Similarity detection for smart and transparent long-range IoT relaying

Congduc Pham
University of Pau, France
congduc.pham@univ-pau.fr

Abdallah Makhoul
Univ. Bourgogne Franche-Comté, FEMTO-ST, France
abdallah.makhoul@univ-fcomte.fr

Mamour Diop
University of Gaston Berger, Senegal
serigne-mamour.diop@ugb.edu.sn

***Abstract*—LPWAN refers to highly energy-efficient wireless communication over very long distances. Nonetheless, even with the increased range, 1-hop connectivity can be difficult to achieve in real-world deployment scenario. Especially for remote and rural areas where density of gateways is low and where devices/gateway are usually deployed for a specific application. Therefore, a smart and transparent 2-hop approach has been proposed in a previous work to leverage these connectivity issues. This article extends this approach with similarity detection features in order to (i) reduce the power consumption when waking-up to relay packets and (ii) reduce the radio activity time when running under duty-cycle regulated constraints.**

***Index Terms*—LoRa, Low-power IoT, Similarity detection**

I. Introduction

Low-Power Wide Area Networks (LPWAN) refers to highly energy-efficient wireless communication over very long distances. These technologies are particularly adapted to battery-operated small Internet-of-Things (IoT) devices. The most popular LPWAN technologies (e.g. SigFox, LoRa) can achieve more than 20km in line-of-sight (LOS) conditions.

In a previous work, a 2-hop LoRa approach has been proposed to extend the LoRa network coverage [1] as feedbacks from pilot deployments highlighted the fact that even with the longer range offered by LoRa, many of these deployment campaigns suffer from connectivity issues with the gateway as clear LOS communication is difficult. For instance there can be hard constraints on gateway and gateway's antenna placement with possible installation often limited to the farm office where power supply and wired Internet are available. Due to field/terrain configuration, some devices can also be isolated from the other devices and direct transmission to the gateway is impossible to achieve.

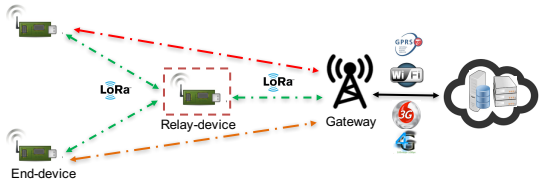

Fig. 1: Long-range 2-hop connectivity architecture

In addition, there can also be severe limitations on the maximum transmission power allowed in some countries that reduce the transmission range. The 2-hop LoRa approach was to design a smart, transparent and battery-operated intermediate node – relay – that can be added after a deployment campaign to seamlessly provide an extra hop between the remote devices and the gateway as illustrated in Figure 1. The red link indicates no direct connectivity and the orange link indicates unstable connectivity. When a stable connection can be realized with the gateway, the link shows in green.

The smart and transparent relay will wake-up at appropriate moments to catch uplink transmissions from specific devices in order to perform the relay operation. However, by nature, a relay device has higher energy consumption than end-devices as they are transmitting more oftenly. In addition, in some countries, a transmitter can be constrained by duty-cycle limitations. In Europe, following the ETSI EN300-220-1 recommendations [2], a transmitter is limited to 1% duty-cycle (i.e. 36s/hour) in the general case, even if it can change to another frequency channel. A relay forwarding uplink packets from n end-devices can have to transmit n packets to the gateway. Assuming each transmission takes about 1.5s (approximately the time-on-air of a 20-byte payload packet – header included – when using the LoRa settings for achieving the longest range) then a relay can rapidly be limited by the duty-cycle if end-devices send packets at a higher rate than 1 packet/hour. To reduce energy consumption and radio activity a relay can very simply decide to not forward a new measure from a device when this measure is close to the last measure received from that device. However, when simply doing so, the relay still need to wake-up to receive the new measures. In addition, as there is no possibility to detect possible redundancy between different devices the mitigation of radio activity remains very limited.

In this article, we extent the smart relay with automatic embedded similarity detection features in order to (i) reduce the power consumption when it has to wake-up to forward packets from child devices and (ii) reduce the radio activity time when running under duty-cycle regulated constraints. We propose similarity detection features based on distance function to evaluate the similarity between datasets sent by child devices. The main constraint is to propose a low-complexity and small memory footprint approach.

The rest of the article is organized as follows. Section II reviews related works in redundancy/similarity detection. Section III presents the distance-based similarity detection mechanism. Section IV describes our relay with similarity

detection features, explaining the proposed approach for low-complexity similarity detection. Implementation and experimental results are presented in Section V and Section VI concludes.

II. Related works on similarity detection

In the literature we can find several approaches that aim to detect and reduce data redundancy [3], [4], [5], [6]. The main idea is to propose a quantitative approach in identifying two similar data measures by studying similarity functions. Such function measures the degree of similarity between two measures and returns a value between 0 and 1. A higher similarity score indicates that the data measures are more similar, thus we can consider these measures as redundant.

