Journal of
Sensor and Actuator Networks

MDPI

Article

A Spreading Factor Congestion Status-Aware Adaptive Data Rate Algorithm

Charles Lehong [1,*], Bassey Isong [1,*], Francis Lugayizi [1] and Adnan Abu-Mahfouz [2]

[1] Computer Science Department, North-West University, Mafikeng 2745, South Africa; francis.lugayizi@nwu.ac.za
[2] Council for Scientific and Industrial Research (CSIR), Pretoria 0001, South Africa; a.abumahfouz@ieee.org
* Correspondence: lehongcharles@gmail.com (C.L.); bassey.isong@nwu.ac.za (B.I.)

Abstract: LoRaWAN has established itself as one of the leading MAC layer protocols in the field of LPWAN. Although the technology itself is quite mature, its resource allocation mechanism, the Adaptive Data Rate (ADR) algorithm is still quite new, unspecified and its functionalities still limited. Various studies have shown that the performance of the ADR algorithm gradually suffers in dense networks. Recent studies and proposals have been made as attempts to improve the algorithm. In this paper, the authors proposed a spreading factor congestion status aware ADR version and compared its performance against that of four other related algorithms to study the performance improvements the algorithm brings to LoRaWAN in terms of DER and EC. LoRaSim was used to evaluate the algorithms' performances in a simple sensing application that involved end devices transmitting data to the gateway every hour. The performances were measured based on how they affected DER as the network size increases. The results obtained show that the proposed algorithm outperforms the currently existing implementations of the ADR in terms of both DER and EC. However, the proposed algorithm is slightly outperformed by the native ADR in terms of EC. This was expected as the algorithm was mainly built to improve DER. The proposed algorithm builds on the existing algorithms and the ADR and significantly improves them in terms of DER and EC (excluding the native ADR), which is a significant step towards an ideal implementation of LoRaWAN's ADR.

Keywords: adaptive data rate; LoRaWAN; LPWAN; energy consumption; data extraction rate; spreading factor; LoRaSim

Citation: Lehong, C.; Isong, B.; Lugayizi, F.; Abu-Mahfouz, A. A Spreading Factor Congestion Status-Aware Adaptive Data Rate Algorithm. *J. Sens. Actuator Netw.* **2021**, *10*, 70. https://doi.org/10.3390/jsan10040070

Received: 17 August 2021
Accepted: 14 October 2021
Published: 3 December 2021

1. Introduction

Over the years Long-Range Wide Area Networking (LoRaWAN) has grown to become one of the leading Media Access Control (MAC) layer protocols in the Low Power Wide Area Networking (LPWAN) space [1]. The LoRaWAN protocol typically runs on top of the LoRa physical layer developed by Semtec. The modulation technique used in the LoRa physical layer is a derivative of the Chirp Spread Spectrum (CSS) modulation and allows technologies such as LoRaWAN to achieve robust multi-kilometre communications while consuming minimal amounts of energy [2]. The popularity of LoRaWAN is owed to its simplicity, openness and cost-effectiveness. Because of the CSS modulation technique which allows for robust long-distance wide-area wireless communications, LoRaWAN is deployed following a simple star topology which keeps it both simple and cost-effective. At the centre of the network is a high capacity gateway surrounded by and serving up to thousands of end devices. The end devices are typically battery-powered sensors and actuators that spend most of their time sleeping except when they have data to transmit to the gateway. This makes the end devices energy-efficient and ensures that they can survive on batteries for years [3]. The gateway is connected, through a non LoRaWAN backhaul, to a centralized network server which performs all the complex operations of the network such as filtering duplicate data and performing data rate optimization for the end devices [3].

Despite presenting itself as one of the most desirable network technologies in the LPWAN space, one of LoRaWAN's biggest limitations is that it operates in the freely available Industrial, Scientific and Medical (ISM) bands [4], which it has to share with other technologies while at the same time adhering to duty cycle regulations in those bands. Furthermore, LoRaWAN networks are often deployed in very challenging networking conditions. Therefore, LoRaWAN must have a network management mechanism to handle these network constraints. Furthermore, the responsibility of this mechanism must go beyond network management, it must also include the intelligence to manage the end devices' transmission parameters to handle these network constraints, as well as improve network reliability and efficiency [1]. LoRaWAN already has one such mechanism, the Adaptive Data Rate (ADR) algorithm. The ADR is an algorithm used by LoRaWAN's network server to optimize the end devices' transmission powers and data rates. When the algorithm is enabled, it ensures that the end devices use the highest data rates and the lowest transmission powers possible, thus enforcing network reliability and energy efficiency and improving network capacity [5].

