respect to the delay and the energy consumption. Here again, the algorithm was found to have better performance in terms of success probability compared to the legacy access method.

The authors in [9] developed the Distributed Queuing Algorithm Access for LTE (DQAL). Compared to the DQ variant for LTE from Laya [3], in DQAL, different groups with a collision can re-transmit their preambles in the same random access slot. The results showed that under certain success probability constraints, the DQAL access delay was superior to the standard Extended Baring Access (EAB). Moreover, in [10], the authors gave two analytical models of the Laya DQ protocol for LTE: the maximum number of attempts before a device can get access to the channel and the access delay.

The conventional (sequential) DQ algorithm applied to LTE networks for M2M communications have been found to cause significant access delay [11]–[13]. In [11], instead of resolving each contention in each group sequentially, the author proposed to resolve those contention groups in parallel. Compared to the sequential method, the proposed scheme was shown to reduce the average number of random access slots with the average number of preambles slightly increasing.

Bui et al. proposed the Free Access Distributed Protocol (FADQ) in [12]. First, the algorithm divides the number of colliding devices into a subset of smaller groups and then applies the conventional DQ algorithm to each group. Secondly, the free access method is applied for each newly arrived device. According to the free access method, new devices can participate in the next random access opportunity without waiting for the collision resolution to end. The results showed that the access delay was significantly reduced compared to the 3GPP Access Class Barring (ACB) mode while still maintaining a high access success probability.

A traffic prioritization method in LTE networks with the DQ protocol is introduced in [14]. The authors showed that the length of the CRQ could be used to retrieve relevant information about network congestion. Thus, based on the length of the CRQ, news arrivals can be delayed according to their priority classes. Compared to both the baseline and dynamic ACB, the proposed method was observed to offer excellent performance in terms of access delay and energy consumption independent of the traffic prioritization.

In [13], the authors also proposed to divide the contending devices into a given number of groups and to apply the conventional DQ algorithm to each group in parallel. However, the newly arrived devices were only allowed to contend in the groups without collisions. The method proposed showed a reduction in the overall access delay and a significant optimization in the use of preamble resources compared to the baseline EAB and the DQ variants proposed separately by Laya and Yoon in [3] and [11] respectively.

## 2.4. DQ in low power wide area networks

Currently, the DQ algorithm is also being proposed as a candidate for the media channel access for Low Power Wide Area Networks (LPWAN).

In [4], Xing et al. brought the DQ algorithm to NB-IoT networks. The authors proposed the Resource Grouping Distributed Queuing algorithm (RGDQ) to resolve the massive connectivity issue in NB-IoT. Aware of the inherent significant delay caused by the conventional DQ scheme, a suggestion of grouping devices from each coverage enhancement was adopted. Thus, devices were divided into three different groups, with different sub-carriers shared evenly between groups. Then, the DQ was applied separately to each group. Each group has its CRQ. It was observed that the RGDQ scheme, compared to the standard ACB mechanism, presented a significant improvement in terms of the access probability, the access delay, and the average number of random access attempts.

In [5], the authors proposed to replace the pure Aloha algorithm used in LoRA networks by the DQ algorithm. They introduced the DQ-LoRA, where the Distributed Queuing protocol was used at the media access layer, and the LoRa technology was applied at the physical layer. Their results showed that DQ-LoRA could achieve better performance compared to the current LoRaWAN Aloha-based scheme. A 2.6-fold gain for throughput was obtained, while the energy consumption and the latency were reduced by 48% and 54%, respectively. Besides, it was observed that for better throughput performance, the number of contention slots should be four, while 28 contention slots were needed for minimum energy consumption. Thus, in a LoRa network with the DQ scheme used for channel access, a trade-off has to be made between energy consumption and throughput.

In [15], the authors presented DQ-N for crowd-sourced low power wide area networks. Unlike the conventional DQ algorithm, DQ-N divides the contention-free period into multiple data slots instead of one to support a massive number of nodes with a low data rate. The results showed that the DQ-N outperforms the LPDQ protocol in terms of channel utility and latency.

## 2.5. DQ and MIMO technology

Finally, the DQ algorithm has also been exploited together with the Massive Input Massive Output (MIMO) technology.

In [16], Yuan et al. proposed a Distributed Queuing Random Access Massive Input Massive Output (DQRA MIMO) system to obtain higher throughput under delay constraint and limited time-frequency resources. The basic principle of the system is to send a maximum number of packets, as there are available antennas at the base station during each frame. It was assumed that each device could only transmit one packet per frame. Once optimized and given specific requirements of average delay, the system was found to provide a better throughput performance compared to that of a Hybrid Random Access and Data Transmission Protocol (HRADTP) and the conventional DQ algorithm.

## 2.5. Research gaps

Despite many existing works on the DQ algorithm in a massive connectivity environment, all the proposed performance metrics (throughput, energy consumption, access delay, number of attempts) were evaluated based on a given physical layer. This situation makes the results biased in favour of specific physical technology considered. Moreover, an analysis of the behaviour of the different queues and other performance metrics that may be used to improve the performance of the algorithm was absent. We have also observed that most of the works have only evaluated the DQ algorithm considering only a given number of contention slots.

In this paper, we propose a statistical analysis of the DQ algorithm without any prior consideration of any physical layer. Furthermore, we introduce new performance metrics in addition to the existing ones for an extended analysis of the DQ algorithm. These are the collision resolution and the data transmission times, the interval of time the contention resolution queue reaches the maximum, the number of contending sensors per frame, and the distribution of contention slots into collided, successful and empty.

## 3. Description of the DQ Algorithm

Xu and Campbell first introduced the DQ algorithm in 1992 for use in broadcast channels shared by an infinite number of bursty devices [17]. The algorithm is a tree-splitting based protocol that iteratively resolves one initial collision by dividing a large group of contenders into multiple smaller groups to reduce the collisions.

Sensors are placed in two virtual queues depending on whether they succeeded in placing an access request or they are still contending for channel access. The first queue is the collision resolution queue (CRQ) containing the groups of sensors that have not succeeded in sending their access requests. The second queue is the data transmission queue (DTQ) containing sensors that have successfully resolved their contention and are waiting for their turn for the transmission of their data.