Similarity functions were used in various domains and applications in order to identify near duplicate objects (data). For instance, for Web search engines [7], Web mining applications [8], detecting plagiarism [9], collaborative filtering in data mining [10], *etc*. To the best of our knowledge, we are the first to use these functions for data aggregation in sensor networks. Recently, many studies have proposed new algorithms that define the similarity between objects or records. These algorithms are classified into three categories, inverted index based methods [11], prefix filtering methods [12], and signature based methods [13]. Most of these methods are quite complex for wireless sensor networks that usually generate large amount of candidate pairs, all of those need to be verified by the similarity function.

In [14], the authors presented a work for data collection in industrial sensor networks. They proposed three different methods based respectively on, sets similarity functions, distance functions and the k-means clustering algorithms. Their results show that their method reduces collected and transmitted data while preserving data integrity.

The authors in [4] propose a local similarity imputation method for eliminating missing data. The proposed technique is based on a fast clustering algorithm and top k-nearest neighbors. The top k-nearest neighbor hybrid distance weighted imputation is proposed to fill in missing values in clusters. The obtained results show that the proposed method can impute the missing data values effectively.

A flexible overlay protocol for duplicate sensitive aggregation functions (FOMA) is proposed in [15]. It aggregates partial results in two layers (routing and aggregation) with no computation error in a highly energy-efficient manner. It uses four phases to collect data from sources: creating a routing tree structure, finding proper data aggregation nodes, creating signatures, and data collection. The obtained results show that the proposed approach outperforms other existing ones (e.g. TAG) in terms of energy consumption and data accuracy.

Researchers in [16] presented a data aggregation technique based on the Jaccard sets similarity function and several optimizations in [17]. The aim is to study the similarity between several datasets generated by neighboring nodes at the cluster-head level. They provided a new prefix filtering method and results show that the approach reduces data size by eliminating in-network redundancy and sending only necessary information to the sink.

In [18] the authors propose two algorithms PARTENUM and WTENUM for evaluating set-similarity joins. Their approach handles a large subclass of set-similarity predicates allowed by the definition of the set-similarity join operator. Their algorithms guarantee that two highly dissimilar sets will not appear as a candidate pair with a high probability. The authors demonstrate the efficiency of their approach through experimental evaluation on real and synthetic data sets.

Our main objective in this paper is to study and use similarity functions for the simple and resource-constrained 2-hop relay node in LoRa IoT networks where the topology is usually a simple star centered on a single gateway.

III. Similarity detection proposal

In our approach, we consider that the relay collects measures from the devices and identifies pairs of measures whose similarities are above a given threshold. A similarity function uses a threshold of value between 0 and 1. A higher similarity score indicates that the measures are more similar, thus we can consider that the devices are generating redundant data. If the compared data are found to be similar to each other, the relay does not need to transmit all data and can also avoid waking-up for those redundant devices.

A. Identification of redundant nodes

A relay node receives several data measures coming from different child nodes. Our idea is to use data similarity and distance functions in order to identify neighboring nodes generating similar data and to not wake up for those redundant nodes. To find the similarity between data measures, several similarity functions can be used for data comparison, such as edit distance, Euclidean distance, cosine distance, Jaccard similarity, Bray-Curtis distance, Camberra, etc. In our approach we propose to use the Euclidean distance which is widely used in various domains.

In mathematics, the Euclidean distance is the straight line distance between two vectors of data. Let us first consider two data sets R_i and R_j generated by the two child devices S_i and S_j respectively in the same period p. Then, in order to compute the Euclidean distance between R_i and R_j. We consider for instance $R_i = \left[r_{i_1}, r_{i_2}, r_{i_3}, r_{i_4}, \ldots, r_{i_\mathcal{T}}\right]$ and $R_j = \left[r_{j_1}, r_{j_2}, r_{j_3}, r_{j_4}, \ldots, r_{j_\mathcal{T}}\right]$ with $|R_i| = |R_j| = \mathcal{T}$. Finally, we can calculate the Euclidean distance between the two vectors R_i and R_j based on the following equation:

$$E_d(R_i, R_j) = \sqrt{\sum_{k=1}^{\mathcal{T}} (r_{i_k} - r_{j_k})^2} \tag{1}$$

where $r_{i_k} \in R_i$ and $r_{j_k} \in R_j$.