Although the ADR was introduced into LoRaWAN as a solution towards improving the efficiency of the end devices' transmissions, studies have shown that the algorithm leads to increases in traffic collisions, especially in large networks [6–8]. These findings have received significant attention from academia: numerous studies and proposals have been produced as potential solutions to this problem. These solutions have been critically studied and summarized by the authors in [4]. The authors showed that these solutions have constantly brought improvements to the ADR. However, the authors also have argued that these improvements are still not adequate and that arriving at an ideal implementation of the ADR remains an open research area. Therefore, in this paper, the researchers used the knowledge gained from studying the findings in [4] to design an enhanced version of the ADR that uses congestion statuses of spreading factors to determine optimized data rates for end devices. The proposed algorithm was then evaluated in a series of simulated experiments and its performance was compared against four other algorithms with the same characteristics from the papers [7–9] in terms of data extraction rate (DER) and energy consumption (EC) [4]. The evaluation results show that the proposed algorithm achieves at least 40 per cent better DER performance and at least 100 per cent better EC performance over the other four algorithms. The process of evaluating the proposed algorithm was guided by the following research objectives:

(i) To modify the Adaptive Data Rate algorithm to consider congestion statuses of data rates during the data rate optimization process to improve the collision probabilities of end devices' traffic in Long-Range Wide Area Networks.
(ii) To use simulations to evaluate the effectiveness of the proposed modifications to the Adaptive Data Rate algorithm in mitigating the occurrence of traffic collisions in Long-Range Wide Area Networks.

The rest of this paper is organized as follows: Section 2 discusses the ADR and related algorithms proposed by other researchers. Section 3 presents the proposed algorithm. Section 4 presents the experimental design that guided this research. Sections 5 and 6 present the results and their discussion, respectively, while Section 7 concludes the paper and provides insight for future work.

2. Adaptive Date Rate Algorithms in LoRaWAN

2.1. Adaptive Date Rate Overview

ADR is an algorithm employed in the LoRaWAN network to optimize the transmission parameters for stationery LoRa end devices [4]. The LoRaWAN provides the transmission parameters spreading factor (SF), bandwidth (BW), transmission power (TP), and coding rate (CR), that the ADR can manipulate to optimize the end devices' data rates, energy consumptions, airtimes and the capacity of the network [4]. The specifications of the ADR are provided in detail in [10]. Semtec provides the simplified descriptions and implementation recommendations of the ADR in [5,11] respectively.

The process by which the ADR optimizes transmission parameters for end devices is as follows: initially, the end devices choose transmission parameters randomly, usually the highest TP and lowest SF possible. When the device wants to enable the ADR, it alerts the network server by setting the ADR bit on one of its transmissions to the network server. Once the network server receives the alert, it starts collecting up to 20 uplink messages. From the 20 uplinks, the Frame counter values, and the Signal-to-Noise ratios (SNRs) are collected to be used in the ADR process. From the SNRs collected, the network server determines the maximum SNR (SNR_{max}) and uses it to calculate the margin using Equation (1) [4]:

$$\text{margin} = SNR_{max} - SNR_{limit} - \text{margin}_{default} \quad (1)$$

where the SNR_{limit} is the minimum SNR required to demodulate a signal and the $\text{margin}_{default}$ is the installation margin specific to the device being used and in most cases is 10 dB. The calculated margin is then used to calculate the number of times (i.e, the N_{step}) the algorithm is executed using Equation (2):

$$N_{step} = \text{int}(\text{margin}/3) \quad (2)$$

where int is the integer value and based on the computed Nstep value, one of the following actions can be executed:

- If the $N_{step} = 0$, the transmission parameters are already optimal, so the algorithm terminates.
- If the $N_{step} > 0$, the algorithm starts by reducing the SF to its lowest possible value, and then it reduces the TP to its lowest possible value.
- If the $N_{step} < 0$, the algorithm increases the TP to its highest possible value. Nothing is done to the SF at this stage since LoRa end devices already have automatic data rate decay.

Once the optimal TP and SF have been determined by the algorithm, the network server waits for the end device to transmit an uplink message and then attaches the information about the optimized transmission parameters to the downlink response to the end device. Once received, the end device can then switch to the newly determined transmission parameters. Furthermore, to ensure that connectivity to the network server with the new parameters is still possible, the end device uses an ADR acknowledgement counter every time an uplink message is transmitted. If a certain limit is reached without a response from the server, the end device set an ADR acknowledgement delay. If a response from the network server is received before the delay is reached, the ADR acknowledgement counter is reset. If no response is received, the end device assumes that connectivity is lost and tries to regain it by gradually increasing the TP until a response is received. If still no response is received, the SF is increased each time the delay is reached until a response is received or the SF reaches its maximum possible value [4].

2.2. Related Adaptive Date Rate Algorithms

The researchers collected, studied, and critically analyzed nine papers that proposed algorithms meant to improve the ADR algorithm and more details can be found in [4]. From the nine papers, four algorithms were selected for empirical analysis in this paper. These algorithms were selected for evaluation in this paper as they share the same characteristics as the algorithm being proposed in this paper. To be exact, the selected algorithms are based on modified versions of the ADR and were mainly intended to improve LoRaWAN in terms of reliability by improving the overall network DER as network sizes increase. Table 1 provides a summary of these algorithms and the rest of this section discusses them in detail.

Table 1. Summary of selected ADR algorithms [4].