The DQ algorithm operates in a star network topology with a given number of sensors all around a network coordinator. The role of the network coordinator is to broadcast beacon messages containing information about the status of the queues and other DQ operational information. Based on the information received from the network coordinator, the sensors can decide and update the information needed for the DQ operation.

The DQ frame is structured in a manner that the contention period is much less than the data transmission period to optimize the throughput characteristics of the algorithm. The frame is composed of three parts:

(i) The contention period with *m* contention slots; these contention slots are used by sensors to send their access request signals to the network coordinator (uplink channel)
(ii) The data transmission period, which may be divided into one or multiple data slots used to send data to the network coordinator (uplink channel)
(iii) The feedback period, which may be divide also into one or multiple slots and is used by the coordinator to send all the DQ related information and other data from the coordinator to the sensors (downlink channel).

The DQ algorithm can be implemented in time, as well as in the frequency domain [6]. In the time domain, the one considered in this paper, the operation of the DQ algorithm assumes a perfect frame and a contention slot synchronization.

At the beginning of each DQ frame, sensors with data to send and not engaged in any contention process send an access request signal in one of the *m* contention slots. The choice of a given contention slot by a device is random. After the contention period, the status of each of the contention slots may be:

(i) Empty, if no sensors have chosen the contention slot;
(ii) Successfully, if only one sensor has placed an access request signal in the contention slot;
(iii) Collision, if more than one sensor has chosen the same contention slot.

At the end of the transmission period, the coordinator broadcasts the status of each of the contention slots during the feedback period. Based on the received information from the coordinator, sensors that contended in the previous frame can execute the DQ algorithm rules and decide whether to enter the CRQ or the DTQ. Sensors that registered a collision in their chosen contention slots enter the CRQ, while those with successful requests enter the DTQ. Then, sensors in the CRQ execute a tree-splitting algorithm to resolve their collisions. For the devices in the DTQ, as there is no priority policy applied to their data, a First-In First-Out (FIFO) model is the one adopted, and their frames are sent during the data transmission period of the subsequent DQ frames.

At each sensor, four integer numbers are used to monitor the status of the queues and the position of the sensor in them. These integer numbers are:

(i) *RQ* representing the length of the CRQ; it indicates the number of groups of sensors with collisions waiting to re-transmit their access requests;

(ii) *TQ* representing the length of the DTQ; it indicates the number of sensors waiting for data transmission in the DTQ;

(iii) *pRQ* indicates the position of a given sensor in the CRQ;

(iv) *pTQ* indicates the position of a given sensor in the DTQ.

The network coordinator updates the first two integer numbers after each frame and broadcasts them during the feedback period. In contrast, each sensor, at the end of a contention process during which it was involved, individually updates the remaining integer numbers.

Three sets of rules are defined and executed at each sensor after the feedback period. Here, they are presented in order of their execution:

(i) The Data Transmission Rules (DTRs), indicating which sensor can transmit data in the following frame;

(ii) The Request Transmission Rules (RTRs), implementing the collision resolution algorithm;

(iii) The Queuing Discipline Rules (QDRs), managing the updates of the queues.

The DTRs are:

(i) If there are no sensors scheduled either for transmission ($TQ$ = 0) or for collision resolution ($RQ$ = 0), any sensor with data ready to be sent transmits an access request in any randomly selected contention slots and its payload in the data slot as it is empty. It is the only possible occasion when the DQ algorithm may register a collision during the data transmission period.

(ii) The sensor at the head of the DTQ ($pTQ$ = 1) is the only one allowed to transmit in the next frame.

The RTRs are:

(i) If there are no sensors scheduled for collision resolution ($RQ$ = 0) and the DTQ is not empty ($TQ$ > 0), any sensor with data ready to be sent chooses randomly one of the contention slots and transmits a channel access request in the next frame.

(ii) If a sensor is at the head of the CRQ ($pRQ$ = 1), it is allowed to select any of the $m$ contention slots and send an access request in it in the next frame.

The QDRs are as follows:

(i) Each sensor increments by one the value of *TQ* for each successful contention slot.

(ii) Each sensor decrements by one the value of *TQ* for each payload successfully transmitted in the data slot.

(iii) The value of *RQ* is decremented by one every time the CRQ ($RQ$ > 0) is not empty.

(iv) Each sensor increments by one the value of *RQ* for each contention slot with collisions.

(v) Each sensor updates its position in the queues (values for *pTQ* and *pRQ*). If the chosen contention slot is successful, then the sensor sets its *pTQ* to the corresponding value at the end of the *TQ*. If a collision occurs in the chosen contention slot, the sensor updates its position in the CRQ and sets its *pRQ* to the corresponding value at the end of *RQ*. If a node has not sent an access request, *pTQ* and *pRQ* follows the same update rules as *TQ* and *RQ* respectively, as long as their initial values are non-zero. It should be noted that the sensors enter the queues following a time arrival criterion.

The DQ algorithm operation flow chart for a node is illustrated in Fig. 1.

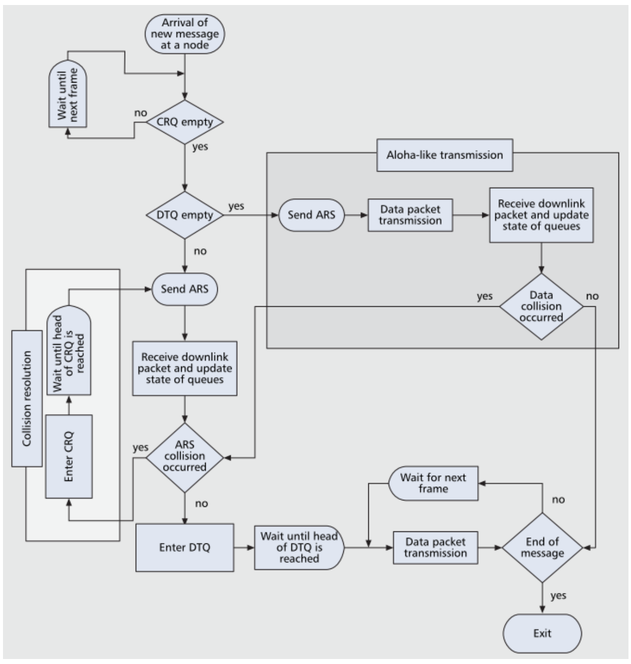

**Figure 1.** Distributed Queuing algorithm flow chart[20]

## 4. System model

To analyze and evaluate the performance metrics of the DQ algorithm, a discrete-event based simulation model was developed and executed in Matlab.

