A Novel Collision-Aware Adaptive Data Rate Algorithm for LoRaWAN Networks

A Novel Collision-Aware Adaptive Data Rate Algorithm for LoRaWAN Networks

Riccardo Marini, Walter Cerroni, Chiara Buratti

***Abstract*—Low Power Wide Area Network technologies are used to interconnect a number of devices in a simple and efficient way. One of these technologies, LoRaWAN, is deemed as one of the most promising due to its capability to allow long range communications with very small energy consumption. LoRaWAN networks are managed by a network server implementing an Adaptive Data Rate (ADR) algorithm to allocate proper data rates to end devices. However, the standard ADR solution focuses only on the link-level performance and assigns transmission parameters to end devices one-by-one in an independent way. In this paper we propose a novel and more efficient ADR algorithm, denoted as Collision-Aware ADR (CA-ADR), which tries to minimize the collision probability when assigning data rates by considering the entire set of end devices in the network and keeping the link-level performance under control. The performance of CA-ADR is characterized and benchmarked against the standard solution as well as another proposal presented in the literature. An integrated simulation-experimental approach is used to assess results for large-scale networks and to compare two architectures based on cloud and fog computing. Results show that CA-ADR outperforms standard solutions when connectivity is good, whereas it behaves similarly in large areas. It is also shown that the improvement w.r.t. the benchmark solutions does not depend on the channel model considered (no shadowing, uncorrelated and correlated shadowing). Finally, a fog-based architecture is proved to be feasible, with the advantage of reducing the end-to-end latency.**

***Index Terms*—Adaptive Data Rate, Cloud/Fog Architectures, Internet of Things, LoRa, LoRaWAN, Low Power Wide Area Networks.**

I. Introduction

The Internet of Things (IoT) is a system of interrelated computing devices and everyday objects which are able to transfer data over a network without requiring human-to-human or human-to-machine interaction, by communicating over the Internet [1]. The idea behind IoT is to revolutionize the way we live and work: smart city or smart home applications [2], as well as smart agriculture [3] or Industry 4.0 [4], are a few among all the possible applications. Some of these applications require short-range radio communication technologies (like IEEE 802.15.4/ZigBee [5]), being the network nodes confined in restricted areas; others, like precision agriculture or smart city, may require communication solutions able to cover distances up to (or more than) 10 km in rural areas.

Recently, the interest of industry toward the latter category, denoted as low power wide area networks (LPWANs), grew significantly [6]–[8]. Some proprietary solutions, working on license-exempt spectrum bands, are already deployed in some regions: as an example, Sigfox operates both as a technology and a service provider for LPWANs [9]; another big player is the LoRa Alliance, which was officially established in Mobile World Congress 2015 and produced a proprietary solution known as LoRaWAN working in the license-exempt band and available in many countries. Due to its simplicity and flexibility and thanks to the fact that it allows long range communication with very low power consumption, LoRaWAN is considered nowadays one of the most promising LPWAN technology, suitable to many of the above cited applications (see e.g., [10], [11]).

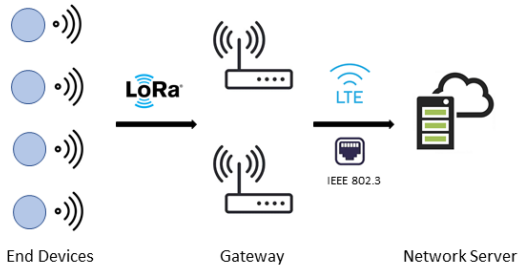

Fig. 1. LoRaWAN Network Architecture

In a LoRaWAN network (see Figure 1) End Devices (EDs), representing IoT nodes, send packets to Gateways (GWs), which, on their turn, are connected via other communication technologies (e.g., Ethernet, 3G/4G), to a centralized Network Server (NS) responsible for functions such as association and traffic management. Being in charge of collecting data from the GWs and forwarding them to relevant applications, the NS is typically deployed in a cloud computing environment, which is rarely located close to the source of data. Therefore, in case of latency-critical services or limited connectivity toward the Internet, deployments of LoRaWAN network components based on edge and fog computing solutions have also been considered [12], [13], bringing services, such as advanced analytics and distributed storage, closer to the EDs [14].

One of the key operations performed by the NS is to implement an Adaptive Data Rate (ADR) algorithm to dynamically set the data rate and transmit power to be used by a given ED during its communication. This algorithm is specifically designed to allow connectivity between EDs and GWs. However, the standard algorithm does not take into account the interference problems that may be present when the offered traffic increases. Indeed, since a simple ALOHA-based protocol is used by EDs to access the channel, collisions and

Department of Electrical, Electronic, and Information Engineering, University of Bologna, and CNIT, Italy. Email: {r.marini, walter.cerroni, c.buratti}@unibo.it

packet losses may dramatically impair the network throughput, especially when low data rates are used for transmissions [15].

In this paper we propose a novel ADR algorithm to be implemented at the network server, denoted as Collision-Aware ADR (CA-ADR), aiming at finding a set of data rates to be assigned to EDs such that packet success rate and network throughput improve w.r.t. the standard ADR solution. The algorithm exploits orthogonality of signals emitted with different data rates [15] and tries to minimize interference, while maintaining connectivity toward the GW. The algorithm has been tested and compared with the standard solution as well as another approach presented in the literature [16], [17], adopting an integrated methodology involving both simulations and experiments. In particular, a small testbed composed of one ED sending data to a GW, connected to a NS, has been used to characterize delays, both in terms of processing time and transmission time in the different links (wireless and wired). A traffic emulator has been developed to stress the NS and characterize its performance in the presence of high network traffic. The outcomes of these experiments have been provided as input to a simulator, evaluating the network-level performance when considering a large number of EDs deployed in a given area.

The performance of CA-ADR has been evaluated assuming the LoRaWAN network deployed in both a cloud and a fog computing scenario. In particular, two different setups for the NS, in terms of computational capabilities and location, have been considered. The impact of the NS deployment on the network performance has been assessed.

The rest of the paper is organized as follows. Section II reviews the state of the art, with particular emphasis on LoRaWAN and the current research trends. Section III is dedicated to the description of the system model considered. Section IV reviews the ADR algorithm, providing description of the current implementation as well as of the novel CA-ADR solution. Section V reports the identified Key Performance Indicators and, finally, Sections VI and VII present numerical results and conclusions, respectively.

II. Background and Related Work

In this section we first describe the LoRa/LoRaWAN standard and then we review some related work.

A. LoRaWAN Overview

LoRaWAN networks rely on two main components, LoRa and LoRaWAN, each corresponding to a different layer in the protocol stack. The former is a proprietary physical layer solution developed by Semtech Corporation[1], while the latter, described in an open specification issued by the LoRa Alliance, constitutes the medium access control (MAC) and network layers [18].

LoRa physical layer is based on Chirp Spread Spectrum modulation, which enables long range communications, even though this results in low data rate, with integrated forward error correction; in addition, it brings robustness against channel effects such as interference, frequency selectivity and Doppler effect. LoRa can work in different bands depending on regional constraints, as described in [19]. The main transmission parameters are: i) the Spreading Factor, hereafter denoted as SF, that is the ratio between the symbol rate and the chirp rate, which is directly proportional to the communication range and inversely proportional to the bit rate; ii) the bandwidth, hereafter denoted as BW, which is the spectrum occupied by a symbol; and iii) the coding rate, hereafter denoted as CR. As a result, the bit rate, R_b, changes by modifying the above parameters, according to the following formula [20]:

$$R_\mathrm{b} = SF \cdot \frac{\frac{CR}{4+CR}}{\frac{2^{SF}}{BW}} \quad [\mathrm{bit/s}] \tag{1}$$

The bit rate has a direct impact on the Time on Air (ToA), that is the time needed to transmit a packet on the wireless channel.

