MADERE: Mobile Adaptive Datarate for LoRaWAN

Anaïs Durand*, Nancy El Rachkidy*, Alexandre Guitton*†
*Université Clermont-Auvergne, CNRS, Mines de Saint-Étienne, Clermont-Auvergne-INP, LIMOS, 63000 Clermont-Ferrand, France
†Univ Lyon, INSA Lyon, Inria, CITI, F-69621 Villeurbanne, France
Emails: {anais.durand, nancy.el_rachkidy, alexandre.guitton}@uca.fr

***Abstract*—Low-power wide area networks (LPWANs) are being increasingly used in Internet of Things applications, including smart city and environmental monitoring, as they enable communications from low-power end-devices to distant gateways. LoRaWAN is the most common protocol for LPWANs, and is able to automatically trade-off throughput and reliability thanks to an algorithm called Adaptive DataRate (ADR). However, the LoRaWAN standard imposes mobile nodes to disable the ADR. In this paper, we propose a protocol called MADERE (for Mobile ADR) that attempts to adapt the LoRaWAN parameters for mobile end-devices. We show that MADERE performs well compared to the few existing algorithms from the literature, with limited overhead.**

I. Introduction

Low-power wide area networks (LPWANs) consist of low-power end-devices that can communicate with gateways that are located several kilometers away. LPWANs are being increasingly used in Internet of Things applications (such as smart city [1], smart agriculture [2], or forest monitoring [3]), for several reasons: their deployment cost is low and their lifetime is high. Several LPWANs technologies exist, including LoRa (Long Range), Sigfox, NB-IoT, or LTE-M. LoRa is currently one of the most prominent technologies.

LoRa uses a chirp spread modulation where data is encoded into frequency sweeps called chirps. The time duration of a chirp depends on the spreading factor (SF) parameter: the longer the SF, the lower the datarate, but the better the communication range. LoRa can enable communications of about 5 km in urban scenarios, and of about 20 km in rural scenarios, with a datarate varying between 290 bps for SF=12, to 11 kbps for SF=7. LoRaWAN (Long Range Wide Area Network) [4] defines a MAC protocol on top of LoRa, and a network topology composed of end-devices, gateways that act as transparent relay nodes, and a central network server in charge of network operations. LoRaWAN also takes into account the regional regulations on the use of the ISM band: in Europe for instance, it enforces a maximum duty-cycle of 1% by forbidding immediate transmissions after each transmission of duration t, during $99t$.

The Adaptive DataRate (ADR) algorithm is a mechanism of LoRaWAN in charge of selecting the best SF for each end-device. It ensures that each end-device can communicate as fast as possible. To do so, the network server has to estimate the link quality between end-devices and gateways, based on a short history of previous successful communications. This estimation is relatively simple for static end-devices, but difficult for mobile end-devices. Hence, the *LoRaWAN specification requests to disable the ADR for mobile end-devices*, and they are forced to use the largest SF. This causes two main issues: (i) mobile end-devices have to use the largest SF, which consumes a lot of energy to transmit frames, and (ii) mobile end-devices experience a large delay when sending frames.

In this paper, we adapt the ADR algorithm for mobile end-devices. Our aim is to ensure that mobile end-devices can use the best SF, and thus spare energy while benefiting from fast datarates. We decide to rely mostly on the reception of acknowledgements from the gateway, assuming that all gateway locations are known beforehand. Thus, we make the following contributions:
(1) We discuss the drawbacks of choosing a fixed (or a blind) SF for mobile end-devices.
(2) We propose a lightweight, efficient mobile ADR protocol, called MADERE.
(3) We compare the performance of the mobile ADR protocols of the literature with our MADERE protocol, and show the benefits of MADERE.

The rest of this paper is as follows. Section II presents the existing ADR algorithms. Section III introduces our MADERE algorithm. Section IV details the comparison of these algorithms in a simulation environment. Finally, Section V concludes our work.

II. State of the art

A. ADR for static end-devices

The ADR algorithm is defined in the LoRaWAN specification [4]. Its goal is to adapt the SF and the transmission power of end-devices depending on the current channel conditions. It is enabled only for static end-devices. It uses two modules: one run by the network server, and one run by the end-devices.

