Analysis and Enhancement of the LoRaWAN Adaptive Data Rate Scheme

Joseph Finnegan, *Student Member, IEEE*, Ronan Farrell, *Member, IEEE*, and Stephen Brown

***Abstract*—The adaptive data rate (ADR) algorithm is a key component of the LoRaWAN protocol which controls the performance of a LoRaWAN Network. This modifies the data rate parameter of a device based on the current wireless conditions. In this article, we present substantive enhancements for the End Device and Network Server which reduce the convergence time for LoRaWAN devices to reach their optimal data rate. We extend the LoRaWAN module in ns-3 by adding ADR, enabling the simulation of realistic LoRaWAN networks, and add the implementation of the new enhancements in this module. The simulations show that these modifications can result in a significant reduction of the data rate convergence time for LoRaWAN devices and lead to an increased overall packet delivery rate for the network in a dynamic network environment. Our contributions are a publicly available implementation of ADR in ns-3, an analysis of the original algorithm behavior, and a novel version of the algorithm with enhancements that improve performance in every case while remaining easily integrable into an existing LoRaWAN system.**

***Index Terms*—Adaptive data rate (ADR), LoRaWAN, low-power wide-area network (LPWAN), ns-3, scalability.**

I. INTRODUCTION

THE NEW wireless paradigm of low-power wide-area networks (LPWANs) provides energy-efficient, long-range connectivity at a low cost, enabling the development of new pervasive Internet-of-Things applications. The performance target of LoRaWAN, a key LPWAN technology, is for a single LoRaWAN gateway to handle the traffic of thousands of low-power devices. The performance of a LoRaWAN network is controlled by the adaptive data rate (ADR) algorithm, which modifies the data rate of a device based on the current network conditions. The optimal choice of the data rate for transmissions ensures that devices operate in an energy-efficient manner and increases the maximum number of devices that can be managed by a single gateway.

In this article, we extend the ns-3 module introduced in [1] to implement ADR. As LoRaWAN networks are not realistically scalable without ADR [2], any thorough simulation of the protocol requires an implementation of the algorithm. In our analysis, we find that the two concurrently running sides of the ADR algorithm do not converge at the same rate; the Network Server-side converges faster. In addition, the two sides do not necessarily converge a device to the same data rate. The Network Server-side algorithm is not explicitly defined in the LoRaWAN protocol specification, which results in some cases where the LoRaWAN devices fail to converge to a suitable data rate. Furthermore, we propose and implement modifications to the ADR scheme. These modifications fit within the existing mechanism and do not require any overhaul of the existing LoRaWAN protocol. They result in the reduction of the convergence time of the LoRaWAN nodes in reaching a steady data rate. In addition, the modifications also result in devices converging to data rates that result in a higher overall packet delivery rate (PDR) for the network.

This article is outlined as follows. Section II gives an overview of LoRaWAN. Section III describes related work. Section IV describes, in particular, the ADR algorithm. Section V provides the simulation results and analysis, describes limitations of the algorithm, and proposes improvements. Section VI evaluates these proposed enhancements in comparison to the original algorithm. Section VII summarizes our contributions and concludes this article.

Manuscript received November 20, 2019; revised March 3, 2020; accepted March 17, 2020. Date of publication March 23, 2020; date of current version August 12, 2020. This work was supported in part by the Science Foundation Ireland through the European Regional Development Fund under Grant 13/RC/2077. *(Corresponding author: Joseph Finnegan.)*

Joseph Finnegan and Stephen Brown are with the Department of Computer Science, Maynooth University, W23 F2H6 Maynooth, Ireland (e-mail: joseph.finnegan@mu.ie).

Ronan Farrell is with the Department of Electronic Engineering, Maynooth University, W23 F2H6 Maynooth, Ireland.

Digital Object Identifier 10.1109/JIOT.2020.2982745

II. LoRaWAN

LoRa is an LPWAN technology developed by Semtech which targets the Internet-of-Things applications with energy constraints and long-range requirements. The LoRa uses a form of frequency-shifted chirp spread spectrum modulation, and allows a tradeoff between range and bitrate which can be manipulated through a choice of four factors: 1) spreading factor; 2) bandwidth; 3) coding rate; and 4) transmission power. LoRaWAN is a MAC layer for LoRa developed and maintained by the LoRa Alliance. A LoRaWAN network is structured in a star-of-stars topology, where individual devices communicate purely with LoRaWAN gateway nodes via LoRaWAN transmissions, and gateway nodes act purely as relays to a central Network Server. LoRaWAN networks in Europe are deployed in the 433- and 868-MHz ISM bands, which are license-free but mandate the adherence to duty cycle limits. The duty cycle limits for the 868-MHz band are shown in Table I.

The LoRaWAN defines three different device classes. Class A is for devices with energy constraints and minimal downlink requirements; devices sent uplink frames via

TABLE I
SPECTRUM ACCESS PER SUB-BAND [3]

Sub-band	Spectrum Access	Edge Frequencies	Max EIRP
g	1 %	865-868MHz	10mW
g1	1 %	868-868.6MHz	25mW
g2	0.1%	868.7-869.2MHz	25mW
g3	10 %	869.4-869.65MHz	500mW
g4	No Limit	869.7-870MHz	5mW
g4	1 %	869.7-870MHz	25mW

TABLE II
LORAWAN DATA RATES FOR EU863-870

Data Rate	Spreading Factor	Bandwidth	Bit Rate	Receiver Sens.
DR0	12	125kHz	250	-136dBm
DR1	11	125kHz	440	-133dBm
DR2	10	125kHz	980	-132dBm
DR3	9	125kHz	1760	-129dBm
DR4	8	125kHz	3125	-126dBm
DR5	7	125kHz	5470	-123dBm
DR6	7	250kHz	11000	-120dBm

ALOHA, and only receive downlink frames as responses to uplink frames. Class B extends Class A to include the reception of beacon frames at predetermined intervals and adds the potential of reception of server-initiated downlink frames at predetermined intervals. Class C devices remain in receive mode when not transmitting, and so this class is not suitable for the devices with energy constraints. LoRaWAN organizes the bandwidth and spreading factor options in LoRa transmissions into defined data rates (see Table II).

As the data rates provide different bit rates and are quasi-orthogonal [4], the optimal choice of data rate is essential in extending the lifetime of individual devices and the scalability of the network as a whole. The LoRaWAN specification defines an ADR scheme that enables devices to modify the chosen data rate based on the recent traffic conditions. A full description of the LoRaWAN ADR is provided in Section IV.

III. RELATED WORK

Implementations and analysis of ADR have previously been undertaken in LoRaWANSim [5], MATLAB [6], and OMNET++ [7]. It has been found that network convergence time increases greatly with network size and that the current ADR algorithm is effective in stable channel conditions but not in highly variable conditions. In addition, it has been found that the performance of the algorithm is ultimately limited in scale by the duty cycle regulations and that in a lossy channel, the End Device-side algorithm convergence time to an optimal state is very slow.