A LoRaWAN network is deployed as a star-of-stars topology, in which the communication between EDs and GWs is realized via LoRa, whereas the NS is connected with GWs via standard Internet Protocol (IP) connections, such as Wi-Fi, Ethernet, 3G/4G (see Figure 1). In particular, EDs are not associated to a specific GW but rather to a NS and all GWs receiving data from an ED forward it to the NS. As a result, the complexity of the network is concentrated at the NS, in charge of controlling ED transmission parameters (SF and transmit power) via the ADR algorithm (see Section IV), discarding duplicated packets (i.e., same data packets received from multiple GWs) and selecting the GW to be used for sending the acknowledgement to a given uplink data packet (if requested). Indeed, uplink transmissions can either be confirmed or unconfirmed. In the former case, the ED waits for an acknowledgment (ACK) packet from the NS (through the GW) after its transmission, and it keeps sending the same message if no ACK is received; in the latter case, no ACK and retransmissions are used.

LoRaWAN defines three classes of devices: A, B and C.

- Class A devices allow for bidirectional communication whereby each uplink transmission is followed by two short downlink receive slots after two different fixed RECEIVE_WINDOW_DELAY intervals (by default 1 s delay between the uplink communication and the first receive window and 2 s delay between the uplink communication and the second receive window), which last at least the amount of time needed for demodulating the received packet. This class of devices is characterized by the lowest power consumption, so it is intended for sensor nodes, which are usually battery powered.
- Class B devices allow for more than two receive slots, opening extra receive windows at scheduled times.
- Class C devices have nearly continuous open receive window except for the time spent in uplink transmission, so they are always reachable, at the cost of high power consumption.

B. Related work

Many recent studies focus on the evaluation of performance of LoRaWAN networks. In [21] and [22], LoRaWAN technol-

[1]https://www.semtech.com/lora

ogy was compared to other LPWAN technologies, that are Sigfox and NB-IoT, to point out the advantages of LoRaWAN in terms of battery lifetime and capacity in many scenarios. The coverage of LoRa communications has been addressed in [23], where Authors show that the maximum communication range can reach up to 10 km with a small percentage of packets lost. A theoretical analysis of the achievable uplink throughput has been carried out in [24], where the effect of the SF allocation and the impact of SF imperfect orthogonality on the overall throughput are taken into account.

There exist also a number of works focused on the ADR mechanism. In [16], [17] it is proposed to estimate the link quality to be used as input to the ADR algorithm based on the average signal-to-noise ratio (SNR), rather than the maximum SNR as done in the standard solution. This approach is also considered as a benchmark to evaluate the CA-ADR algorithm. In [17], Authors also propose a hysteresis algorithm to mitigate the problem of a link for which the link margin lies approximately at the midpoint between two decision levels and the ADR algorithm may lead to oscillations between a sub-optimal and an optimal solution for the transmission parameters. Another study worth mentioning is [25], where Authors propose that each ED chooses its transmission parameters locally, to minimize the Time on Air and maximize lifetime. This approach, though, does not take into account the possible connection problem the ED could experience without a proper mechanism to set such parameters according to the conditions the ED is working in. Authors in [26] propose a way to optimize the throughput with respect to the standard implementation by increasing the number of devices using small SFs, even though this leads to lower packet success probability. In [27] two SF allocation mechanisms are proposed: EXP-SF, where SF are equally distributed among nodes (i.e., each cluster of $N/6$ nodes is assigned a specific SF, where N is the total number of nodes), whereas in EXP-AT, SFs are assigned to EDs so as to attempt to achieve the same Time on Air for each group of (potential) interferers.

In contrast with the above cited works, CA-ADR takes into account the collision probability at the network level in order to improve the overall packet success probability. This is something other works do not focus on, since their goal is to improve performances in terms of energy consumption and link-level throughput, but eventually reducing the network success probability. This new approach, instead, allows to achieve fairness among all nodes distributed in the network, whatever the SF they are using, reducing the number of collisions they will experience, which is particularly suitable for large dense networks.

The advantages of deploying LoRaWAN network components on edge and fog computing infrastructures have also been investigated, especially in case of latency-critical services or limited connectivity toward the Internet. An edge-assisted IoT architecture has been proposed and demonstrated with proof-of-concept implementation, showing how to enable advanced services and distributed storage closer to the end-devices [14]. Another approach consists in augmenting the existing LoRaWAN architecture with a middleware solution that enables IoT data analytics at the edge, allowing application data sharing among multiple customers [28]. To this purpose, a proof-of-concept prototype named IoTRACE has been implemented, which allows multiple data subscribers to deploy their analytics applications at the edge, demonstrating how edge analytics can be combined with cloud analytics. An edge-fog-IoT architecture has also been proposed and applied to the use case of urban traffic management and monitoring [29]. Devices located at edge and fog layers are used to offload time-sensitive data processing tasks and to provide localized gateway and storage functions. Another work presents the design and deployment of a LoRaWAN infrastructure capable of providing novel applications in a smart campus [30]. The architecture enables to deploy fog computing nodes throughout the campus to support physically distributed, low-latency, and location-aware applications that decrease the network traffic and the computational load of traditional cloud computing systems. However, in most of the previous works the NS functionality is still deployed in the cloud, with processing-enabled GWs located at edge or fog nodes. Although the IoTRACE architecture assumes that NS can be deployed even at the edge, this solution has not been quantitatively evaluated. In general, to the best of our knowledge, no previous studies have been carried out to characterize the difference between a cloud-based and a fog-based deployment of a LoRaWAN NS in terms of performance of the ADR algorithm.

III. System Model

In this section we report assumptions and models according to the scenario we considered. We consider a set $\mathcal{N}$ of EDs, whose size is $N = |\mathcal{N}|$, randomly and uniformly distributed in a square area of side D [m]. A set $\mathcal{G}$ of GWs are deployed in the area, with $G = |\mathcal{G}|$. We assume that EDs generate a packet of payload B [bytes] in an instant that is randomly and uniformly distributed within a time period T [s]. As a result, the offered throughput, O, defined as the number of bits per second generated in a network of N EDs, is given by[2]:

$$O = \frac{8 \cdot B \cdot N}{T} \quad [\text{bit/s}] \tag{2}$$

A. Channel Model

We compute the power received by a GW, P_{R}, as a function of the transmit power, P_{T}, as: $P_{\mathrm{R}}[dBm] = P_{\mathrm{T}}[dBm] - L[dB]$. The path loss, L, in dB is modeled using the well known Okumura-Hata model [31]:

$$\begin{aligned} L =& 69.55 + 26.16\, log_{10}(f) - 13.82\, log_{10}(h_b) - C_H + \\ & (44.9 - 6.55\, log_{10}(h_b)) \cdot log_{10}(d) + s \end{aligned} \tag{3}$$

where f is the frequency in MHz; h_b and h_m represent the height of the GW and the ED in m, respectively; d is the transmitter-receiver distance in km and C_H is the antenna height correction factor, which depends on the size of the city and the frequency. In particular, for large urban areas, that

[2]We do not explicitly consider duty cycle constraints, but the results shown in the paper are valid even in the presence of duty cycle, provided that the offered throughput O given by eq. (2) is generated each time an ED wakes up to transmit its data.

represent the scenarios of interest here, and $f = 868$ MHz, it is given by [31]:

$$C_H = 3.2(log_{10}(11.75\, h_m))^2 - 4.97 \quad (4)$$

Finally, s represents random channel fluctuations due to shadowing, modelled via a Gaussian random variable, with zero mean and standard deviation σ, that is $s \sim \mathcal{N}(0, \sigma^2)$. Both uncorrelated and correlated shadowing samples are considered in the following. For the correlated case, we used the well-known Gilbert-Elliot model [32], [33], since it gives a negative exponential correlation function, which is exactly the correlation function that can be observed for shadowing in outdoor environments [33]. According to [32] the channel is modelled via a two-state Markov chain (see Figure 1 in [32]), where the two states represent: *Good* channel, where shadowing is characterized by a low value of the standard deviation, σ_{G} and *Bad* channel, where shadowing is characterized by a high value of the standard deviation, σ_{B}. We also set the probability of remaining in a given state (*Good* or *Bad*) equal to p and we analyze performance by varying p.

