# Probabilistic and distributed traffic control in LPWANs

Kawtar Lasri, Yann Ben Maissa, Loubna Echabbi, Oana Iova, Fabrice Valois

# Probabilistic and Distributed Traffic Control in LPWANs

Kawtar Lasri[1], Yann Ben Maissa[2], Loubna Echabbi[3], Oana Iova[4], and Fabrice Valois[5]

[1,4,5]Univ Lyon, INSA Lyon, Inria, CITI, EA3720, 69621 Villeurbanne, France
[1,2,3]INPT Rabat, Morocco

**Abstract**

Low-power, low data transmission rates, and long-range wireless networks, also known as LPWANs, are intended to work best with equipment that uses few resources and can be used for many years thanks to their long battery life operation. This type of networks can handle traffic from nearly 1,000 nodes while maintaining a duty cycle of less than 1%. However, as the nodes become denser, the number of collisions increases and network traffic management becomes mandatory. To address this concern, we propose a Distributed and Probabilistic Traffic Control algorithm (DiPTC) that allows nodes to change their traffic in response to the application requirements (e.g., acquiring $K$ measurements over a period of time) while being agnostic to the number of nodes or the network topology. When this requirement is not achieved, the gateway sends a feedback message to all the nodes so that they may adapt their traffic. We compare the proposed solution to LoRaWAN and to a Centralized Optimal Traffic Control solution (COTraC), in simulation. Compared to LoRaWAN, our algorithm proved successful in achieving the objective while minimizing collisions and extending the network lifetime threefold.

## 1 Introduction

One of the primary applications for deploying sensors in smart buildings and smart cities is data collection. Low Power Wide Area Networks (LPWANs), such as LoRaWAN [1] or Weightless [2], are quickly becoming a popular choice because they provide an infrastructure for data collection at a low cost while still covering large area [3–5].

Although they are appealing, these networks have some major drawbacks, such as the used frequencies (e.g., 868 MHz in Europe, 915 MHz in USA) and medium access protocols (e.g., Aloha) [6]. For large-scale deployments that need to be able to support a lot of traffic, controlling the traffic sent is necessary. This is especially true for two different types of scenarios. The first one is in applications that aim to

obtain a sampling of a certain situation in an area (e.g., environmental monitoring): only a specific amount of measurements per time unit is required. Obtaining more data has no effect and would simply result in network congestion, collisions, and energy waste. On the other hand, receiving less information results in an inefficient application. The second scenario is when an LPWAN is operated by a telecommunication company providing coverage and connectivity to their customers. A Service-Level Agreement (SLA) specifies the maximum capacity provided by the company and establishes the client traffic model. Unfortunately, LPWANs lack an algorithm for managing user traffic and the number of nodes linked to the network. As a result, a client may transmit more data than allowed and establish connections with additional end devices, resulting in network congestion. We believe that these situations will become increasingly common in LPWANs. While the goal of LoRaWAN Adaptive Data Rate (ADR) [7–9] is to minimize energy consumption and adapt transmission data rates in response to radio link budget and environmental factors, this algorithm is not capable of correctly addressing network congestion, nor impact traffic across multiple nodes.

To solve this problem, we presented in [10] a new Distributed Probabilistic Traffic Control algorithm (DiPTC) for LPWANs that enables the network manager and the applications to better control data collection. In DiPTC, when the application requires a specific amount of data (e.g., $K$ measurements over a period of time), a central server operates a control loop to impose a traffic policy that complies with the requirements. The central server uses a downlink message to inform the nodes if the target was achieved or not. After receiving this control message, nodes use a local algorithm to modify their traffic intensity without having any neighborhood knowledge. We present here a major improvement of DiPTC that considerably decreases the energy consumption of end devices. We also study the impact of the traffic control parameters on DiPTC performance, and we make an extensive performance evaluation by comparing DiPTC against LoRaWAN and a Centralized Optimal Traffic Control solution (COTraC).

The remainder of the paper is structured as follows: we discuss the state of the art for traffic control in wireless sensor networks and LPWANs in Section 2, and we introduce and explain our proposed solution in Section 3. Section 4 describes the simulation setup and scenarios used for the performance evaluation of DiPTC in sections 5, 6, and 7 to validate and compare our results to those of the baseline LoRaWAN and the COTraC solutions. We discuss our findings in Section 8 and we conclude our work in Section 9.

# 2 Related work

While traffic control can have several benefits in a network (e.g., reducing the number of collisions, decreasing energy consumption, and limiting congestion), there are several methods through which this can be achieved. We present in this section, a selection of approaches for this purpose, from the literature.

1. *Data aggregation.* Many researchers in wireless sensor networks propose spatial and temporal data aggregation as a solution for traffic and thus collision reduction. Spatial data aggregation is based on network node organization and is accomplished by selecting a leader or a group of leader devices that are accountable for transmitting the gathered data to the gateway [11, 12]. To collect the data,

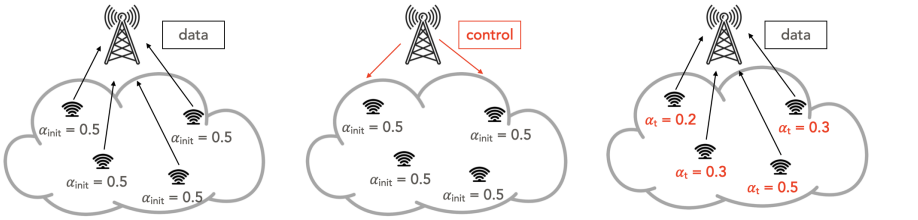
communication between the messenger(s) and the nodes is essential. [13–15] describe temporal data aggregation based on data prediction, with the authors focusing on temporal aggregation functions using ARIMA, ARMA, or LMS-PCS prediction models. When a threshold error occurs, the nodes modify their prediction model and send their new model coefficients to the gateway. The gateway uses these coefficients to predict data.

To solve the stated problem using data aggregation methods, nodes must have a large memory capacity, neighborhood knowledge, and the ability to communicate with one another. Unfortunately, LPWANs cannot fulfill these criteria. As a result, temporal aggregation techniques cannot be applied to our scenarios: limiting traffic to collect only $K$ samples per time period $\Delta T$.

2. *Resource allocation.* In LoRaWAN, researchers try to reduce the number of collisions and improve scalability by optimizing resource allocation, such as spreading factor and transmission power [8, 9, 16, 17]. Ta *et al.* present LoRaMAB, a flexible decentralized learning resource allocation approach based on the Multi-Armed-Bandit reinforcement algorithm [9]. Their approach outperforms the traditional Adaptive Data Rate (ADR) algorithm used in LoRaWAN. Another important resource that can be optimized is the frequency channel, as transmissions in different channels do not interfere with each other. Shen *et al.* propose to optimize the channel assignment and the backoff time between two consecutive transmissions [18]. Chinchilla *et al.* propose a combination of channel allocation and spreading factors [19], and Qin *et al.* combine channel allocation with the optimization of the transmission power [20]. However, these approaches only adapt the physical layer parameters of the existing traffic to the environmental conditions. While they can indeed improve the capacity of the network, they do not control the amount of traffic sent by the nodes, and they do not ensure that the traffic respects the constraints of the application. Still, they can easily be combined with traffic reduction algorithms for better efficiency.

3. *Traffic scheduling.* Collisions can be reduced by scheduling all the traffic in the network. This schedule can be done by a central entity that assigns transmission slots to each end-node based on several parameters [17, 21, 22], or in a distributed manner, where each end-node chooses its transmission slot [23]. However, both approaches add unwanted overhead in the network (due to the dissemination of the transmission schedule) and demand high clock accuracy of end-nodes (which is hard to meet in real life). A simpler approach would be to only offset the transmission timings, to avoid unnecessary packet transmission, as proposed by Kaburaki *et al.* [24]. The authors propose an autonomous decentralized traffic control system that uses Q-learning to automatically manage transmission times in order to reduce the probability of packet collision while improving communication quality in event-driven traffic. Q-learning is used to determine the transmission time offset and the probability. Assuming an ideal downlink, the transmission probability is automatically controlled based on the downlink acknowledgment. They use transmission probability to limit the number of nodes that transmit event packets, thereby reducing the probability of packet collision. However, none of these schemes actually reduces the amount of traffic in the network and cannot be used to control the generated traffic.

4. *Traffic control.* While not present in current IoT approaches as deemed unnecessary, traffic control algorithms are heavily used in the Internet, where networks have to handle huge data flows. Transport layer protocols such as TCP [25]




(a) Initial setup. (b) Feedback control sent by the gateway. (c) Local update of the transmission probability.

Figure 1: DiPTC in a nutshell. (a) Initial transmission considering the initial weighting coefficient equal to 0.5 (for instance). (b) Feedback control send by the gateway to adapt the uplink traffic intensity. (c) Local computation of the new weighting coefficient and the new number of messages to send, based on the control feedback and the previous weighting coefficient.

and QUIC [26] use feedback control algorithms to adapt the sending rates to the network states and limit congestion in the network. A popular ones is the Additive-Increase/Multiplicative-Decrease (AIMD) [27] algorithm that combines linear growth of the transmission traffic when there is no congestion, with an exponential reduction when congestion is detected. Related schemes exist, such as Multiplicative-Increase/Multiplicative-Decrease (MIMD) and Additive-Increase/Additive-Decrease (AIAD). However, they do not reach stability. To the best of our knowledge, no such traffic control algorithm exists today for LPWANs, and this is where our inspiration comes from.

