

A Self-Healing Mesh Network without Global-Time Synchronization

Alphan Şahin* and Hüseyin Arslan†

*University of South Carolina, Columbia and †Istanbul Medipol University, Istanbul
E-mails: asahin@mailbox.sc.edu, huseyinarslan@medipol.edu.tr

Abstract—In this paper, we propose a slot-based protocol that does not rely on global-time synchronization to achieve a self-healing mesh network. With the proposed protocol, each node synchronizes with its neighbors locally by adjusting its time to transmit based on the reception instant of a decoded beacon signal. Also, it determines its slots without any coordinator to avoid collisions. Finally, to communicate the messages over the mesh network, it identifies the forwarding nodes on the shortest path without knowing the entire communication graph. We show that the proposed protocol can effectively resolve collisions over time while enabling nodes to synchronize with each other in a distributed manner. We numerically analyze the performance of the proposed protocol for different configurations under a realistic channel model considering asymmetrical links. We also implement the proposed method in practice with Long-Range (LoRa) devices. We demonstrate that the nodes adapt themselves to changes in the network and deliver a message from a sensing node to a reference node via multi-hop routing.

Index Terms—LoRa, multi-hop routing, mesh networks.

I. INTRODUCTION

Long-Range (LoRa) devices are often used via Long Range Wide Area Network (LoRaWAN) standard relying on a star topology, where the nodes communicate with the gateways connected to the Internet [1]. The gateways forward the data acquired from nodes to the central servers, and users can interact with the central servers by accessing the data. Although this approach is useful for a wide range of applications, the coverage area of a LoRaWAN network is limited to the coverage area of the gateways. Also, the coverage range of a typical LoRa node can degrade in non-line-of-sight (NLoS) conditions since the narrowband LoRa signals can be affected severely by the fading in multi-path channels. Hence, a star topology may not address the communication scenarios where the signals are obstructed. To address the coverage problem, an alternative solution is a mesh topology [2], [3]. With mesh, the communication is achieved by routing the packets over multiple nodes. However, building a large-scale reliable mesh network with low-cost LoRa devices in a distributed manner is not a trivial task because of the collision problems, lack of time-synchronization among nodes, asymmetrical links due to the non-identical transmit powers or hardware impairments, time-variant communication channels, low-end microprocessors, and intermittent nodes. In this study, we address these issues with a new protocol compatible with LoRa devices.

In the literature, LoRa-based mesh networks have been receiving increasing attention. For instance, in [4], it is proposed to modify LoRaWAN with an event-driven approach to support a mesh network. In [5], parallel transmissions with multiple spreading factors are exploited and the nodes are assigned to several subnets to improve the capacity of the mesh network. In [6], a peer-to-peer mesh network without using LoRaWAN is proposed, where the clocks of the nodes are assumed to be synchronized. In [7], to achieve a synchronous LoRa mesh network, the authors utilize the Global Positioning System (GPS) or DCF77 long-wave time signaling. Similarly, in [8], nodes acquire real-time clock information from auxiliary signals and broadcast the acquired information for synchronization. A useful library that allows LoRa nodes to keep the routing tables updated based on routing messages among the nodes is developed in [9] for mesh networks. In [10], the CottonCandy protocol, where the nodes organize themselves in a spanning-tree topology in a distributed fashion, is proposed. In this protocol, the root is a gateway device and all nodes, including the root, start their duty cycles at about the same time. The nodes are assumed to be roughly synchronized within the range of seconds. Meshtastic is one of the successful open-source LoRa mesh projects. It is shown that it can support more than 200 km of communication range [11]. However, this project adopts a plain flooding approach, and the underlying protocols are not particularly optimized for communications in NLoS. We refer the readers to [2] and the references therein for further discussions on LoRa mesh networks.

In this study, to achieve a self-healing mesh network where the nodes adapt themselves to the changes in the network, we propose a slot-based protocol that locally synchronizes the nodes with their neighbors by exploiting the reception instants of the beacons while allowing nodes to choose their slots to mitigate the collisions. We also introduce a technique for a node to obtain its hopping distance to a reference node (e.g., a gateway) to find the shortest routing without knowing the communication graph. With comprehensive simulations and a proof-of-concept demonstration with LoRa devices, we show the proposed protocol resolves the collision events over time, synchronizes the nodes without GPS signals, and coordinates a dynamic network in a distributed manner.