Challenges Addressed	Proposed Solutions	Critical Analysis
Reduction in network throughput when the ADR is employed [9]	A contention-aware ADR with restricted usage of data rates.	The proposed algorithm used the gradient projection method to determine distribution ratios that minimize contention in each data rate and maximizes network throughput.
The increase in collisions in ADR-enabled LoRa networks [7]	An EXPLoRa-SF algorithm that creates orthogonal channels to avoid network collisions.	EXPLoRa-SF divides the network into channels and then assigns a data rate to each. The channels are orthogonal and therefore, collisions cannot occur between traffic from the different channels.
The increase in collisions and unfairness amongst network traffic [7]	An EXPLoRa-AT algorithm that promotes fairness amongst network traffic.	EXPLoRa-AT extends EXPLoRa-SF to include the intelligence to equalize airtimes for all traffic in different transmission channels and to enforce channel usage fairness and thus reduce network collisions.
Collision unfairness between end devices that are near and far from the gateway [8].	A fair adaptive data rate algorithm that promotes data extraction fairness at the gateway.	The authors used mathematical modelling to derive the fairest data rate distribution ratios and set up the ADR to use these ratios to make data rate optimization decisions.

Kim and Yoo [9] discovered that the native ADR algorithm performed poorly under large LoRa network conditions. They found that the algorithm reduced the network's overall throughput. To mitigate this problem, the authors proposed a contention-aware approach to the ADR. The solution used the gradient projection method to compute optimized data rate distributions that optimized throughput for each data rate group. This in turn reduced network contention and optimized the overall network throughput. The solution was evaluated using simulation representing the European ISM band with a one per cent duty cycle. The simulation networks consisted of 5 to 10,000 end devices each transmitting 50 to 100 bytes of data. The evaluation was based on throughput and its effectiveness was compared against the traditional ADR algorithm and to a version of the ADR with a balanced number of end devices (NEDs) in each data rate group. The proposed solution outperformed the traditional ADR algorithm in terms of overall network throughput, however, the solution performed poorly in terms of end device transmission success rate. Hence, in terms of transmission reliability, the solution still needs to be improved.

Abdelfadeel et al. [8] also revealed that when enabled in a network, the ADR algorithm leads to data transmission unfairness where packets from end devices closer to the gateway are more likely to be extracted by the gateway than those from the end device further away from the gateway. Consequently, the authors proposed a fair ADR algorithm. The algorithm's goal is to ensure fairness amongst network traffic regardless of the end devices' distances from the gateway and to optimize the overall network energy consumption. Accordingly, the author used mathematical modelling to derive the fairest data rate ratios to be used by the algorithm to compute end devices' data rates, and balanced transmission powers across end devices to mitigate the effects of the capture effect. The algorithm was evaluated using LoRaSim and compared against algorithms from [12] and [13] in terms of data extraction rate and energy consumption. The results revealed the overall superiority of the algorithm over both algorithms except in terms of energy consumption, where [12] was more superior.

The last two algorithms considered in this paper are presented by Cuomo et al. [7]. In their quest to find a way to improve the ADR, Cuomo et al. [7] exposed that in some cases, forcing an end device to use a lower data rate can reduce collisions in a LoRa net-

work. Accordingly, they proposed two implementations of the ADR. The simpler of the two, EXPLoRa-SF, was built to reduce network collisions by segregating end devices into channels based on their distance from the gateway. The second approach, EXPLoRa-AT, was built on the functionalities of EXPLoRa-SF and included an intelligence mechanism to equalize airtimes for all traffic in different transmission channels and to enforce channel usage fairness. Both algorithms were implemented and tested using LoRaSim and compared against the native ADR algorithm in terms of throughput and data extraction rate. The simulation environment was a network operating in the European ISM band consisting of 500 to 2000 end devices each transmitting a 160-byte packet every 5 to 3600 s. The overall results showed superior performances of both algorithms over the native ADR.

3. Spreading Factor Congestion-Aware ADR

In this section, the researchers present the proposed algorithm. The algorithm was named spreading factor congestion-aware ADR approach and is specified in Algorithm 1. The name of the algorithm came from the fact that it uses the congestion statuses of each spreading factor in the network to determine the best one for the end device that requested the ADR. The development of this algorithm was performed to answer the question: how can a modified Adaptive Data Rate algorithm be developed to efficiently manage and reduce signal interferences in LoRaWAN? And to carry out the defined research objective (i). The proposed algorithm has seven main global variables (i.e., snrList, sfList, uplinkCnt, optimizedSF, defaultMargin, snrLimit, sfUsageIndex). These variables are explained as follows:

Algorithm 1: Spreading Factor Congestion-Aware Adr Approach

```
Inp.: Signal-to-noise ratio and spreading factor values of the uplink messages.
Outp.: An optimized spreading factor value for the end device that requested ADR.
Steps:
snrList ← 0
sfList ← 0
uplinkCnt ← 0
optimizedSF ← 12
defaultMargin ← 10
snrLimit ← [7: −7.5, 8: −10, 9: −12.5, 10: −15, 11: 17.5, 12: −20]
sfUsageIndex ← associative array of spreading factors and their usage indexes

function optimizeDataRate(uplinkSNR, uplinkSF)
        snrList[uplinkCnt] ← uplinkSNR
        sfList[uplinkCnt] ← uplinkSF
        uplinkCnt ← uplinkCnt + 1
        if uplinkCnt = 20 then
                maxSNR ← max(snrList)
                maxSF ← sfList[indexOf(maxSNR)]
                minSF ← minSpreadingFactor(maxSNR, maxSF)
                optimizedSF ← optimizeSpreadingFactor(minSF, maxSF)
                uplinkCnt ← 0
        else
                exit
        end if
        return optimizedSF
end function

function minSpreadingFactor(maximumSNR, maximumSF)
        margin ← maximumSNR − snrLimit[maximumSF] − defaultMargin
        steps ← round (margin / 3)
        SF ← maximumSF
```