# 3 Towards a distributed and probabilistic approach

In low power wide area networks, such as LoRaWAN or SigFox, there is no traffic control for the end-nodes, except the limitations due to the duty cycle. No algorithms are provided to allow a network provider to control the uplink traffic. Nevertheless, it is useful from a point of view of (1) an application that requires collecting a certain number of samples periodically or (2) an IoT telecommunication operator requiring a maximum sample reception rate through several service level agreements. To that end, we propose a Distributed and Probabilistic algorithm for the Traffic Control of end nodes (DiPTC).

## 3.1 DiPTC: the big picture

The goal is to meet the application requirement, i.e., to collect $K$ packets per time period $\Delta T$ over a given area, regardless of the number of end-nodes, without adding excessive control which overloads the network, and without adding technical features at the end-node level.

In LPWANs, the data transmission is only limited by the duty cycle. In DiPTC, we propose to limit the data transmission in order to reach the application requirement, while respecting the duty cycle limitation. To this end, rather than trans-

mitting new data according to the duty cycle, in DiPTC, each node draws first a transmission probability, in order to decide if it can send data.

Assuming node $n$ is allowed to transmit, then it uses its weighting coefficient, $\alpha_n$, to determine the number of messages it should send (see Fig. 1, step (a)). At the end of the time period $\Delta T$, the network server compares the number of received packets $k$ with the objective $K$: if lower ($k < K$), the network server sends a feedback message to the gateway, which then forwards it to all nodes in the area, to increase their weighting coefficient, therefore their number of transmitted messages; if higher ($k > K$), the opposite occurs (weighting coefficient decreases - see Fig. 1, step (b)). When the feedback message is received by the end nodes, according to the DiPTC algorithm, an updated weighting coefficient, $\alpha_n$, is locally computed by each end node, and the number of messages to send is re-evaluated (see Fig. 1 step (c)).

## 3.2 Assumptions

In this paper, we examine the scenario of a single gateway covering a given area. The multi-gateway scenario can be simply analyzed since the network server has the DiPTC functionality, not gateways. We also consider that the number of nodes associated to the network, denoted by $N$, is unknown. This number may change over time as a result of subsequent deployments, hardware failures, or depleted batteries. Our proposal can be useful for monitoring applications that require samples of a physical quantity over time, such as pollution [28]. These applications only require $K$ measurements over a certain area at regular intervals (every $\Delta T$), regardless of how many nodes are in the network.

In LPWANs, the downlink is primarily used to provide control information to change the physical layer configuration, as in LoRaWAN, and to acknowledge correctly received packets by the gateway [1]. In DiPTC, we propose to use also the downlink as a source of feedback information, indicating if the application goal is reached. This feedback may include the quantity of packets received over the previous period $\Delta T$, as well as the quantity of $K$ packets required by the application, or it may include less detailed data, such as whether the quantity of packets received falls within the application objective, or not. In general, any other type of message could be considered in order to notify goal achievement. Each node can adjust its traffic to match the requirements of the application by relying on the information received and always abiding by local laws, such as the duty cycle. The implementation of the broadcast or multicast required to send this feedback message in an LPWAN is not covered in this paper but will be discussed in section 4 for the case of LoRaWAN.

## 3.3 The DiPTC algorithm

We propose DiPTC, based on the additive increase multiplicative decrease algorithm. We use a decrease factor, $x_D$, to apply a reduction strategy when the network server receives excessive amounts of data. On the other hand, we use an increasing policy with an increase factor, given by $x_I$, when the network server does not get enough data. This algorithm provides a prompt response by rapidly altering the number of transmitted messages by every node, depending on the information sent by the network server through the gateway in each time period $\Delta T$. To lighten the downlink message payload, which is highly constrained, the feedback message

sent by the network server through the gateway is a binary value. In other words, the gateway will either spread 0, meaning that too many packets were received or 1 if not enough packets were sent. When the goal is met, the network server and gateway send no downlinks, and the nodes continue to send the same number of messages for the next time period. The traffic intensity of each node rises linearly, taking the coefficient $x_I$ into account, as long as the network server does not receive enough $K$ packets required by the application per time period $\Delta T$. Otherwise, when the network server receives more than $K$ packets, the nodes reduce their traffic intensity exponentially using the coefficient $x_D$.

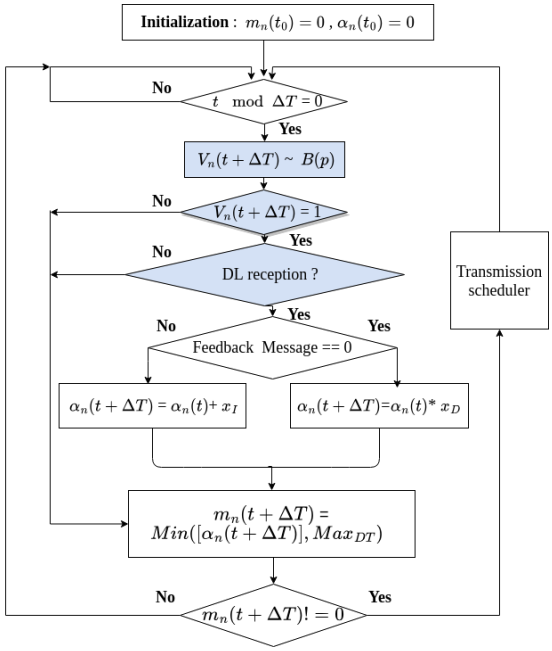



Figure 2: Finite state machine of the DiPTC algorithm, for each node.

The finite state machine in Fig. 2 describes the DiPTC algorithm adopted by each node in the network. We define $m_n(t+\Delta T)$ as the number of messages to send by the node $n$ during the next time period $\Delta T$, $\alpha_n(t+\Delta T)$ the weighting coefficient for node $n$ which is used to compute the number of messages to send, $Max_{DT}$ is the maximum number of messages a node can transmit during $\Delta T$ w.r.t. its duty cycle. Initially, the number of messages to be sent is null for all nodes: it avoids a burst of uplink messages when new end nodes are deployed in the network. The node then waits for the feedback message for the next time period $\Delta T$ before adjusting the number of messages to send. The network server only sends the feedback message via the gateway if the number of received messages differs from the required number. This reduces downlink traffic.

To avoid an event-based synchronization between nodes which would lead to a situation where all the nodes react identically with the same number of messages to send, a node decides to take into account the next feedback message following a Bernoulli distribution ($B(p)$, see the stochastic variable $V_n(t)$). If the outcome of

the Bernoulli distribution is positive, the node will wake up to receive the feedback message; otherwise, it will sleep and wake up to transmit its messages without adjusting its traffic intensity. Since receiving a downlink message consumes a considerable amount of energy in LPWANs, in this new version of DiPTC, a node opens its reception window to receive the feedback message from the gateway only if it should adapt its traffic as a result of a favorable draw ($V_n(t) = 1$), which saves energy compared to the previous version of the algorithm [10]. The finite state machine of the new version (Fig. 2) shows how we modified these three steps to improve the energy efficiency of end nodes and the network lifetime.

When the feedback message is received and taken into account, each node increases or decreases its number of transmitted messages considering the weighting coefficient $\alpha_n(t)$ according to the binary value of the feedback message received from the gateway. On the one hand, each node increases the number of messages to transmit using $x_I$, $x_I \in ]0,1]$, if the value of the feedback message is equal to 1. On the other hand, it uses the multiplication factor $x_D$ to reduce the number of transmitted messages exponentially, $x_D \in ]0,1]$. The additive factor $x_I$ and the multiplicative factor $x_D$ are constant in time and identical for every node. They represent respectively, the increase or decrease in traffic compared to the last period. When the value of $x_I$ is greater, the amount of traffic at each node increases more quickly; conversely, when the value of $x_D$ is greater, the amount of traffic decreases more slowly. Next, the node schedules its $m_n(t+\Delta T)$ messages, which is the integer portion of $\alpha_n(t+\Delta T)$. A node is said to be active when $m_n(t+\Delta T) > 0$. Optimized scheduling is out of the scope of this work, and any scheduling algorithm could be applied.

# 4 Simulation Model

While DiPTC can be used with any LPWAN technology, we focus here on applying it to LoRaWAN. To implement and test our proposal, we used LoRaSim [29], a well-known discrete event-based network simulator, as it allows to simulate collisions, capture effect, and interference in the network. We improved this simulator by adding downlink communication and a battery depletion model, as we discuss below.