The network-server module stores the received signal strength indicator (RSSI) of the last 20 frames, for each end-device that has the ADR enabled. The network server computes the maximum RSSI of these frames, denoted $RSSI_{max}$, as well as the required demodulation threshold $RSSI_{req}$ based on the current datarate of the end-device. Then, it computes $N_{step} = round(RSSI_{max} - RSSI_{req} - margin)/3$, where $margin$ is a configurable margin equal to 10 dBm by default. If N_{step} is negative, the network server increases the transmission power by $3 \times N_{step}$, up to the maximum allowed transmission power. If N_{step} is positive, the network server first reduces the SF by this amount N_{step} if possible, but without reducing SF below SF=7, and then reduces the transmission power by 3 dBm for each remaining value of N_{step}.

The end-device module periodically checks its connectivity with the network server by requesting an acknowledgment every 20 frames. Whenever an acknowledgement is received, the end-device resets a counter denoted `ADR_ACK_CNT`. When an acknowledgement is requested but not received, the end-device increases `ADR_ACK_CNT`. When this counter exceeds 20, the end-device assumes that the network server is unreachable and increases its transmission parameters by one step (first, by increasing the transmission power by 3 dBm, and then, by increasing the SF by one), and the next frames are sent without resetting `ADR_ACK_CNT`, that is by continuing to send acknowledgement requests. Further transmission parameter increases occur after 20 more missed acknowledgements.

The LoRaWAN specification states that mobile end-devices should not run the ADR, and should instead use SF12 with the maximum transmission power for all their transmissions. This is hindering for mobile devices, as they experience a low datarate and a high energy consumption.

The performance of the ADR algorithm has been evaluated in a short experiment with mobile devices in [5], by forcibly enabling the ADR mechanism. The authors show that the benefits of ADR decrease as a function of the mobility speed and that a network-side scheme is required.

B. Blind ADR for mobile end-devices

Blind ADR is an algorithm proposed in [6] by Semtech for mobile end-devices. Its goal is to balance the benefits of large SFs (that is, long-range connectivity) with those of small SFs (that is, long battery life) for mobile end-devices. It consists in blindly sending (hence the name) frames with varying parameters. The parameters are predefined and used in a round-robin manner. In [6], it is stated that each end-device is able to transmit every hour: one frame with SF=12, two frames with SF=10, and three frames with SF=7.

C. ADR algorithms for mobile end-devices

Gaussian filter-based ADR (G-ADR) and Exponential Moving Average-based ADR (EMA-ADR) have been proposed in [7]. These two algorithms are designed to reduce the convergence speed of the ADR for static networks, by reducing the high-variability of the signal-to-noise ratio (SNR). In some simulation scenarios (see Subsubsections 4.3.2, 4.4.2, and 4.5.2 of [7]), the authors studied the convergence ratio of their protocols in a mobile scenario, with the ADR forcibly enabled. They show that the packet delivery ratio with G-ADR and EMA-ADR converges to 65% relatively quickly (*i.e.*, in a few hours). Thus, G-ADR and EMA-ADR can be considered the first mobility-aware ADR mechanisms.

Enhanced-ADR (E-ADR) has been proposed in [8], [9]. It computes an estimation of the distance from an end-device to a gateway by mapping the RSSI of a frame into a distance, using an *a priori* knowledge of the propagation channel. Then, the position of the end-device is estimated by trilateration from several gateways. The position of each end-device is computed frequently, and the n last known positions are used to estimate the current position. Then, the network server decides the best network configuration to reach the mobile end-device. The assumptions of E-ADR are strong. First, E-ADR assumes that it is possible to estimate the distance between two nodes based on the RSSI. Second, E-ADR assumes that each possible position is in range of at least three gateways, in order to perform the trilateration. Third, E-ADR requires frequent position computations so that it can accurately estimate the current position. This yields a high overhead. It can also be seen as a requirement of low mobility for the end-devices.