Algorithm 1: Spreading Factor Congestion-Aware Adr Approach

```
29:                         while steps > 0 and SF > sfMin do
30:                             SF ← SF − 1
31:                             Steps ← steps − 1
32:                         end while
33:                         return SF
34:     end function
35:
36:     function optimizeSpreadingFactor(minimumSF, maximumSF)
37:                         optSF ← minimumSF
38:                         counterSF ← minimumSF
39:                         while counterSF <= maximumSF do
40:                             if sfUsageIndex[optSF] > sfUsageIndex[counterSF]
                                then
41:                                 optSF ← counterSF
42:                             end if
43:                             counterSF ← counterSF + 1
44:                         end while
45:                         sfUsageIndex[optSF] = sfUsageIndex[optSF] + 1
46:                         return optSF
47:     end function.
```

snrList: a list of SNR values to be used in determining the best data rate. The list is initially empty and is updated each time the end device sends an uplink to the network server.
sfList: a list of SFs that corresponds to the collected SNR values. The list is also initially empty and is updated each time the end device sends an uplink to the network server.
uplinkCnt: a counter to keep track of the number of uplinks that the end device has sent to the network server since the ADR was requested.
optimizedSF: the SF that will represent the optimized data rate. This variable stores the value that the algorithm will calculate once enough data has been collected.
defaultMargin: the device-specific margin. This value is given in the devices' datasheets. A value of 10 is used in this experiment as it is generally the value for most devices.
snrLimit: a list of limits required for each SF for the receiver to be able to demodulate the received signal. These values are used in determining the margin required for optimizing the data rate.
sfUsageIndex: a list of stored values representing the number of devices using each SF. The values will be used to determine the SF with the fewest devices using it. This list is one of the most important pieces required by the algorithm.

As it can be seen from the pseudocode in Algorithm 1, the algorithm is split into three functions that perform specific tasks. All the functions are vital to the operation of the algorithm:
optimizeDataRate function: The optimizeDataRate function is the main entry point to the algorithm. It accepts the uplink SNR value as well as the uplink SF, stores them in their respective lists, and then updates the counter. Just like in the native ADR, once the counter has reached 20 uplink messages, the algorithm starts to calculate the optimized data rate. In the optimizeDataRate function, the maximum SNR value and the maximum SF are computed. The results are then passed in as parameters to the minSpreadingFactor function.
minSpreadingFactor function: As the name suggests, this function is used to determine the lowest value of the SF that the end device can support while being able to reach the gateway. This SF is what will represent the optimized data rate. To compute this value, the function first calculates the margin and the number of steps the algorithm needs to run from the passed in parameters and the default margin using Equations (3) and (4) respectively:

$$\text{margin} = \text{maximumSNR} - \text{snrLimit[maximumSF]} - \text{defaultMargin} \quad (3)$$

$$\text{steps} = \text{round}\,(\text{margin}/3) \quad (4)$$

where snrLimit[maximumSF] is the SNR limit from the SNR limit list that corresponds with the value stored in the maximumSF variable and the defaultMargin is the device-specific margin given in the device datasheet. And the "round" in Equation (4) represents the integer part of the resulting calculated value. From here, the function then loops from the calculated value of steps until the value reaches zero and tries to minimize the value of the SF to its possible optimal value. Once the loop exits, the remaining value of the SF is considered as the lowest value that the end devices can support. This value is returned to the main function and stored in the minSF variable. Once the mainFunction has the value of the minimum SF, the optimized SF can be computed. To do this, the optimizeSpreadingFunction function is called by passing in the minimum SF and the maximum SF values to it as parameters.

optimizeSpreadingFunction function: Using these values (i.e., minimum SF and the maximum SF), the function runs a loop from the minimum SF to the maximum SF and tries to find the least utilized SF value using the SF usage indexes stored in the sfUsageIndex list. If the least utilized SF is found, it's returned. If not, the minimum SF value is returned instead. This ensures that the returned value is optimized even when the least utilized SF is not found. The optimized SF is then returned to the main function and stored in optimizedSF variable, and the counter is then reset. The value stored in the optimizedSF variable is then sent down as part of the response to the end device the next time the device sends an uplink.