B. Distance Normalization

The normalization of data is an essential process when using the distance functions. The objective of the normalization process is to scale all vectors of data to have the same variation

then, to perform an exact comparison among these vectors. In this paper, we use Gaussian normalization to normalize data generated by the sensors. First, we calculate the Euclidean distance for each pair of data vectors in the network:

$$\mathbb{E}_d = \{E_d(R_1, R_2), E_d(R_1, R_3), \ldots, E_d(R_{N-1}, R_N)\}$$

where N is the total number of sensors. Then, we can apply the Gaussian normalization using the following formula when $E_d(R_i, R_j) \neq 0$:

$$E'_d(R_i, R_j) = \frac{E_d(R_i, R_j) - \overline{Y}}{6 \times \sigma} + \frac{1}{2} \qquad (2)$$

where $\overline{Y}$ is the mean of all distances and σ is the standard deviation of pairwise distance over all data. $\overline{Y}$ and σ are calculated as follows:

$$\overline{Y} = \frac{\sum_{k=1}^{|d|} d_k}{|d|} \quad \text{and} \quad \sigma = \sqrt{\frac{\sum_{k=1}^{|d|}(d_i - \overline{Y})^2}{|d|}},$$

$$\text{where } |d| = \frac{n \times (n-1)}{2}$$

Thus, R_i and R_j are said to be redundant if $E'_d(R_i, R_j) \leq \epsilon$, where $\epsilon \; in[0, 1]$ is a user defined threshold based on the application requirement.

IV. 2-HOP RELAYING WITH SIMILARITY DETECTION

A. Review of the smart relay node

Our 2-hop LoRa approach [1] adds an extra hop between some end-devices and the gateway. The addition of the relay is performed after the deployment of both end-devices and the gateway. Logically, the relay node does not take part in any data sensing tasks and one of the major considerations should be its appropriate location to cover areas where connectivity is either lost or unstable after the network deployment. We designed the relay-device with the following requirements:

- *Low-cost, low power*: the objective is to avoid additional hardware such as Real-Time Clock (RTC) to make the relay only different from end-devices with the uploaded software. Therefore, an end-device can be "recycled" and reprogrammed to act as a relay. As it is also desirable that *relay*s run on battery, their longevity must be similar to the longevity of end-devices.
- *Smart*: relays are designed to remain in low-power mode most of the time, only waking-up at appropriate moments to catch uplink transmissions from end-devices. Therefore, a relay-device must be able to switch from deep sleep to active mode by smartly analyzing the uplink pattern from end-devices during an observation phase activated at relay's startup.
- *Transparent*: relays are transparent to the rest of the network: (a) no change in hardware or software for end-devices or gateway to support the new 2-hop approach; (b) no additional signaling traffic between relays and end-devices or gateway. Therefore, end-devices are not be aware of the 2-hop relay mode, and do not have to perform any discovery and binding process to a nearby relay. The relay also does not need to exchange parameters with the gateway for advertising its presence. On the gateway side, no scheduling mechanism for end-devices and relay is required. Furthermore, withdrawal or failure of a relay leaves the network as functional as before its integration in the network.

We can summarize the behavior of the relay node as follows for the purpose of the rest of the paper: after an initial observation phase where uplink transmission time patterns from end-devices are collected, the relay will autonomously and periodically wake up from deep sleep mode to catch uplink packets and perform the relaying task.

B. Extending with similarity detection

The relay node basic behavior described in [1] and summarized in previous sub-section is extended with the distance-based similarity detection features. Messages from end-devices are supposed to use the following format: "SH/44" which means for instance that the soil humidity is 44. An example with 3 sensor nodes, S1, S2 and S3, is shown below.

```
S1:       S2:       S3:
SH/44     SH/45     SH/46
SH/44     SH/43     SH/44
SH/46     SH/43     SH/43
SH/46     SH/43     SH/43
SH/46     SH/44     SH/44
SH/45     SH/44     SH/43
...       ...       ...
```

After m sensing periods (equivalent to a number of messages from devices), e.g. $m = 3$, we have $R_1 = [44, 44, 46]$, $R_2 = [45, 43, 43]$ and $R_3 = [46, 44, 43]$. Then, the relay computes $\mathbb{E}_d = \{E_d(R_1, R_2), E_d(R_1, R_3), E_d(R_2, R_3)\}$ with Eq. 1 and computes $E'_d(R_1, R_2)$, $E'_d(R_1, R_3)$ and $E'_d(R_2, R_3)$ with Eq. 2 to determine whether (a) sensor 1 is redundant with sensor 2, (b) sensor 1 is redundant with sensor 3 and (c) sensor 2 is redundant with sensor 3.

Figure 2 illustrates the similarity detection mechanism with respect to the device's sensing period and the relay's observation phase. Here we illustrate with 8 devices with a sensing period of 10 minutes, $m = 3$ and a relay's observation phase of twice the device's sensing period. We also show the 1-hour cycle that can be of importance when dealing with radio duty-cycling regulations.

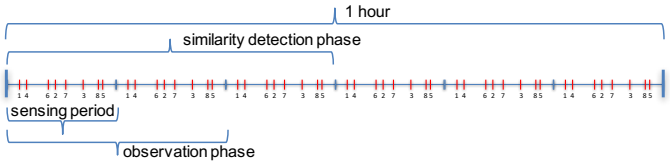

Fig. 2: Overall timing schematic

As indicated previously, the relay can also simply choose to not forward the last measure from a device as shown in Figure 3 for $r_{1_2} = 44$ and $r_{2_3} = 43$. However, even though these measures are not forwarded to the gateway, they are taken into account for the similarity detection mechanism.