B. Packet Capture Model

We assume a packet is correctly received at the receiver side, if:

1) $P_{\mathrm{R}} \geq P_{\mathrm{R}_{\min}}(SF)$ and
2) $P_{\mathrm{R}} / \sum_i P_{\mathrm{R}_i} \geq \gamma$

where P_{R} is the useful received power, that is the power received by the GW when the ED is transmitting (in case of multiple GWs, we consider the strongest one, that is the one receiving the largest received power); $P_{\mathrm{R}_{\min}}$ is the receiver sensitivity; $P_{\mathrm{R}_{\mathrm{i}}}$ is the power the GW is receiving from the $i-th$ interfering ED.

The second condition represents the capture effect: even in the presence of collisions, there is a possibility that the receiver (i.e., the GW) is able to capture the frame if the signal-to-interference ratio (SIR) is above a given threshold (capture threshold) [34]. The term $P_{\mathrm{R}} / \sum_i P_{\mathrm{R}_i}$ represents the SIR, where the denominator includes the sum of the interfering powers, while γ is the capture threshold, which varies by varying the spreading factor. γ has been measured in [15] via experiments performed with the same devices as the ones used in this paper (see Table III of [15] for details). Finally, note that, as stated above, LoRaWAN uses a simple ALOHA-based multiple access protocol, meaning that an ED sends its data whenever available and interferers will be represented by all the other transmitters in the area whose packets are partially or entirely overlapping in time with the useful one.

According to specifications [35], the receiver sensitivity, $P_{\mathrm{R}_{\min}}$, is defined as:

$$P_{\mathrm{R}_{\min}}[dBm] = 10\, log_{10}(k \cdot BW \cdot T_0(F-1)) + SNR_{\min} \quad (5)$$

where the first term represents the noise power, given that k is the Boltzmann constant, BW is the bandwidth, $T_0 = 290$ Kelvin degrees is the standard noise temperature, $F = 6$ dB is the receiver noise figure; $SNR_{\min}$ is reported in Table I and it represents the minimum SNR required for demodulation [36].

TABLE I: $SNR_{\min}$ values for different SF with BW=125 kHz.

Data Rate	SF	$SNR_{\min}$[dB]
0	12	-20
1	11	-17.5
2	10	-15
3	9	-12.5
4	8	-10
5	7	-7.5

Since a collision happens if two packets are overlapping in time, it is important to derive the ToA, being the time needed to transmit a packet, which depends on the SF value and it is computed as follows [35]:

$$T_{\mathrm{symbol}}(SF) = \frac{2^{SF}}{BW} \quad (6)$$

$$T_{\mathrm{preamble}}(SF) = (L_{\mathrm{preamble}} + 4.25) \cdot T_{\mathrm{symbol}}(SF) \quad (7)$$

$$L_{\mathrm{payload}} = 8 + \left\lceil \frac{(8B - 4SF + 28 + 16 - 20H)}{(4SF)} \right\rceil \cdot (CR + 4) \quad (8)$$

$$T_{\mathrm{payload}}(SF) = L_{\mathrm{payload}} \cdot T_{\mathrm{symbol}}(SF) \quad (9)$$

$$ToA(SF) = T_{\mathrm{preamble}}(SF) + T_{\mathrm{payload}}(SF) \quad [\mathrm{s}] \quad (10)$$

where B is the payload size, H can be 0 or 1 and represents the presence of the physical layer header, L_{preamble} and L_{payload} are the length in symbols of the preamble and the payload respectively.

IV. Adaptive Data Rate Algorithm

A. Standard ADR Algorithm

The standard ADR tries to assign the lowest value of SF, allowing connectivity between the ED and the GW, in order to reduce as much as possible the energy consumption. The algorithm working on the NS is designed by the server developer, while the one working on the ED is specified by LoRa Alliance [18]. For the NS algorithm, we refer to the most widespread implementation, that is the one used by The Things Network or ChirpStack, which is based on Semtech's recommended algorithm [36].

EDs are in charge of deciding if ADR should be used or not. When activated, the NS will control the transmission parameters of the device sending ADR-specific commands. Besides, the device should periodically check whether the NS still receives its uplink frames, otherwise it should autonomously set its SF.

Algorithm 1 shows the implementation at the NS used to decide about the transmission parameters (specifically, SF and P_{T}) each ED connected to the NS should set. The NS runs the algorithm each time an uplink packet from a given ED is received (i.e., every T in our scenario) and decisions on data rates are based on measurements performed over the last k packets received from the ED (by default, $k = 20$). In particular, at first the Signal-to-Noise ratio (SNR) margin, SNR_{margin}, is measured as reported in Algorithm 1, where

Algorithm 1: NS ADR Algorithm for a given ED in a given instant, t.

Input: $SF_{\text{temp}} = SF(t)$, $P_{\text{T}_{\text{temp}}} = P_{\text{T}}(t)$, $SF_{\min} = 7$
$SNR_{\max} = max\{last\ k\ uplink\ packets\ received\}$,
$SNR_{\min}$ given in Table I by setting $SF(t)$,
$SNR_{\text{margin}} = SNR_{\max} - SNR_{\min} - M$,
$N_{\text{step}} = \left\lfloor \frac{SNR_{\text{margin}}}{3} \right\rfloor$
Output: $SF(t+T)$, $P_{\text{T}}(t+T)$

if $N_{\text{step}} > 0$ **then**
  while $N_{\text{step}} > 0$ & $SF_{\text{temp}} > SF_{\min}$ **do**
    $SF_{\text{temp}} = SF_{\text{temp}} - 1$
    $N_{\text{step}} = N_{\text{step}} - 1$
  while $N_{\text{step}} > 0$ & $P_{\text{T}_{\text{temp}}} > P_{\text{T}_{\min}}$ **do**
    $P_{\text{T}_{\text{temp}}} = P_{\text{T}_{\text{temp}}} - 3\,dB$
    $N_{\text{step}} = N_{\text{step}} - 1$
else
  while $N_{\text{step}} < 0$ & $P_{\text{T}_{\text{temp}}} < P_{\text{T}_{\max}}$ **do**
    $P_{\text{T}_{\text{temp}}} = P_{\text{T}_{\text{temp}}} + 3\,dB$
    $N_{\text{step}} = N_{\text{step}} + 1$
return $SF(t+T) = SF_{\text{temp}}$, $P_{\text{T}}(t+T) = P_{\text{T}_{\text{temp}}}$

$SNR_{\max}$ is the maximum SNR among the last k collected packets, M is a margin set a priori ($M = 10$ dB by default), and $SNR_{\min}$ is the minimum SNR required to correctly demodulate the received signal, computed considering the initial value of SF (see Table I). Then, an iterative process starts. SNR_{margin} is used to compute N_{step}, which indicates how many times the iteration will run. In particular, if N_{step} is greater than 0, both N_{step} and the SF are decremented by one unit at each step, until either the minimum SF is reached ($SF = 7$) or N_{step} reaches zero. If the iteration has not been completed yet ($N_{\text{step}} > 0$), then the transmitted power is also decremented by 3 dB at each further step, until either it reaches the minimum value (2 dBm in our scenario) or the iteration terminates ($N_{\text{step}} = 0$). On the other hand, if the initial value of N_{step} is lower than 0, then both N_{step} and the transmitted power are incremented at each step (by one unit and 3 dB, respectively) until either the maximum power (14 dBm) or the last iteration ($N_{\text{step}} = 0$) is reached. In both cases, the algorithm stops at most after N_{step} steps.

As far as the ED is concerned, it will receive information from the NS about the SF to be used, but then it will continuously track the connection with the NS. Algorithm 2 reports the implementation at each ED. We consider here the case of confirmed transmission mode (extension to the unconfirmed mode is straightforward [18]). Each time an ED sends an uplink packet without receiving an ACK, the counter ADR_ACK_CNT is incremented; after ADR_ACK_LIMIT (by default 64) messages without any downlink response, the device sends a request to NS, which must respond within the next ADR_ACK_DELAY (by default 32) frames with a downlink frame. If no reply is received, the ED must try to reconnect to the network, by first setting the transmitted power to its default value and then possibly switching to the next lower data rate. The device must lower its data rate every time the ADR_ACK_DELAY expires.

Algorithm 2: ED ADR Algorithm.