4. Evaluation Setup

The goal of this research experiment was to empirically evaluate the spreading factor congestion-aware ADR algorithm and to study its performance in terms of reliability and energy efficiency against that of the traditional ADR algorithm and the related algorithms presented in Section 2.2. Data was collected using network simulations via LoRaSim, "a discrete-event simulator based on SimPy for simulating collisions in LoRa networks and to analyse scalability" [12]. The end devices in the network setup were varied in steps of 100 from 100 to 2000. This was done to represent an increase in network size and allowed the researchers to study the performance implications of the algorithm in varying network sizes. The European ISM band was chosen for the network simulation setup. This ISM band was chosen as currently, it is the only band that is supported by all existing implementations of the ADR. The parameters used for the simulations are shown in Table 2 and are specific to the chosen ISM band except for the coding rate. A coding rate of 4/5 was chosen as it is the lowest coding rate available in LoRaSim and it the researchers wanted to ensure that it is kept constant so that it had little to no effect on the achieved evaluation results. In the simulations, the end devices chose transmission parameters: TP and SF based on the algorithms' specifications. In the scenario without the ADR algorithm, the transmission parameters were chosen randomly by the end devices. The simulation scenarios were set up such that all end devices could reach the gateway with their selected transmission parameters. This was done to ensure that each end devices traffic was accounted for in the overall evaluation results. The simulation network setup consisted of a single gateway at the centre of the network with the end devices randomly scattered around it, as illustrated by the simplified topology in Figure 1. In the topology, the end devices were connected to the gateway through a LoRa network with LoRaWAN as the communication protocol, while the gateway, the network server and the application server were connected through an internet protocol backhaul.

The chosen simulation scenario represented a simple sensing application where end devices sense and transmit data to the gateway at 1-h intervals. This type of application was relevant enough for the researchers to use to collect performance data about the different algorithms. In this research experiment, this application involved the end devices sending 20-byte packets to the gateways every 1 h. The end devices and the gateway adhered to the 1% duty cycle required by applications operating in the European ISM band [14]. All simulation scenarios mentioned lasted for one week and were executed ten times as in [8].

The results were reported as averages from the iterations and visualized using graphs. The evaluated parameters from the results were the Data Extraction Rate (DER) and Energy Consumption (EC).

Table 2. Simulation parameters.

Parameter	Value
Carrier Frequency	868 MHz
Bandwidth	125 KHz
Coding Rate	4/5
Transmission Powers	2 dBm to14 dBm
Spreading Factor	7 to 12

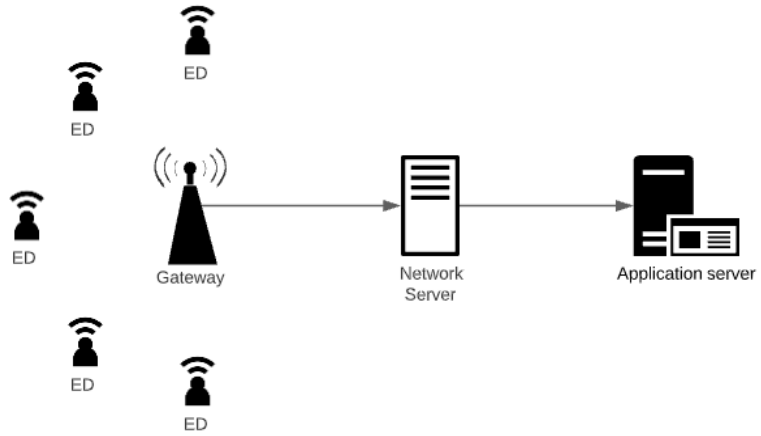

Figure 1. Experimental design.

5. Results

This section presents the results of the simulated experimental study performed to carry out the objectives presented in the introduction and to answer the following research questions:

(1) How can a modified Adaptive Data Rate algorithm be developed to efficiently manage and reduce signal interferences in Long-Range Wide Area Networks?
(2) How effective are the proposed modifications to the Adaptive Data Rate algorithm in reducing signal interferences in Long-Range Wide Area Networks?

DER is the ratio of traffic sent by end devices to the traffic received by the gateway. The value ranges from 0 to 1 and the closer the value is to 1, the better the DER. At 100 end devices in the network, the results obtained for the proposed algorithm are closely matched with those of the other algorithms presented in Section 2.2. At this point, the proposed algorithm's performance displays a slight superiority over the rest of the algorithms, achieving a DER of 0.97, a value that is 0.02 units higher than that of the closest competition from ADR-Dense. Overall, the performance relationship displayed by the results is an inverse proportionality relationship. The performance of the proposed algorithm starts high when the network size is small and exponentially degrades as the network size increases. Throughout all network sizes, the proposed algorithm has shown superior performance over the rest of the other algorithms including the native ADR and network with no ADR. As the network size continues to increase, the performance of the proposed algorithm starts to pull away from the rest, degrading less and less compared to the rest of the algorithm. ADR-Dense and FADR, which are the closest competitors to the proposed algorithm, begin to achieve DER results below 0.5 at about 1200 end devices. At this point, the proposed algorithm still achieved an ADR value of over 0.6. What is significant from the results presented in Figure 2 is that throughout all the network sizes studied, the proposed algorithm managed to achieve a DER of over 0.5. At the maximum number of end devices,

the DER achieved by the proposed algorithm stands at about 0.52. Surprisingly, the closest result to this is that of a network with no ADR algorithm employed and is at 0.36. That means at 2000 end devices, the proposed algorithm achieves a 44% better performance over its closest competitor. Overall, the proposed algorithm has shown superior performance over the discussed algorithms in Section 2.2, the ADR and the network with no ADR.