## 4.1 Wireless environment

Since downlink communication is not implemented by LoRaSim in LoRaWAN, we first enhanced the simulator by simulating downlink communication. Second, we used a random variable that follows a Bernouilli distribution of parameter $P_{DL}$ to describe the reliability of the downlink. Third, we took into account the work of Roedig *et al.* [29] to describe collisions and interference on the uplink:

$$P_{rx} = P_{tx} + GL - L_{pl}(d) \tag{1}$$

$$L_{pl}(d) = \bar{L}_{pl}(d_0) + 10\gamma\log(d/d_0) + X_\sigma \tag{2}$$

where $P_{rx}$ is the power of the received signal, $P_{tx}$ the transmission power, $GL$ the accumulated general gains losses along the communication path, $L_{pl}(d)$ the path loss in dB at the communication distance $d$, $\bar{L}_{pl}(d_0)$ the mean path loss at the

reference distance $d_0$, $\gamma$ the path loss exponent, and $X_\sigma \sim N(0,\sigma^2)$ the normal distribution with zero mean, and $\sigma^2$ the variance to account for shadowing.

## 4.2 Energy consumption

No energy consumption model for the end nodes is implemented by LoRaSim. We expanded the simulator by adding the one mentioned in [30]. After sending $m$ messages to each node $n$, the energy $E_c$ used depends on the time on air $TOA$, the power used in the reception mode $P_{w_{Rx}}$, and the power used in the transmitter mode $P_{w_{Tx}}$.

$$E_c = TOA * (P_{w_{Tx}} * m + P_{w_{Rx}}) \text{ for DiPTC} \quad (3)$$

$$E_c = TOA * m * (P_{w_{Tx}} + P_{w_{Rx}}) \text{ for LoRaWAN} \quad (4)$$

The values used for the parameters in these models are presented in Table 2. The LoRa parameters (*SF, CR, BW, Freq.*) are chosen randomly at the beginning, and they do not change during the simulation. In our setup, we do not use channel hopping to be robust in case of interference and to avoid collisions, in order to study DiPTC behavior in the worst case (i.e., high collision probability).

## 4.3 Feedback message support in LPWANs

As discussed previously, the downlink message sent by the gateway is necessary to disseminate the feedback information to the end-devices in order to increase / decrease the traffic intensity to meet the application requirement. In LoRaWAN, multicast is available for Class B and Class C, but not in Class A. How to implement a multicast downlink in Class A is nowadays an open question. We could define a multicast address for a group of end-devices belonging to the same application and we could use the reception windows RX1/RX2 to disseminate the feedback message.

# 5 Overall Performance Evaluation

This section covers the performance evaluation of DiPTC. We describe here the simulation scenarios, the DiPTC configurations, the LoRa [31] settings, and comparative solutions. Then, considering several metrics (success rate, collisions rate and network lifetime), we compare the performance of DiPTC against the baseline LoRaWAN and an optimal solution.

## 5.1 Scenarios and network parameters

We consider the case of a single central gateway surrounded by $N$ randomly distributed nodes with LoRa parameters chosen at random. We evaluate and compare the performance of DiPTC with baseline LoRaWAN using the three simulation scenarios listed in Table 1. Each scenario differs from the number of nodes $N$ in the network, the number of measurements $K$ required by the application in the time period $\Delta T$, the increasing and decreasing parameters $(x_I, x_D)$, the adaptation probability $P_{adapt}$, and the simulation time *Simtime*. The adaptation parameters $x_I$, $x_D$, and $P_{adapt}$ depend on the number of nodes in the network and the traffic intensity

and were chosen accordingly. We present a detailed study on how to choose these parameters in Section 7.

To evaluate the performance of DiPTC, we consider two different scenarios: INTENSIVE and DENSE. In the INTENSIVE scenario, we consider a significantly higher traffic load to test the scalability of our proposal. In the DENSE scenario, we consider a significantly larger number of nodes to test the spatial scalability of our proposal. By testing DiPTC under these different scenarios, we are confident that it will perform well in a variety of situations. To the best of our knowledge, we are the first ones to propose a traffic control algorithm for LPWANs, so there are not other existing solutions against which we can compare. Still, we consider that it is important to have a comparison benchmark against the existing LoRaWAN solution and against an optimal solution that we call the Centralized Optimal Traffic Control algorithm (COTraC). To propose a fair comparison and to respect the application constraints (i.e., receiving $K$ measurements every time period $\Delta T$), in baseline LoRaWAN, nodes send their packet following a Poisson distribution with the rate: $AVG = \frac{\Delta T}{K} \times N$. In the COTraC solution, we assume that the network server is a central entity with access to all the information about the end devices in the network: residual energy, duty cycle, and LoRa parameters. The network server establishes an optimal scheduler, where in a round-robin way, each node sends no more than the $K$ messages per $\Delta T$ required by the application. If a node can no longer send messages, because of energy depletion or because of the duty cycle constraint, the next one takes over in sending the remaining ones.

Table 1: Simulation scenarios.

| **Scenario** | INTENSIVE | DENSE |
|---|---|---|
| DiPTC | | |
| **N** | 150 | 500 |
| **K** | 10 | 1 |
| **ΔT** | 1 min | 10 min |
| $\mathbf{x_I}$ | 0.5 | 0.5 |
| $\mathbf{x_D}$ | 0.5 | 0.5 |
| $\mathbf{P_{adapt}}$ | 0.06 | 0.5 |
| *Simtime* | 1 year | 1 year |
| LoRaWAN | | |
| Interarrival time | 15 min | 5000 min |

Table 2, summarizes the values of the different parameters used in our simulation. The values of the propagation model are determined empirically in [29], and those of the energy consumption model come from the datasheet of the Semtech SX1272 LoRa transceiver [32].

Moreover, when a packet is lost, baseline LoRaWAN is configured to re-transmit it a maximum of 8 times before dropping it, a value commonly used in different implementations given that the standard specifies a maximum of 15 re-transmissions. In DiPTC and in the COTraC there are no re-transmissions, as the nodes do not receive an unicast acknowledgment from the gateway like in LoRaWAN.

Table 2: LoRa and network parameters.

| **Parameters** | **Values** |
|---|---|
| Payload size (PL) | 20 bytes |
| Header length (H) | 0 |
| Preamble symbols | 8 |
| Downlink reception probability $P_{DL}$ | 0.99 |
| Transmission power $P_{Tx}$ | 14dBm |
| Gain and Loss $GL$ | 0 |
| Path Loss exponent $\gamma$ | 2.08 |
| Reference distance $d_0$ | 40m |
| Max. distance to the gateway | 300m |
| Path Loss at the reference distance $L_{pl}(d_0)$ | -127.41dB |
| Normal distribution $X_\sigma$ | N(0,3.57) |
| Current drawn during the receive mode $I_{Rx}$ | 11.2mA |
| Current drawn during the Transmission mode $I_{Tx}$ | 90mA |
| Current drawn during the sleep mode $I_{Sleep}$ | 1µA |
| Supply voltage | 3V |
| Battery capacity | 30J |

## 5.2 DiPTC vs. Baseline LoRaWAN vs. COTraC

In this section, we compare the overall performance evaluation of DiPTC against baseline LoRaWAN and COTraC, using the following metrics:

- **Success rate** $\mu_c$: measures the number of times the base station receives exactly the $K$ required measurements per period $\Delta T$.
- **Collisions rate** $\tau_c$: measures the number of packet collisions in the network divided by the number of transmitted packets. The packet reception rate is $1-\tau_c$.
- **Network lifetime** $t_l$: measures the time the network is able to support the application requirements (i.e., the nodes are able to send the $K$ required measurements per period $\Delta T$ before the exhaustion of their battery).

Table 3: DiPTC, Baseline LoRaWAN and COTraC performance

| Sc. | Measures | baseline LoRaWAN | DiPTC | COTraC |
|---|---|---|---|---|
| Intensive | $\mu_c$ | 0.90% | 58% | 95.3% |
| | $\tau_c$ | 0.70% | 0.074% | 0% |
| | $t_l$ | 25096min | 34702min | 54334 min |
| Dense | $\mu_c$ | 28.99% | 97.62% | 98.6% |
| | $\tau_c$ | 0.085% | 0.026% | 0% |
| | $t_l$ | 8912.83h | 13393.5h | 291033.33h |

Table 3 summarizes the simulation results for these metrics considering the two scenarios INTENSIVE and DENSE.

Despite the high traffic in the INTENSIVE scenario, DiPTC has a success rate of 58%, which is 64 times higher than LoRaWAN. Furthermore, DiPTC has a collision rate 32 times lower than LoRaWAN and a longer network lifetime, thanks to our traffic control mechanism which limits the medium use.

However, the COTraC solution outperforms DiPTC and LoRaWAN. In fact, with the overall knowledge of the network parameters and the end nodes configuration in the COTraC solution, the network server schedules the nodes traffic with respect to their energy so that the next one takes over immediately after the active node's death or before the excess of its duty cycle limit without going through a transient regime. That explains the near-perfect success rate. Note that for all scenarios the success rate in the COTraC is not 100% because the uplink reliability is taken into consideration in the evaluation. The collision rate is zero in all scenarios because only one or two nodes are active simultaneously, and send the message(s) which lowers the collision probability to zero. Furthermore, in this scenario, the application requires 10 messages every minute. Those ten messages are sent by one or two nodes in COTraC. In DiPTC, however, they are sent by one, two, or even ten different nodes. As a result, reaching the application requirement in DiPTC takes more time and energy than in COTraC. Furthermore, as previously stated, COTraC does not have a transient regime, whereas DiPTC does. All these facts explain COTraC's high success rate when compared to DiPTC.