II. PROBLEM STATEMENT

Consider a scenario where N_{sen} sensing nodes spread in an area. Also, suppose that N_{ref} reference nodes are deployed

in the same region. We assume that all nodes use the same carrier frequency and can either transmit or receive signals at a given time. The ultimate goal of the network formed by all nodes is to transfer a message initiated by a sensing node to one of the reference nodes or vice versa, reliably.

A. Finding and tracking neighbors

In practice, the link between any two nodes can be severely affected by the wireless channel. Also, the transmit power of the nodes may not be identical. Hence, without loss of generality, the network can be abstracted as a directed dynamic graph changing over time, where the vertices and the arcs correspond to the nodes and the directed communication links, respectively. To express the graph, we let the positive integer a_n to denote the n th node identity (ID) for $a_n \neq a_m$, $\forall n, m \in \{1, 2, \dots, N_{\text{node}}\}$ and $N_{\text{node}} \triangleq N_{\text{sen}} + N_{\text{ref}}$. Let $c_{n,m}^{(t)}$ show if the n th node can decode the m th node's packet or not at time t . If the m th node's signals can be decoded by the n th node, $c_{n,m}^{(t)}$ is 1. Otherwise, $c_{n,m}^{(t)}$ is set to 0. We define the set of nodes that can reach the n th node and the set of nodes that can communicate bi-directionally with the n th node as $\hat{\mathcal{R}}_n^{(t)} \triangleq \{a_m | c_{n,m}^{(t)} = 1, \forall m \in \{1, 2, \dots, N_{\text{node}}\}\}$ and $\mathcal{N}_n^{(t)} \triangleq \{a_m | c_{n,m}^{(t)} = c_{m,n}^{(t)} = 1, \forall m \in \{1, 2, \dots, N_{\text{node}}\}\}$, respectively. Hence, $\mathcal{N}_n^{(t)}$ is a subset of $\hat{\mathcal{R}}_n^{(t)}$ in general.

The first challenge that needs to be addressed is that $\mathcal{N}_n^{(t)}$ and $\hat{\mathcal{R}}_n^{(t)}$ are not available to the n th node. Hence, the n th node needs to learn and actively track the sets $\hat{\mathcal{R}}_n^{(t)}$ and $\mathcal{N}_n^{(t)}$ to achieve a self-healing network. To this end, let $\mathcal{R}_n^{(t)}$ and $\mathcal{N}_n^{(t)}$ denote the set of discovered nodes that can be heard by the n th node at time t and the set of discovered nodes that can communicate with the n th node at time t , respectively. Both $\mathcal{R}_n^{(t)}$ and $\mathcal{N}_n^{(t)}$ are empty sets for $t = 0$ and need to be obtained by a protocol, as close as possible to $\hat{\mathcal{R}}_n^{(t)}$ and $\mathcal{N}_n^{(t)}$.

B. Synchronizing nodes and mitigating collisions

Due to the absence of global-time synchronization among nodes, a transmitted packet may not be received by the intended receiver as the node may not be listening to the channel at that particular time. This issue can be more severe with typical LoRa nodes as they are often equipped with low-end microprocessors to increase battery life. Hence, the microprocessor may need to allocate some time to handle other functionalities, such as a computation task for sensing, causing blind periods. To address this issue, we consider *periodically alternating roles (PAR)*, where we allocate fixed durations for processing, i.e., T_{proc} , and communications, i.e., T_{comm} , and alternate them. We further divide the communication duration into N_{slot} slots, where the duration of each slot is $T_{\text{slot}} = T_{\text{comm}}/N_{\text{slot}}$ as can be seen in Fig. 1. Let $s_n^{(t)} \in \{1, \dots, N_{\text{slot}}\}$ denote the slot index chosen by the n th node at time t . We define three roles for all nodes: 1) Processing: The node cannot transmit or receive as the microprocessor is busy with some other processing tasks. 2) Responder: The node actively listens to the communication channel to receive a packet. 3) Initiator: The node initiates a task by transmitting a packet.

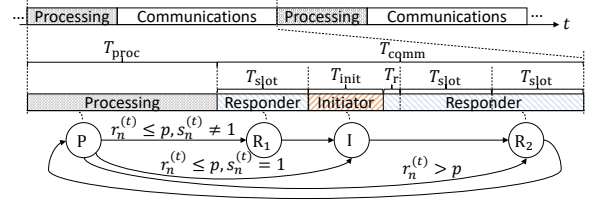


Fig. 1. PAR for $N_{\text{slot}} = 4$ and the corresponding states. In this example, the node uses $s_n^{(t)} = 2$ when it is an initiator and changes its state when $T_{\text{state},n}^{(t)}$ is zero in addition to the conditions given in the state diagram.