The system model is composed of a network coordinator and $n$ sensors trying to get access to the wireless channel using the DQ algorithm. All the $n$ sensors are in the range of the network coordinator. As the downlink channel, only dedicated to the gateway, is collision-free, the channel from the sensors to the gateway (uplink channel) is the one considered. Thereby, the DQ frame is composed of two parts: the contention period subdivided into *m* equal contention slots and the data transmission period of one data slot with a fixed size.

Before the operation of the DQ algorithm, we assumed that all the queues are empty (*CRQ* = 0 and *DTQ* = 0), and no sensors are scheduled to contend (*pRQ* = 0) or to transmit its data (*pTQ* = 0). Moreover, sensors have a perfect frame and contention slot synchronization.

In the beginning, every sensor chooses a contention slot randomly among the *m* available slots in a DQ frame and sends an access request signal to the chosen contention slot. However, at the same time, as the data slot is empty, sensors that generate the same random number as the network coordinator, are allowed to contend for transmission in the data slot. It is supposed that at the beginning of every empty frame, the network coordinator generates a random number. Every sensor knows the random number from the previous feedback period, and it has to generate its random number. This mechanism is the one adopted every time the data slot is empty.

If two or more sensors choose the same contention slot, a collision will occur. Thus, the sensors with a collision are sent in the collision resolution queue. Sensors that collided in the same contention slot are put in the same CRQ contention group. Groups of sensors in the CRQ are formed following the number of the contention slots in which the collisions occurred. Then, each group is scheduled to re-enter the channel access following its position in the CRQ in the subsequent frames. A sensor with a successful access request is put in the DTQ following the number of successes in the previous contention slots. If a sensor has successfully sent its payload in the data slot as it was empty, it leaves both queues. At the end of every frame and after every contention process, a sensor updates its integer numbers following the rules of the DQ algorithm.

The simulation time equals the time it requires for all the *n* sensors to secure a data slot and send their payload according to the DQ algorithm. It is a transient simulation model. A sensor needs only one data slot to send its payload.

In Table I, we present the simulation parameters. It is assumed that the sensors choose the contention slot to send in their access request signal following a discrete uniform distribution *U*(*a*;*b*) with the parameters *a* and *b* equal to 1 and *m* respectively. Moreover, the random number, chosen by both the network coordinator and the sensors at the beginning of an empty data slot, also follows a discrete uniform distribution *U*(*c*;*d*) with the parameters *c* and *d* equal to 1 and 5000 respectively. The parameters *c* and *d* are chosen arbitrarily.

To evaluate the performance of the DQ algorithm, we used the following metrics:

(i) the collision resolution time;
(ii) the data transmission time;
(iii) the time it takes the CRQ to reach the maximum;
(iv) the collision resolution time per sensor;
(v) the data transmission time per sensor;
(vi) the distribution of data slots into successful, empty and collided;
(vii) the number of random access attempts per sensor;
(viii) the number of sensors contending per frame;
(ix) the distribution of the contention slots into successful, empty and collided during the collision resolution time.

Table I. Simulation parameters

| Parameter | Description |
|---|---|
| Number of sensors | *N=250,750,…,5000* |
| Number of contention slots | *m=2,3,4,8,12,16* |
| Number of data slot | 1 |
| Distribution of the chosen contention slot by each sensor | Discrete uniform with parameters 1 and *m* |
| Distribution of the random number at the gateway and sensor level | Discrete uniform with parameters 1 and 5000 |
| Simulation runs | *R* =100 |

## 5. Performance evaluation and discussion

In this section, we present the simulation results obtained for each performance metric evaluated.

Matlab was used to perform all the simulations. It is launched in a Virtual Private Server (VPS) with eight cores and 12 GB of Random Access Memory (RAM). Due to the time constraints and the limited computing power available, the DQ algorithm is evaluated in a network with up to 5,000 nodes which are trying to get access to the channel simultaneously. Moreover, the number of contention slots in the DQ frame is limited to 16.

### 5.1. Evolution in the time of both CRQ and DTQ

From Fig. 2, a description of the evolution in the time of each of the queues of the DQ algorithm is given. It can be seen that based on the moments $t_{crqmax}$, $t_{crq}$ and $t_{dtq}$ the duration of each queue can be divided into three intervals of time:

- The first interval goes from the beginning of the collision resolution time up to the moment $t_{crqmax}$ when the CRQ reaches its maximum. The increasing number of groups of contenders in the CRQ can be explained by a high number of contention slots with collisions occurring during this period. The collisions are due to a high number of contending sensors in each CRQ group at the beginning of the contention process. Besides, during this interval of time, the

DTQ is "nearly" empty most of the time as a small number of sensors is being successful in accessing the channel.

- The second interval goes from the moment $t_{crqmax}$ up to the end of the collision resolution time at $t_{crq}$. A medium to a low number of contention slots with collisions characterizes this interval of time. In the beginning, the CRQ may grow slightly compared to its value at $t_{crqmax}$, and then it would decrease linearly over time. At the end, the CRQ is empty, as all the sensors have secured access to the channel. At the same time, the DTQ begins to grow linearly until it reaches its maximum value at the end of the interval at $t_{crq}$.
- Lastly, the third interval begins at $t_{crq}$ to end at the moment $t_{dtq}$ when the DTQ is empty. During this interval of time, the CRQ is empty as no sensors are trying to get access to the channel. However, the DTQ is characterized by a linearly decreasing number of sensors leaving it.

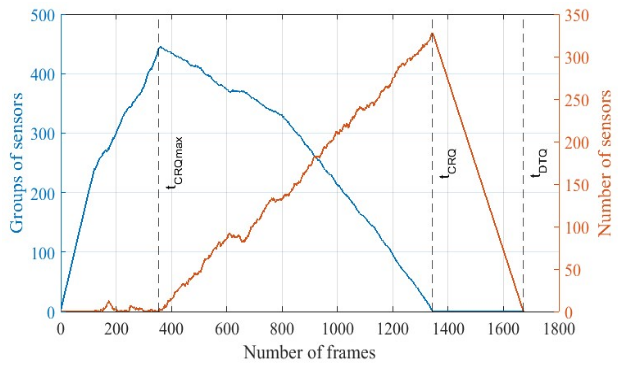

**Figure 2.** Evolution in time of the collision resolution queue (blue) and data transmission queue (red) when n =1500 sensors and m = 3

It should be noted that those three intervals of time are only distinguishable for the cases when the number of contention slots in the frame is more than two. For the case when there are two contention slots, the DTQ never reaches a moment when it grows linearly in a monotonous way except at the end of the collision resolution time. It is "nearly" empty most of the time. Due to a small number of contention slots ($m < 3$), a high number of contention slots with collisions is registered up to near the end of the collision resolution time.