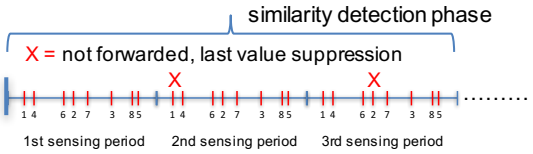

Fig. 3: Single device last measure suppression

C. Similarity detection in action

After the initial similarity detection phase (e.g. with $m = 3$), some redundancy patterns can already be identified and the relay can decide to not forward some messages. With the previous datasets example, the relay computes $\mathbb{E}_d = \{3.31, 3.60, 1.41\}$ with Eq. 1 and computes $E'_d(R_1, R_2) = 0.59$, $E'_d(R_1, R_3) = 0.64$ and $E'_d(R_2, R_3) = 0.26$ with Eq. 2 where $\overline{Y} = 2.77$, $\sigma = 0.97$ and $|d| = 3$. If the application defines $\epsilon = 0.3$ then the relay can decide that devices S2 and S3 are redundant each other. Figure 4 shows for the next sensing periods the decision of the relay node.

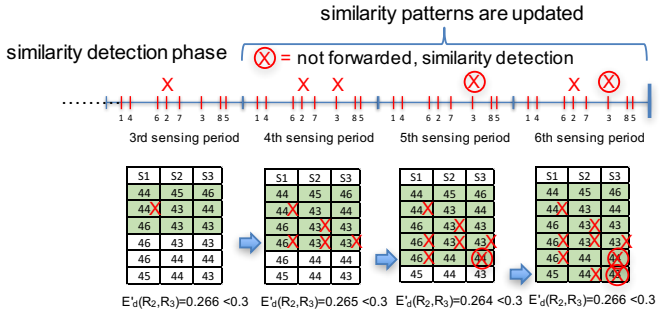

Fig. 4: Similarity detection: avoiding message forwarding

In the 4th sensing period, both messages from S2 and S3 are not forwarded by the relay because they are similar to the last received values. This is indicated by the "X" symbol on the time line. Note that measures from S2 and S3 are still used to update future similarity detection computations. In the 5th sensing period, the relay forwards the message from S2 while the message from S3 is not forwarded because S3 is detected to be redundant with S2. This is indicated by the circled "X" on the time line. In the 6th sensing period the message from S2 is not forwarded as it is similar to the last received value from S2. The message from S3 is also not forwarded, but thanks to the similarity detection mechanism. These informations are also illustrated by the tables below the time line in Figure 4. Each cell at row k and column i store the measure r_{i_k} which is the measurement from device i in sensing period k.

D. Saving more energy by not waking-up

1) Principle: The proposed similarity detection mechanism can allow for more energy saving by avoiding the relay to wake up for redundant devices. In Figure 5, we indicate similarity computations between 2 devices i and j to obtain $E'_d(R_i, R_j)$ by a double arrow in the measurement table, between the 2 corresponding columns, except for devices 1 and 3 that are indicated by the double arrows on the right side of the S3 column. We can see that after determining that S2 and S3 are redundant, at the end of the 3rd period, the relay chose to not wake up for the next message from S2 in the 4th sensing period. This is indicated by the red cell in the measurement table and by the "z" symbol on the time line. The relay wakes up for the next message from S3 but as this message is the same than the previous one, it is not forwarded by the relay. However, this measure will be used to update $E'_d(R_1, R_3)$ which is the only similarity computation that can be performed because the measure for S2 is not available. Compared to the previous case, $E'_d(R_2, R_3) = 0.286$ because $\overline{Y}$ and σ are different since both $E'_d(R_1, R_2)$ and $E'_d(R_2, R_3)$ cannot be computed at this step.

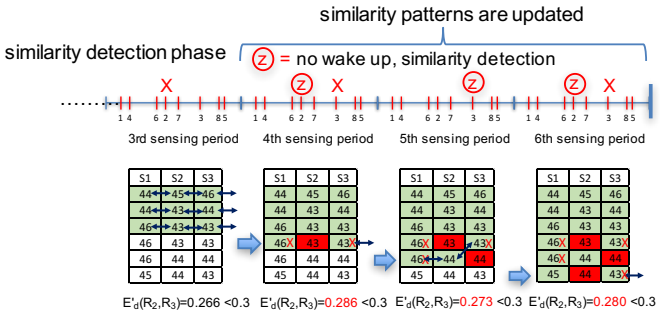

Fig. 5: Similarity detection: avoiding wake-up

In the 5th sensing period, the relay will now not wake up for S3 but will wake up for S2, therefore alternating between devices in the redundant set. We can see that $E'_d(R_1, R_2)$, and especially $E'_d(R_2, R_3)$, can now be updated by adding the measures indicated by the double arrows. Continuing with the 6th sensing period, it is similar to the 4th sensing period where the relay skips to wake up for S2 but wakes up for S3. Again, here, the measure from S3 is not forwarded because it is similar to the last received measure.