Variable order Hidden Markov Models (VHMM)-based E-ADR is an extension of E-ADR, and is proposed in [10]. It uses a hidden Markov chain to estimate the

current position from the n last known positions. Apart from this, it is very similar to E-ADR.

Linear Regression-ADR (LR-ADR) and Linear Regression + (LR+ADR) have been proposed in [11]. LR-ADR aims to smooth the SNR of received packets and to predict the SNR of the next transmission, while LR+ADR operates at the end-device side and attempts to quickly regain connectivity in case of failure. They both assume that the end-devices transmit frames with a fixed period. This periodicity is used in the estimation of the next SNR, by adding a constant margin which corresponds to having the end-device moving away from the gateway with a maximum distance.

Resource Management ADR (RM-ADR) has been proposed in [12]. The proposed network server algorithm uses the reception power of the previous received frame in order to determine the SF and transmission parameter for the next frame. The proposed end-device algorithm reduces the retransmission attempts, which makes the algorithm react faster in case an unsuitable SF is chosen.

D. Summary of ADR algorithms for mobile end-devices

Table I gives a summary of the existing ADR algorithms, with the third column indicating which protocols we compare to. Blind ADR is fully implemented. E-ADR is modeled as the eDistance algorithm, which assumes that the exact position of an end-device is known at all time. G-ADR, EMA-ADR, VHMM-based E-ADR, and LR-ADR are modeled as the aDistance algorithm, which assumes that the exact position of an end-device is known only periodically. Finally, LR+ADR and RM-ADR are not implemented, as they only modify the end-device side of the ADR algorithm.

Table I
SUMMARY OF THE EXISTING MOBILE ADR ALGORITHMS.

Name	Hypotheses	Compared here
Blind ADR [6]	none	yes
G-ADR [7]	low mobility	aDistance
EMA-ADR [7]	low mobility	aDistance
E-ADR [8], [9]	accurate distance estimation	eDistance
VHMM-based E-ADR [10]	low mobility	aDistance
LR-ADR [11]	low mobility	aDistance
LR+ADR [11]	low mobility	no
RM-ADR [12]	low mobility	no

III. ADAPTIVE DATARATE FOR MOBILE END-DEVICES: THE MADERE PROTOCOL

In this section, we describe our proposed MADERE protocol. Our aim is to have no maximum transmission period for the end-devices, contrarily to most existing works, while making better SF estimations than static approaches and than random approaches.

A. Hypotheses and Mobility Model

We consider a finite rectangular area with a set $\mathcal{G}$ of at least one gateway and $\mathcal{D}$ mobile end-device. Each end-device knows the limits of the area, the location of the gateways, and its own initial position.

The MADERE protocol makes a single general assumption on the mobility of the end-devices: their speed is at most v_{max}. They can move freely within the area, along straight lines, circles, randomly, *etc*.

We use two functions, called $SNRToDist()$ and $DistToSNR()$. These functions are used to translate SNR into distance and distance into SNR, respectively. Those two functions use a coarse estimation of the propagation model. They are parameterized by an error margin e, to take into account the shadowing effect. They will also serve to determine the SF to use for transmissions, according to the LoRaWAN standard [4].

Finally, we assume that each transmission of a mobile end-device is confirmed, that is, the end-device expects to receive an acknowledgement for each frame. This acknowledgement is used to determine whether the end-device is within the communication range of a gateway and to estimate the distance to this gateway.

B. Description of the MADERE Protocol

The main idea behind the MADERE protocol is to estimate the location of the end-device by computing the probability to be at each point of space. Since it is not possible to compute such probabilities for every point of a continuous space, the area is discretized into a grid of $n \times m$ points, each of them representing a square area of width w, where w is a parameter of the protocol.