In Fig. 3, the simulation results of the performance evaluation of these three metrics are presented:

(i) The average collision resolution time (Eq. 1), i.e., $t_{crq}$, represents the average time it takes all the contending sensors to get access to the wireless channel and secure a frame for the data transmission.

(ii) The average data transmission time (Eq. 2), i.e., $t_{dtq}$, is the average time from the beginning of the collision resolution time up to the moment when the DTQ is empty.

(iii) The average time it takes the CRQ to reach its maximum (Eq. 3), i.e., $t_{crqmax}$, represents the average interval of time from the beginning of the contention process up to the moment when the CRQ is at its maximum. It should be noted that the maximum considered here corresponds to the number of groups of sensors in the CRQ. At the same time, the data transmission subsystem has a "near-zero" registration in its queue. The "near-zero" means that the DTQ is observed until the time it begins to grow linearly.

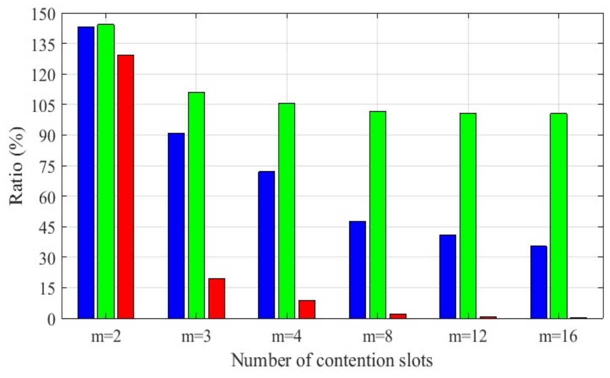

**Figure 3.** Average collision resolution time (blue). Average data transmission time (green). Average time for the CRQ to reach the maximum (red)

The performance metrics $t_{crq}$, $t_{dtq}$ and $t_{crqmax}$ are measured as ratios in percentages and are defined as follows:

$$t_{crq} = \sum_{i=1}^{R} \frac{(t_{crq})_i}{RN} \tag{1}$$

$$t_{dtq} = \sum_{i=1}^{R} \frac{(t_{dtq})_i}{RN} \tag{2}$$

$$t_{crqmax} = \sum_{i=1}^{R} \frac{(t_{crqmax})_i}{RN} \tag{3}$$

where

- $t_{crq_i}$ is the duration of the collision resolution time for an *i*th observation;
- $t_{dtq_i}$ is the duration of the data transmission for an *i*th observation;
- $t_{crqmax_i}$ is the interval of time it takes the CRQ to reach the maximum for an *i*th observation;
- $R$ is the total number of observations;
- $N$ is the number of frames required to send data from $n$ sensors with a perfect scheduling algorithm.

In (1), (2) and (3), the absolute values of the performance metrics $t_{crq}$, $t_{crq}$ and $t_{crqmax}$ are compared to $N$, respectively. By a perfect scheduling algorithm, it is assumed that each sensor requires only one frame to be sent. If $n$ sensors are present at the beginning of the collision resolution time, $n$ frames will be required to send all the data from the sensors.

From Fig. 3, it can be observed that both $t_{crq}$ and $t_{dtq}$ do not vary with the number $n$ of sensors present in the network, whereas they decrease logarithmically with the number $m$ of contention slots. For $m = 2$, the average collision resolution time is 143%, meaning that it requires a number of frames greater than the number $n$ of sensors to resolve all the collisions. On the other hand, for $m \geq 3$, the algorithm needs less than $n$ frames to resolve all the contentions. As for the $t_{dtq}$, it is always found to be over 100%, confirming that $n$ sensors cannot send their data in less than $n$ frames. Except for $m$=2, where $t_{dtq}$ is considerably high (144%), it is found to vary between 112% and 101%, respectively, for $m$ varying from three to 16.

Same as for $t_{crq}$ and $t_{dtq}$, the performance metric $t_{crqmax}$ does not change with the number of sensors, but it decreases considerably with the number of contention slots. A value of 129% for $t_{crqmax}$ is observed for $m$=2, whereas it varies from 19% to 0,4% when the number of contention slots changes from three to 16, respectively.

The results obtained for $t_{crq}$ and $t_{dtq}$ show that, irrespective of the number of the contention slots, $t_{dtq}$ is always greater than $t_{crq}$. Thus, confirming the results presented earlier in [17], that the speed of the contention resolution is faster than the speed of data transmission. For better performance metrics in terms of $t_{crq}$ and $t_{dtq}$, it is preferable to have a frame with more than two contention slots. When the number of contention slots is equal to two, the values of $t_{crq}$ and $t_{dtq}$ are relatively high compared to their respective values when the number of contention slots is three or more. We observed that a further increase in the number of contention slots improves the values of $t_{crq}$ and $t_{dtq}$. However, such an increase implies a more precise synchronization technique. That may result in an issue in terms of the cost and the size of the sensors. A better synchronization system implies a more complex sensor.

In general, we realized that CRQ reaches the maximum very quickly as the number of contention slots increases. As $m$ grows, the number of contention slots with collisions diminishes, leading to a small over-division of the CRQ. The performance metric $t_{crqmax}$ can be used to improve the access delay characteristics of the algorithm. Sensors from the previous contention process can be allowed to send their data during the $t_{crqmax}$ interval of time of the current contention process.

## 5.2. Average access delay per sensor

To assess the access delay introduced by the algorithm for each sensor, two performance metrics were estimated. These are the average collision resolution time per sensor and the average data transmission time per sensor:

(i) The average collision resolution time per sensor (Eq. 4), i.e., $t_{crqsensor}$, is the average time it takes a sensor to get access to the wireless channel and to secure a data slot before leaving the CRQ.