Compared to the previous case where 5 message transmissions were avoided during the 4th→6th sensing periods, here the relay additionally did not wake up 3 times, saving more energy.

2) Similarity computation with missing measurements: As the relay may not wake up to receive measurements from some devices that are detected to be redundant, the similarity detection procedure must be able to cope with missing measures. In the example described in Figure 5, that happens in the 5th sensing period. For instance $E_d(R_1, R_2)$ was computed with $R_1 = [r_{1_1}, r_{1_2}, r_{1_3}, r_{1_5}]$ and $R_2 = [r_{2_1}, r_{2_2}, r_{2_3}, r_{2_5}]$ because r_{2_4} was missing. $E_d(R_2, R_3)$ was computed with $R_2 = [r_{2_1}, r_{2_2}, r_{3_3}, r_{2_5}]$ and $R_3 = [r_{3_1}, r_{3_2}, r_{3_3}, r_{3_4}]$ because both r_{2_4} and r_{3_5} are missing. The objective is to only take real measures to compute the similarity patterns. Although the mechanism described previously and illustrated in Figure 5 is still valid, we want to propose a low-complexity approach and simple implementation compared to for instance the approach proposed in [4].

First, the relay keeps a measurement table with M rows

and N columns to store the r_{i_k}, i refers to the device which will be translated into a column index and k refers to the sensing period which will be translated into a row index. The table rows can be logically indexed from 0 to $M-1$. The relay starts at sensing period 1 as shown in Figure 3 and the similarity detection computations can begin when m messages from devices have been received. Each row l of the table contains measures from devices for a given sensing period k and we have $l = k \mod M$. At the end of sensing period k, the relay performs the similarity detection computations by taking into account sensing periods $[k-M+1, k]$, if $k > M$, and sensing periods $[1, k]$ if $k \leq M$.

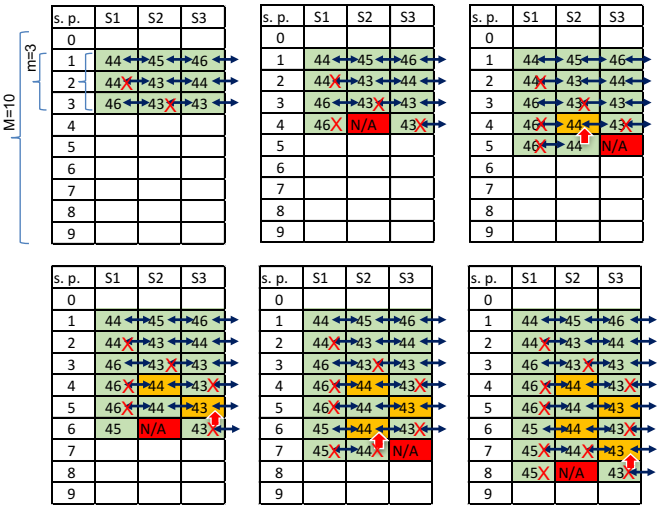

Fig. 6: Similarity detection with missing measures

If the relay decides in sensing period k to not wake up for a device i, it will indicate so in the table with a special value, e.g. $r_{i_l} = N/A$, with $l = k \mod M$. This is illustrated in Figure 6 for $k = 4$ and $k = 5$. Obviously, $E'_d(R_i, R_j)$ up to k can only be computed if both r_{i_k} and r_{j_k} are different from N/A. In Figure 6, the $E'_d(R_i, R_j)$ that can be computed are again shown with the double arrows.

We propose the following simple algorithm. In sensing period k, when receiving r_{i_k}, the relay determines $l = k \mod M$ and replaces in the table all r_{i_j}, $j < l$, that were set to N/A by r_{i_k} thus replacing the missing measures in previous steps by the newly received measure at step k. This is illustrated in Figure 6 when $k = 5$ and where r_{2_4} takes the value of r_{2_5} for device S2. The corresponding cell is shown in orange color. By doing so, $E_d(R_1, R_3)$ and $E_d(R_2, R_3)$ can be updated up to $k = 4$ and up to $k = 5$ for $E_d(R_1, R_2)$. When $k = 6$, the relay does not wake up for S2 again (alternating between S2 and S3 that have been detected as redundant) and receives a real measure from S3. Now, the relay sets $r_{3_5} = r_{3_6}$ for device S3. The cases when $k = 7$ and $k = 8$ are similar to the case when $k = 5$ and $k = 6$ respectively.

E. Similarity detection to mitigate duty-cycling

Similarity detection in the context of LPWAN can provide some sort of duty-cycling mitigation technique when radio duty-cycle is limited to a very small ratio as explained in Section I (e.g. 36s/hour). As the relay knows the number of child devices with their respective transmission interval, it can determine the total radio time needed for relaying all the uplink messages from child devices during a 1 hour period (assuming for instance the ETSI regulation explained previously). If this total radio time is greater than the allowed radio time, e.g. 36s, then the relay knows how many uplink messages it has to ignore. During the observation phase, a First Come First Serve policy can be used to relay uplink messages and then when similarity scores are available, the decision of relaying or not can be optimized based on number of similarity pairs and number of messages that should be ignored. These issues will be addressed in future works.