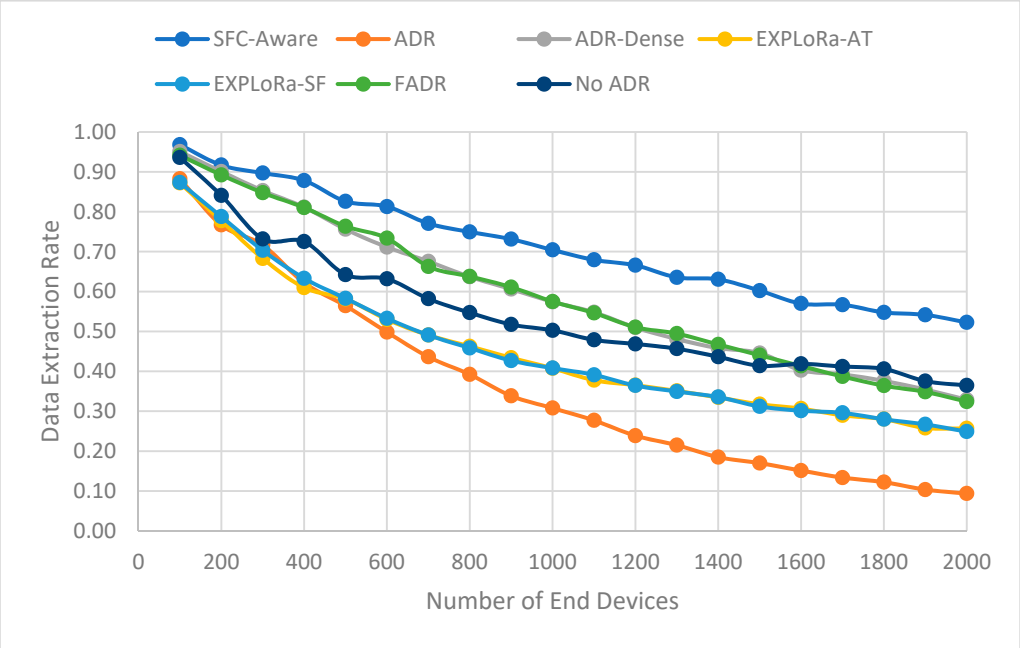

Figure 2. Data extraction rate as the network size increases.

As can be seen from Figure 3, the proposed algorithm's energy consumption is directly proportional to the size of the network. As the number of end devices in the network increases so does the network's overall energy consumption. At a hundred end devices, the proposed algorithm's energy consumption is very low and closely matched with that of both the ADR and the algorithms presented in Section 2.2. From the results, EXPLoRa-SF and EXPLoRa-AT were the worst performing algorithms from the experiment, achieving the highest energy consumption in the network. At 2000 end devices, both algorithms had energy consumptions 700 and 450 per cent higher than that of the native ADR and the proposed algorithm, respectively. The algorithms whose performances were the closest to those of the native ADR and the proposed algorithm were FADR and ADR-Dense. However, at 2000 end devices, both algorithms still managed to consume energies that were 200 and 100 per cent higher than those of the native ADR and the proposed algorithm, respectively. Although the proposed algorithm had lower energy consumptions as compared to the other algorithms across all network sizes, overall, the EC performance of the algorithm was still inferior to that of the native ADR, which had the lowest energy consumptions across all network sizes as compared to the rest of the algorithm. However, the proposed algorithms energy consumptions were not too high. In fact, at 2000 end devices, the algorithm's energy consumption was a little over 40 per cent higher than that of the ADR.

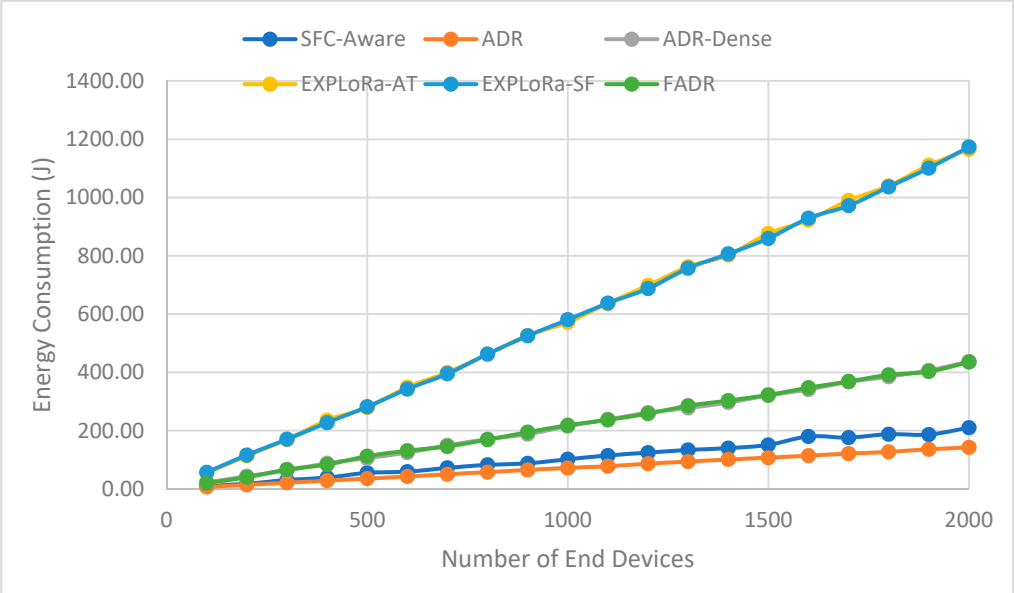

Figure 3. Energy consumption rate as the network size increases.