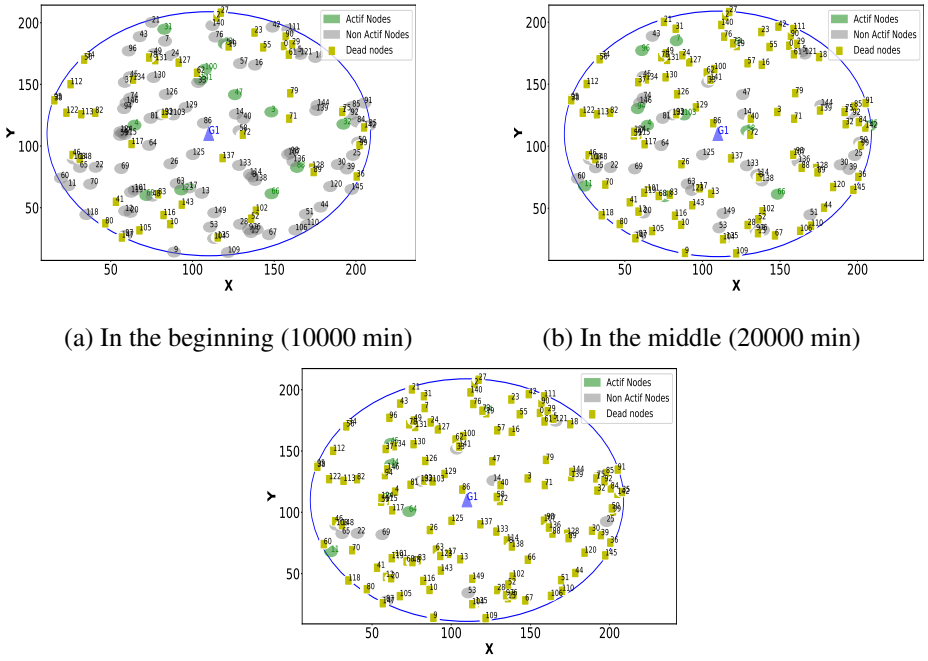
In the DENSE scenario, DiPTC presents a success rate of 97%, which is three times that of LoRaWAN. This demonstrates that, despite the high number of nodes, DiPTC can meet the application requirements more often than baseline LoRaWAN. In addition, when compared to LoRaWAN, DiPTC reduces collision rates threefold and increases network lifetime. Compared to COTraC, DiPTC performs as well as the COTraC solution in terms of success rate and collision rate with a shorter network lifetime. In fact, in this scenario one message is required by the application each period of time 10min. This message can be sent only by one node which lowers the nodes competition in DiPTC and fits perfectly with the COTraC solution where the transmission is done by one node all the time.

# 6 Horizontal and vertical scalability

=In this section, we take a deep dive into the behavior of DiPTC in order to better understand the results that we obtained in the previous section. To be more precise, we look at the number of sent and received packets, number of collisions, dead nodes, and downlink/uplink losses throughout the whole network lifetime. We evaluate these metrics in the scenarios that put the most strain on the network, allowing us to study the impact of network densification and traffic intensification on DiPTC performance.

## 6.1 Impact of the traffic intensity

In this section, we focus on the INTENSE scenario, to test the vertical scalability of DiPTC. First, we want to ensure that using DiPTC does not damage data collection by introducing a spatial bias; a problem that could be extremely detrimental to the requirements of this application. To do so, we plotted out the node activity at different times throughout the simulation in Fig. 3. A green-colored node indicates its activity. As we can see from these figures, messages sent to the gateway come

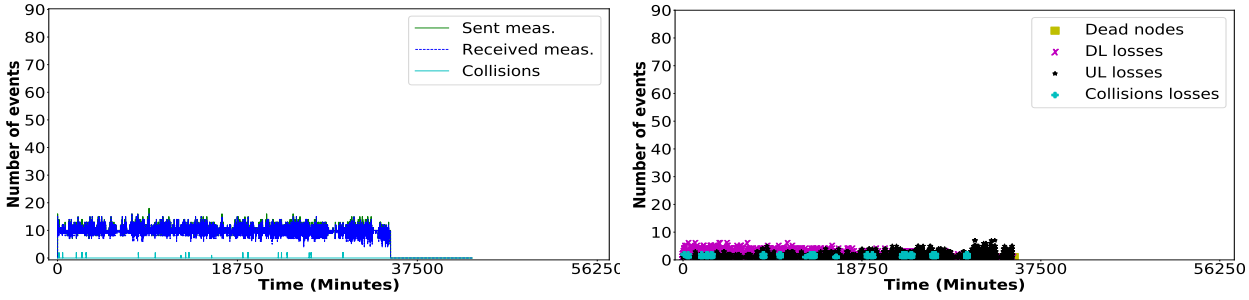



(a) In the beginning (10000 min)

(b) In the middle (20000 min)

(c) In the end (30000 min)

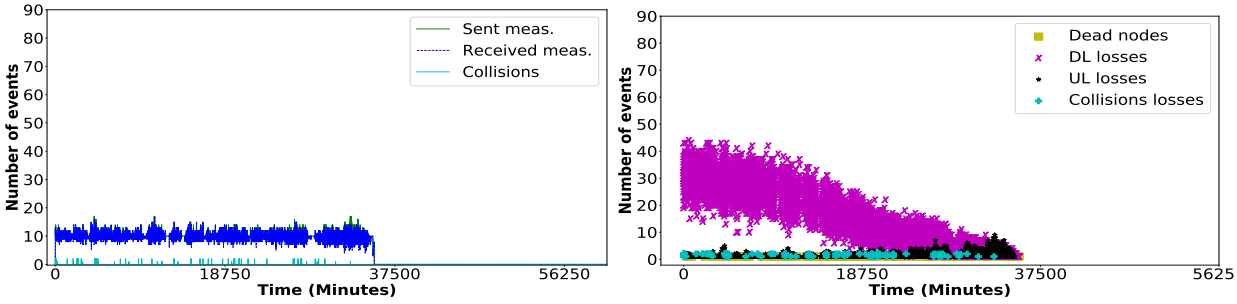
Figure 3: Spatial distribution of active nodes in the INTENSIVE scenario over 60 minutes - DiPTC config. (0.5, 0.5, 0.06).




(a)

(b)

Figure 4: Traffic evolution in the INTENSIVE scenario for DiPTC (0.5, 0.5, 0.06) and a downlink probability reception of 0.99.




(a) (b)

Figure 5: Traffic evolution in the INTENSIVE scenario for DiPTC (0.5, 0.5, 0.06) and a downlink probability reception of 0.70.

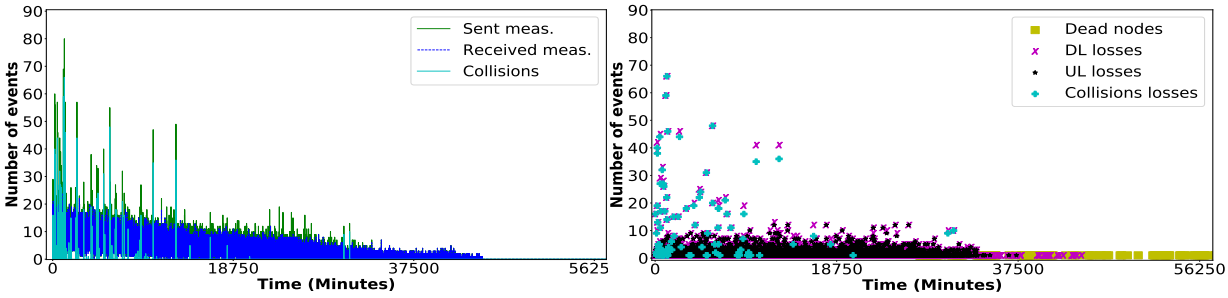
from all over the targeted area, which shows that DiPTC is able to provide a form of spatial load-balancing. That is not the case in COTraC, where one or two nodes take over data transmission during a time period $\Delta T$.

Fig. 4 illustrates the evolution of the number of collisions, sent and received packets, dead nodes, downlink, and uplink losses, in the case of the INTENSIVE scenario for DiPTC. In the latter, the results for DiPTC show multiple alternations between long transient and short stationary states. These oscillations are caused by instabilities brought by numerous variables, including frequent and consecutive uplink and downlink losses, active node deaths, and a high collision probability. The probability is higher in this scenario since we shorten the time period ($\Delta T =$ 1 min) while increasing the number of measurements required by the gateway ($K$=10).

If we look at the traffic evolution in the zoomed area (Fig. 7) we notice that the number of packets received fluctuates at around 10 packets every minute with an error of at least three packets. These oscillations are triggered by the loss of an uplink packet from node 6 at 10175 min and two other packets belonging to nodes 6 and 96 at 10182 min. They persist for about twenty minutes, especially with the loss of downlink messages and collisions. At the end of the zoom, we can notice a 5 minutes delay in the transient regime due to the downlink reliability. However, our proposal manages to converge with an acceptable level of error given the density of the network.