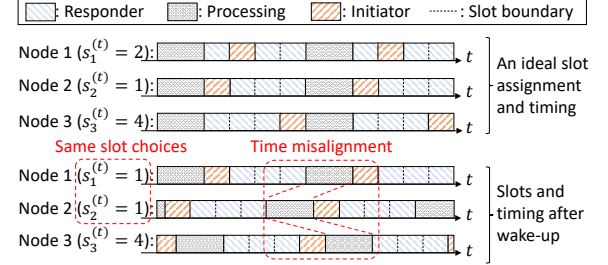


Fig. 2. The nodes can choose the same slot index and they are out-of-sync since they wake up at different times almost surely ($N_{\text{slot}} = 4$).

We denote the initiator duration as T_{init} for $T_{\text{init}} \leq T_{\text{slot}}$. For $T_r \triangleq T_{\text{slot}} - T_{\text{init}} > 0$, the node behaves as a responder for the remaining duration within the slot.

In this study, we assume that each node chooses its slots randomly after it wakes up. Also, the n th node is granted to be an initiator only within the slot $s_n^{(t)}$ with the probability of p . To express this probabilistic behavior, let $r_n^{(t)}$ be a uniform random variable between 0 and 1, drawn during the processing duration. If $r_n^{(t)} > p$ holds, the node behaves as a responder on the corresponding communication duration. Otherwise, it is granted to be an initiator. To show the transitions between the aforementioned roles, we denote the n th node state as $S_n^{(t)} \in \{P, R_1, I, R_2\}$, where P, R_1 , I, and R_2 symbolize the processing, the responder role before being an initiator, the initiator role, and the responder role after being an initiator, respectively. Let $T_{\text{state},n}^{(t)}$ denote the remaining time for the n th node at time t to switch its behavior for the next role. When $T_{\text{state},n}^{(t)}$ is zero, the node changes its behavior. For instance, before entering the processing state, the node sets $T_{\text{state},n}^{(t)}$ to T_{proc} seconds. In the processing state, the node completes the computation tasks and waits until $T_{\text{state},n}^{(t)}$ becomes zero to go to the next state. We illustrate the states of the n th node in Fig. 1. It is worth noting that the node can stay in the same state by extending $T_{\text{state},n}^{(t)}$, which is exploited in Section III-A. Also, the state transitions based on $T_{\text{state},n}^{(t)}$ can be easily implemented in practice [12].

With PAR, all nodes run the same cycle in their microprocessors as there is no hierarchical difference among the nodes. However, since the nodes wake up at different instants almost surely, the processing and communication durations of the nodes are not aligned. Also, the slot indices are chosen ran-

domly initially. Hence, as exemplified in Fig. 2 for $N_{\text{slot}} = 4$, the collisions or missed packets can occur arbitrarily, and the nodes may not discover or communicate with their neighbors. Suppose that global time synchronization is achieved and PAR is used in the network. The slot assignment problem can then be expressed as

$$\{\hat{s}_n^{(t)} | \forall n\} = \arg \min_{\{s_n^{(t)} | \forall n\}} \sum_{n=1}^{N_{\text{node}}} \int_{t-T_{\text{init}}}^t \mathbb{I}[\mathcal{M}_n(t') > 1] dt', \quad (1)$$

for $\mathcal{M}_n(t) = \left| \{a_m | a_m \in \mathcal{R}_n^{(t)}, S_m^{(t)} = 1\} \right|$, where the function $\mathbb{I}[\cdot]$ results in 1 if its argument holds, otherwise, it is 0. The function $\mathcal{M}_n(t)$ quantifies the number of transmitting nodes such that their signals reach the n th node at time t . If $\mathcal{M}_n(t') > 1$ for $t' \in [t - T_{\text{init}}, t]$, the n th node can receive none of the transmitted packets on this duration for a synchronized network. The problem in (1) is combinatorial and it is not trivial to solve even for a coordinated static network. In this work, we address the same slot assignment problem in a distributed manner when there is no global time synchronization and the nodes do not know their neighbor sets.