(ii) The average data transmission time per sensor (Eq. 5), i.e., $t_{dtqsensor}$, is the average time that a sensor spends in the DTQ while waiting to transmit its data after it has secured a frame.

Both metric metrics are measured as ratios in percentages and are defined as follows:

$$t_{crqsensor} = \sum_{j=1}^{R} \sum_{i=1}^{n} \frac{(t_{crqsensor})_{ij}}{nRN} \quad (4)$$

$$t_{dtqsensor} = \sum_{j=1}^{R} \sum_{i=1}^{n} \frac{(t_{dtqsensor})_{ij}}{nRN} \quad (5)$$

where

- $t_{crqsensorij}$ is the duration of the collision resolution time for a given $i$ sensor during a $j$th observation;
- $t_{dtqsensorij}$ is the duration of the data transmission time for a given $i$ sensor during a $j$th observation;
- $n$ is the total number of contending sensors;
- $R$ is the total number of observations;
- $N$ is the number of frames required to send data from $n$ sensors with a perfect scheduling mechanism.

Here again, in (4) and (5), the performance metrics $t_{crqsensor}$ and $t_{dtqsensor}$ are compared to $N$, respectively.

From Fig. 4, it can be noticed that both $t_{crqsensor}$ and $t_{dtqsensor}$ are invariant to the number of sensors. However, they vary logarithmically with the number of the contention slots in opposite directions: $t_{crqsensor}$ decreases while $t_{dtqsensor}$ is growing. Except for $m = 2$, on average it takes nearly a half or less of $n$ frames for a sensor to get access to the channel.

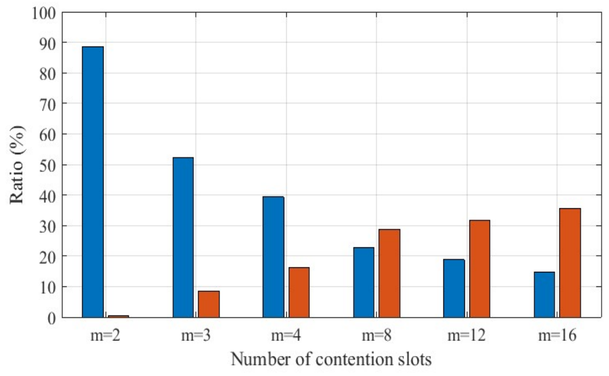

**Figure 4.** Access delay per sensor. Average collision resolution time (blue). Average data transmission time (red).

As for the data transmission, each sensor requires from 8% to 35%, when *m* varies from three to 16, respectively, while $t_{dtqsensor}$ is 0.3% when *m* = 2.

In general, except for the case when *m* =2, it takes a sensor on average from 60% to 50% of *n* frames to get access to the channel and to transmit its data. That is if *n* sensors are considered at the beginning of the contention process, and the number of contention slots varies from three to 16, respectively. When the number of contention slots is two, it requires on average 80% of *n* frames for a sensor to leave both queues. A small number of contention slots implies an over-division of the CRQ, leading to a high number of groups of contenders in it. Consequently, this causes a significant delay in the CRQ. At the same, as no sensors or only a small number of them enter the DTQ, the time a sensor spends in the DTQ is negligible compared to the time it spent in the CRQ. Thus, it is efficient to work with a DQ frame with more than two contention slots for higher performance in terms of delay access per sensor.

## 5.3. Average distribution of data slots into successful, empty, and collided

The throughput analysis is carried out from the beginning of the collision resolution process up to the time when all the sensors have sent their data. For the throughput evaluation, the contention period and the contention-free period are analyzed separately.

A data slot may register a success, a collision or may be empty. Based on those three situations, three performance metrics were used to evaluate the throughput. These are the average percentage of successful data slots *SD* (Eq. 6), the average percentage of empty data slots *ED* (Eq. 7) and the average percentage of data slots with collisions *CD* (Eq. 8):

$$SD = \sum_{i=1}^{R} \frac{SD_i}{R(t_{dtq})_i} \tag{6}$$

$$ED = \sum_{i=1}^{R} \frac{ED_i}{R(t_{dtq})_i} \tag{7}$$

$$CD = \sum_{i=1}^{R} \frac{CD_i}{R(t_{dtq})_i} \tag{8}$$

where

- *SDi* is the number of successful data slots registered in an *i*th observation;
- *EDi* is the number of empty data slots registered in an *i*th observation;
- *CDi* is the number of data slots with collisions registered in an *i*th observation;
- $t_{dtqi}$ is the number of data slots required by the DQ algorithm to clear the CRQ and DTQ queues for an *i*th observation;
- *R* is the total number of observations.

From Table II, it is shown that *SD*, *ED* and *CD* are all invariant to the number *n* of sensors present at the beginning of the contention process. As the number of contention slots increases, the metrics *SD* and *ED* change logarithmically in opposite directions. *SD* increases from 69.3% to 99.4%, while *ED* decreases from 30.5% to 0.5%. Except for *m*=2, *SD* is always above 90.16%, whereas *ED* is below 9.75%. As for the *CD*, it is always less than 0.15% for all the scenarios.

Table II. Average distribution of data slots into successful, empty, and collided

| Number of contention slots | *SD*, % | *ED*, % | *CD*, % |
|---|---|---|---|
| *m* =2 | 69.32 | 30.55 | 0.13 |
| *m* =3 | 90.16 | 9.75 | 0.09 |
| *m* =4 | 94.65 | 5.28 | 0.07 |
| *m* =8 | 98.45 | 1.51 | 0.04 |
| *m* =12 | 99.44 | 0.53 | 0.03 |
| *m* =16 | 99.49 | 0.48 | 0.03 |

In general, for *m* ≥ 3, all the empty data slots are registered during the moment when the CRQ has not reached its maximum $t_{crqmax}$. Therefore, that interval of time can be used to send data from the previous contention process. As for *m*=2, the observed high percentage of empty data slots (30.5%) can be explained by the low probability of a sensor to secure a data slot when there is a high number of sensors present in each CRQ group. Consequently, several data slots are empty because collisions are occurring more often.