In order to better evaluate the effects of downlink loss on DiPTC, we lower the downlink reception probability to $P_{DL} = 0.70$. As we can see in Fig. 5, we have the same general evolution pattern as in Fig. 4 where the downlink reception probability is higher $P_{DL} = 0.99$. Even though the transient state could experience some delays, the stationary state is unaffected, showing that DiPTC is resilient to downlink loss.

DiPTC performs significantly better than LoRaWAN in the INTENSIVE scenario, as shown in Fig. 4 and Fig. 6. Due to dead nodes and downlink losses, Baseline LoRaWAN sees a decrease due to an important amount of uplink losses and collisions. The convergence of our proposal depends mainly on the amount of lost uplink messages. Furthermore, in this case, the maximum absolute error is 7,




(a) (b)

Figure 6: Traffic evolution in the INTENSIVE scenario for Baseline LoRaWAN

which means that at worst, 17 messages are received at the gateway. With network density taken into account, this error is rather small.

Fig. 8 represents the application error over time in the case of the INTENSIVE scenario, respectively for baseline LoRaWAN and DiPTC. The application error in DiPTC has lower values and frequencies than LoRaWAN. An application error of zero means that the application received exactly $k = 10$ measures. Note that the success rate of DiPTC is 19 times higher than baseline LoRAWAN. This indicates that our algorithm can fulfill the demands of the application more often than baseline LoRaWAN, even in the case of intensive traffic.

Finally, Fig. 9 shows the nodes traffic evolution in the INTENSIVE scenario for DiPTC and baseline LoRAWAN. In DiPTC (Fig. 9(a)), the traffic adaptation rate increases when the number of active node deaths increases, as expected. Unlike baseline LoRaWAN (Fig. 9(b)), in DiPTC, the average network traffic increases with the nodes death accumulation. In baseline LoRaWAN, the average network traffic is still regular after the nodes death accumulation, because of a traffic distribution generation that is fair on average.

## 6.2 Impact of the nodes scalability

In this section, we focus on the DENSE scenario, to test the horizontal scalability of DiPTC. We increased the number of nodes from 150 to 500, and decreased the traffic intensity compared to the INTENSIVE scenario.

Note in Fig. 10(a) that there is an alternation of transient and stationary states, with the former only lasting for a short time due to factors such as node death or packet loss. What this suggests is that DiPTC can quickly converge to the desired number of measurements, known as $K$, despite obstacles like fluctuating environments. As in the INTENSIVE scenario, the transient states are generally caused by issues including node death and uplink loss. Even if a node downlink is lost, our algorithm will continue to function as the node itself keeps the same traffic intensity until it hears back from the gateway. The reason we see a lot of long stationary states is because of our algorithm which provides stability and reduces collisions. There are fewer collisions in general because there is less traffic overall (since $K$=1). Low traffic has also a positive effect on the lifetime of a

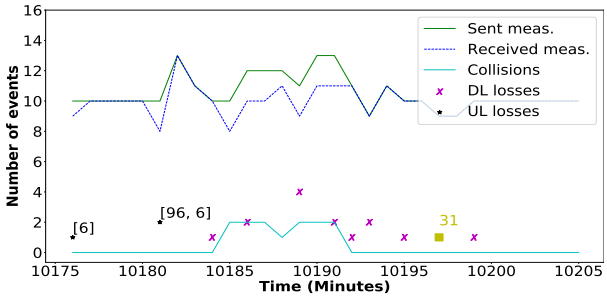



Figure 7: Zoom on the traffic evolution in the INTENSIVE scenario for DiPTC(0.5, 0.5, 0.06).

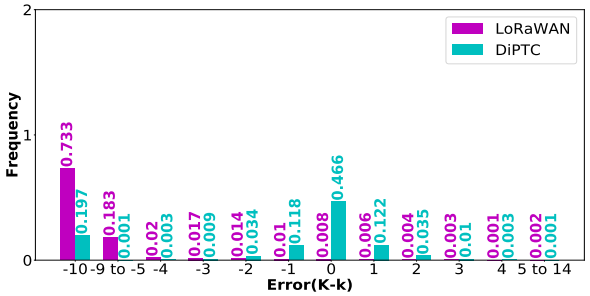
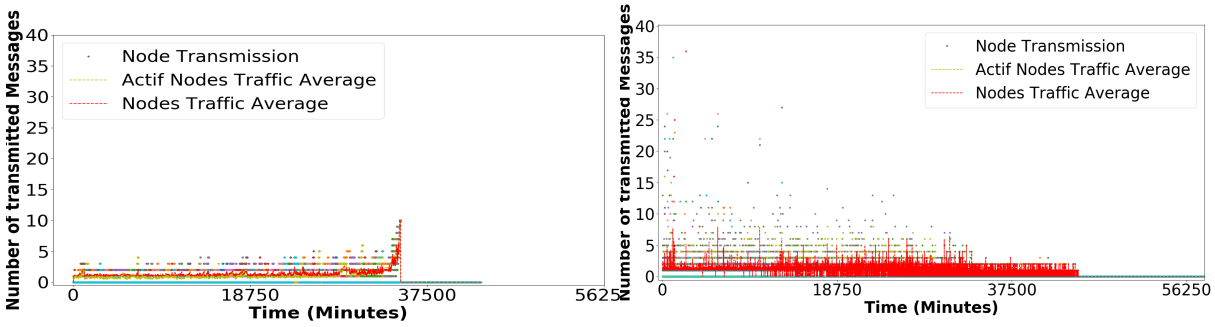


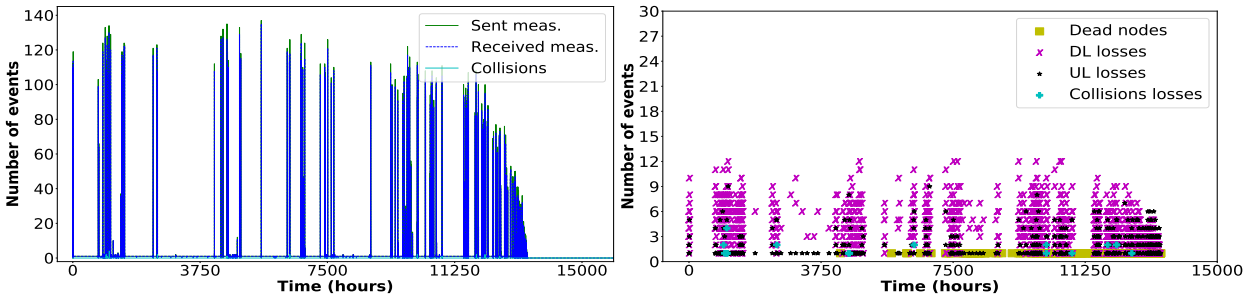

Figure 8: Frequency error in the INTENSIVE scenario for DiPTC(0.5, 0.5, 0.06) and Baseline LoRaWAN.




(a) DiPTC(0.5, 0.5, 0.06)

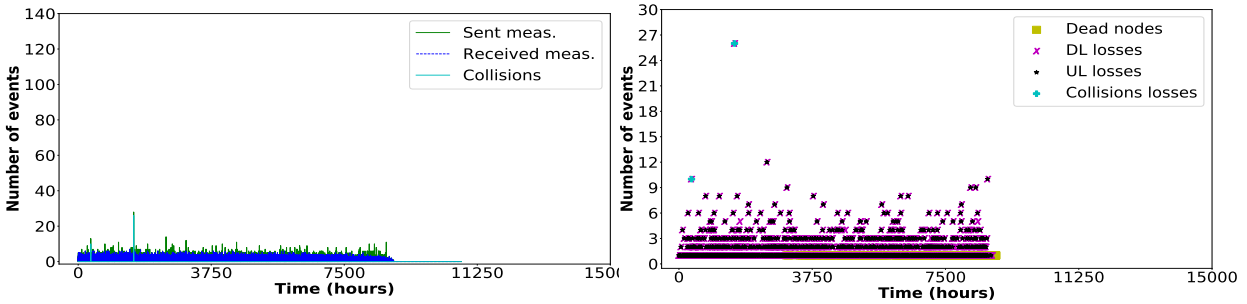
(b) Baseline LoRaWAN.

Figure 9: Nodes traffic evolution in the INTENSIVE scenario.




(a) (b)

Figure 10: Traffic evolution in the DENSE scenario for DiPTC(0.5, 0.5, 0.5).