The current NS-side ADR algorithm allocates the data rate to a device based purely on uplink frames received from that particular device; and so, data rate allocation approaches using global network knowledge have been investigated, which overhaul the existing LoRaWAN ADR. Different methods have attempted to achieve greater scalability through aiming to achieve an equal collision probability across nodes [8], through overall throughput maximization [9], through average system packet success probability maximization [10], through the balancing of the link load across channels [11], and through the optimization of the packet error rate of edge nodes [12]. Other approaches considered are the equal allocation of time-on-air to nodes (while also taking into account detected collisions of neighboring nodes) [4], the use of message replicas to minimize the outage probability while avoiding high collision probabilities and the otherwise use of higher Data Rates [13], the consideration of inter-SF and co-SF interference in data rate allocation [14], and the use of mesh topologies in longer range networks to enable the avoidance of the use of higher data rates [15].

The alternative beacon-based approaches which remove the need for the ADR algorithm have also been considered. In [16], the data rates are allocated to the devices based on the RSSI from a previous uplink. Then, the devices are grouped by their data rate, and the grouped devices transmit simultaneously and are packed in one group acknowledgment. In [17], the beacon frames are introduced to indicate the allowed data rates and RSS limit of each channel for the following beacon period. This effectively groups the devices by distance from the gateway, minimizing the capture effect. A different approach to communications optimization is the game-theoretical solution used to optimize the use of LoRaWAN as a fail-over communications channel for public safety networks [18]. This contrasts with our work in that it takes a multiprotocol approach rather than enhancing the capabilities of LoRaWAN.

IV. MODELING OF THE ADAPTIVE DATA RATE SCHEME

The ADR is used to optimize the data rate of stationary LoRaWAN nodes. The use of ADR is optional but highly recommended by the LoRaWAN protocol specification [19], and LoRaWAN networks do not realistically scale without the use of ADR [2]. The ADR is split into two independent algorithms, one of which runs on the device (hereby referred to as the End Device-side algorithm, or ED-side algorithm), and the other which runs on the Network Server (hereby referred to as the Network Server-side algorithm, or NS-side algorithm).

A. ADR—End Device-Side

The End Device-side algorithm is explicitly defined in the LoRaWAN protocol specification [19]. The algorithm reacts to a lack of requested downlink feedback by decreasing the data rate (and thus increasing the range). Thus, on the End Device-side, the data rate is only decreased (and therefore slowed). The pseudocode of this algorithm is provided in Algorithm 1. If an End Device is not currently using the slowest available data rate, a device requests a downlink response after ADR_ACK_LIMIT uplink frames by setting the *ADRAckReq* bit in the LoRaWAN header. For each ADR_ACK_DELAY frames sent without a downlink response, the data rate of the device is decremented. Upon receipt of any downlink frame, the *ADRAckReq* bit is unset and all counters are cleared. By

Algorithm 1 End Device-Side ADR

```
1: on downlink frame receive:
2: counter ← 0.
3: adr_ack_req_bit ← 0.
4: on uplink frame send:
5: if data_rate ≠ DATA_RATE_MIN then
6:     counter ← counter + 1.
7:     if adr_ack_req_bit = 0 and counter = ADR_ACK_LIMIT then
8:         adr_ack_req_bit ← 1.
9:     else if counter = ADR_ACK_LIMIT + ADR_ACK_DELAY then
10:        counter ← ADR_ACK_LIMIT
11:        if tx_power ≠ TX_POWER_MAX then
12:            tx_power ← TX_POWER_MAX
13:        else if data_rate ≠ DATA_RATE_MIN then
14:            data_rate ← data_rate − 1
15:        else
16:            reenableDefaultUplinkChannels()
```

TABLE III
SNR_TABLE(DR)

Data Rate	Required SNR (dBm)
DR0	-20
DR1	-17.5
DR2	-15
DR3	-12.5
DR4	-10
DR5	-7.5

default, ADR_ACK_LIMIT and ADR_ACK_DELAY are both equal to 32.

B. ADR—Network Server-Side

The second algorithm runs simultaneously on the Network Server. There is no official version of the algorithm defined in the LoRaWAN protocol, but Semtech does provide a recommended algorithm [20], which has been adopted by open source projects, such as the Things Network [21] and the LoRa Server [22], and has been the assumed algorithm for previous research analyzing ADR [5], [7], [8]. In this algorithm, the Network Server records the highest SNR value for each incoming packet, and then, if the end device is not already using the fastest data rate and the lowest transmission power, it calculates the expected most suitable data rate for the device. The new data rate is calculated based on the previously received SNR values, relative to the current data rate (see Table III) and transmission power level (see Table IV). An added margin_db parameter (usually set to 5 dB) prevents the oscillation between data rates. The new data rate and transmission power for the device is communicated through the *LinkADRReq* MAC command in a downlink frame, and this is acknowledged with the piggybacking of a *LinkADRAns* MAC command in the device's next uplink frame. This algorithm is described in the pseudocode in Algorithm 2. Note that this algorithm only increases the data rate.

TABLE IV
TX POWER TABLE

tx_power	Configuation (EIRP)
0	Max EIRP
1	Max EIRP - 2dB
2	Max EIRP - 4dB
3	Max EIRP - 6dB
4	Max EIRP - 8dB
5	Max EIRP - 10dB
6	Max EIRP - 12dB
7	Max EIRP - 14dB
8..14	Reserved For Use
15	Defined in LoRaWAN

Algorithm 2 Network Server-Side ADR

```
1: on uplink frame receive (device, packet):
2: if length(device.packets) = 20 then
3:     packets.pop()
4: device.packets.push(packet.snrMax).
5:
6: on downlink frame send (device, packet):
7: if packet.frameCounter%20 = 0 then
8:     snr_max ← max(device.packets.snrMax)
9:     snr_margin ← int(snr_max − SNR_TABLE(data_rate) − margin_db)
10:    n_step ← snr_margin/3
11:    while n_step ≠ 0 do
12:        if n_step > 0 then
13:            if data_rate < 5 then
14:                data_rate ← data_rate + 1.
15:            else
16:                if tx_power ≠ TX_POWER_MIN then
17:                    tx_power ← tx_power + 1.
18:            n_step ← n_step − 1.
19:        else
20:            if tx_power ≠ TX_POWER_MAX then
21:                tx_power ← tx_power − 1.
22:                n_step ← n_step + 1.
```

The frequency of the NS-side algorithm calculates the most suitable data rate for each device that is not specifically defined in the Semtech document, but two recommendations are provided: 1) to run the algorithm either every time the *ADRAckReq* bit is set by a device or 2) to run the algorithm once every Nth uplink frame from a device. The second approach (with $N = 20$) was taken in the implementation of the Things Network and the LoRa Server, and in [5] and [7], and as such is the approach taken here. In addition, the format of the *LinkADRReq* MAC command is region specific; the implementation described in this article is for the EU868 band. In all other implementation-level differences, unless otherwise stated, the approach taken was similar to that in the LoRa Server.