Input: SF_{temp} set to the current SF value assigned, $ADR\_ACK\_CNT = 0$
Output: New value of SF

while *uplink transmission* **do**
  if *no ACK received* **then**
    ADR_ACK_CNT=ADR_ACK_CNT+1
    if *ADR_ACK_CNT=ADR_ACK_LIMIT* **then**
      request downlink response from NS
      if *ADR_ACK_CNT* $\geq$ *ADR_ACK_LIMIT+ADR_ACK_DELAY* **then**
        $SF_{\text{temp}} = SF_{\text{temp}} + 1$
  else
    $ADR\_ACK\_CNT = 0$
return SF_{temp}

B. Collision-Aware ADR Algorithm

The idea behind the new algorithm is to assign SF in order to guarantee a given success probability in delivering a packet, accounting not only for the link-level performance (connectivity with the GW), but also for the collision probability (MAC-level performance). The algorithm exploits the orthogonality among signals transmitted with different SFs, as demonstrated in previous work (see, e.g., [15], [37]).

Algorithm 3: Collision-Aware ADR Algorithm

Input: $\hat{p}_{\text{mac}} = 1$, $\mathcal{SF} = \{7, 8, 9, 10, 11, 12\}$
Output: $SF(n)$ for $n \in \mathcal{N}$

foreach $SF \in \mathcal{SF}$ **do**
  $n_{\text{count}}(SF) = 0$
  $n_{\max}(SF) = \left\lfloor \left(\frac{1}{2} \log_{(1 - \frac{ToA(SF)}{T})} (\hat{p}_{\text{mac}}) \right) + 1 \right\rfloor$
foreach $n \in \mathcal{N}$ **do**
  $SF_{\min}(n) = min\{SF\ s.t.\ P_{\text{R}}(n) \geq P_{\text{R}_{\min}}(SF)\}$
  $SF_{\text{temp}}(n) = SF_{\min}(n)$
foreach $n \in \mathcal{N}$ **do**
  if $n_{\text{count}}(SF_{\text{temp}}(n)) < n_{\max}(SF_{\text{temp}}(n))$ **then**
    $SF_{\text{res}}(n) = SF_{\text{temp}}(n)$
    $n_{\text{count}}(SF_{\text{temp}}(n)) = n_{\text{count}}(SF_{\text{temp}}(n)) + 1$
  else
    $SF_{\text{temp}}(n) = SF_{\text{temp}}(n) + 1$
    if $SF_{\text{temp}}(n) \leq 12$ **then**
      go to 8
if $\sum_{SF \in \mathcal{SF}} (n_{\text{count}}(SF)) = N$ **then**
  return $SF(n) = SF_{\text{res}}(n)$ for $n \in \mathcal{N}$
else
  if $\hat{p}_{\text{mac}} > 0.01$ **then**
    $\hat{p}_{\text{mac}} = \hat{p}_{\text{mac}} - 0.01$
    go to 1

In the algorithm we denote with $p_{\text{mac}}(SF)$ the packet success probability for an ED using spreading factor SF, taking into account collisions at the MAC layer. In particular, $p_{\text{mac}}(SF)$ is the probability that no other transmissions will occur during the vulnerability period of ALOHA, that coincides with $2 \cdot ToA$, being ToA given by eq. (10). Therefore

we have [38]:

$$p_{\mathrm{mac}}(SF) = \left(1 - \frac{ToA(SF)}{T}\right)^{2(n(SF)-1)} \tag{11}$$

where $n(SF)$ is the number of devices using spreading factor SF. In the algorithm, for the sake of simplicity, we do not account for the capture effect; therefore, we try to assign SFs in order to limit as much as possible the collision probability. When running the simulation, after the SF values assignment, we then account for the capture effect, as described in section III, to evaluate if a packet is correctly received by the GW.

The CA-ADR algorithm is reported in Algorithm 3 and it works as follows. We first set $p_{\mathrm{mac}}(SF)$ to a target value, $\hat{p}_{\mathrm{mac}} = 1$, and then we compute the maximum number of EDs that can use a specific SF, $n_{\max}(SF)$, according to the equation reported in line 3 of the algorithm (i.e., reversing eq. (11)). Then, for each node n we derive the minimum value of SF the ED can use, that is the minimum SF satisfying $P_{\mathrm{R}}(n) \geq P_{\mathrm{R}_{\min}}(SF)$ (see line 5 of the algorithm). $P_{\mathrm{R}}(n)$ is the average power received by the GW when the ED is transmitting (averaged over the last k packets received) and $P_{\mathrm{R}_{\min}}(SF)$ is reported in Table II. The algorithm then tries to assign the obtained minimum SF to each ED, as long as the number of nodes already assigned that SF value ($n_{\mathrm{count}}(SF)$) does not exceed the maximum allowed ($n_{\max}(SF)$). If the limit is reached, then the algorithm tries to assign a higher SF following the same process, and keeps increasing it up to the maximum value ($SF = 12$). When $n_{\mathrm{count}}(SF) = n_{\max}(SF)$ even for the highest SF value, no SF assignment is carried out for that specific ED. At the end of the process, the algorithm checks if it was able to assign a SF to each ED, by comparing the sum of the $n_{\mathrm{count}}(SF)$ to N. If they are equal, it means that each ED has a correct SF assigned, otherwise $\hat{p}_{\mathrm{mac}}$ is decremented by 0.01 and the algorithm runs again from the beginning, as long as $\hat{p}_{\mathrm{mac}} > 0.01$.

Differently from the reference ADR implementation, the CA-ADR algorithm running on the NS goes through a number of iterations to make sure that all EDs are assigned the best SF values such that $n_{\max}(SF)$ is never exceeded. It is thus important to assess the complexity of the CA-ADR algorithm. In the worst case when all the iterations must be executed, the number of operations to be performed is given by $(|SF| + N \cdot |SF| + N + |SF|) \cdot 100$. Considering that the set SF is very small, the CA-ADR complexity is $\mathcal{O}(N)$.

TABLE II: SF sets for different $P_{\mathrm{R}_{\min}}$ with BW=125 kHz.

$P_{\mathrm{R}_{\min}}$[dBm]	$SF_{\min}$	$\mathcal{SF}$
-137	12	$\{12\}$
-134.5	11	$\{11, 12\}$
-132	10	$\{10, 11, 12\}$
-129.5	9	$\{9, 10, 11, 12\}$
-127	8	$\{8, 9, 10, 11, 12\}$
-124.5	7	$\{7, 8, 9, 10, 11, 12\}$

V. Key Performance Indicators

The performance of the CA-ADR algorithm is evaluated in terms of: i) Packet Success Rate, P_{s}; ii) Network Throughput, S; iii) Latency, L, and iv) Round Trip Time, RTT.

P_{s} is the percentage of packets that are correctly received at the GW, given that both conditions 1 and 2 in Subsection III-B are satisfied. Consequently, the Network Throughput, S, has been defined as:

$$S = \frac{B \cdot N \cdot P_{\mathrm{s}}}{T} \quad [\mathrm{bit/s}] \tag{12}$$

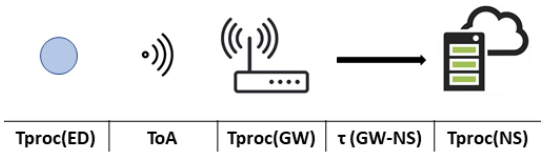

Fig. 2. Latency

Latency L has been defined as the interval of time between the generation of a packet at the ED and its reception at the NS. As shown in Fig. 2, L can be expressed as

$$\begin{aligned} L =& T_{\mathrm{ED}}^{(proc)} + ToA_{UL}(SF) + T_{\mathrm{GW}}^{(proc)} \\ &+ \tau_{GW-NS} + T_{\mathrm{NS}}^{(proc)} \quad [\mathrm{s}] \end{aligned} \tag{13}$$

where $T_{\mathrm{ED}}^{(proc)}$ is the time needed by an ED to generate a packet, $ToA_{UL}(SF)$ is the ToA for transmitting the uplink data for a given value of SF, $T_{\mathrm{GW}}^{(proc)}$ is the processing time needed by the GW (which just performs packet forwarding), τ_{GW-NS} is the propagation delay needed to reach the NS from the GW (via Internet), and $T_{\mathrm{NS}}^{(proc)}$ is the processing time required at the network server.