(a) (b)

Figure 11: Traffic evolution in the DENSE scenario for Baseline LoRaWAN

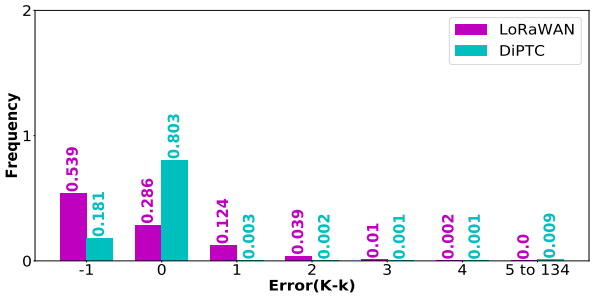
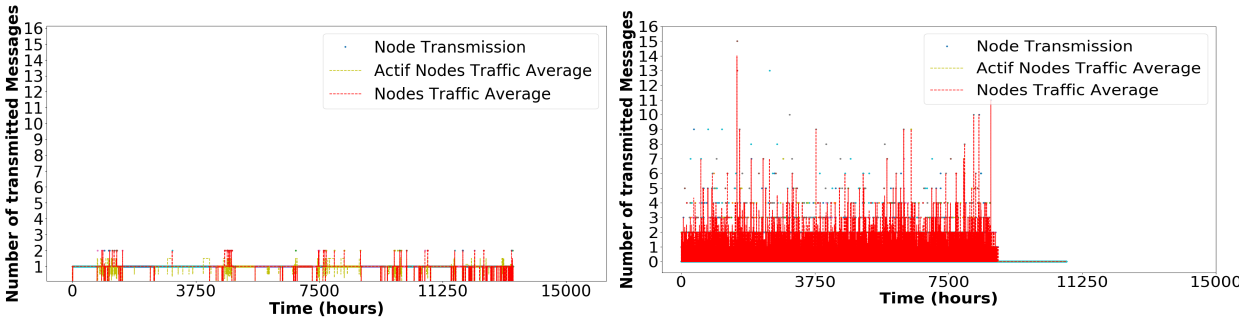


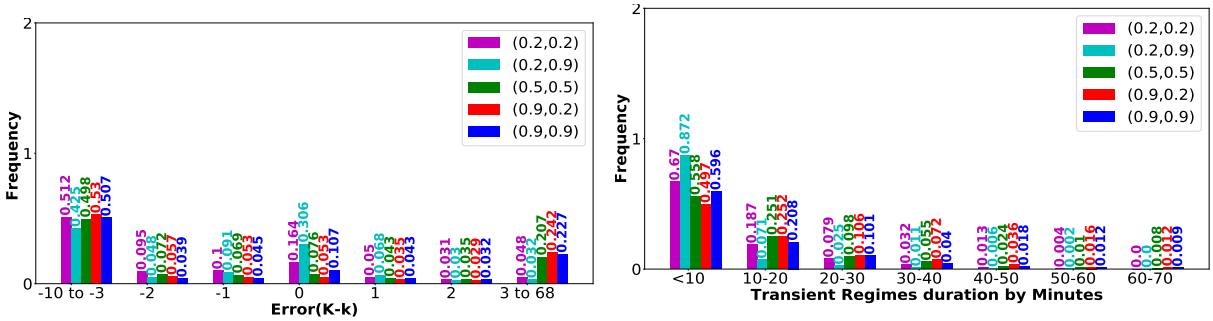

Figure 12: Error frequency in the DENSE scenario for DiPTC(0.5, 0.5, 0.5) and Baseline LoRaWAN




(a) DiPTC(0.5, 0.5, 0.5)

(b) Baseline LoRaWAN

Figure 13: Nodes traffic evolution in the DENSE scenario.




(a) Error frequencies

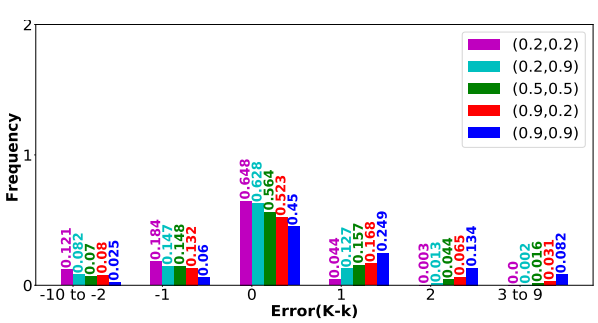
(b) Transients regimes duration frequencies

Figure 14: A comparison of error frequency and transient regimes duration frequencies, in the INTENSIVE scenario for the DiPTC when $P_{adapt}$=0.5.

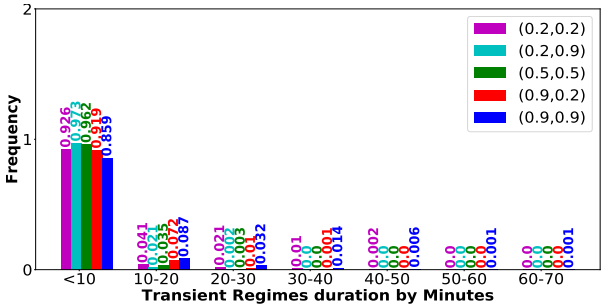
node, as it reduces the battery depletion process. Despite the significant increase in the number of nodes, our approach can quickly converge to the target number of measurements, $K$, and maintain stability for a considerable amount of time.

In comparison to DiPTC, we notice that the traffic in LoRaWAN appears to be lower (Fig. 11). In fact, the maximum number of received messages per period is only 28 ($\Delta T$=10 minutes), while in DiPTC it is 138. The baseline LoRaWAN oscillations, on the other hand, are more frequent. This is due to the fact that LoRaWAN packet generation follows a Poisson distribution of rate 0.0002, which implies that one packet is sent every 5,000 minutes on average, causing the nodes to traffic less often but more *regularly*. As illustrated in Fig. 12, DiPTC outperforms LoRaWAN when it comes to application error frequency. The null application error ($e = 0$) occurs three times more frequently with DiPTC than with LoRaWAN.

Finally, Fig. 13 shows node traffic evolution in the DENSE scenario for baseline LoRaWAN and DiPTC. It illustrates the necessity to have a frame transmission control in LoRaWAN especially when the number of nodes is important. We can




(a) Error frequencies




(b) Transient regimes duration frequencies

Figure 15: A comparison of error frequency and transient regimes duration frequencies, in the INTENSIVE scenario for the DiPTC when $P_{adapt}$=0.06.

notice that the maximum number of packets a node can send in DiPTC remains significantly small (2 messages), although the number of nodes increases in the network, contrary to LoRaWAN. In fact, compared to baseline LoRaWAN, the maximum number of packets a node may send in DiPTC is 7 times lower, and the network traffic average is generally smoother.

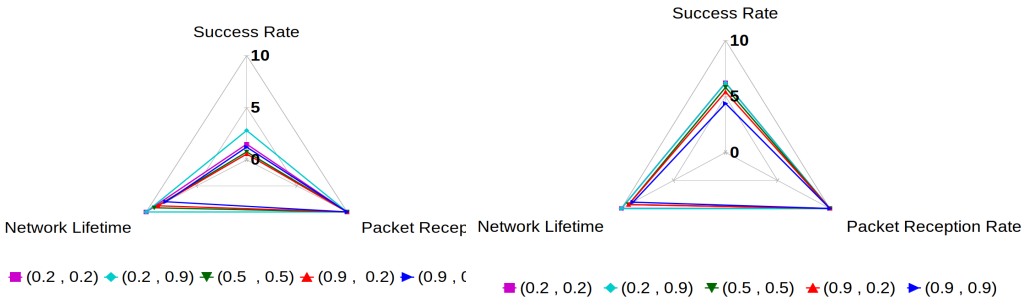
# 7 Impact of traffic control parameters

Table 4: DiPTC configurations.

| $P_{adapt}$ | $x_I$ | $x_D$ |
|---|---|---|
| 0.1 or 0.5 | 0.2 | 0.2 |
| | 0.2 | 0.9 |
| | 0.5 | 0.5 |
| | 0.9 | 0.2 |
| | 0.9 | 0.9 |

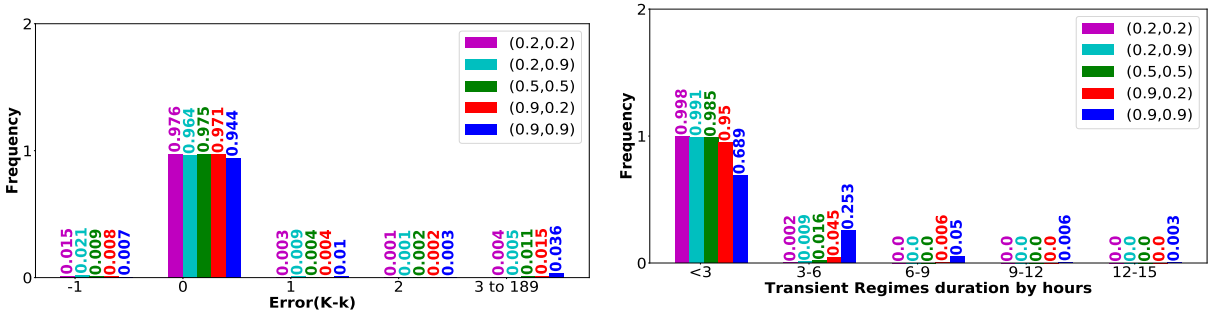
In the previous sections, DiPTC parameters ($x_I$, $x_D$, and $P_{adapt}$) were chosen and fixed according to the characteristics of the simulated network that took into account node density and traffic intensity. As a reminder, $x_I$ and $x_D$ represent how much the traffic will increase or decrease from the previous period. The larger the value of $x_I$, the faster the increase, and the greater the value of $x_D$, the slower the decrease. The last parameter, $P_{adapt}$, controls nodes adaptation traffic. This section investigates the effects of these three parameters on DiPTC performance. We investigate the different values described in Table 4.