Each end-device maintains a two-dimensional table $\mathcal{P}_t$ such that $\mathcal{P}_t[i][j]$ is the estimated probability to be on the discrete point (i, j), with $i \in \{0, .., n-1\}$, $j \in \{0, .., m-1\}$, at time $t \geq 0$. We assume that the end-device knows its initial position (i_0, j_0). Thus, $\forall i \in \{0, .., n-1\}$, $j \in \{0, .., m-1\}$, we have: $\mathcal{P}_0[i][j]$ equals to 1 if $i = i_0$ and $j = j_0$, and equals to 0 otherwise. These probabilities are updated in two different ways:

(a) When an end-device has a frame to transmit, it updates the probabilities. Since the end-device does not know its actual movement, it computes the maximum distance traveled during the elapsed time since the last position update at t' using v_{max}. Then, for each discrete point (i, j) with a non-null probability, MADERE computes every point that is within the maximum distance travelled of (i, j) (see Fig 1(a)). Then, the previous

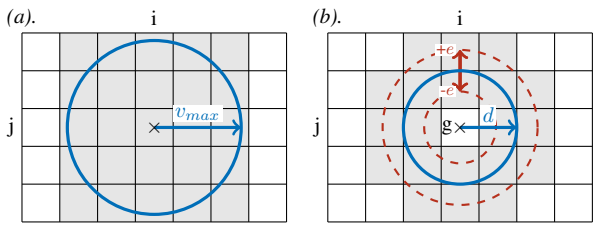

Figure 1. *(a).* Reachable discrete points (grayed out on the figure) during a position update. *(b).* Possible positions (grayed out on the figure) around gateway g after receiving an acknowledgement. d is the distance computed with $SNRToDist()$ and e is the error margin.

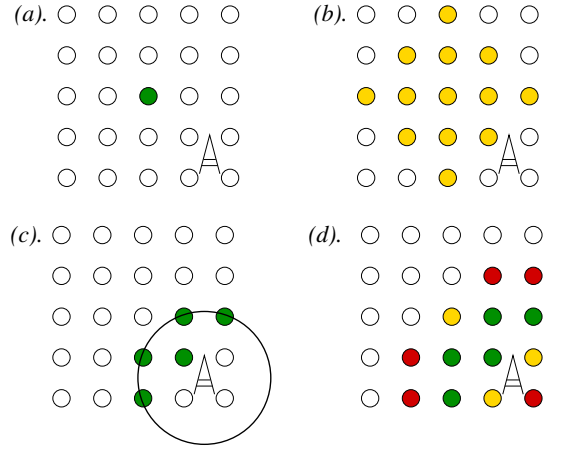

Figure 2. Evolution of the probability of presence for a mobile end-device, where the colors indicate the probability: white is a null probability, red is a low probability, yellow is a medium probability, and green is a high probability.

probability $\mathcal{P}_{t'}[i][j]$ is distributed uniformly among all the reachable discrete points. Note that the distances are computed from the center of the square represented by the discrete point (i, j).

(b) Every time the end-device receives an acknowledgment from a gateway, it computes its approximate distance to this gateway, using function $SNRToDist()$ with an error margin e. This function computes a range of distances from the gateway. Similarly to the previous case (see Fig 1(b)), MADERE computes a set D of discrete points that are in a ring around the gateway. Then, the probabilities are updated in two steps:
step1. the probabilities outside of D are collected, *i.e.*, let $p = \sum_{(i,j)\notin D} \mathcal{P}_t[i][j]$, before being set to zero.
step2. p is uniformly distributed on any point of D.

Before each transmission, an end-device chooses the best SF to use according to the computed probabilities. To do so, it computes a weighted average distance $\hat{d}_t$, by computing the distance $d(i, j)$ between the closest gateway of each discrete point (i, j), with a weight equal to the computed probability to be in the square represented by this discrete point (i, j): $\hat{d}_t = \sum_{(i,j)} d(i, j) \times \mathcal{P}_t[i][j]$. Then, MADERE applies the function $DistToSNR()$ to $\hat{d}_t$, and uses the corresponding SF for the transmission.