Finally, RTT has been defined as the interval of time between the generation of a query at the NS, to be sent in downlink to a given ED, and the instant when the NS receives the reply in uplink from the ED itself. This delay strongly depends on the operating class used by the ED. In particular, since in Class A the receive window is opened only after an uplink message, the NS is able to send a downlink message (which contains the query) only after the correct transmission by the ED. This means that if an uplink packet is lost, the NS will not know that the receive window of the ED is open and it will not send the downlink message, waiting for the next uplink packet. Therefore, in the case of Class A devices, the packet to be sent in downlink remains at the NS for a certain amount of time, denoted as $T_{\mathrm{NS}}^{(wait)}$, which depends on the periodicity with which packets are generated and on the probability that they are successfully received at the GW/NS. Therefore, RTT for Class A devices, RTT_{A}, is given by:

$$\begin{aligned} RTT_{\mathrm{A}} =& T_{\mathrm{NS}}^{(wait)} + ToA_{UL}(SF) + T_{\mathrm{ED}}^{(RXwind)} + \\ &+ ToA_{DL}(SF) + L \quad [\mathrm{s}] \end{aligned} \tag{14}$$

where $T_{\mathrm{NS}}^{(wait)} = \frac{T}{2} + T \cdot (1 - P_{\mathrm{s}})$ is the average waiting time of the packet at the NS, given that EDs generate packets in uplink every T and these packets have a probability to be

correctly received P_{s}, and $T_{\mathrm{ED}}^{(RXwind)}$ is the interval between the uplink transmission and the beginning of the first receive window opened by the ED, denoted in the standard as RECEIVE_DELAY1.

In Class C, instead, as described in Section II, the NS can send a packet to the ED at almost every instant (except during ED transmissions and taking into account possible duty cycle limitation), therefore, there is no need to wait for an uplink message from the ED. The RTT for Class C devices, RTT_{C}, is given by:

$$RTT_{\mathrm{C}} = \tau_{NS-GW} + ToA_{DL}(SF) + L \quad [\mathrm{s}] \qquad (15)$$

VI. Numerical Results

A. Methodology and Experimental Results

CA-ADR algorithm has been tested and compared with the standard solution, considering an integrated approach exploiting simulations and experiments.

As far as the simulation is concerned, a proprietary MATLAB® simulator, implementing the models presented in Section III and the ADR algorithms reported in Section IV, has been developed. At each transmission, after the SF assignment, the simulator checks if collisions among packets sent with the same SF happen and if they result in a loss (see condition 2 in Subsection III-B). This allows to compute P_{s} and S.

In addition, experiments have been conducted to characterize delays. For this purpose, we used the Chirpstack project to setup our LoRaWAN network (i.e., the network server). As for the GW we used the LoRaWAN™ EMB-GW1301-O gateway, based on the *Semtech SX 1301* chipset working at 868 MHz, with network connectivity provided via Ethernet. Finally, concerning EDs, we used a board (Idesio Rigers Board 1.0) equipped with the microchip RN2483 radio transceiver, fully certified 433/868 MHz SX1276 LoRa module and supporting LoRaWAN Class A devices.

The above described setup has been replicated in two different architectures: i) cloud-based case, where the NS and all its components were installed on a powerful computing machine and we assumed the NS was located in the cloud (see below); ii) fog-based case, where the NS was running on a Raspberry Pi 3 Model B, connected via the University Local Area Network (LAN) to the GW.

We started measuring the propagation delay, τ_{GW-NS}, introduced when considering the two architectures. For this purpose, *ping* sessions of 30 minutes each have been carried out considering a public remote NS provided by A2A Smart City[3], which can be reasonably assumed as a cloud server, as well as the Raspberry Pi server implementation, connected via our LAN to the GW for the fog case. Results of the measurements are provided in Table III.

We then measured the processing time at the EDs, $T_{\mathrm{ED}}^{(proc)}$: results are reported in Table IV and, as we can see, this time increases with SF. As for the processing time at the GW and NS, $T_{\mathrm{GW}}^{(proc)}$ and $T_{\mathrm{NS}}^{(proc)}$, negligible times have been measured in all traffic conditions analyzed.

[3]https://www.a2asmartcity.it/

TABLE III: Propagation Delays

τ	Average	Standard Deviation
τ_{GW-NS} (Cloud)	26.2 ms	16.9 ms
τ_{GW-NS} (Fog)	0.4 ms	0.2 ms

TABLE IV: $T_{\mathrm{ED}}^{(proc)}$

SF	Average	Standard Deviation
SF7	61,6 ms	4,26 ms
SF8	68,5 ms	1,9 ms
SF9	90,8 ms	2,78 ms
SF10	132,6 ms	2,33 ms
SF11	260,3 ms	2,6 ms
SF12	389,9 ms	0,98 ms

Finally, in order to understand the limits, in terms of computing capabilities, of the Raspberry Pi implementation for the NS, we have investigated a situation of high traffic conditions. Since we had at our disposal only 15 real EDs, we used the Lorhammer traffic emulator[4], that is an open-source tool able to emulate LoRaWAN traffic and redirect it to the NS, that was implemented on our Raspberry Pi. Such tool offers the possibility to specify the number of GWs and EDs in the network, the frequency with which EDs send packets and the packets size, that is the offered throughput as defined in eq. (2).

The outcomes of the above experiments have been provided as input to a simulator, characterizing the network-level performance, to derive the latency and RTT as defined above. Figure 3 shows such methodology.

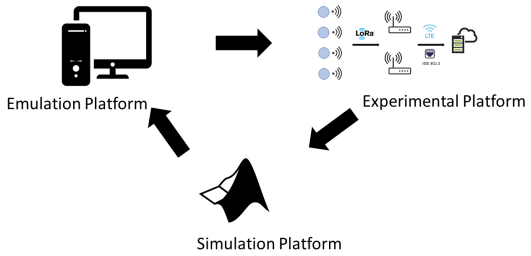

Fig. 3. Evaluation methodology.

B. Benchmarking the CA-ADR algorithm

The considered scenario is shown in Figure 4, where two GWs are deployed in a fixed position and EDs are randomly distributed in a squared area. The parameters used in the simulation are reported in Table V. If not otherwise specified, the case of uncorrelated shadowing with standard deviation σ is considered.

In order to understand the limitations of the new algorithm in terms of the number of devices it is able to handle, we plotted the Packet Success Rate and the Network Throughput

[4]http://lorhammer.itk.fr/

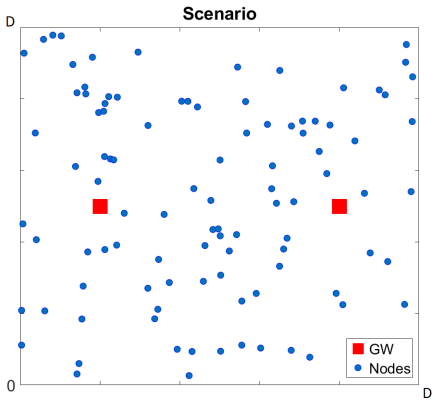

Fig. 4. Reference Scenario.

TABLE V: Simulation Parameters.

f	868, 5 MHz	BW	125 kHz
CR	4/5	P_{T}	14 dBm
N	100	G	2
T	5 s	D	3 km
γ	1.5	B	16 bytes
β	3	σ	3 dB
$P_{\mathrm{T}_{\mathrm{max}}}$	14 dBm	$P_{\mathrm{T}_{\mathrm{min}}}$	2 dBm
L_{preamble}	8	H	1
h_b	30 m	h_m	1 m
σ_{G}	1 dB	σ_{B}	6 dB

as a function of N. In Figure 5, the Packet Success Rate as a function of the number of the EDs in the network is presented. As expected, the success probability decreases with N due to the increasing of the collision probability. The new CA-ADR algorithm performs notably better because it takes into account the overall distribution of EDs in the network, assigning SFs in a more fair way. Collisions are not taken into account by the standard algorithm, which assigns the lowest SF possible to all devices by looking only at the link-level performance. Our algorithm, instead, reduces as much as possible the set of nodes using the same spreading factor, by keeping track of the number of EDs already using a given SF and assigning (when possible from the connectivity viewpoint) a different value of SF, reducing the collision probability. In the figure we report also results obtained with the ADR algorithm proposed in [16], [17]. As expected, the latter protocol performs better than the standard one, since it assigns spreading factors based on the measure of the average SNR, rather than the maximum one. However, our solution outperforms also this benchmark, since it uses average SNR measure as done in [16], [17] and, in addition, includes features to reduce collisions, not considered in [16], [17].