In line with the previous works [2]–[6], [15], the results obtained confirm that the DQ algorithm can offer better performance in terms of throughput. We found that the percentage of successful data slots was over 90% for *m*≥3. It should also be noted that as *m* grows, $t_{crqmax}$ decreases, leading to a high number of successful data slots as more sensors are succeeding in their attempts to access the channel.

Although collisions can still occur in the DQ algorithm, with respect to the first data transmission rule, their impact on the throughput of the algorithm is minimal. The percentage of data slots with collisions is negligible compared to both the percentages of successful and empty data slots. For a better throughput performance, a frame with more than two contention slots is necessary.

## 5.4. Average number of random access attempts per sensor

During the collision resolution time, each sensor tries to access the wireless media channel by sending access request signals to the network coordinator. As long as a sensor fails to secure a position in the DTQ for data transmission, it sends access request signals every time its CRQ is scheduled to retry the access. The number of random access attempts that a sensor needs to send before it is granted access to the channel is evaluated through the average number $RA_{attempts}$ of random access attempts per sensor and is determined as follows:

$$RA_{attempts} = \sum_{j=1}^{R} \sum_{i=1}^{n} \frac{(RA_{attempts})_{ij}}{nR} \qquad (9)$$

where $RA_{attempts ij}$ is the number of random access attempts required by a sensor $i$ during a $j$th observation to leave the CRQ, $n$ is the total number of sensors and $R$ is the total number of observations.

From Fig. 5, it can be observed that the average number of attempts per sensor increases logarithmically with the number of sensors. At the same time, it also decreases logarithmically with the number of contention slots. For $m = 2$, a high number of attempts that vary from ten to 14 attempts per sensor is registered. However, that number decreases to change from three to four attempts per sensor when the number of contention slots is 16.

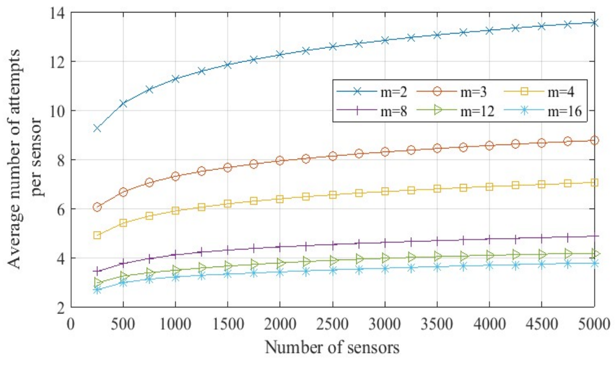

**Figure 5.** Average number of attempts per sensor during the collision resolution time

The high average number of attempts per sensor obtained in the case with two contention slots is related to the value of $t_{crq}$. A more significant value of $t_{crq}$ implies that sensors have to make several attempts before getting access to the channel. As $m$ increases, the number of attempts per sensor quickly decreases, following the same trend as $t_{crq}$.

A small number of attempts per sensor is better because each new attempt would require additional energy spending. Thus, for a better energy consumption performance, a frame with more than two contention slots needs to be considered. The average number of attempts per sensor varies from three to five attempts per sensor for $m \geq 8$. However, an increase in the number of contention slots would imply a more sophisticated sensor in terms of synchronization, leading to an expensive and large in size sensor. Therefore, a trade-off has to be made depending on the importance of the performance metric considered.

## 5.5. Average number of sensors contending per frame

Another important metric for the characterization of the DQ algorithm is the number of sensors contending in each frame during the collision resolution time. This performance metric is evaluated through the average number $S$ of sensors contending per frame and is defined as follows:

$$S = \sum_{j=1}^{R} \sum_{i=1}^{(t_{crq})_j} \frac{S_{ij}}{R(t_{crq})_j} \qquad (10)$$

where $S_{ij}$ is the number of contending sensors during an $i$th frame of the $j$th observation, $t_{crqj}$ is the duration of the collision resolution time for the $j$th observation and $R$ is the total number of observations.

From Fig. 6, the results show that the average number of sensors contending per frame increases logarithmically with the number of sensors. At the same time, it changes very slightly with the number of contention slots. It grows from seven sensors contending per frame for $n$ = 250 sensors to reach ten sensors contending per frame for $n$ = 5000 sensors.

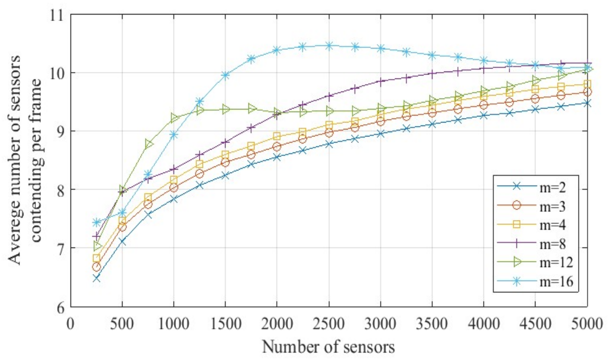

**Figure 6.** Average number of contending sensors per frame during the collision resolution time

At the beginning of the collision resolution time, a high number of contending sensors per frame characterizes the algorithm. However, as time progresses, due to the over-division of the CRQ into multiple groups with a small number of sensors, the number of contending sensors per frame decreases. The over-division of the CRQ into smaller groups decreases with the number $m$ of contention. As $m$ increases, the sensors in the CRQ groups have more opportunity to succeed in channel access. Therefore, a high number of sensors contending per frame is obtained when $m$ is high because the collisions are resolved quickly. However, for small values of $m$, it takes more time to resolve all the contentions. Consequently, the CRQ is composed of multiple groups with a small number of sensors in them. In general, it can be observed from Fig. 6 that the average number of sensors contending per frame depends slightly on the number of contention slots.

The increase of the number *n* of sensors in the network leads to an augmentation of the number of contention groups in the CRQ for a given *m*. Therefore, the number of sensors contending per frame is also increased. The number of sensors contending per frame is essential as the status of each of the contention slots depends on it. The number of sensors contending per frame needs to be optimized to maximize the number of successful contention slots.