C. Data Rate Change Latency

The time to change down from a data rate *drx* to a slower data rate *dry* is based on the ED-side algorithm and the transmission rate of the LoRaWAN device (t_{period}), and in the best case can be calculated as

$$t_{drx}^{dry} = 64 * \left[t_{\text{period}}\right] + \sum_{d=dry}^{drx-1} 32 * \left[t_{\text{period}}\right]. \quad (1)$$

The latency is also dependent on the relative activity of the nearest LoRaWAN gateway, as when the device finally begins transmitting at the most suitable data rate, a downlink frame is required from the Network Server to inform the device to maintain this data rate. However, the gateway also has to adhere to the duty cycle limits and as such may not be able to respond to the first request of a downlink frame seen.

The time to change from a data rate *drx* to a faster data rate dry is based on the NS-side algorithm and in the best case can be calculated as

$$t_{drx}^{\text{dry}} = 20 * \left[t_{\text{period}}\right]. \quad (2)$$

Similarly to the other case, this latency is also dependent on the relative activity of the nearest LoRaWAN gateway, as the gateway has to adhere to the duty cycle limits and as such may not be able to send the downlink MAC commands in response to the first uplink frame seen from the device since the running of the algorithm.

Note that the minimum value of t_{period} is restricted by the need to adhere to the duty cycle regulations of the EU868 band, and thus is dependent on the data rate and the channel allocation of the device.

These equations describe the latency behavior of ADR and their effect on large-scale LoRaWAN networks is explored next through simulation.

V. Simulation and Analysis of Standard ADR

The simulator used in the analysis was the ns-3 LoRaWAN module introduced in [1], with the MAC layer modified to include an implementation of the ADR algorithm. We have released the modified module for public use on GitHub.[1] The LoRa PHY layer error model in [1] is based on measuring the bit error rate for different configurations over an AWGN channel. The packets received below a cutoff SINR value (which includes thermal noise), based on the data and coding rate, are immediately discarded. A chunk-based approach to packet reception is taken, similar to the modeling of IEEE 802.11 and IEEE 802.15.4 in ns-3. In this approach, every time the SINR changes during the reception of a packet, the reception of the bits since the last SINR change is evaluated, based on the error model and SINR (with the ns-3 default propagation loss model, *LogDistancePropagationLoss*, used). If the reception of any chunk fails, the packet is failed to be received. This approach models propagation loss as well as enabling the modeling of interference from LoRa transmissions as well as the transmissions from other devices operating in the same band. The simulator also ensures that devices and gateways adhere to the duty cycle regulations of the EU868 band. As shown in [23], LoRa exhibits a remarkable immunity to multipath interference.

[1]https://github.com/ConstantJoe/ns3-lorawan-adr

TABLE V
Simulation Parameters

Gateways	1, 4
End Devices	100, 500, 1000, 2500
Disk Radius	5000m
Uplink Period	600s, 3600s
Downlink Period	No data plane DL
Packet Size	8 bytes (excluding header)
Random Streams	5
Simulation Time	250 * Uplink Period, or till convergence

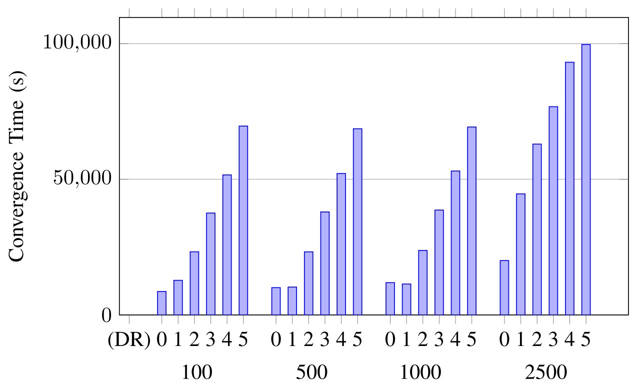


Fig. 1. Average node convergence time for varying starting data rates and network sizes, uplink period = 600 s.

The simulations consist of a LoRaWAN network, where *N* LoRaWAN Class A nodes are equally distributed across a disk of radius 5 km. The key parameters in the simulations are outlined in Table V. The devices transmit their first frame at a random time between 0 and Uplink_Period, and then transmit once every Uplink_Period.

A. Effect of Starting Data Rate on Convergence Time

In each simulation, every device begins transmitting using the same data rate. Fig. 1 shows the amount of time for every device in the network to reach a steady data rate using the ADR algorithm, for each starting data rate case. The data from simulations with a network size of 100, 500, 1000, and 2500 devices are provided. It can be seen that the convergence time for all devices to reach a steady data rate is dependent both on the starting data rate and the number of devices in the network. As discussed in Section IV, the ADR is composed of two separate algorithms that run concurrently: 1) the NS-side algorithm only increases the data rate of a device and 2) the ED-side algorithm only decreases the data rate. As such, the simulations where all devices begin using *DR*0 is the case where the data rate of all devices (at least initially) are manipulated by the NS-side algorithm. Similarly, the ED-side algorithm manipulates the data rate of all nodes in the simulations where all nodes begin using *DR*5.

Using this information, from Fig. 1, it can be seen that the two algorithms do not converge at the same rate; the NS-side algorithm converges much faster than the ED-side algorithm. The starting data rate of a device does have a large effect on the convergence time, with the NS-side algorithm converging

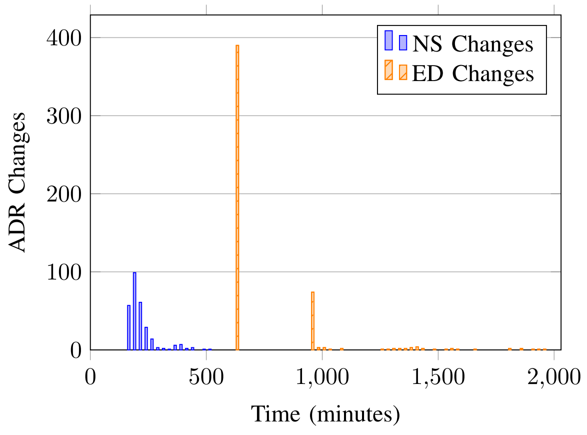


Fig. 2. ADR changes across time, starting DR = 2, uplink period = 600 s, 1000 nodes.

to the ideal data rate much quicker than the ED-side. This is intuitive when we consider the structure of the two algorithms (see Algorithms 1 and 2): the NS-side algorithm may modify the data rate of a device to use any faster rate; conversely, the ED-side algorithm only changes the data rate of a device one step at a time.