III. PROPOSED METHODS

A. Joint time-and-slot adjustment and neighbor discovery

For a node to find its neighbors while addressing the synchronization and slot assignment problems, we propose a "listen-and-adjust" protocol relying on local adjustments that occur at any node that receives a beacon signal from one of its neighbors. Hence, the proposed protocol inherently runs in parallel at all nodes. We introduce the following rules:

- If the n th node is an initiator, it transmits a beacon signal that indicates its ID and slot index and its neighbors' IDs and slot indices, i.e., $\{(a_n, s_n^{(t)}), \{(a_m, s_m^{(t)}) | a_m \in \mathcal{R}_n^{(t)}\}\}$. We set $T_{\text{init}} = T_{\text{beacon}}$ seconds, where T_{beacon} is the duration of the beacon signal.
- If the n th node is a responder and can decode the m th node's beacon signal:
 - Adding nodes to neighbor lists: The n th node adds a_m to the set $\mathcal{R}_n^{(t)}$. If a_n is listed as a neighbor in the beacon, i.e., $a_n \in \mathcal{R}_m^{(t)}$, (implying that the m th node can decode the n th node's packets), a_m is also added to $\mathcal{N}_n^{(t)}$.
 - Removing the nodes from neighbor lists: To keep their neighbor lists up-to-date (i.e., tracking the changes in the graph), the n th node removes a node from their $\mathcal{R}_n^{(t)}$ and $\mathcal{N}_n^{(t)}$, if its beacon signal is not heard for $N_{\text{max}}(T_{\text{proc}} + T_{\text{comm}})$ seconds.
 - Slot adjustment: If $s_n^{(t)}$ is reported to be utilized at another node (i.e., $s_n^{(t)} \in \{s_m^{(t)} \cup \{s_l^{(t)} | a_l \in \mathcal{R}_m^{(t)}\}\}$ – slot index collision), the n th node chooses another slot from $\{1, \dots, N_{\text{slot}}\} \setminus \{s_l^{(t)} | a_l \in \mathcal{R}_n^{(t)} \cup \mathcal{R}_m^{(t)}\}$, randomly. If there is no available slot, collisions are inevitable, and the slot index is kept the same.
 - Local time synchronization: After the slot adjustment, the n th node re-calculates $T_{\text{state},n}^{(t)}$ as

$$T_{\text{state},n}^{(t)} = \begin{cases} (s_n^{(t)} - s_m^{(t)} - 1)T_{\text{slot}} + T_r & S_n^{(t)} = R_1, s_n^{(t)} > s_m^{(t)}, \\ (N_{\text{slot}} + s_n^{(t)} - s_m^{(t)} - 1)T_{\text{slot}} & S_n^{(t)} = R_1, s_n^{(t)} < s_m^{(t)}, \\ \quad + T_{\text{proc}} + T_r & \\ (N_{\text{slot}} - s_m^{(t)})T_{\text{slot}} + T_r & S_n^{(t)} = R_2, \end{cases} \quad (2)$$

In Fig. 3, we provide an example illustrating how the proposed protocol works for $N_{\text{node}} = 2$ nodes and $N_{\text{slot}} = 4$ slots. Assume that the slot indices are $s_1^{(t)} = 1$ and $s_2^{(t)} = 1$ (i.e., the slot indices collide), $\mathcal{N}_1^{(t)}$, $\mathcal{R}_1^{(t)}$, $\mathcal{N}_2^{(t)}$, and $\mathcal{R}_2^{(t)}$ are empty sets, and the nodes are out-of-sync for $t = 0$. The first beacon (indicating $(a_2, s_2^{(t)} = 1)$) transmitted from Node 2 cannot be received by Node 1 as Node 1 is in the processing state. However, the first beacon (indicating $(a_1, s_1^{(t)} = 1)$) transmitted from Node 1 is received by Node 2. Hence, Node 2 adds a_1 to $\mathcal{R}_2^{(t)}$. Since the beacon does not report a_2 as a neighbor of Node 1, $\mathcal{N}_2^{(t)}$ is still an empty set (i.e., Node 2 can listen Node 1, but it does not know if its signals can be decoded by Node 1). From the same beacon, Node 2 learns that Node 1 also uses the first slot index. Hence, it chooses another slot (i.e., $s_2^{(t)} = 2$) among three available slots to avoid collision. Subsequently, it re-calculates $T_{\text{state},2}^{(t)}$ as $3T_{\text{slot}} + T_r$ seconds (the third case in (2)) and behaves as a responder before transitioning to the processing state. Hence, Node 2 becomes synchronous to the timing of Node 1. Similarly, for the second beacon (indicating $(a_1, s_1^{(t)} = 1)$) transmitted from Node 1, Node 2 re-adjusts $T_{\text{state},2}^{(t)}$ (the first case in (2)) and waits for T_r seconds. The second beacon transmitted from Node 2 indicates $(a_2, s_2^{(t)} = 2)$ and $(a_1, s_1^{(t)} = 1)$. Hence, Node 1 updates $\mathcal{R}_1^{(t)}$ and also adds a_2 to $\mathcal{N}_1^{(t)}$ as its ID is listed in the beacon. It also calculates $T_{\text{state},1}^{(t)}$ (the third case in (2)) as $2T_{\text{slot}} + T_r$ seconds. Similarly, Node 2 adds a_1 to $\mathcal{N}_2^{(t)}$ after a_2 is listed in the third beacon transmitted from Node 1. In the following periods, the nodes re-calculate $T_{\text{state},1}^{(t)}$ and $T_{\text{state},2}^{(t)}$ when they receive a beacon signal, and they stay in sync with each other.