In Figure 6 we show the Network Throughput as a function of the number of EDs. Performance, in this case, is a direct consequence of the behavior of the Packet Success Rate. Indeed, for low values of N, S increases with N, being proportional to it (see eq. (12)) and being P_{s} high; on the contrary, when N becomes too large, P_{s} starts decreasing

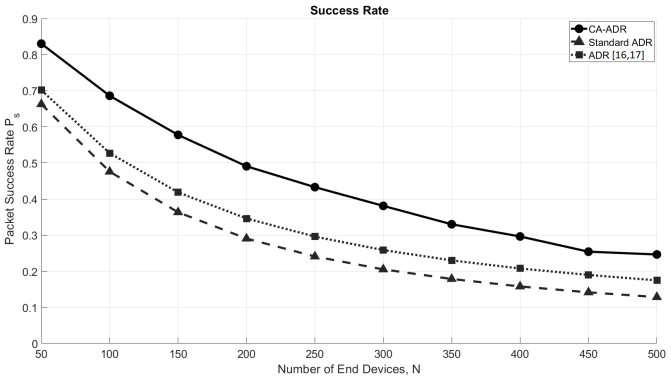

Fig. 5. Packet Success Rate, P_{s}, as a function of the number of EDs, N.

dramatically, resulting in a saturation effect for S. Therefore, the improvement of the proposed solution w.r.t. the standard one increases by increasing N.

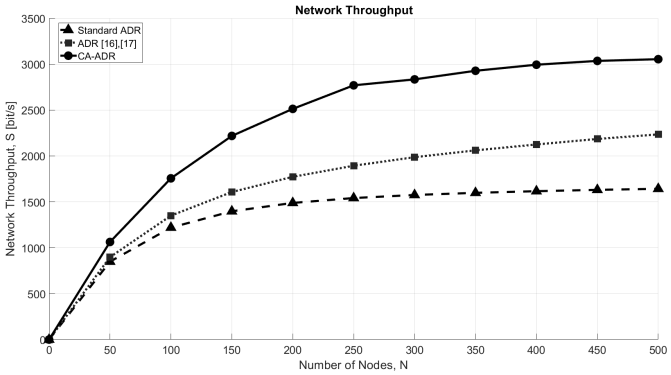

Fig. 6. Network Throughput, S, as a function of the number of EDs, N.

In Figure 7 we report the behaviour of the Network Throughput by varying the area size. The new algorithm performs better w.r.t. the standard solution, until a given area size is reached, after which they converge. S for the CA-ADR case decreases with D, since this algorithm is only limited by connectivity problems, so its performance is maximized for small areas. With increasing values of D, the algorithm needs to assign $SF = 12$ to more and more nodes in order to reach the GW, resulting in an increased number of collisions. On the contrary, the standard solution presents an optimum value of D maximizing S: for small area sizes, it tends to assign $SF = 7$ to all nodes, which is enough to reach the GW, and this results in many collisions (many nodes using the same SF), while for large areas, as in the case of CA-ADR, the algorithm tends to assign $SF = 12$ to all nodes to overcome connectivity problems, resulting again in many collisions (this is the reason why the two algorithms converge for $D > 8000$ m). The peak value of S for the standard solution is reached for $D = 6000$ m, where some EDs use a higher SF to maintain connectivity, so the overall number of nodes using the same SF is lower with respect to smaller areas, resulting in less collisions. The solution in [16], [17]

shows some improvements, because it takes into account the average of the SNR values, so it is more conservative and it assigns a low SF value with less probability w.r.t. the standard. However, CA-ADR still performs better for areas with side $D < 5000$ m.

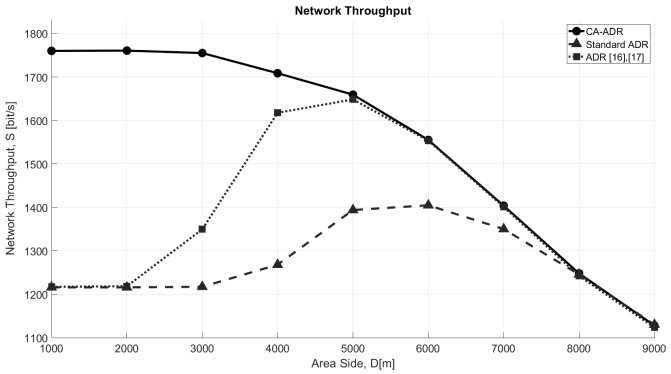

Fig. 7. Network Throughput, S, as a function of the area side, D [m].

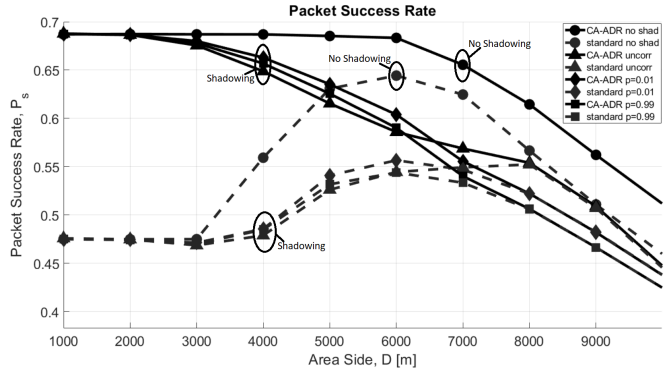

Fig. 8. Packet Success Rate, P_{s}, as a function of the area side, D [m] with shadowing analysis.

Figure 8 shows the impact of the shadowing on the performance of the algorithms. The inclusion of shadowing strongly reduces performance for both solutions, standard and CA. However, performance does not change significantly when considering different channel models in the presence of shadowing, that is for the uncorrelated or correlated channels by varying p (i.e., the probability of remaining in *Good* or *Bad* channel conditions). However, it is important to notice that, in all cases, the proposed solution is always better than the standard one, because the assignment of the SF is always carried out in order to reduce collisions, when possible.

Figure 9 shows the Packet Success Rate as a function of T. Since the CA-ADR algorithm is specifically designed to reduce collisions, the lower is T, the higher will be the offered throughput, and the larger will be the gap w.r.t. the standard, because collisions are better managed by CA-ADR. In addition, the figure shows the impact of the presence of external interference. A situation where $N_{\mathrm{i}} = 50$ interfering nodes are randomly and uniformly deployed in the area has been simulated; they generate a data packet, assumed to be transmitted with the same transmit power used by LoRa

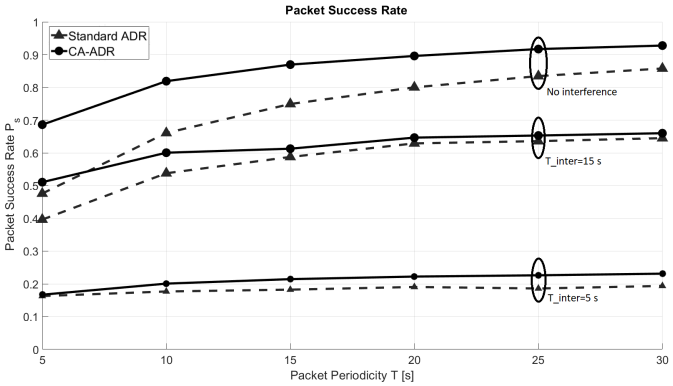

Fig. 9. Packet Success Rate, P_{s}, as a function of the packet periodicity, T [s].

devices and assuming that the packet occupies the channel for 100 ms. Results have been obtained by modifying the frequency of generation of packets from the interferers, that is varying the level of interference generated in the network; in particular, they are assumed to transmit every $T_{\mathrm{i}} = 5$ s and $T_{\mathrm{i}} = 15$ s. As it can be seen, even though the performance is generally worse due to the presence of external interference, the CA-ADR still performs better, although the difference with the standard algorithm becomes smaller.