V. Experimentations

A. Implementation

The relay node described in [1] is based on an Arduino ProMini running at 3.3v and 8MHz. The similarity detection mechanism is added as follows:

- an additional measurement table stores the last measures from end-devices. The size of this table is rather small, typically to store the last 10 measures from end-devices.
- each measure received by the relay will be added in the table according to the device's address and the current sensing period k, as illustrated previously in Figure 4.
- a similarity table of size $|d| = \frac{n \times (n-1)}{2}$ will store the similarity scores computed following Eq.2, n being the number of connected end-device. Similarity scores are computed at the end of each sensing period and, based on the value of the ϵ threshold, similarity between 2 devices will be marked true (T) or false (F) as illustrated in figure below for $n = 5$.

	2	3	4	5	3	4	5	4	5	5
	F	F	T	F	T	F	T	F	T	F
	1				2			3		4
index	1	2	3	4	5	6	7	8	9	10

The grey cells represent the first device in the similarity pair. The values on the first line represent the second device in the similarity pair. For instance, the similarity score $E'_d(R_2, R_3)$ between device 2 and 3 will be stored at index 5, and so on. Here, device pairs (1,4), (2,3), (2,5) and (3,5) are similar. Note that we have $E'_d(R_2, R_3) = E'_d(R_3, R_2)$ therefore the table only considers $E'_d(R_i, R_j)$ where $j > i$.

- a schedule table will be used to store the wake-up schedule for the next periods. As explained in Section IV.D, the relay node will alternate wake-up between devices in a redundant set. Therefore it is necessary to identify for which node the relay will wake up at a given sensing period.

We explain below how the schedule table is filled once similarity scores are computed and the similarity table filled.

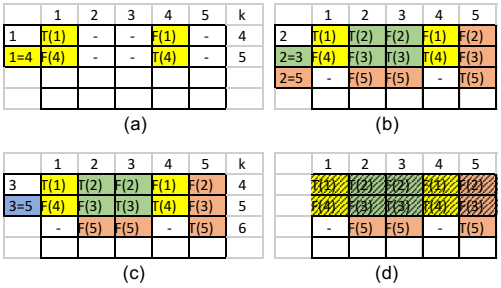

Fig. 7: Schedule table: starting a new schedule

Figure 7 illustrates 4 steps from (a) to (d) to start a new schedule when $m = 3$ messages have been received from end-devices (therefore at the end of sensing period $k = 3$). The starting configuration is similar to the one illustrated previously in Figure 5.

The relay node uses the similarity table to sequentially determine all the similar pairs. Starting with device 1, see Figure 7(a), the relay determines that device 1 and 4 are similar and then indicates for next period $k = 4$ that it will wake up for device 1 (marked with T as true) and that because of device 1, noted $T(1)$. For the same sensing period, the relay will indicate for device 4 that it will not wake up (marked with F as false) and that because of device 1, noted $F(1)$. The relay will then indicate for sensing period $k = 5$ the opposite wake up pattern: i.e. $F(4)$ for device 1 and $T(4)$ for device 4. Since device 1 is only similar to device 4, only the next 2 sensing periods are scheduled. Going to device 2, the relay first finds that device 2 is similar to device 3 and then to device 5. In Figure 7(b), we can see in green the cells that are filled when processing the (2,3) pair and in beige the cells that are filled when processing the (2,5) pair. The information on the first device's id in the pair, e.g. device 2, is kept in each cell to be able to link together cells that are involved in the same similarity set, e.g. $\{2, 3, 5\}$. Going to device 3, see Figure 7(c), the relay finds that 3 is similar to 5. However, when trying to fill in the corresponding cells the relay also finds that these cells have already been filled. Therefore no new schedule is created for device 3 as shown in Figure 7(c).

After steps (a)..(c), the relay will perform the regular wake-up from deep sleep based on timestamp data gathered during the observation phase. However, with the addition of the similarity detection mechanism, the relay will wake up for a device i at sensing period k only if the schedule table indicates T for device i at sensing period k. If the schedule table indicates F then the relay node will indicate N/A in the measurement table as explained in Section IV.D.2. In the example illustrated in Figure 7, the relay will wake up for device 1 and 2 at sensing period $k = 4$, then for device 3 and 4 at sensing period $k = 5$. At the end of sensing period $k = 5$ some cells in the schedule table for sensing period $k = 6$ are not defined, see Figure 7(d). The relay will therefore use the similarity table to fill in the empty cells but only for partially filled sensing periods, therefore stopping at $k = 6$ as illustrated in Figure 8(a). Figure 8(b) shows the end of a schedule cycle where the relay has finished processing all defined schedules and where it will re-start creating new schedules for another schedule cycle as depicted previously in Figure 7(a).