Fig. 2 illustrates these mechanisms. The area is represented as a grid of 5x5 points. MADERE stores the probability of the presence of a given end-device at each point. Initially, the location of the end-device is known with certainty, as shown in Fig. 2(a). As time passes without additional information, the location of the end-device becomes uncertain, and becomes a large disk centered around the initial location, as shown in Fig. 2(b). Then, the end-device receives an acknowledgement from a gateway at a known location. Thus, the set of possible locations becomes the intersection of the previous possible locations with the communication range of the gateway for the particular SF, as shown in Fig. 2(c). Finally, as time passes, the possible locations extend in all directions, as shown in Fig. 2(d). Thanks to these mechanisms, without any a priori knowledge of their mobility, the end-devices can estimate their position when transmitting a frame to refine their SF choice.

IV. Simulation Results

In order to evaluate the performance of MADERE with the algorithms from the state of the art, we used a homemade network simulator written in Java. In this section, we describe the parameters setting of the simulations and we analyze the performance of our proposed MADERE algorithm in terms of goodput, latency, as well as energy consumption.

A. Simulation setup

We used an area of $1km^2$, with one gateway randomly located. Each end-device is mobile and follows a random waypoint mobility model. We choose this mobility model as it is the most popular to evaluate mobile ad-hoc networks, due to its simplicity and wide availability. The movement speed varies between v_{min}=1 m/s and v_{max}=10 m/s, and the pause time between segments varies between 1 s and 10 s. The initial location and intermediate destinations of end-devices are randomly chosen within the area.

The channel propagation model we use is the shadowing model, with a path loss exponent of 4 and a standard deviation of 2 dB. Our implementation takes into account the capture effect of LoRa, as well as interferences and the quasi-orthogonality of SFs. We focus on a single LoRaWAN channel (instead of the mandatory three channels in Europe). Each end-device transmits as many frames as possible, while respecting the duty-cycle limitation of 1%. The first frame of each end-device is randomly chosen within $(0; 100t)$, where

t is the time on air of a frame for SF12. Simulation results are averaged by performing ten independent runs. Each simulation lasts for 2000 seconds. We also vary the number of mobile nodes between 5, 10, and 15 nodes.

In our implementation of Blind ADR, we consider that each mobile end-device can send data as fast as possible, while respecting the duty-cycle of 1%, with the following parameters used in a round-robin manner: SF7, SF10, SF7, SF12, SF7, and SF10.

The default algorithm uses a fixed SF12 for all end-devices, and the Blind ADR algorithm uses random SFs. The aDistance-k algorithm models a class of algorithms including G-ADR, EMA-ADR, VHMM-based E-ADR, and LR-ADR, where the exact location is periodically known, with a period of k seconds. This class of algorithms yields a large overhead when k is small. Moreover, it is not always feasible to know the exact location (for instance, when the number of gateways in the vicinity is smaller than three). In our simulations, we choose to set k to one hour. We believe that this value would be realistic for some monitoring applications. In addition to these algorithms from the state of the art, we include the exact distance algorithm, called eDistance on the figures, which uses an oracle to inform the node of its exact location at each frame sent. The behaviour of E-ADR follows the behaviour of eDistance, as E-ADR has strong requirements on the location of each end-device. Finally, we also implemented the MADERE algorithm.

B. Analysis of the SF

Fig. 3 shows the proportion of frames sent with the perfect SF, for all algorithms. Using a perfect SF means that the frame would have been dropped by all gateways if sent with a strictly smaller SF. The best algorithm for this metric is the exact distance algorithm, as it uses the theoretical distance to compute the SF. The next best algorithm is MADERE. Indeed, the computation of the probabilities allows an accurate decision in choosing the suitable SF. It can be seen that the other algorithms yield a low percentage of perfect SFs.

Fig. 4 shows the proportion of frames sent with an over-estimated SF, for all algorithms. Using an over-estimated SF means that the frame is received with the chosen SF, but would also have been received by at least one gateway if it was sent with a strictly lower SF. Choosing an over-estimated SF keeps the reception rate high, but reduces the throughput of the end-device, as well as increases its consumed energy. It can be seen that apart from the exact distance algorithm and MADERE, all algorithms significantly overestimate the SF. This also explains why most of the existing algorithms have a good reception rate.