Highlighting this is Fig. 2, which shows the number of ADR changes across time for a 1000 device network (where all nodes initially transmit using DR2). The bars in blue show the number of devices that changed the data rate because of the NS-side ADR algorithm, at each point in time. The bars in orange show the number of devices that changed the data rate because of the ED-side ADR algorithm. The ED-side changes occur in time in steps of ADR_ACK_DELAY * uplink_period. Conversely, the NS-side changes begin to occur after N uplink frames are received. The choice of a particular N value is quite arbitrary and in some situations, it may become clear quickly that the data rate of a device should be increased. One case, in particular, is when a device first joins the network: in the join procedure, any data rate can be used, and whichever data rate is used in the join procedure becomes the first set data rate of the device. So, when a device first joins the network, it is possible that the device is using a much slower data rate than required.

The ED-side ADR algorithm is slower to converge as it has been designed to minimize the amount and maximize the flexibility of the control plane downlink traffic, which is limited by the duty cycle regulations applied to LoRaWAN devices, including LoRaWAN gateways. The design of the algorithm enables a greater number of devices to be handled by a single LoRaWAN gateway in the general case (when a device is currently using the most suitable data rate) and enables nodes that lose all connection to the gateway to eventually re-establish reliable communication by gradually stepping up the data rate. This flexibility is at the expense of a slow convergence time in the worst case (i.e., when a node is currently using a much lower range data rate than needed to reach the nearest gateway).

Overall, in the general case, the NS-side algorithm converges to the ideal data rate for a device at a faster rate. Therefore, a reasonable strategy can be to attempt to join the network using a conservative data rate and allow the node to converge to the ideal data rate itself, rather than attempting to join the network using the fastest data rate possible and potentially ultimately joining the network using a data rate that cannot allow the node to reliably communicate with the nearest gateway.

TABLE VI
PERCENTAGE OF DEVICES SETTLING TO EACH DATA RATE, $N = 1000$, UPLINK PERIOD = 600 s

	Starting Data Rate	
Ending Data Rate	**DR0**	**DR5**
0	36%	8.3%
1	23.4%	32.5%
2	14.2%	20.9%
3	8.7%	12.6%
4	5.5%	8.3%
5	12.2%	17.4%

TABLE VII
PDR AND AVERAGE CURRENT CONSUMPTION OF DEVICES AFTER CONVERGENCE, $N = 1000$, UPLINK PERIOD = 600 s

Starting DR	**Average PDR**	**Average Curr. Cons.**
DR0	90.79%	$73.74\mu A$
DR1	83.53%	$58.60\mu A$
DR2	80.23%	$54.32\mu A$
DR3	78.95%	$52.25\mu A$
DR4	77.18%	$50.96\mu A$
DR5	75.85%	$49.05\mu A$

B. Effect of Starting Data Rate on Final Data Rate

In addition to having a direct effect on the data rate convergence time of devices, the starting data rate has an effect on the final data rate of a device; i.e., the ED-side and NS-side algorithms do not necessarily converge to the same data rate assignment. Fig. 3(a) and (b) displays a top-down view of the network, with the single LoRaWAN gateway located at the center and the location of the individual devices marked according to the final steady data rate of each device. These figures show the difference in data rate assignment for two simulations, one of which all devices begin transmitting at $DR0$, and the other transmitting at $DR5$. Table VI shows the percentage of nodes using each data rate at convergence for these two simulations.

These figures show that the NS-side algorithm converges devices to a slower, higher range but less energy-efficient data rate than the ED-side algorithm. Table VII shows the average PDR and current consumption results from a set of six similar simulations. The energy consumption results make use of an ns-3 extension previously developed by Finnegan *et al.* [24], and models the energy consumption of the transceiver only (assumed to be the SX1272). The average current consumption for each data rate appears low; note that this is an average across time and even in the worst case, LPWAN devices spend $> 99\%$ of the time in sleep mode. The NS-side algorithm converges devices to a data rate that provides a higher PDR,

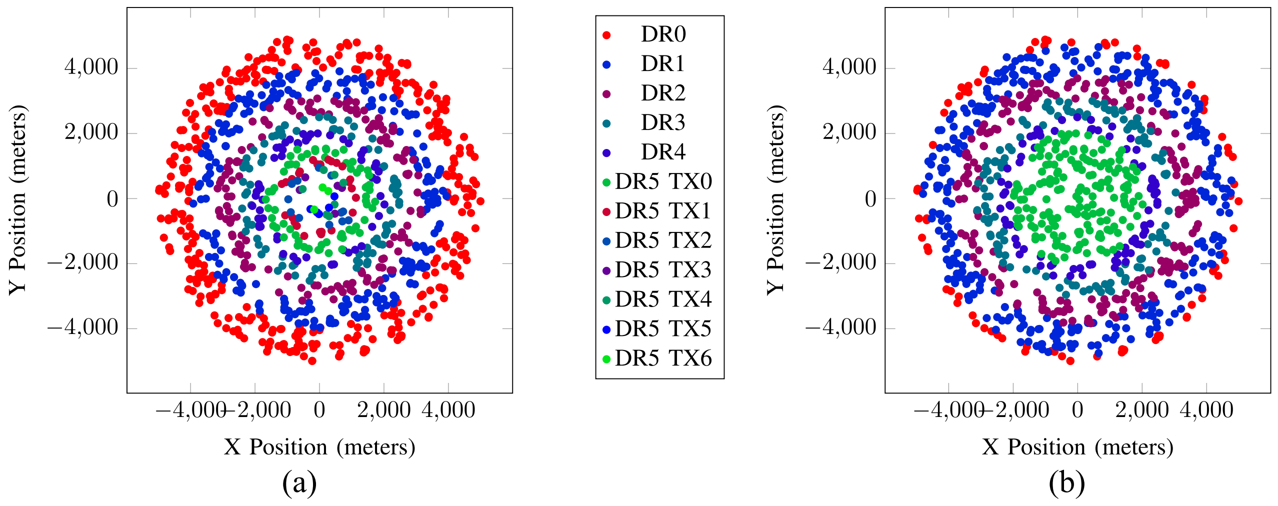


Fig. 3. Map of final data rate versus device location of different starting data rates. (a) 1000 devices, starting DR = 0. (b) 1000 devices, starting DR = 5.

but also a higher energy consumption. The poorer quality of the converged ED-side links is a result of the method used in the ED-side algorithm in maintaining links: the algorithm only requires one of a potential 32 uplink frames to be received and responded to; though, as previously discussed, this does enable flexibility and greater scalability, it also does not take into account the quality of the link when decided to maintain that current data rate. Therefore, the ED-side algorithm can converge devices to a data rate that provides a poor quality link, as the device cannot distinguish between a somewhat lossy link and a completely loss-less link.