We also tried to investigate the differences in terms of delay the two algorithm provide as a result of their decision. Figure 10 shows the RTT as a function of the number of EDs. In case of Class A, the factor which has more relevance is the collision probability, because an uplink packet needs to be received by the NS before it could be able to transmit a packet in downlink. Therefore, the CA-ADR algorithm performs better (i.e., the RTT is lower) w.r.t. the standard solution. In case of Class C, instead, RTT does not depend on P_{S} because the ED is always listening, so the NS can send downlink packets even if it has not received anything from the ED. Therefore, the standard solution performs better, because it tends to assign lower SFs (i.e., minimum value of SF allowing connectivity), resulting in lower transmission times, on average. Such results have been computed considering the cloud implementation.

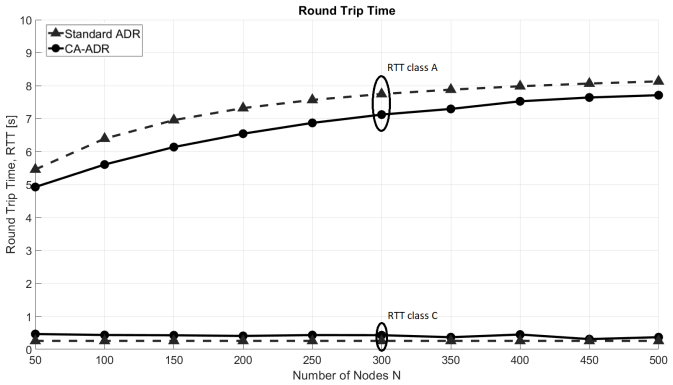

Fig. 10. Round Trip Time, RTT, as a function of the number of EDs, N.

As a final benchmark, considering that the complexity of the

CA-ADR algorithm grows linearly with the number of nodes N, we measured during our experiments with the simulator that in the worst case analyzed ($N = 500$ and $T = 5$ s), the algorithm execution time never exceeded a few milliseconds.

C. Comparing Cloud and Fog Architectures

In our scenario we consider two GWs, which can either be managed by a single centralized NS (cloud case), or are associated to two separated NSs (fog case). Therefore, in the first case, CA-ADR algorithm runs considering all EDs in the area, while in the second case, EDs are divided into two subsets according to the RSSI value (for the sake of simplicity, the strongest GW is selected) and the algorithm runs on the two separated subsets. No substantial differences have been observed in terms of SF allocation to EDs, therefore no significant differences in terms of P_{s} can be observed (see Table VI). In particular, it can be seen that the advantage achievable in the cloud configuration (where a joint optimization of the entire area is considered) is negligible. On the other hand, it is expected that the cloud architecture will bring to larger latency, and this is analyzed in the following.

TABLE VI: Success rate in Cloud and Fog architectures

N	P_{s} Cloud	P_{s} Fog
10	0.997	0.986
50	0.962	0.960
100	0.922	0.921
150	0.884	0.884
200	0.852	0.849

Figure 11 shows how the latency evolves when increasing the area size, for the two architectures. By increasing D, higher SF are needed to reach the GWs, therefore the ToA of the packet and then the latency increase. In addition, as expected, the cloud architecture results in larger latency, due to the time needed to reach the cloud infrastructure, where the NS is deployed.

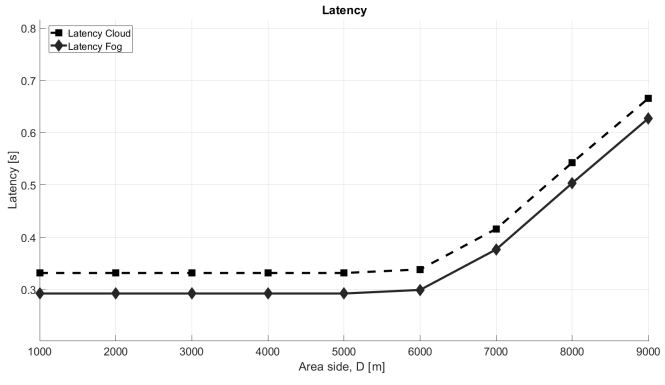

Fig. 11. Latency, L, as a function of the area side, D

The above results show that the cloud architecture works worsening in terms of latency, while slightly improving the network throughput performance. However, to draw final conclusions, it is necessary also to check if a fog-based solution is feasible also in terms of computing capabilities. To this aim, we derived an upper bound in terms of the maximum amount of traffic (in bit/s) the network server is able to process without crashing or causing relevant delay for the user. In our experiments we fixed the packet size to 100 bytes and the packet periodicity equal to 0.1 s and we varied the number of EDs generating data. For each value of N we emulated 10000 packets transmissions to check the CPU usage and to check possible crashing of the NS. The Raspberry Pi implementation proved to handle up to an offered traffic of 40 Mbit/s, reached by setting $N = 5000$, with almost 100% of CPU usage. Such limit is largely compliant with all LPWAN applications, as identified in [15], which demonstrates that the fog solution is sufficiently powerful to manage a LoRaWAN network and this architecture does not present particular drawbacks.

VII. Conclusion

In this paper, a new Adaptive Data Rate algorithm for LoRaWAN network, called Collision-Aware ADR, has been proposed, which takes into account collision probability at the MAC layer to assign data rates to end devices. Simulation and experimental approaches have been jointly used in order to compare the new algorithm with two benchmark solutions, considering different performance metrics. Results show that CA-ADR outperforms the standard solution for networks that are not strongly limited by connectivity issues (in this case the two algorithms provides the same performance). This is because CA-ADR exploits orthogonality of signals emitted with different data rates, a fact that allows to drastically reduce collisions among transmissions. It is also shown that random channel fluctuations degrade the performance, but this happens with any solution, as any ADR algorithm presents some issues when applied to dynamic channel scenarios.

In addition, cloud- and fog-based architectures have been setup and compared in terms of network throughput, latency and processing capabilities. Results demonstrate that the fog architecture is feasible, since even if the NS is deployed on a common Raspberry Pi, it is still able to manage a sufficient amount of traffic for many real-life IoT applications. In addition, the fog architecture allows to reduce the end-to-end latency as expected, while maintaining very good performance in terms of network throughput when compared to the cloud-based scenario. Finally, we underline that the lower latency achievable with the fog architecture can help also in reacting to dynamic environmental changes, i.e. the NS can quickly send commands to EDs to change transmission parameters.

VIII. Acknowledgment

This work was carried out in the framework of the CNIT National Laboratory WiLab and of the COST Action CA15104 "IRACON".

References

[1] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of things: A survey on enabling technologies, protocols, and applications," *IEEE Communications Surveys Tutorials*, vol. 17, no. 4, pp. 2347–2376, 2015.

[2] P. Datta and B. Sharma, "A survey on iot architectures, protocols, security and smart city based applications," in *2017 8th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*, 2017, pp. 1–5.

[3] M. Dholu and K. A. Ghodinde, "Internet of things (iot) for precision agriculture application," in *2018 2nd International Conference on Trends in Electronics and Informatics (ICOEI)*, 2018, pp. 339–342.

[4] C. Lu, A. Saifullah, B. Li, M. Sha, H. Gonzalez, D. Gunatilaka, C. Wu, L. Nie, and Y. Chen, "Real-Time Wireless Sensor-Actuator Networks for Industrial Cyber-Physical Systems," *Proceedings of the IEEE*, vol. 104, no. 5, pp. 1013–1024, May 2016.

[5] M. R. Palattella, N. Accettura, X. Vilajosana, T. Watteyne, L. A. Grieco, G. Boggia, and M. Dohler, "Standardized protocol stack for the internet of (important) things," *IEEE Communications Surveys Tutorials*, vol. 15, no. 3, pp. 1389–1406, 2013.

[6] C. Goursaud and J.-M. Gorce, "Dedicated networks for IoT: PHY/MAC state of the art and challenges," *EAI Endorsed Trans. Internet Things*, vol. 1, no. 1, pp. 1–11, Oct. 2015.