6. Discussion

6.1. Data Extraction Rate

The goal of the proposed algorithm was to reduce collisions in LoRaWAN by using SF congestion statuses to determine the optimized data rates for end devices with ADR enabled. It is important to note that in a network like LoRaWAN it is impossible to eliminate collisions, even with network management mechanisms such as the ADR or any versions thereof, especially in dense networks. That is because LoRaWAN has a limited number of transmission parameters that these mechanisms can manipulate to give network traffic the best chances of successful transmission. And as network sizes increase, the number of potential colliders with every transmission also increases. Therefore, how well an algorithm performs in terms of DER is judged based on the extent to which the algorithm reduces the number of collisions when it is employed in a LoRaWAN network. The DER results presented in Section 4 indicate that the proposed algorithm has an overall superior performance over both the native ADR and the related algorithms presented in Section 2.2.

This helped the authors to answer research question 2: the proposed algorithm has proven to be highly effective in mitigating the occurrence of collisions in LoRaWAN, achieving the highest DER of all the algorithms that were under investigation in all networks sizes. In small network sizes, the proposed algorithm performed closely with the other algorithms. However, as the network size continued to grow, the performance of the proposed algorithm became more and more superior. When the DERs of the other algorithms had dropped below 0.5, that of the proposed algorithm was above 0.7. That is a 40 per cent or better performance than the other algorithms. This shows the significant performance boost the proposed algorithm can bring to LoRaWAN.

Surprisingly, at the largest network size (of 2000 end devices) the proposed algorithm managed to achieve a DER above 0.5 and still maintain a 40% superior performance over the rest of the algorithms. This appealing performance of the proposed algorithm in dense network settings is owed to how it performs data rate optimization. The algorithm keeps track of the congestion statuses of each SF. And when optimizing the data rate for an end device, it uses the congestion statuses to find out which of the computed optimized data rates uses the SF with the lowest congestion status. This ensures that the end device's traffic experiences the lowest collision probability possible. And while the data rate may not be the most efficient at times, this is compensated by better transmission reliability.

6.2. Energy Consumption

Low energy consumption is essential in LoRaWAN as most of the end devices are battery powered and are expected to survive on those batteries for years. As such, even though the focus of the proposed algorithm was on DER, it was still essential that the authors measure its EC performance to evaluate and study its implications on the overall LoRaWANs EC and end devices' battery lives. The overall EC results indicate that the proposed algorithm has a superior performance over the algorithms discussed in Section 2.2. Despite this, the results also indicate that the algorithm is still outperformed by the native ADR. The inferior performance against the native ADR is however expected as the proposed algorithm was not designed to improve EC but rather DER.

As it can be seen from Figure 3, the relationship displayed by the results is a directly proportional one. As the network size increases, the overall network EC also increases. This behaviour is expected as more end devices in the network lead to more contributors to the overall network EC. Also, an increase in network EC cannot be eliminated as end device transmissions depend on energy consumptions. Furthermore, higher energy usages are essential for achieving long-distance transmissions without corrupting the transmitted traffic. Lastly, end devices depend on increasing transmission powers to try and regain connectivity to the network server.

Even though the native ADR outperformed all the algorithms it was compared with, the proposed algorithm still managed to achieve reasonable EC performances closest to those of the native ADR. For small network sizes, the EC performance differences between the proposed algorithm and the native ADR were barely noticeable. From the experimental results, it is evident that the performance differences become most noticeable at 2000 end devices in the network. At this network size, the proposed algorithm consumed 70 joules more energy than the native ADR. The extra energy consumption by the proposed algorithm was to be expected as, in addition to computing optimized data rates for end devices, it also must go a step further and ensure that the data rates provide the end devices' traffic with optimized collision probabilities. Therefore, considering this reason, it can be argued that the proposed algorithm performed desirably. Especially because the algorithm was only built to improve DER.

7. Conclusions

ADR, which is the resource allocation mechanism currently being employed in the LoRaWAN, has proven to be quite immature and poor performing, especially in dense network scenarios. This paper proposes a modified version of ADR and presents an empirical analysis of its performance in terms of DER and EC. The evaluation was performed using simulations to study and critically analyze the proposed algorithm's performance implications on LoRaWAN as network sizes increase and how well the algorithm performed against the native ADR and four other related algorithms proposed by different researchers. The results have shown that when network sizes are small, the proposed algorithm and the algorithms considered in this paper perform fairly well and close to each other. Only when the network sizes get too large that the proposed algorithm's superiority starts to be most noticeable. That is, when the network sizes gradually increase, the proposed algorithm completely outperforms the rest of the algorithms, including the native ADR. At around 1000 end devices in the network, all the other algorithms were achieving DERs that were below 0.5 while the proposed algorithm managed to achieve a DER of about 0.7. That is over 40% better performance compared to all the algorithms that were considered in the experiment. The proposed algorithm continued to maintain this superiority across all network sizes and at 2000 end devices in the network it still managed to achieve a DER that was above 0.5. For EC, the ADR still has superior performance over the proposed algorithm and the other related algorithms that were considered in this research. Of course, that is to be expected as the algorithms are mostly targeted towards improving the DER. Based on the results discussed and arguments made here, the authors can further argue that the ADR performs poorly under dense networks and that although attempts have

been made to improve it, these attempts still need to be supplemented. And conclude that the proposed algorithm builds on the existing algorithms and the ADR and significantly improves them in terms of DER and EC (excluding the native ADR), which is a significant step towards an ideal implementation of LoRaWAN's ADR.