As the time to transmit the same LoRaWAN packet using different data rates is $\propto 2^{12-\mathrm{DR}}$, there is a significant difference in the energy consumption of the transmission of two frames using two different data rates. However, since confirmed LoRaWAN frames are not scalable [25], either the use of a slower-than-necessarily required data rate to maintain a higher PDR, or the sending of redundant frames (i.e., *NbTrans* > 1) is advisable instead of the fully confirmed traffic. Naive approaches to this can have a cascading negative effect on the PDR of devices because of the increased probability of collisions and is complicated by the fact that data rates are quasi-orthogonal [4]. This has led to the proposal of overhauls of the ADR to perform assignment of data rates using global knowledge. Current approaches have been outlined in Section III but are otherwise out of scope for this article.

C. Issue in the Things Network Implementation

As previously mentioned, the NS-side algorithm is not explicitly defined in the LoRaWAN protocol specification. This has led to differences in the available implementations of the Network Server. In LoRa Server, if there is a MAC-layer message queued to be sent to an End Device, if at the time of one of the device's downlink receive windows there is no data queued to be sent to that device, a new downlink frame with no data payload is generated and the MAC-layer message is piggybacked in that downlink frame. However, in the Things Network, in the same situation, a new downlink frame is *not* generated, and the MAC-layer message is delayed to be sent in the next downlink frame for that device. In devices without downlink data requirements, this results in MAC-layer messages only being sent in downlink frames after the *ADRAckReq* bit is set in a preceding uplink frame, as these are the only downlink frames.

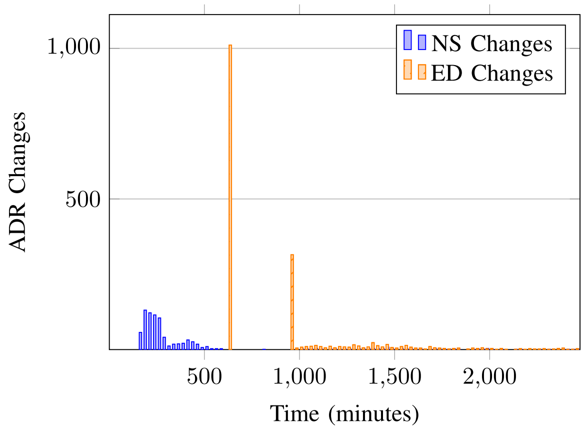


Fig. 4. ADR changes across time, starting DR = 2, uplink period = 600 s. $N = 2500$, single gateway.

For devices using *DR*0, this has the consequence that in applications with no downlink data requirements, no downlink frames of any kind will ever be sent, as in the current implementation of the ED-side ADR the *ADRAckReq* bit is never set for nodes using *DR*0 (see line 5 of Algorithm 1). As a result, no MAC-layer messages are ever sent, and thus no data rate change ever takes place no matter the quality of the link, limiting the energy efficiency and scalability of the network.

D. Effect of Multiple Gateways

Fig. 4 shows the equivalent ADR change graph for a 2500 device simulation, also with a starting data rate of DR2. The peak of the NS-side changes becomes limited both by the duty cycle regulations that the gateway must adhere to, and simultaneously arriving ADR requests. The final node rebalancing

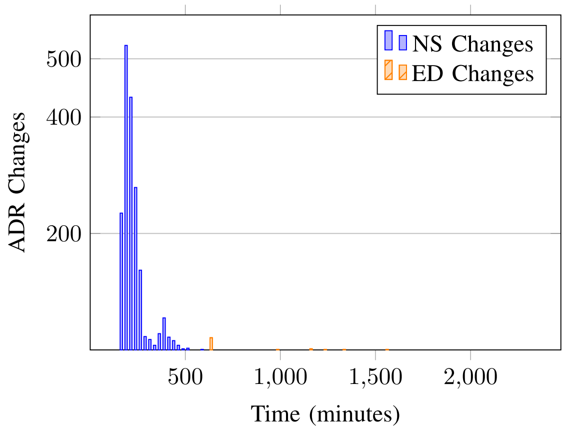


Fig. 5. ADR changes across time, starting DR = 2, uplink period = 600 s. $N = 2500$, multiple gateways.

seen in earlier equivalent graphs is extended in time much further as the probability of node transmissions consistently colliding in time and space is increased.

Fig. 5 shows the equivalent graph for the multigateway case, where instead of one gateway located at (0, 0), in the network there exist four gateways at (3000, 3000), (3000, −3000), (−3000, 3000), and (−3000, −3000). It can be seen that the average convergence time drops across each simulation in comparison to the single gateway case. This is because of the dual factors of 1) devices are less likely to be located far away from a gateway, so $DR0$ and $DR1$ are used less frequently and 2) the increased number of gateways increases the amount of time the Network Server can send downlink frames, enabling ADR commands to be sent faster. As the comparison between Fig. 6(a) and (b) shows a greater density of gateways enabling more devices to use faster data rates. This also enables the gateways to use faster data rates in response, enabling a larger amount of traffic to be sent per gateway. Note that in LoRaWAN, the gateways act purely as relays to the Network Server, so the addition of more gateways does not complicate the management of a LoRaWAN network and thus is the simplest method for increasing the scalability of a network.

E. Conclusion

Based on the simulations and analysis, we make the following conclusions about the LoRaWAN ADR.

1) The ADR consists of two concurrent algorithms, which do not converge at the same rate. The ED-side ADR algorithm is slower to converge as it has been designed to minimize the amount and maximize the flexibility of the control plane downlink traffic, which is limited by the duty cycle regulations applied to LoRaWAN devices (including gateways). The two algorithms do not necessarily converge to the same data rate assignment; the NS-side algorithm generally converges devices to a slower, higher range but less energy-efficient data rate than the ED-side algorithm. The ED-side algorithm can potentially converge devices to a lossy link.
2) The NS-side algorithm is not explicitly defined in the LoRaWAN protocol specification, and this results in inconsistent behavior between different implementations of the NS. In the Things Network implementation, the NS-side algorithm fails to function for nodes using DR0 and without any downlink data requirements.
3) The overall convergence time eventually becomes extended because of the duty cycle regulations applied to the gateway. Since the gateways act purely as relays to the central Network Server, the simplest solution to increasing the scalability of the network is to increase the density of gateways.

Based on this, we can identify enhancements of the ED-side and NS-side ADR algorithms with the following justifications.

1) End Device-Side ADR Enhancement: The main identified issue with the ED-side algorithm is the convergence of devices to lossy links; however, the NS is actually able to calculate the PDR of devices by analyzing the frame numbers included in the LoRaWANFrameHeader. We propose that prior to sending a downlink frame in response to a set *ADRAckReq* bit, the NS calculates the PDR of the device since the device last changed the data rate. If the PDR is below a threshold value, a *LinkADRReq* MAC-layer message is sent to the device to request the use of the next slowest data rate. In our simulations, a threshold value of 80% is chosen, representing a plausible minimum rate of successful data transfer for a LoRaWAN network, with a range of up to 5 km [26].