Note that if two nodes are in sync and choose the same slots, their signals can interfere with each other for all periods for $p = 1$, and the other nodes cannot learn and broadcast that the corresponding slot is utilized. For $p < 1$, such collisions can be resolved and the slot information can be learned. Secondly, the proposed protocol addresses the hidden node problem since the nodes that receive a beacon packet avoid using the reported slots in the beacon.

B. Transferring the sensing messages to the reference nodes

We consider a similar approach to the Bellman-Ford routing [13]. Let us define the *hopping number* of the n th node, i.e., $h_n^{(t)}$, as the number of hops to reach a reference node, calculated at time t . By definition, the hopping number of a reference node is always 0. For $t = 0$, the sensing nodes do not know their hopping numbers and we set $h_n^{(t)}$ to $h_{N/A}$ if the n th node does not know its hopping number, where $h_{N/A}$ is a pre-determined number larger than the maximum degree of separation of the graph. We extend the rules discussed in Section III-A as follows:

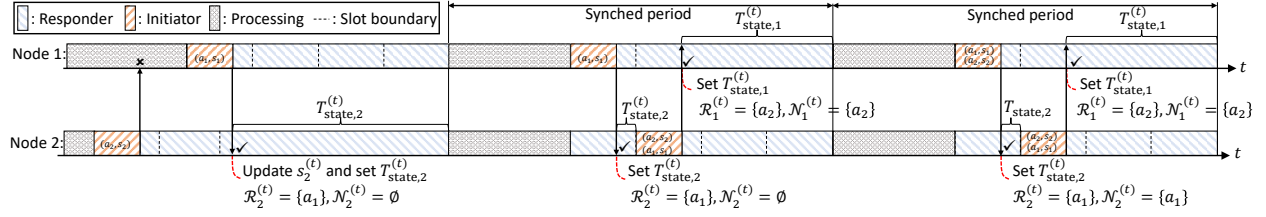


Fig. 3. An example of the proposed time-and-slot adjustment and neighbor discovery protocol ($N_{\text{node}} = 2$, $N_{\text{slot}} = 4$).

- Broadcasting hopping number: If the n th node transmits a beacon signal, it indicates $h_n^{(t)}$ in the beacon.
- Calculating hopping number: If the n th node decodes the beacon signal transmitted from the m th node, it registers $h_m^{(t)}$. It re-calculates $h_n^{(t)}$ as

$$h_n^{(t)} = 1 + \min_{a_l \in \mathcal{N}_n^{(t)}, h_l^{(t)} < h_{N/A} - 1} h_l^{(t)}, \quad (3)$$

if $\{l | a_l \in \mathcal{N}_n^{(t)}, h_l^{(t)} < h_{N/A} - 1\}$ is not an empty set. Otherwise, $h_n^{(t)}$ is $h_{N/A}$.

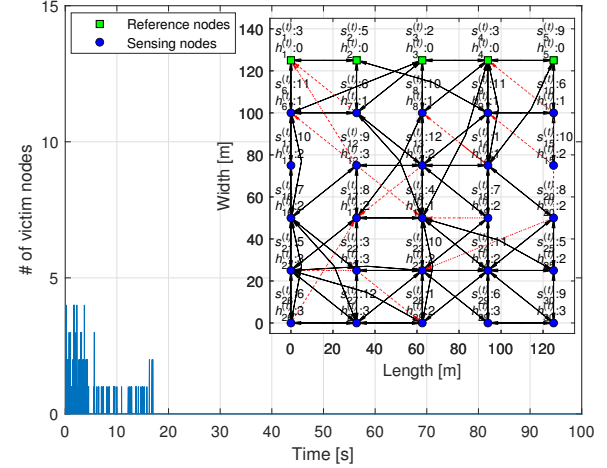
- Identifying the forwarding node: If the n th node receives a message, it forwards it to the node with the smallest hopping number in $\mathcal{N}_n^{(t)}$, i.e., $a_l' = \arg \min_{a_l \in \mathcal{N}_n^{(t)}} h_l^{(t)}$. If there are multiple nodes with the smallest hopping number, the node forwards the sensing message to a randomly chosen node with the smallest hopping number.