Fig. 14 compares the error and transient regimes duration frequencies, in the INTENSIVE scenario for DiPTC, with a 0.5 adaptation probability. We notice that the five configurations have significant different results. In this scenario, the adapting probability is set to 0.5, meaning that at least 75 nodes are capable of adapting




(a) $P_{adapt} = 0.5$ (b) $P_{adapt} = 0.06$

Figure 16: A comparison of performance measures of different DiPTC configurations $(x_I, x_D, P_{adapt})$ in the INTENSIVE scenario.




(a) Error frequencies

(b) Transients regimes duration frequencies

Figure 17: A comparison of error frequency and transient regimes duration frequencies, in the DENSE scenario for the DiPTC when $P_{adapt}$=0.5.

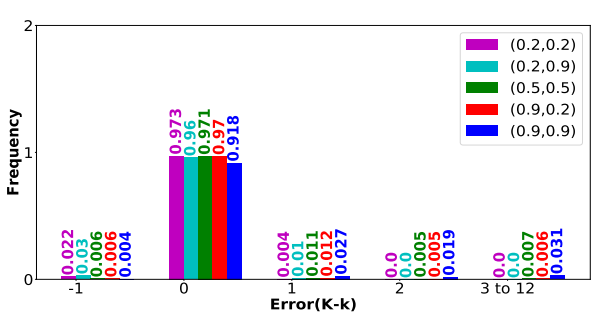
their traffic, hence competing to send the 10 required measurements. Fig. 14(a) displays that the best configurations, with regards to receiving exactly 10 messages each period of time $\Delta T = 1min$ from 150 nodes i.e., ($error = 0$), are in order (0.2, 0.9, 0.5), (0.2, 0.2, 0.5), (0.9, 0.9, 0.5) and finally (0.9, 0.2, 0.5). Even with intensive traffic, the configurations with a low increase give the best results in terms of error frequency. A weak decreasing coefficient boosts the performance. Since nodes increase and decrease their traffic slowly, it is more likely to achieve the application requirements, i.e., receiving 10 measurements in a period of time $\Delta = 1min$. Those configurations also show more frequent short regimes than the other ones 14(b).

Fig. 15 compares the error and transient regimes duration frequencies, in the INTENSIVE scenario for DiPTC, with a 0.06 adaptation probability. As we can see, decreasing the adaptation probability increases significantly not only the frequency of sending $\pm 10$ messages per time period $\Delta T = 1min$ (i.e., error 0, 1 and $-1$) but also the frequency of the short transient regimes duration. In fact, in this scenario, at least 9 nodes are competing to send the $\pm 10$ messages each $1min$. There are greater chances to meet the application requirements with 9 ($P_{adapt} = 0.1$) nodes than with 75 nodes ($P_{adapt} = 0.5$). Note that if the goal of the application is receiving exactly $\pm 10$ messages each period of time $\Delta T = 10min$, then the best configurations are in order (0.2, 0.2, 0.06), (0.2, 0.9, 0.06), (0.5, 0.5, 0.06), (0.9, 0.2, 0.06) and finally (0.9, 0.9, 0.06). With fewer competing nodes, the best configuration is (0.2, 0.2, 0.06). However, if the application allows an error of $\pm 1$, (0.2, 0.9, 0.06) is by far the best one.

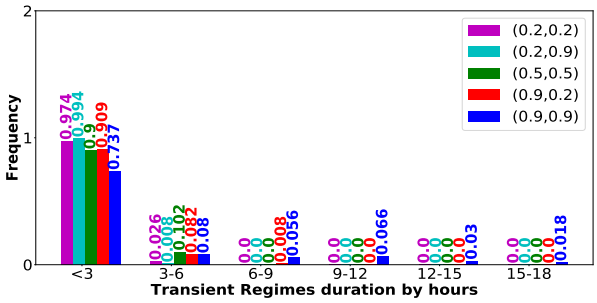
Fig. 16 compares the performance measures of the different DiPTC configurations in the INTENSIVE scenario. As we already notice in the previous figures, the configuration (0.2, 0.9, 0.06) performs better in comparison to the others (Fig. 16(a)). In this scenario (Fig. 16(b)) where at most 9 nodes compete to meet the application requirements ($\pm 10$ message(s) at each period of time $\Delta T = 1min$), configurations with a weak increasing coefficient $x_I$ traffic and consume less energy, therefore providing longer lifetime.

Fig. 17 compares the error and transient regimes duration frequencies, in the DENSE scenario for DiPTC, with a 0.5 adaptation probability. The best configurations, with regards to receiving exactly 1 message each time period $\Delta T = 10min$ from 150 nodes, (i.e., $error = 0$) are in order (0.2, 0.2, 0.5), (0.5, 0.5, 0.5), (0.9, 0.2, 0.5) and finally (0.2, 0.9, 0.5). If the application allows an error of $\pm$ 1, the configurations (0.2, 0.9, 0.5) and (0.2, 0.2, 0.5) and (0.5, 0.5, 0.5) take the lead. We also observe an error increase and the apparition of longer transient regimes. When the number of nodes increases, the traffic increases, leading to more errors and longer transient regimes.

Fig. 18 compares the error and transient regimes duration frequencies, in the DENSE scenario for DiPTC, with a 0.1 adaptation probability. As we can see, the decrease in the adaptation probability decreases the errors and increases the duration of the transient regimes. If the adaptation probability has a small value $P = 0.1$, the number of nodes allowed to adapt their traffic each period of time is smaller (50 nodes), compared to when the adaptation probability is 0.5 (250 nodes). Thus they make smaller errors, but take a longer time to adjust their sending. The best configurations are characterized by small frequencies for larger errors, important frequencies when the error is 0, and finally short transient regimes duration. In the DENSE scenario and for an adaptation probability of 0.1, the best configurations are (0.2, 0.2, 0.1), (0.5, 0.5, 0.1), (0.9, 0.2, 0.1).

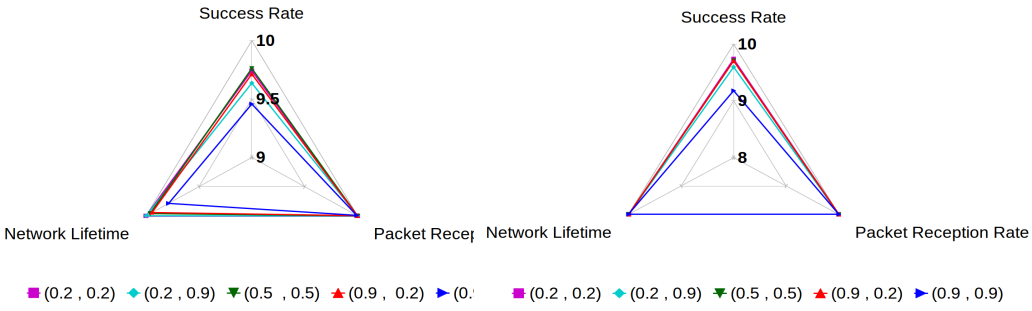



(a) Error frequencies




(b) Transient regimes duration frequencies

Figure 18: A comparison of error frequency and transient regimes duration frequencies, in the DENSE scenario for the DiPTC when $P_{adapt}$=0.1.




(a) $P_{adapt} = 0.5$ (b) $P_{adapt} = 0.1$

Figure 19: A comparison of performance measures of different DiPTC configurations $(x_I, x_D, P_{adapt})$ in the DENSE scenario.

Fig. 19 contrast the performance measures of the different DiPTC configurations in the DENSE scenario. These figures display the presence of a trade-off between the network lifetime and the success rate. Actually, in this scenario where the application objective is 1 or 2 message(s) at a period of time $\Delta T = 10min$, configurations with a low increasing coefficient $x_I$ traffic less and consume less energy, therefore they perform better, in terms of network lifetime. While the configurations with a strong increasing coefficient adapt their traffic more often to the desired messages and perform better w.r.t. the success rate measure. Notice that the packet reception rate and the average channel inactivity time are equal for the five configurations in both figures.

# 8 Discussion & Insights

The algorithm we propose achieves excellent results in terms of convergence, not just for high traffic networks, but also for those with a large number of nodes. It shows that the use of a control trafic management in LPWAN increase widely the overall performance and allow to respect strong application requirement. Our algorithm is able to converge towards the required measurements $K$ with a reasonable absolute error margin owing to the adaptation algorithm stability feature. In fact, DiPTC is capable of regulating LPWAN traffic such as LoRaWAN with low collision rates and minimal overhead.