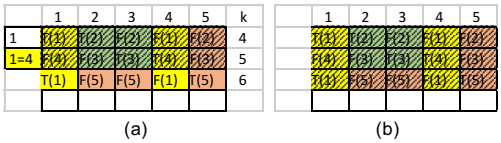

Fig. 8: Schedule table: continuing a schedule

B. Memory footprint

We detail here the main memory usage of the similarity detection mechanism. To handle 10 child devices and keeping the last 10 measures, the measurement table is $10 * 10 * (4 + 1 + 1) = 600B$ where 4 is the size of a double variable type and there are 2 bytes for the device address and the measure's status (whether is it a real measure, a copied one or N/A). If limited in memory, an unsigned int coded on 2 bytes could reduce the table size by 200B. The number of historical measures can also be reduced. Then the similarity table is $(10*(10-1))/2 = 45B$. For the schedule table, when handling 10 devices, the maximum number of similarity pairs is 9. Therefore the maximum number of sensing periods to schedule ahead in a cycle is also 9. So the schedule table is $10 * 9 * 2 = 180B$ where the 2 bytes/cell are for a wake-up status (T, F or empty) and the id of the first device in the similarity pair. In total the memory footprint of the similarity detection mechanism is $600+45+180 = 825B$ where a large part is for the storing past measures. Our relay node based on the Arduino ProMini (ATMega328P) has a total of 2kB of memory which is used at 88% with the similarity features.

C. Deployment tests

We performed field tests to assess the performance of the proposed 2-hop approach with similarity features. Fig. 9 shows the scenario that has been deployed in one of the WAZIUP pilot site in Gaston Berger University's experimental farm, Saint-Louis, Senegal. We deployed a network consisting of 8 soil humidity end-devices (S_1 to S_8), a relay and one gateway (GW). GW was placed in the farm office. End-devices have same transmission intervals set to 10mins. LoRa parameters of the experiments were chosen as follows: spreading factor of 12, bandwidth of 125kHz and coding rate of 4/5, which is the usual setting providing the longest range. The transmission power for all tests has been set to 10dBm (following Senegal regulations).

After some additional stability and calibration tests to also normalize the output of various low-cost conductive soil humidity sensors the ϵ threshold has been set to 0.4. Note that the distance normalization step described in Section III.B is useful to highlight similar pairs when some devices are sending very different measures. For the tests, all 8 sensors are placed in a small area, not in a large area as they would be in a real farm. We watered the area so that soil conditions are most likely to be the same for sensors $\{1, 2, 3 , 4\}$, $\{5, 6\}$ and $\{7, 8\}$.

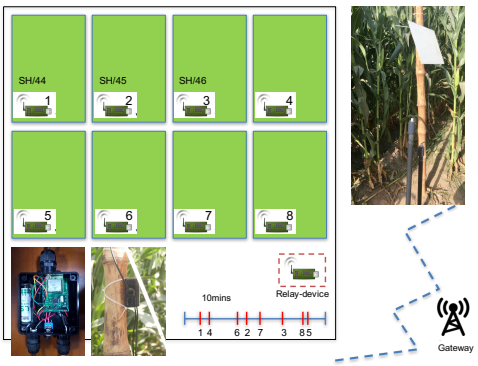

Fig. 9: Deployment scenario

We observed the expected behavior where the relay successfully determines similarity between devices and saves both energy, by reducing the number of wake-ups, and radio time, by not forwarding similar measures. We disabled intentionally the last value suppression mechanism for the same device in order to only focus on the similarity detection between devices as the duration of the test was not long enough for the soil condition to change significantly. The schedule table of the experiment is shown in Figure 10. We can see that for each schedule (sensing period) there are 3 wake-up instead of 8.

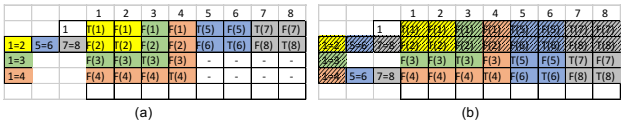

Fig. 10: Deployment scenario

The Arduino ProMini running at 3.3v draws about 15mA in receive mode. In deep sleep mode, the board draws 5uA. When forwarding a packet, the energy consumption is similar to the transmission from an end-device, i.e. 40mA. For each wake-up, there will be a continuous receive for a maximum of 2s (guard time of 500ms and 1.5s of packet reception), then it has to forward the packet during 1.5s (time-on-air of a 20B packet). Therefore, for each uplink packet with wake-up, the relay draws in the average $(2s*15mA+1.5s*40mA)/3.5s \approx 26mA/s$. With 3 wake-up and forwarding every 10 mins (i.e. 6 times in an hour) we have a mean consumption on an 1 hour period of $(6 * (3 * (2s + 1.5s)) * 26mA + (3600s - 63s) * 0.005mA)/3600s \approx 0.460mA/h$ which still allows for more than 226 days of operation compared to the 85 days if the relay node needs to wake-up/forward for the 8 devices. Regarding the radio activity time, from $8 * 6 * 1.5s$ =72s/h which is far greater than the allowed 36s/hour, the relay now consumes $3 * 6 * 1.5s$ =27s/h which is now compatible with ETSI regulations.