2) Network Server-Side ADR Enhancement: There are two issues with the NS-side algorithm: 1) the frequency of the running of the algorithm is quite arbitrary (ran after N uplink frames are received, where N is usually 20) and 2) the previously mentioned Things Network case where the algorithm fails to run for devices using $DR0$. We propose a mechanism that enables the timing of the NS-side algorithm based on the SNR values of recently received frames. In particular, after at least five uplink frames have been received since the last data rate change, if the received SNR values from the previous uplinks will ultimately have the data rate of the device changed, and if the standard deviation across those SNR values is less than 2.5 dB (the difference in signal strength between LoRaWAN data rates), then the ADR algorithm is fired. In addition, after this early fire of the ADR algorithm has been run, if there is no currently scheduled downlink frame for the device, then we propose that in this case, the Network Server is specifically allowed to create a new empty downlink frame for the MAC command to be sent in, in the next open receive window of that device.

VI. Evaluation of Enhancements

In this section, simulations of the ADR with the proposed enhancements are performed and compared with the original scheme. All key parameters remain the same as detailed in Section V; in the figures, for the sake of space and clarity, the results for $DR1$ and $DR4$ are not provided.

A. Effect of Enhancements on the Single Gateway Case

The proposed NS-side enhancement results in a faster convergence time for devices increasing the data rate. As Fig. 7 shows, in networks with a small number of devices, an

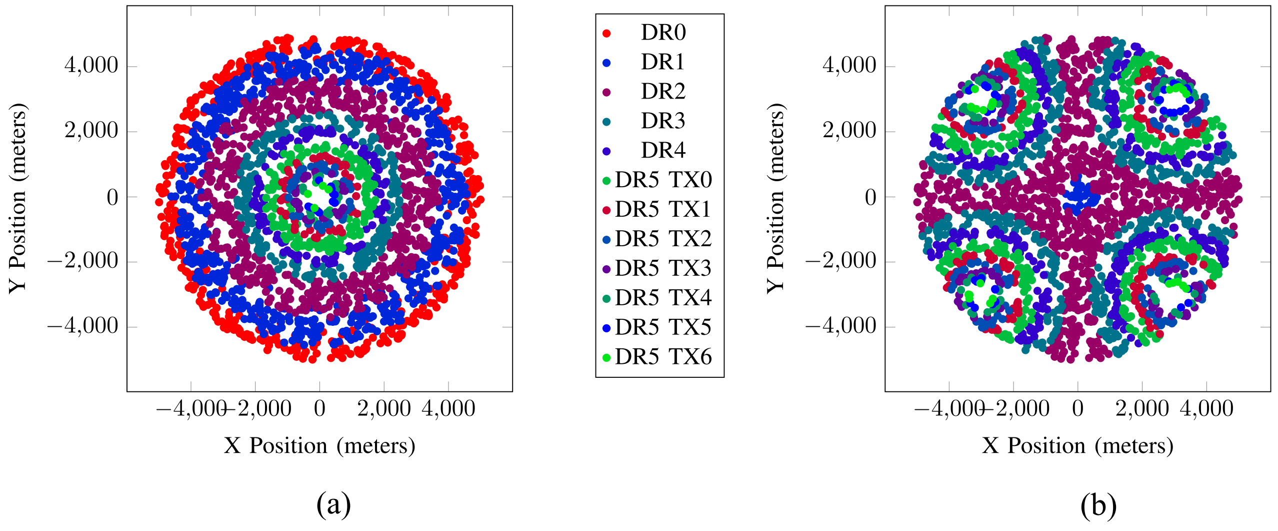


Fig. 6. Map of final data rate versus device location for the (a) single and (b) multiple gateway case, for a starting data rate of DR2.

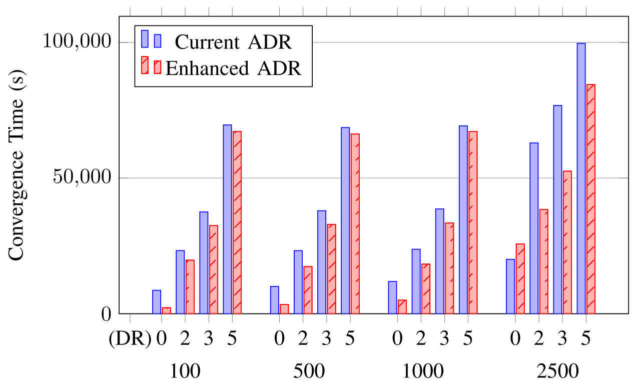


Fig. 7. Average node convergence time, uplink period = 600 s, original ADR versus enhanced ADR.

improvement of $> 50\%$ can be seen in the case where the majority of devices would be directly affected by this modification (i.e., the *DR*0 case). As expected, the benefit of the proposed enhancement is decreased in networks where fewer devices would be affected by the NS-side algorithm (i.e., the *DR*5 case), but there is an improvement seen in all cases.

The performance in terms of the convergence time remains the same for smaller networks, where the gateway on-air time is not at the maximum allowed by the duty cycle regulations. As the network scales, there is an increasing benefit as the more flexible NS-side algorithm reduces the delay for a device to transition to a faster data rate, enabling the transmission of more downlink frames. In addition, the ED-side modification results in devices gravitating away from lossy links faster. A visualization of the change caused by the enhancements can be seen in Fig. 8(a) and (b). The NS-side changes occur earlier, while the ED-side changes happen at the same rate as before.

In addition, the new ED-side enhancement converges devices to better quality links. Table VIII shows the equivalent PDR values after convergence for the simulations shown in Fig. 7; the use of the new link quality-aware mechanism results in a higher average PDR in all cases.

TABLE VIII
COMPARISON OF AVERAGE CONVERGENCE TIME AND PDR AFTER CONVERGENCE OF ORIGINAL AND ENHANCED ADR SCHEMES

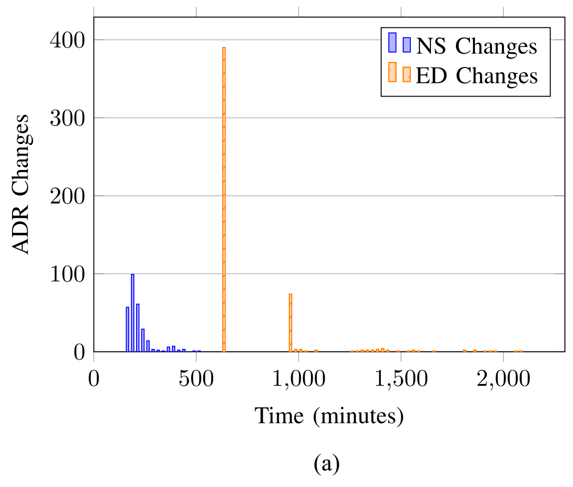
	Original	**Enhanced**	**Improvement**
Nodes	**100**		
DR0	99.02%	99.05%	+0.03%
DR2	87.95%	98.01%	+10.06%
DR3	85.92%	95.82%	+9.9%
DR5	80.77%	95.87%	+15.1%
	500		
DR0	95.44%	95.59%	+0.15%
DR2	84.22%	94.08%	+9.86%
DR3	82.19%	93.01%	+10.82%
DR5	77.63%	93.97%	+16.34%
	1000		
DR0	90.60%	91.05%	+0.45%
DR2	78.63%	89.60%	+10.97%
DR3	77.53%	88.87%	+11.34%
DR5	74.61%	92.87%	+18.26%
	2500		
DR0	79.29%	79.56%	+0.27%
DR2	78.28%	78.72%	+0.44%
DR3	75.09%	79.97%	+4.88%
DR5	75.60%	79.57%	+3.97%

B. Effect of Enhancements on the Things Network Issue

The ED-side enhancement also fixes the issue discussed in the Things Network implementation. Table IX shows that in the *DR*0 case, devices can now converge to a stable data rate, at the same rate as the LoRa Server implementation.