As a result of our analysis, we can draw the following conclusions:

1. DiPTC is unaffected by downlink unreliability, hence it will continue to operate as intended even in the presence of a low downlink reception probability. Even though the transitory state could experience some delays, the stationary state is unaffected.
2. An active node battery depletion leads to its death, which in turn creates a new, short-lived transient mode.
3. The reliability of the uplink has a detrimental influence on the DiPTC convergence since it yields lengthy transient states and brief stationary states.
4. DiPTC converges exactly towards $K$ measurements per time period in the DENSE scenario. In comparison to the INTENSIVE scenario, the stationary states have a relatively long duration.
5. Despite the frequent and longer transient states in the INTENSIVE scenario, DiPTC converges towards the required measurements ($K$) with a reasonable absolute error.
6. The loss of transmission due to a collision or the propagation model has the same effect.
7. In the INTENSIVE scenario (0.2, 0.9, 0.06) is by far the best configuration in terms of success rate and network lifetime.
8. Unlike the INTENSIVE one, in the DENSE scenario, the choice of DiPTC parameters requires a compromise between the network lifetime, the error tolerated by the application, and the success rate.
9. When compared to baseline LoRaWAN, DiPTC is able to achieve the goal within reasonable deadlines, while maintaining a low number of collisions with a longer network lifetime.

10. The performance of DiPTC in the INTENSIVE scenario is lower than the performance of the optimal centralized traffic control solution (COTraC), in terms of success rate and network lifetime. Note that COTRaC works without collision due to the optimal scheduling and without additional control traffic that uses the downlink, which is not realistic.
11. Like the COTraC solution, in the DENSE scenario, our DiPTC is capable of achieving the objective of $K$ measurements per period.
12. Unlike the COTraC solution our DiPTC algorithm presents a spatial load-balancing characteristic.

# 9 Conclusion

In this work, we propose a Distributed and Probabilistic algorithm for Traffic control in LPWANs, called DiPTC. It is based on a multiplicative increase and multiplicative decrease algorithm, as well as a binary feedback message sent by a gateway. This simple yet effective way of controlling traffic yields very good results. Moreover, DiPTC ensures convergence of the application requirements (K measurements per period) and reduces collision risk within acceptable delays. DiPTC also shows a threefold increase in success rate over Baseline LoRaWAN with a significant longevity advantage. The centralized optimal algorithm COTraC outperforms our decentralized DiPTC because of the network knowledge and no feedback message is needed. However, DiPTC ensures a network spatial load balancing which is not the case for COTraC.

In our future work, we plan to improve the formalization of our model by using advanced approaches such as game theory and reinforcement learning, for success rate and network lifetime enhancement.

# Acknowledgment

This research was partially supported by CAMPUS FRANCE (PHC TOUBKAL 2019, French-Morocco bilateral program), Grant Number: 41562UA.

# References


[1] N. Sornin, E. L. M., T. Kramp, and O. Hersent, "LoRaWAN specification," *LoRa Alliance*, 2015.

[2] "Weightless aliance," https://www.weightless-alliance.org/.

[3] K. Mekki, F. Bajic, E.and Chaxel, and F. Meyer, "A comparative study of LPWAN technologies for large-scale iot deployment," *Elsevier ICT express*, 2019.

[4] N. Nurelmadina, M. K. Hasan, I. Memon, R. A. Saeed, K. A. Zainol Ariffin, E. S. Ali, R. A. Mokhtar, S. Islam, E. Hossain, and M. A. Hassan, "A systematic review on cognitive radio in low power wide area network for industrial iot applications," *MDPI Sustainability*, 2021.

[5] M. Swain, D. Zimon, R. Singh, M. F. Hashmi, M. Rashid, and S. Hakak, "Lora-lbo: an experimental analysis of lora link budget optimization in custom build iot test bed for agriculture 4.0," *MDPI Agronomy*, 2021.

[6] B. Chaudhari, M. Zennaro, and S. Borkar, "Lpwan technologies: Emerging application characteristics, requirements, and design considerations," *MDPI Future Internet*, 2020.

[7] R. M. Sandoval, A.-J. Garcia-Sanchez, J. Garcia-Haro, and T. M. Chen, "Optimal policy derivation for transmission duty-cycle constrained lpwan," *IEEE Internet of Things Journal*, 2018.

[8] A.Azari and C.Cavdar, "Self-organized low-power iot networks:a distributed learning approach," in *IEEE GLOBECOM*, Abu Dhabi, UAE, 2018.

[9] K. Ta, D. T. and al, "Lora-mab: a flexible simulator for decentralized learning resource allocation in iot networks," in *IEEE Wireless and Mobile Networking Conference*, Paris, France, 2019.

[10] K. Lasri, Y. Ben Maissa, L. Echabbi, O. Iova, and F. Valois, "A new distributed and probabilistic approach for traffic control in lpwans," in *Springer International Conference on Advanced Information Networking and Applications (AINA)*, 2021.

[11] H. Ennajari, Y. B. Maissa, and S. Mouline, "Energy efficient in-network aggregation algorithms in wireless sensor networks: a survey," in *Springer UNet*, Casablanca, Morocco, 2016.

[12] Y. Ma, Y. Guo, X. Tian, and M. Ghanem, "Distributed clustering-based aggregation algorithm for spatial correlated sensor networks," *IEEE Sensors Journal*, 2010.

[13] L. Tan and M. Wu, "Data reduction in wireless sensor networks: a hierarchical LMS prediction approach," *IEEE Sensors Journal*, 2015.

[14] C. Liu, K. Wu, and M. Tsao, "Energy efficient information collection with the ARIMA model in wireless sensor networks," in *IEEE GLOBECOM*, St. Louis, MO, USA, 2005.

[15] J.-L. Lu, F. Valois, and M. Dohler, "Optimized data aggregation in wsns using adaptive arma," in *IEEE Fourth International Conference on Sensor Technologies and Applications (SensorComm)*, Venice, Italy, 2010.

[16] J. Moraes, N. Matni, A. Riker, H. Oliveira, E. Cerqueira, C. Both, and D. Rosário, "An efficient heuristic lorawan adaptive resource allocation for iot applications," in *IEEE Symposium on Computers and Communications (ISCC)*, 2020.

[17] J. Z.Xu and al, "S-mac: achieving high scalability via adaptive scheduling in lpwan," in *IEEE INFOCOM*, Virtual Conference, 2020.

[18] L.-H. Shen, C.-H. Wu, W.-C. Su, and K.-T. Feng, "Analysis and implementation for traffic-aware channel assignment and contention scheme in lora-based iot networks," *IEEE Internet of Things Journal*, 2021.

[19] N. Chinchilla-Romero, J. Navarro-Ortiz, P. Muñoz, and P. Ameigeiras, "Collision avoidance resource allocation for lorawan," *IEEE Sensors*, 2021.

[20] Z. Qin and J. A. McCann, "Resource efficiency in low-power wide-area networks for iot applications," in *IEEE Global Communications Conference*, 2017.

[21] K. Q. Abdelfadeel, D. Zorbas, V. Cionca, and D. Pesch, "$free$ —fine-grained scheduling for reliable and energy-efficient data collection in lorawan," *IEEE Internet of Things Journal*, 2020.

[22] J. Haxhibeqiri, I. Moerman, and J. Hoebeke, "Low overhead scheduling of lora transmissions for improved scalability," *IEEE Internet of Things Journal*, 2019.

[23] D. Zorbas, K. Abdelfadeel, P. Kotzanikolaou, and D. Pesch, "Ts-lora: time-slotted lorawan for the industrial internet of things," *Elsevier Computer Communications*, 2020.

[24] A. Kaburaki, K. Adachi, O. Takyu, M. Ohta, and T. Fujii, "Autonomous decentralized traffic control using q-learning in lpwan," *IEEE Access*, 2021.

[25] U. o. S. C. Information Sciences Institute, "RFC 793: transmission control protocol, darpa internet program protocol specification," IETF, Tech. Rep., 1981.

[26] J. Iyengar and M. Thomson, "RFC 9000: quic: a udp-based multiplexed and secure transport," IETF, Tech. Rep., 2021.

[27] D. Chiu and R. Jain, "Analysis of the increase/decrease algorithms for congestion avoidance in computer networks," *June Computer Networks and ISDN Systems*, 1989.

[28] A. Boubrima, W. Bechkit, and H. Rivano, "On the deployment of wireless sensor networks for air quality mapping: optimization models and algorithms," *IEEE/ACM Transactions on Networking*, 2019.

[29] M. C. Bor, U. Roedig, T. Voigt, and J. M. Alonso, "Do LoRa low-power wide-area networks scale?" in *IEEE MSWiM*, Malta, 2016.

[30] M. Slabicki, G. Premsankar, and M. Di Francesco, "Adaptive configuration of LoRa networks for dense iot deployments," in *IEEE/IFIP NOMS*, Taipei, Taiwan, 2018.

[31] A. J. Wixted, P. Kinnaird, H. Larijani, A. Tait, A. Ahmadinia, and N. Strachan, "Evaluation of lora and lorawan for wireless sensor networks," in *IEEE SENSORS*, 2016.

[32] SX1272/73, "Semtech datasheet - 860 MHz to 1020 MHz low power long range transceiver, rev. 4," 2019.