In the future, the authors plan to build the algorithm's support for other ISM bands as opposed to just the European ISM band as it is the only band that is currently supported by ADR algorithms. Furthermore, the authors plan to improve the proposed algorithm to consider EC during transmission parameter optimizations. This could improve the algorithm's overall performance and nudge it closer to the ideal implementation of a network management mechanism.

Author Contributions: Conceptualization, C.L. and B.I.; methodology, C.L. and B.I.; software, C.L., and B.I.; validation, C.L., and B.I.; C.L. and B.I.; formal analysis, investigation, resources, and data curation, C.L.; writing—original draft preparation, C.L.; writing—review and editing, B.I.; visualization, C.L.; supervision, B.I. and F.L.; project administration and funding acquisition, A.A.-M. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data presented in this study are available on request from the corresponding author.

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

References

1. Li, S.; Raza, U.; Khan, A. How Agile is the Adaptive Data Rate Mechanism of LoRaWAN? In Proceedings of the 2018 IEEE Global Communications Conference (GLOBECOM), Abu Dhabi, United Arab Emirates, 9–13 December 2018; pp. 206–212. [CrossRef]
2. De Carvalho Silva, J.; Rodrigues, J.J.; Alberti, A.M.; Solic, P.; Aquino, A.L. LoRaWAN—A low power WAN protocol for Internet of Things: A review and opportunities. In Proceedings of the 2017 2nd International Multidisciplinary Conference on Computer and Energy Science (SpliTech), Split, Croatia, 12–14 July 2017.
3. LoRa Alliance, LoRaWAN, What Is It: A Technical Overview of LoRa and LoRaWAN. Available online: https://lora-alliance.org/resource_hub/what-is-lorawan/ (accessed on 21 June 2021).
4. Lehong, C.; Isong, B.; Lugayizi, F.; Abu-Mahfouz, A.M. A Survey of LoRaWAN Adaptive Data Rate Algorithms for Possible Optimization. In Proceedings of the 2020 2nd International Multidisciplinary Information Technology and Engineering Conference (IMITEC), Kimberley, South Africa, 25–27 November 2020.
5. Semtec. *Understanding the Lora Adaptive Data Rate*; Semtec Corporation: Camarillo, CA, USA, 2019.
6. Garlisi, D.; Tinnirello, I.; Bianchi, G.; Cuomo, F. Capture Aware Sequential Waterfilling for LoRaWAN Adaptive Data Rate. *IEEE Trans. Wirel. Commun.* **2020**, *20*, 2019–2033. [CrossRef]
7. Cuomo, F.; Campo, M.; Caponi, A.; Bianchi, G.; Rossini, G.; Pisani, P. EXPLoRa: Extending the performance of LoRa by suitable spreading factor allocations. In Proceedings of the 2017 IEEE 13th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob), Rome, Italy, 9–11 October 2017.
8. Abdelfadeel, K.Q.; Cionca, V.; Pesch, D. Fair Adaptive Data Rate Allocation and Power Control in LoRaWAN. In Proceedings of the 2018 IEEE 19th International Symposium on "A World of Wireless, Mobile and Multimedia Networks" (WoWMoM), Chania, Greece, 12–15 June 2018; pp. 14–15. [CrossRef]
9. Kim, S.; Yoo, Y. Contention-Aware Adaptive Data Rate for Throughput Optimization in LoRaWAN. *Sensors* **2018**, *18*, 1716. [CrossRef] [PubMed]
10. L.A.T. Committee. *LoRaWAN 1.1 Specification. Version no. 1.1.*; L.A.T. Committee: Beaverton, CA, USA, 2017.
11. Semtec. *LoRaWAN—Simple Rate Adaptation Recommended Algorithm: Common Class A/B/C Specification*; Semtec Corporation: Camarillo, CA, USA, 2016.
12. Bor, M.C.; Roedig, U.; Voigt, T.; Alonso, J.M. Do LoRa Low-Power Wide-Area Networks Scale? In Proceedings of the MSWiM '16: The 19th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems, Malta, Malta, 13–17 November 2016; pp. 59–67. [CrossRef]
13. Reynders, B.; Meert, W.; Pollin, S. Power and spreading factor control in low power wide area networks. In Proceedings of the presented at the 2017 IEEE International Conference on Communications (ICC), Paris, France, 21–25 May 2017.
14. LoRa Alliance, LoRaWANTM 1.1 Regional Parameters. Available online: https://lora-alliance.org/resource_hub/lorawan-regional-parameters-v1-1ra/ (accessed on 21 June 2021).