C. Effect of Enhancements on the Introduction of New Nodes

Fig. 9 shows the average node convergence time of 100 newly introduced devices into a network of *N* already converged devices. In smaller networks, the presence of other devices has some small effect because of collisions. There



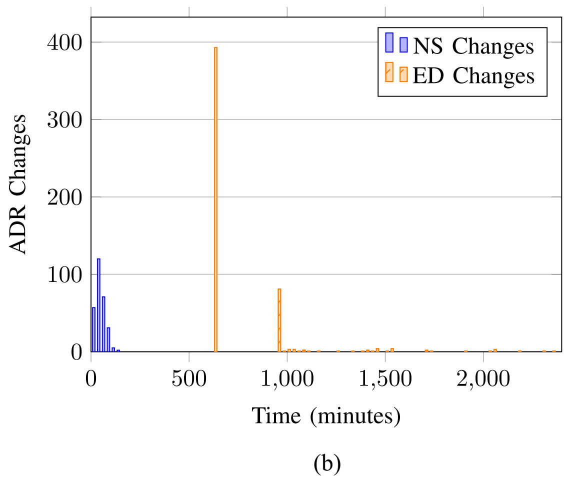


Fig. 8. ADR changes across time, for the (a) original ADR and (b) enhanced ADR.

TABLE IX
AVERAGE NODE CONVERGENCE TIME, UPLINK PERIOD = 600 s, N = 100, THINGS NETWORK IMPLEMENTATION CASE

Data Rate	**Current ADR**	**Proposed ADR**
DR0	∞	**2233.05**
DR2	26056.63	20958.10
DR3	38749.01	36031.38
DR5	69608.57	69349.56

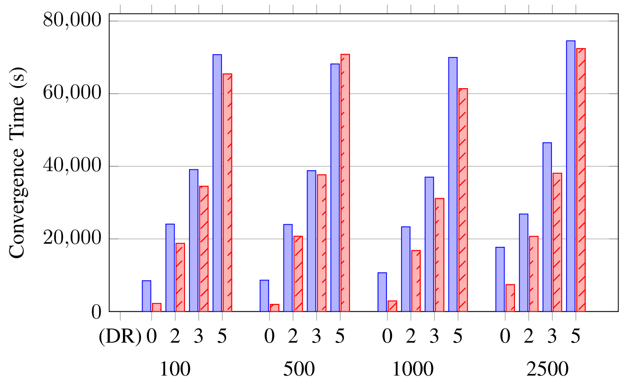


Fig. 9. Average node convergence time of 100 newly introduced devices, uplink period = 600 s.

is a greater effect in larger networks as the gateway begins to reach the duty cycle limit. The proposed enhancements continue to result in a decreased convergence time even in networks already containing many devices.

D. Effect of Enhancements on the Multiple Gateway Case

Fig. 10 shows that the proposed enhancements have increasing gains in the multigateway case.

VII. CONCLUSION

In this article, an extension to a LoRaWAN ns-3 module has been developed to implement ADR and using this a number of conclusions about the scheme have been made. The two concurrently running ADR algorithms do not converge at the same rate; the ED-side ADR algorithm is slower to converge as it has been designed to minimize the amount and maximize the flexibility of control plane downlink traffic, which is limited by the duty cycle regulations applied to LoRaWAN devices, including LoRaWAN gateways. In addition, the two algorithms do not necessarily converge a device to the same data rate; the NS-side algorithm generally converges devices to a slower, higher range but less energy-efficient data rate than the ED-side algorithm.

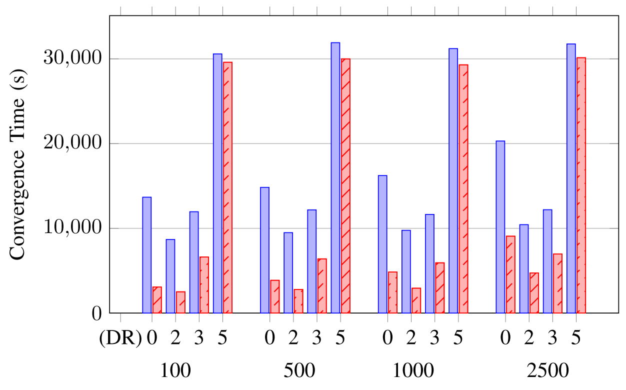


Fig. 10. Average node convergence time, uplink period = 600 s, multigateway case.

The NS-side algorithm is not explicitly defined in the LoRaWAN protocol specification, and this results in inconsistent behavior between different implementations of the Network Server. In particular, in the Things Network implementation, the NS-side algorithm fails for nodes using DR0 and without any downlink data requirements (which is possible in many LPWAN use cases). The overall convergence time for devices eventually becomes extended as the network scales because of the duty cycle regulations applied to the gateway. The simplest method of increasing the scalability of the network is to provide a greater density of gateways.

Based on this analysis, modifications for the NS-side and ED-side of the ADR scheme have been proposed which are easily integrable into the existing LoRaWAN protocol. These were implemented in ns-3, and evaluation through simulation shows that the proposed enhancements lead to a faster convergence rate for devices, and the convergence to data rates that result in a higher overall PDR for the network.

References

[1] F. Van den Abeele, J. Haxhibeqiri, I. Moerman, and J. Hoebeke, "Scalability analysis of large-scale LoRaWAN networks in ns-3," *IEEE Internet Things J.*, vol. 4, no. 6, pp. 2186–2198, Dec. 2017.

[2] M. Bor, U. Roedig, T. Voigt, and J. Alonso, "Do LoRa low-power wide-area networks scale?" in *Proc. 19th ACM Int. Conf. Model. Anal. Simulat. Wireless Mobile Syst. (MSWiM)*, 2016, pp. 59–67.

[3] *Electromagnetic Compatibility and Radio Spectrum Matters; Short Range Devices; Radio Equipment to Be Used in the 25 MHz to 1 000 MHz Frequency Range With Power Levels Ranging Up to 500 mW; Part 1: Technical Characteristics and Test Methods*, ETSI, Sophia Antipolis, France, Jan. 2012.