## 5.5. Average distribution of contention slots into successful, empty and collided

An analysis of the metrics related to the contention period of the DQ frame was also conducted. Those metrics are evaluated only during the collision resolution time as no sensors are contending after the CRQ is empty. Three performance metrics related to the number of contention slots were of interest. These are the average number of empty contention slots per frame *EC* (Eq. 11), the average number of successful contention slots per frame *SC* (Eq. 12) and the average number of empty contention slots per frame *CC* (Eq. 13):

$$EC = \sum_{j=1}^{R} \sum_{i=1}^{(t_{crq})_j} \frac{EC_{ij}}{mR(t_{crq})_j} \tag{11}$$

$$SC = \sum_{j=1}^{R} \sum_{i=1}^{(t_{crq})_j} \frac{SC_{ij}}{mR(t_{crq})_j} \tag{12}$$

$$CC = \sum_{j=1}^{R} \sum_{i=1}^{(t_{crq})_j} \frac{CC_{ij}}{mR(t_{crq})_j} \tag{13}$$

where

- $EC_{ij}$ is the number of empty contention slots during an *i*th frame of the *j*th observation;
- $SC_{ij}$ is the number of successful contention slots during an *i*th frame of the *j*th observation;
- $CC_{ij}$ is the number of contention slots with collisions during an *i*th frame of the *j*th observation;
- m is the total number of contention slots available in a DQ frame;
- R is the total number of observations.

From Fig. 7, it can be noted that *EC*, *SC,* and *CC* do not vary with the number of sensors. The average number of empty contention slots per frame *EC* increases logarithmically from 15% to 76% as the number of contention slots grows from two to 16. At the same time, *CC* decreases logarithmically from 49% to 6%. As for *SC*, it varies from 34% to 17% with a maximum of 37% when *m* equals three.

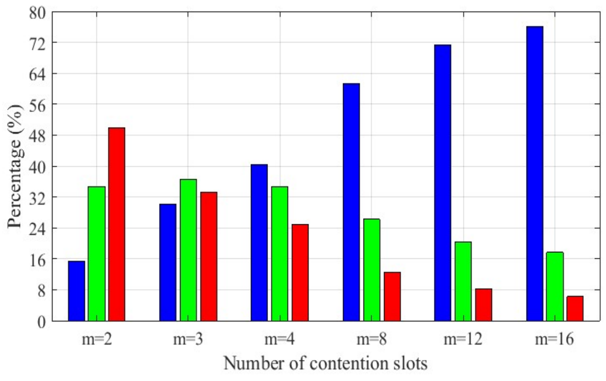

**Figure 7.** Average number of empty (blue), successful (green) and collided (red) contention slots during the collision resolution time

The DQ algorithm is characterized by a high number of contention slots with collisions from the beginning of the contention process up to the moment $t_{crqmax}$. During that interval of time, the CRQ groups contain a high number of sensors that are trying to access the channel. Therefore, the average numbers of successful and empty contention slots are considerably small or zero compared to the average number of contention slots with collisions.

However, as time progresses, the groups of contending sensors become less populated, and the average number of contention slots with collisions begins to decrease while the average numbers of empty and successful contention slots respectively are increasing.

Finally, at the end of the collision resolution time, the average number of empty contention slots always tends to be larger than the remaining two other performance metrics. The average number of successful contention slots turns out to be larger than the average number of contention slots with collisions. This trend is emphasized as the number of contention slots increases.

As time progresses, the number of sensors contending per frame becomes smaller than the number of available contention slots. The performance metric *EC* is found to be always more significant than the remaining two performance metrics for $m \geq 4$. However, an optimal ratio among those three metrics is only observed in the case when the number of contention slots is three. A rise in the number of contention slots (over three) leads to inefficient use of the contention slots, as a large percentage of them are empty. Thus, for better performance in terms of efficient use of the contention slots, a frame with three contention slots is preferable.

## 5. Conclusion, recommendations and future works

In this paper, we presented a statistical performance analysis of the DQ algorithm. To accomplish that goal, we used different performance metrics to evaluate the algorithm.

First, the collision resolution time and the data transmission time were analyzed, and we found that their averages were decreasing with the number of contentions slots. Then, based on the moments when the CRQ was reaching the maximum and when it was empty, the DQ algorithm queues were subdivided into three different intervals of time.

Secondly, we also evaluated the waiting times of each sensor in both the collision resolution and data transmission queues. We observed that the average access delay per sensor was varying from 60% to 50% of *n* frames as *m* increased from three to 16 and if *n* sensors were present in the network at the beginning of the contention process. Also, the throughput of the algorithm was investigated through the number of successful, empty, and collided data slots. We had found that the algorithm could achieve a performance of over 90% in terms of the number of successful data slots when *m* was greater or equal to three contention slots.

Another metric analyzed was the number of random access attempts needed by a sensor to leave the CRQ. On average, six to nine attempts were necessary for a sensor to leave the queue when *m* was three, and we observed that it was decreasing to vary from three to four attempts per sensor when *m* was 16. Additionally, we also realized an evaluation of the average number of sensors contending per frame, and we noticed that it was increasing with the number of sensors from seven to ten sensors contending per frame when *n* varied from 250 to 5000 sensors respectively.

Lastly, we found that the average number of empty contention slots was always more significant than both the average number of successful and collided contentions slots when they were compared separately for $m \geq 4$.

In general, we observed that a DQ frame with more than two contention slots was preferable for better network performance. Moreover, an optimal ratio between the average numbers of collided, empty, and successful contention slots was observed only for $m$=3. Apart from the number of random access attempts per sensor and the number of sensors contending per frame, all the other evaluated performance metrics were invariant to the initial number of sensors in the network when *m* was constant. We have also established that the first interval of time of the DQ algorithm could be used to send data from the previous contention process. It is during that interval of time that all the empty frames were registered.

Another crucial finding was that an increase in the number of contention slots was reducing the number of attempts per sensor, but at the same time, it was leading to inefficient use of the contention slots. Thus, a trade-off was needed, or the number of the sensors contending per frame should be optimized to avoid a waste of the contention slots. However, an increase in the number of contention slots should be carefully considered as it may result in a more complex, expensive, and large in size sensor.

In the future, we plan to perform a steady-state analysis to evaluate the stability criteria of the algorithm in a massive communication environment.

## Acknowledgement

The Pan African University funded this research

## Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

## References

[1] Ericsson, "Ericsson Mobility Report (June 2019)," *Ericsson White Paper*, p. 36, Jun. 2019. [Online]. Available: www.ericsson.com/mobilityreport

[2] F. Vazquez-Gallego, P. Tuset-Peiro, L. Alonso, and J. Alonso-Zarate,´ "Combining distributed queuing with energy harvesting to enable perpetual distributed data collection applications," *Transactions on Emerging Telecommunications Technologies*, vol. 29, no. 7, p. e3195, 2018.