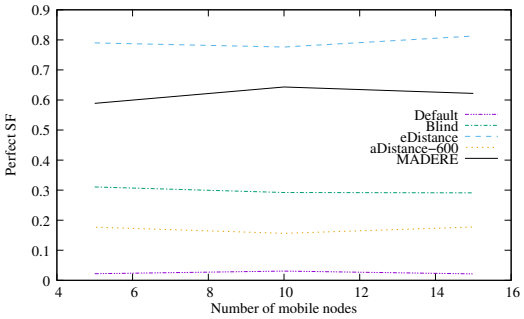

Figure 3. Frames with perfect SF as a function of mobile end-devices.

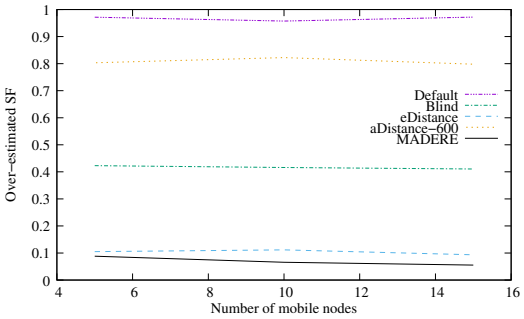

Figure 4. Frames with large SF as a function of mobile end-devices.

Fig. 5 shows the proportion of frames sent with an under-estimated SF, for all algorithms. Using an under-estimated SF means that the frame is not received by any gateway with the chosen SF, but would have been received with a strictly larger SF (thus, the frame loss is not due to collision but to a low SF). Both the aDistance and the MADERE algorithms yield relatively a large proportion of under-estimated SFs, which occurs due to inaccurate location estimation. This yields to an average performance in terms of reception rate as it sends much more frames than the other algorithms.

C. Analysis of the goodput

Fig. 6 shows the goodput of each algorithm, computed as the number of bytes received per second. The best algorithm is the exact distance algorithm, as expected. Then MADERE algorithm comes first. All algorithms based on approximate distance, with a period of $k = 600$ seconds, yield low goodput.

D. Analysis of the latency

Fig. 7 shows the latency of each algorithm, computed as the average delay between the first transmission of a frame and its first correct reception by the gateway. MADERE yields much lower latency than the algorithms

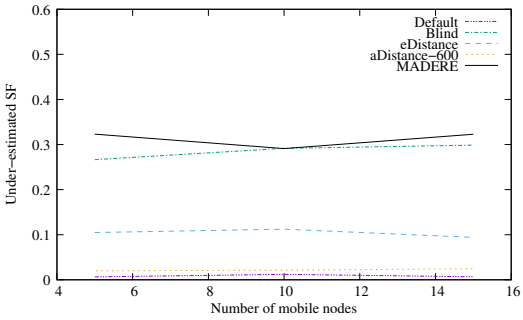

Figure 5. Frames with small SF, as a function of mobile end-devices.

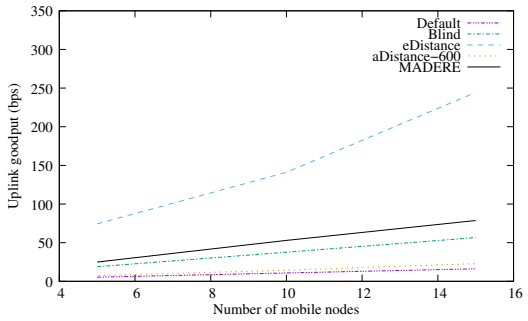

Figure 6. Uplink goodput as a function of mobile end-devices.

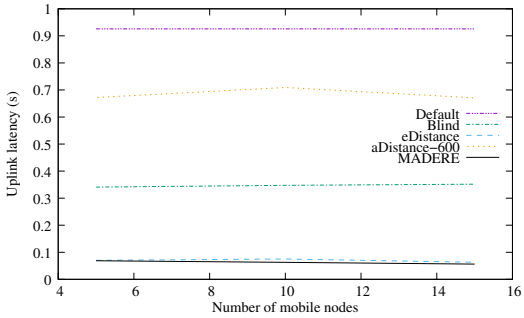

Figure 7. Uplink latency as a function of mobile end-devices.