[7] A. Khalifeh, K. A. Aldahdouh, K. A. Darabkh, and W. Al-Sit, "A survey of 5g emerging wireless technologies featuring lorawan, sigfox, nb-iot and lte-m," in *2019 International Conference on Wireless Communications Signal Processing and Networking (WiSPNET)*, 2019, pp. 561–566.

[8] K. E. Nolan, W. Guibene, and M. Y. Kelly, "An evaluation of low power wide area network technologies for the internet of things," in *2016 International Wireless Communications and Mobile Computing Conference (IWCMC)*, 2016, pp. 439–444.

[9] N. I. Osman and E. B. Abbas, "Simulation and modelling of lora and sigfox low power wide area network technologies," in *2018 International Conference on Computer, Control, Electrical, and Electronics Engineering (ICCCEEE)*, 2018, pp. 1–5.

[10] G. Pasolini, C. Buratti, L. Feltrin, F. Zabini, C. D. Castro, R. Verdone, and O. Andrisano, "Smart city pilot projects using lora and ieee 802.15.4 technologies," *Sensors Journal*, pp. 1–17, 2018.

[11] G. Pătru, D. Trancă, C. Costea, D. Rosner, and R. Rughiniş, "Lora based, low power remote monitoring and control solution for industry 4.0 factories and facilities," in *2019 18th RoEduNet Conference: Networking in Education and Research (RoEduNet)*, 2019, pp. 1–6.

[12] G. Davoli, W. Cerroni, S. Tomovic, C. Buratti, C. Contoli, and F. Callegati, "Intent-based service management for heterogeneous software-defined infrastructure domains," *International Journal of Network Management*, vol. 29, no. 1, p. e2051, 2019.

[13] J. Pan and J. McElhannon, "Future edge cloud and edge computing for internet of things applications," *IEEE Internet of Things Journal*, vol. 5, no. 1, pp. 439–449, 2018.

[14] V. K. Sarker, J. P. Queralta, T. N. Gia, H. Tenhunen, and T. Westerlund, "A survey on LoRa for IoT: Integrating edge computing," in *2019 Fourth International Conference on Fog and Mobile Edge Computing (FMEC)*, Rome, Italy, June 2019, pp. 295–300.

[15] L. Feltrin, C. Buratti, E. Vinciarelli, R. De Bonis, and R. Verdone, "LoRaWAN: Evaluation of Link- and System-Level Performance," *IEEE Internet of Things Journal*, vol. 5, no. 3, pp. 2249–2258, June 2018.

[16] M. Slabicki, G. Premsankar, and M. Di Francesco, "Adaptive configuration of LoRa networks for dense IoT deployments," *NOMS 2018 - 2018 IEEE/IFIP Network Operations and Management Symposium*, pp. 1–9, April 2018.

[17] V. Hauser and T. Hégr, "Proposal of Adaptive Data Rate Algorithm for LoRaWAN-Based Infrastructure," *2017 IEEE 5th International Conference on Future Internet of Things and Cloud (FiCloud)*, pp. 85–90, 2017.

[18] LoRaAlliance, *LoRaWAN Specification (V1.0.2)*, [Online] Available: https://lora-alliance.org/resource-hub/lorawanr-specification-v102, 2016.

[19] LoRa Alliance, *LoRaWAN 1.1 Regional Parameters*, [Online] Available: https://lora-alliance.org/resource-hub/lorawanr-regional-parameters-rp002-100, 2017.

[20] Semtech Corporation, "LoRa Modulation Basics," [Online] Available: http://wiki.lahoud.fr/lib/exe/fetch.php?media=an1200.22.pdf, 2015.

[21] J. de Carvalho Silva, J. J. P. C. Rodrigues, A. M. Alberti, P. Solic, and A. L. L. Aquino, "LoRaWAN — A low power WAN protocol for Internet of Things: A review and opportunities," *2017 2nd International Multidisciplinary Conference on Computer and Energy Science (SpliTech)*, pp. 1–6, July 2017.

[22] R. S. Sinha, Y. Wei, and S.-H. Hwang, "A survey on LPWA technology: LoRa and NB-IoT," *ICT Express*, vol. 3, pp. 14–21, March 2017.

[23] 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," *2015 14th International Conference on ITS Telecommunications (ITST)*, pp. 55–59, Dec 2015.

[24] A. Waret, M. Kaneko, A. Guitton, and N. El Rachkidy, "Lora throughput analysis with imperfect spreading factor orthogonality," *IEEE Wireless Communications Letters*, vol. 8, no. 2, pp. 408–411, 2019.

[25] M. Bor, U. Roedig, T. Voigt, and J. Alonso, "Do LoRa Low-Power Wide-Area Networks Scale?" *Proceedings of the 19th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, p. 59–67, November 2016.

[26] S. Kim, Y. Yoo, "Contention-Aware Adaptive Data Rate for Throughput Optimization in LoRaWAN," *Sensors*, vol. 18, no. 6, May 2018.

[27] F. Cuomo, M. Campo, A. Caponi, G. Bianchi, G. Rossini, and P. Pisani, "Explora: Extending the performance of lora by suitable spreading factor allocations," in *2017 IEEE 13th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, 2017, pp. 1–8.

[28] H. Truong, "Enabling edge analytics of IoT data: the case of LoRaWAN," in *2018 Global Internet of Things Summit (GIoTS)*, Bilbao, Spain, June 2018.

[29] T. Nguyen Gia, J. P. Queralta, and T. Westerlund, "Exploiting LoRa, edge, and fog computing for traffic monitoring in smart cities," in *LPWAN Technologies for IoT and M2M Applications*, B. S. Chaudhari and M. Zennaro, Eds. Academic Press, 2020, pp. 347 – 371.

[30] P. Fraga-Lamas, M. Celaya-Echarri, P. Lopez-Iturri, L. Castedo, L. Azpilicueta, E. Aguirre, M. Suárez-Albela, F. Falcone, and T. M. Fernández-Caramés, "Design and Experimental Validation of a LoRaWAN Fog Computing Based Architecture for IoT Enabled Smart Campus Applications," *Sensors*, vol. 19, no. 15, p. 3287, July 2019.

[31] M. Hata, "Empirical formula for propagation loss in land mobile radio services," *IEEE Transactions on Vehicular Technology*, vol. 29, no. 3, pp. 317–325, 1980.

[32] M. Aydinlik and M. Salehi, "16-qam turbo coded modulation for the gilbert-elliot channel model," in *2002 IEEE International Conference on Communications. Conference Proceedings. ICC 2002 (Cat. No.02CH37333)*, vol. 3, 2002, pp. 1393–1397 vol.3.

[33] M. Chiani, E. Milani, and R. Verdone, "A semi-analytical approach for performance evaluation of tcp-ip based mobile radio links," in *Globecom '00 - IEEE. Global Telecommunications Conference. Conference Record (Cat. No.00CH37137)*, vol. 2, 2000, pp. 937–942 vol.2.

[34] D. Bankov, E. Khorov, and A. Lyakhov, "Mathematical model of lorawan channel access with capture effect," in *2017 IEEE 28th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*, 2017, pp. 1–5.

[35] Semtech Corporation, "LoRa Design Guide," [Online] Available: https://www.rs-online.com/designspark/rel-assets/ds-assets/uploads/knowledge-items/application-notes-for-the-internet-of-things/LoRa\%20Design\%20Guide.pdf, 2014.

[36] SemtechCorporation, *LoRaWAN – simple rate adaptation recommended algorithm*, [Online] Available: https://www.thethingsnetwork.org/forum/uploads/default/original/2X/7/7480e044aa93a54a910dab8ef0adfb5f515d14a1.pdf, 2016.

[37] D. Croce, M. Gucciardo, S. Mangione, G. Santaromita, and I. Tinnirello, "Impact of LoRa Imperfect Orthogonality: Analysis of Link-Level Performance," *IEEE Communications Letters*, vol. 22, no. 4, pp. 796–799, 2018.

[38] N. Abramson, "The aloha system: Another alternative for computer communications," *Proceedings of the November 17-19, 1970, Fall Joint Computer Conference*, p. 281–285, 1970. [Online]. Available: https://doi.org/10.1145/1478462.1478502