Currently, the data platform is not aware of possible similarity relationship between devices, therefore measures that were not forwarded thanks to the similarity detection mechanism appear as missing measures in the graphics. It is possible for the relay to insert the list of similar devices when sending a measure from a given device i. The IoT gateway can then be enhanced with the possibility to reconstruct and upload data for missing devices. It can also do so by using a specific tag to allow the data platform to identify and display to the end-user the similarity attribute.

VI. Conclusion

We presented an extension of a 2-hop LoRa relay node with similarity detection features in order to (i) reduce the power consumption when waking-up to relay packets from child devices and (ii) reduce the radio activity time when running under duty-cycle regulated constraints. The implementation and experimental tests demonstrate the effectiveness of our approach, especially validating the usage of the distance-based similarity function running on low-cost hardware.

Acknowledgments

This work is supported by the WAZIUP (grant No 687607) and WAZIHUB (grant No 780229) projects funded by EU Horizon 2020 research program.

References

[1] M. Diop and C. Pham, "Increased flexibility in long-range iot deployments with transparent and light-weight 2-hop lora approach," in *Proceedings of the IEEE WD'2019, Manchester, UK*, 2019.

[2] ETSI, "Electromagnetic compatibility and radio spectrum matters (erm); short range devices (srd) [...] part 1: Technical characteristics and test methods," 2012.

[3] H. Harb, A. Makhoul, D. Laiymani, and A. Jaber, "A distance-based data aggregation technique for periodic sensor networks," *ACM Transactions on Sensor Networks (TOSN)*, vol. 13, no. 4, pp. 32:1–32:40, 2017.

[4] L. Zhao, Z. Chen, Z. Yang, Y. Hu, and M. S. Obaidat, "Local similarity imputation based on fast clustering for incomplete data in cyber-physical systems," *IEEE Systems Journal*, vol. 12, no. 2, pp. 1610–1620, 2018.

[5] H. Harb, A. Makhoul, and C. A. Jaoude, "A real-time massive data processing technique for densely distributed sensor networks," *IEEE Access*, vol. 6, pp. 56 551–56 561, 2018.

[6] H. Lin, X. Liu, X. Wang, and Y. Liu, "A fuzzy inference and big data analysis algorithm for the prediction of forest fire based on rechargeable wireless sensor networks," *Sustainable Computing: Informatics and Systems*, vol. 18, pp. 101–111, 2018.

[7] M. Henzinger, "Finding near-duplicate web pages: a large-scale evaluation of algorithms," *Proceedings of the 29th ACM SIGIR*, 2006.

[8] A. Z. Broder, S. C. Glassman, M. S. Manasse, and G. Zweig, "Syntactic clustering of the web," *Computer Networks and ISDN Systems*, vol. 29, no. 8-13, pp. 1157–1166, 1997.

[9] T. C. Hoad and J. Zobel, "Methods for identifying versioned and plagiarized documents," *Journal of the American Society for Information Science and Technology*, vol. 54, no. 3, pp. 203–215, 2003.

[10] R. J. Bayardo, Y. Ma, and R. Srikant, "Scaling up all pairs similarity search," *Proceedings of 16th WWW'07*, pp. 131–140, 2007.

[11] S. Sarawag and A. Kirpal, "Efficient set joins on similarity predicates," *Proceedings of ACM SIGMOD*, pp. 743 – 754, 2004.

[12] S. Chaudhuri, V. Ganti, and R. Kaushik, "A primitive operator for similarity joins in data cleaning," *Proceedings of the 22nd ICDE'2006*.

[13] S. Sarawag and A. Kirpal, "Efficient exact set-similarity joins," *Proceedings of the 32nd VLDB'06*, pp. 918–929, 2006.

[14] H. Harb and A. Makhoul, "Energy-efficient sensor data collection approach for industrial process monitoring," *IEEE Trans. Industrial Informatics*, vol. 14, no. 2, pp. 661–672, 2018.

[15] M. Ashouri, H. Yousefi, A. M. A. Hemmatyar, and A. Movaghar, "Foma: Flexible overlay multi-path data aggregation in wireless sensor networks," in *Proceedings of the IEEE ISCC'2012*.

[16] J. Bahi, A. Makhoul, and M. Medlej, "An optimized in-network aggregation scheme for data collection in periodic sensor networks," in *Proceedings of the ADHOC-NOW'2012*, pp. 153–166.

[17] J. M. Bahi, A. Makhoul, and M. Medlej, "A two tiers data aggregation scheme for periodic sensor networks," *Ad Hoc & Sensor Wireless Networks*, vol. 21, no. 1-2, pp. 77–100, 2014.

[18] A. Arasu, V. Ganti, and R. Kaushik, "Efficient exact set-similarity joins," in *Proceedings of the 32nd VLDB'2006*.pp. 918–929.