[3] A. Laya, L. Alonso, and J. Alonso-Zarate, "Contention resolution queues for massive machine type communications in LTE," *IEEE 26th annual international symposium on personal, indoor, and mobile radio communications (PIMRC), Hong Kong, China*, pp. 2314–2318, 2015.

[4] S. Xing, X. Wen, Z. Lu, Q. Pan, and W. Jing, "A novel distributed queuing-based random access protocol for Narrowband-IoT," *IEEE International Conference on Communications (ICC), Shanghai, China*, pp. 1–7, 2019.

[5] W. Wu, Y. Li, Y. Zhang, B. Wang, and W. Wang, "Distributed Queueing Based Random Access Protocol for LoRa Networks," *IEEE Internet of Things Journal*, vol. 7, no. 1, pp. 763–772, 2020.

[6] A. Laya, C. Kalalas, F. Vazquez-Gallego, L. Alonso, and J. AlonsoZarate, "Goodbye, aloha!" *IEEE access*, vol. 4, pp. 2029–2044, 2016.

[7] P. Tuset-Peiro, F. Vazquez-Gallego, J. Alonso-Zarate, L. Alonso, and X. Vilajosana, "LPDQ: A self-scheduled TDMA MAC protocol for one-hop dynamic low-power wireless networks," *Pervasive and Mobile Computing*, vol. 20, pp. 84–99, 2015.

[8] F. Vazquez-Gallego, J. Alonso-Zarate, P. Tuset-Peiro, and L. Alonso, "Energy analysis of a contention tree-based access protocol for machine-to-machine networks with idle-to-saturation traffic transitions," *IEEE International Conference on Communications (ICC) 2014, Sydney, Australia*, pp. 1094–1099, 2014.

[9] A. Samir, M. M. Elmesalawy, A. S. Ali, and I. Ali, "An Improved LTE RACH Protocol for M2M Applications," *Mobile Information Systems*, vol. 2016, 2016.

[10] R. G. Cheng, Z. Becvar, and P. H. Yang, "Modeling of Distributed Queueing-Based Random Access for Machine Type Communications in Mobile Networks," *IEEE Communications Letters*, vol. 22, no. 1, pp. 129–132, 2018.

[11] C. Yoon, "Distributed queuing with preamble grouping for massive IoT devices in LTE random access," *International Conference on Information and Communication*

*Technology Convergence (ICTC), Jeju Island, South Korea*, pp. 103–105, 2016.

[12] A.-T. H. Bui, C. T. Nguyen, T. C. Thang, and A. T. Pham, "Free Access Distributed Queue Protocol for Massive Cellular-based M2M Communications with Bursty Traffic," *IEEE 88th Vehicular Technology Conference (VTC-Fall), Chicago, USA*, pp. 1–5, 2018.

[13] K. Lee and J. U. W. Jang, "An Efficient Contention Resolution Scheme for Massive IoT Devices in Random Access to LTE-A Networks," *IEEE Access*, vol. 6, pp. 67118–67130, 2018.

[14] H. A. T. Bui, C. T. Nguyen, T. C. Thang, and A. T. Pham, "A Comprehensive Distributed Queue-based Random Access Framework for mMTC in LTE/LTE-A Networks with Mixed-Type Traffic," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 12, pp. 12107– 12120, 2019.

[15] A. Marchiori, "Maximizing coverage in low-power wide-area IoT networks," *IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops), Kona, Big Island, HI, USA*, pp. 467–472, 2017.

[16] J. Yuan, H. Shan, A. Huang, T. Q. Quek, and Y. D. Yao, "Massive machine-to-machine communications in cellular network: Distributed queueing random access meets MIMO," *IEEE Access*, vol. 5, pp. 2981– 2993, 2017.

[17] W. Xu and G. Campbell, "A near perfect stable random access protocol for a broadcast channel," *IEEE International Conference on Communications, Chicago, IL, USA, 1992. ICC'92, Conference record, SUPERCOMM/ICC'92, Discovering a New World of Communications.*, pp. 370–374, 1992.

[18] H.-J. Lin and G. Campbell, "PDQRAP-Prioritized Distributed Queueing Random Access Protocol." *Proceedings of 19th Conference on Local Computer Networks, Minneapolis, MN, USA*, pp. 82–91, 1994.

[19] C.-T. Wu and G. Campbell, "Extended DQRAP (XDQRAP) A cable TV protocol functioning as a distributed switch," *Proc.1st International Workshop on Community Networking, San Fransisco, USA. Computer Communication Review*, vol. 23, no. 4, pp. 270–278, 1994.

[20] J. Alonso-Zarate, C. Verikoukis, E. Kartsakli, A. Cateura, and L. Alonso, "A near-optimum cross-layered distributed queuing protocol for wireless LAN," *IEEE Wireless Communications*, vol. 15, no. 1, pp. 48–55, 2008.

[21] L. Alonso, R. Agust´ı, and O. Sallent, "A near-optimum MAC protocol based on the distributed queueing random access protocol (DQRAP) for a CDMA mobile communication system," *IEEE Journal on Selected Areas in Communications*, vol. 18, no. 9, pp. 1701–1718, 2000.

[22] B. Otal, L. Alonso, and C. Verikoukis, "Towards energy saving wireless body sensor networks in health care systems," *IEEE International Conference on Communications Workshops (ICC), Cape Town, South Africa*, pp. 1–5, 2010.

[23] J. Alonso-Zarate, C. Verikoukis, E. Kartsakli, A. Cateura, and L. Alonso, "Saturation throughput analysis of a passive cluster-based medium access control protocol for ad hoc wireless networks," *IEEE International Conference on Communications, Beijing China*, pp. 2348–2352, 2008.

[24] J. Alonso-Zarate, C. Verikoukis, E. Kartsakli, and L. Alonso, "A novel near-optimum medium access control protocol for a distributed cooperative ARQ scheme in wireless networks," *IEEE 19th International Symposium on Personal, Indoor and Mobile Radio Communications, Cannes, France*, pp. 1–5, 2008.