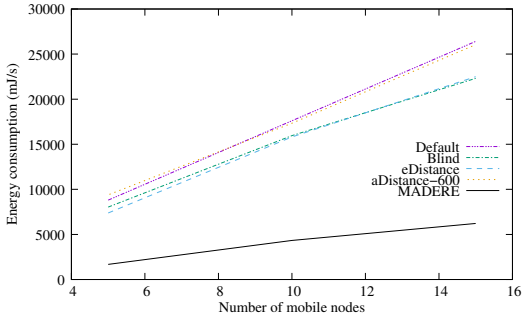

Figure 8. Energy consumption as a function of mobile end-devices.

from the state of the art. Indeed, as MADERE computes a good SF for each frame, the probability of a frame being correctly received is high, thus few retransmissions are needed. This decreases the overall network latency.

E. Analysis of the energy consumption

Fig. 8 shows the overall energy consumption (in mJ/s) of each algorithm. The energy consumption increases with the number of mobile end-devices for all algorithms. We can notice that MADERE greatly outperforms the other existing algorithms. This is because MADERE is able to often allocate a perfect SF with limited overhead, as it does not need specific information about the exact location of each mobile end-device.

V. Conclusion

In LoRaWAN, mobile end-devices are supposed to use fixed transmission parameters with maximum coverage but low data-rate, as it is difficult to correctly estimate their location, and thus the right SF to use. Most existing mobile ADR algorithms assume that the exact location of mobile end-devices is known periodically. This paper showed that it is possible to use indirect information based on the reception of acknowledgements in order to make a good decision for the SF. Our MADERE protocol is able to reach good performance in terms of perfect SF estimation, goodput, latency, and energy consumption.

References

[1] D. Magrin, M. Centenaro, and L. Vangelista, "Performance evaluation of lora networks in a smart city scenario," in *ICC*, 2017.

[2] P. Solanki, D. M. Patel, and D. G. Thakore, "A survey on smart agricultural system using LoRa wireless technology," *IJMTST*, vol. 6, no. 5, pp. 13–18, 2020.

[3] S. Park, S. Yun, H. Kim, R. Kwon, J. Ganser, and S. Anthony, "Forestry monitoring system using LoRa and drone," in *WIMS*, 2018.

[4] LoRa Alliance Technical Committee, "LoRaWAN 1.1 specification," LoRa Alliance, Tech. Rep., 2017.

[5] K. Kousias, G. Caso, O. Alay, and F. Lemic, "Empirical analysis of LoRaWAN adaptive data rate for mobile Internet of Things applications," 2019.

[6] Semtech Corporation, "LoRaWAN mobile applications: Blind ADR," Semtech, Tech. Rep., 2019.

[7] A. Farhad, D.-H. Kim, S. Subedi, and J.-Y. Pyun, "Enhanced LoRaWAN adaptive data rate for mobile internet of things devices," *Sensors*, vol. 20, no. 64, 2020.

[8] N. Benkahla, H. Tounsi, Y.-Q. Song, and M. Frikha, "Enhanced ADR for LoRaWAN networks with mobility," in *IWCMC*, 2019.

[9] ——, "Review and experimental evaluation of ADR enhancements for LoRaWAN networks," *Telecommunication Systems*, vol. 1, no. 77, 2021.

[10] ——, "VHMM-based E-ADR for LoRaWAN networks with unknown mobility patterns," in *IWCMC*, 2021.

[11] V. Moysiadis, T. Lagkas, V. Argyriou, A. Sarigiannidis, I. D. Moscholios, and P. Sarigiannidis, "Extending ADR mechanism for LoRa enabled mobile end-devices," *Simulation Modelling Practice and Theory*, vol. 113, 2021.

[12] K. Anwar, T. Rahman, A. Zeb, I. Khan, M. Zareei, and C. Vargas-Rosales, "RM-ADR: Resource management adaptive data rate for mobile application in LoRaWAN," *Sensors*, vol. 21, no. 23, 2021.