[4] F. Cuomo *et al.*, "Towards traffic-oriented spreading factor allocations in LoRaWAN systems," in *Proc. 17th Annu. Mediterr. Ad Hoc Netw. Workshop (Med-Hoc-Net)*, Jun. 2018, pp. 1–8.

[5] S. Li, U. Raza, and A. Khan, "How agile is the adaptive data rate mechanism of LoRaWAN?" in *Proc. IEEE GLOBECOM*, Dec. 2018, pp. 206–212.

[6] V. Hauser and T. Hégr, "Proposal of adaptive data rate algorithm for LoRaWAN-based infrastructure," in *Proc. IEEE 5th Int. Conf. Future Internet Things Cloud (FiCloud)*, Aug. 2017, pp. 85–90.

[7] M. Slabicki, G. Premsankar, and M. Di Francesco, "Adaptive configuration of LoRa networks for dense IoT deployments," in *Proc. IEEE/IFIP Netw. Oper. Manag. Symp. (NOMS)*, 2018, pp. 1–9.

[8] K. Abdelfadeel, V. Cionca, and D. Pesch, "Fair adaptive data rate allocation and power control in LoRaWAN," in *Proc. IEEE WoWMoM*, Feb. 2018, pp. 14–15.

[9] S. Kim and Y. Yoo, "Contention-aware adaptive data rate for throughput optimization in LoRaWAN," in *Proc. MDPI Sensors*, 2018, pp. 1716–1732.

[10] J. Lim and Y. Han, "Spreading factor allocation for massive connectivity in LoRa systems," *IEEE Commun. Lett.*, vol. 22, no. 4, pp. 800–803, Apr. 2018.

[11] Q. Zhou, J. Xing, L. Hou, R. Xu, and K. Zheng, "A novel rate and channel control scheme based on data extraction rate for LoRa networks," in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, Feb. 2019, pp. 1–6.

[12] B. Reynders, W. Meert, and S. Pollin, "Power and spreading factor control in low power wide area networks," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2017, pp. 1–6.

[13] A. Hoeller, Jr, R. D. Souza, O. A. López, H. Alves, M. de Noronha Neto, and G. Brante, "Exploiting time diversity of LoRa networks through optimum message replication," in *Proc. 15th Int. Symp. Wireless Commun. Syst. (ISWCS)*, Aug. 2018, pp. 1–5.

[14] L. Amichi, M. Kaneko, N. El Rachkidy, and A. Guitton, "Spreading factor allocation strategy for LoRa networks under imperfect orthogonality," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2019, pp. 1–7.

[15] M. N. Ochoa, A. Guizar, M. Maman, and A. Duda, "Evaluating LoRa energy efficiency for adaptive networks: From star to mesh topologies," in *Proc. IEEE 13th Int. Conf. Wireless Mobile Comput. Netw. Commun. (WiMob)*, Oct. 2017, pp. 1–8.

[16] J. Lee, W. Jeong, and B. Choi, "A scheduling algorithm for improving scalability of LoRaWAN," in *Proc. Int. Conf. Inf. Commun. Technol. Convergence (ICTC)*, 2018, pp. 1383–1388.

[17] B. Reynders, Q. Wang, P. Tuset-Peiro, X. Vilajosana, and S. Pollin, "Improving reliability and scalability of LoRaWANs through lightweight scheduling," *IEEE Internet Things J.*, vol. 5, no. 3, pp. 1830–1842, Jun. 2018.

[18] V. Sharma, G. Choudhary, I. You, J. Lim, and J. Kim, "Self-enforcing game theory-based resource allocation for LoRaWAN assisted public safety communications," *J. Internet Technol.*, vol. 19, Mar. 2018, pp. 515–530.

[19] *LoRaWAN Specification V1.1*, LoRa Alliance, Fremont, CA, USA, Jan. 2017.

[20] *LoRaWAN: Simple Rate Adaptation Recommended Algorithm*, Semtech, Camarillo, CA, USA, Oct. 2016.

[21] *The Things Network Stack V2*, Things Netw., Amsterdam, The Netherlands, 2019.

[22] *LoRa Server Open Source Network Server*, LoRa Server, Fremont, CA, USA, 2019.

[23] K. Staniec and M. Kowal, "LoRa performance under variable interference and heavy-multipath conditions," *Wireless Commun. Mobile Comput.*, vol. 2018, Apr. 2018, Art. no. 6931083.

[24] J. Finnegan, S. Brown, and R. Farrell, "Modeling the energy consumption of LoRaWAN in ns-3 based on real wold measurements," in *Proc. Global Inf. Infrastruct. Netw. Symp. (GIIS)*, Oct. 2018, pp. 1–4.

[25] M. Capuzzo, D. Magrin, and A. Zanella, "Confirmed traffic in LoRaWAN: Pitfalls and countermeasures," in *Proc. 17th Annu. Mediterranean Ad Hoc Netw. Workshop (Med-Hoc-Net)*, Jun. 2018, pp. 1–7.

[26] J. Petajajarvi, K. Mikhaylov, A. Roivainen, T. Hanninen, and M. Pettissalo, "On the coverage of LPWANs: Range evaluation and channel attenuation model for LoRa technology," in *Proc. 14th Int. Conf. ITS Telecommun. (ITST)*, Dec. 2015, pp. 55–59.

Joseph Finnegan (Student Member, IEEE) received the B.S. degree in computer science from Maynooth University, Maynooth, Ireland, in 2015, and the M.S. degree in computer science from the University of Bristol, Bristol, U.K., in 2016. He is currently pursuing the Ph.D. degree in wireless communications with Maynooth University.

His current research interests include wireless communications and networking, sensor networks, and Internet of Things.

Ronan Farrell (Member, IEEE) received the Ph.D. degree in electronic engineering from University College Dublin, Dublin, Ireland, in 1998.

Since 2000, he has been with Maynooth University, Maynooth, Ireland, where he is currently a Professor with the Department of Electronic Engineering and also a Co-Principal Investigator with the Science Foundation Ireland, CONNECT Centre for the Internet of Things. He has coauthored over 200 journal and conference papers. He leads a research team in the area of high-frequency radio systems with a particular interest in high-performance systems, power amplifiers, and applications of millimeter-wave communications.

Stephen Brown received the B.A.I. degree in electronic engineering and the M.Sc. degree in computer science from the University of Dublin (TCD), Dublin, Ireland, in 1982 and 1984, respectively, and the Ph.D. degree from University College Cork, Cork, Ireland, in 2010, with a thesis on software updating for wireless sensor networks.

Since 1994, he has been a Research Fellow with Trinity College Dublin, Dublin. He worked for Digital Equipment Corporation in Ireland, USA, and U.K., for ten years. He is currently a Senior Lecturer with Maynooth University, Maynooth, Ireland, where he is a Member of the Irish CONNECT SFI Funded Research Center. His current research interests are in wireless networking for sensor networks and IoT.