Note that the node can forward the message to the one that has the highest received signal strength information (RSSI) or signal-to-noise ratio (SNR) or all of the nodes with the smallest hopping number to improve reliability. Also, if the nodes register the pair of the ID of the original sender of the node and the ID of the forwarding node, the messages can be transferred from the reference node to the originating node via the same route in the reverse direction.

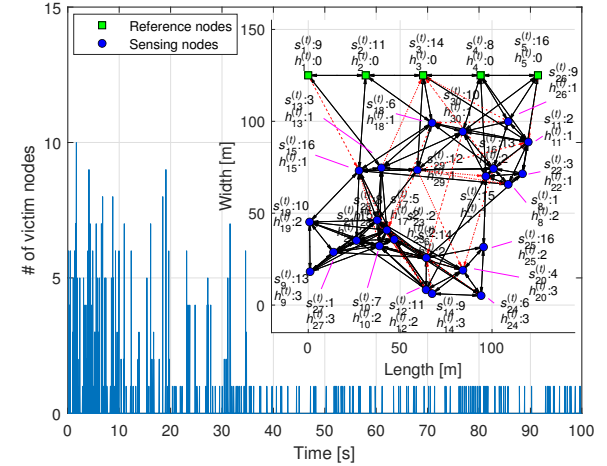
IV. NUMERICAL RESULTS

In this section, we assess the proposed methods with both simulations and a proof-of-concept demonstration. For simulations, we consider $N_{\text{ref}} = 5$ nodes separated apart by 30 m and 25 m on the x-axis and y-axis, respectively. We assume that $N_{\text{sen}} = 25$ sensing nodes are randomly distributed in an area where its size is 125 m by 100 m unless otherwise stated. For the large-scale fading, we assume that the path loss exponent is $\eta = 3.7$ and the variance of the log-normal shadowing coefficient between the n th and m th node, i.e., $\phi_{n,m}$ is 6 dB for $\phi_{n,m} = \phi_{m,n}$. For the small-scale fading, we consider Rayleigh fading, i.e., $h_{n,m} \sim \mathcal{CN}(0, 1)$ for $|h_{n,m}|^2 = |h_{m,n}|^2$. We assume that the minimum SNR is $\gamma_m = -5$ dB to decode a packet and the reference SNR is $\gamma_0 = 20$ dB if the distance between two nodes is $d_0 = 10$ meters. We model the transmit power imbalance of the n th node, i.e., ψ_n , as a zero-mean Gaussian distribution with a variance of 3 dB. Based on this model, we obtain $c_{n,m}^{(t)}$ as

$$c_{n,m}^{(t)} = \begin{cases} 1 & \gamma_0 + \psi_m + \phi_{n,m} + 10 \log_{10} \frac{d_{n,m}^\eta |h_{n,m}|^2}{d_0^\eta} > \gamma_m, \\ 0 & \text{otherwise,} \end{cases}$$



(a) Number of victims over time for a regular deployment ($N_{\text{slot}} = 12$).



(b) Number of victims over time for a random deployment ($N_{\text{slot}} = 16$).

Fig. 4. The proposed protocol reduces the collision events while resolving the hidden-node problems ($p = 0.5$, a dashed arrow: a directional link, a solid double-ended arrow: a bi-directional link).

We choose $T_{\text{beacon}} = 5$ ms, $T_{\text{slot}} = 10$ ms, $T_{\text{proc}} = 10$ ms, and $p = \{0.25, 0.5\}$, $h_{N/A} = 30$, and $N_{\text{max}} = 10$. The nodes awake up at random times between 0 and 100 ms. The behaviors of the nodes are modeled in MATLAB environment.

In Fig. 4, we demonstrate the performance of the proposed protocol by evaluating the number of victim nodes (i.e., a node interfered by a neighboring node) over time, i.e., $\sum_{n=1}^{N_{\text{node}}} \mathbb{I}[\mathcal{M}_n(t) > 1]$ in (1), for two deployment scenarios. In Fig. 4(a), we consider a rectangular tessellation. We also

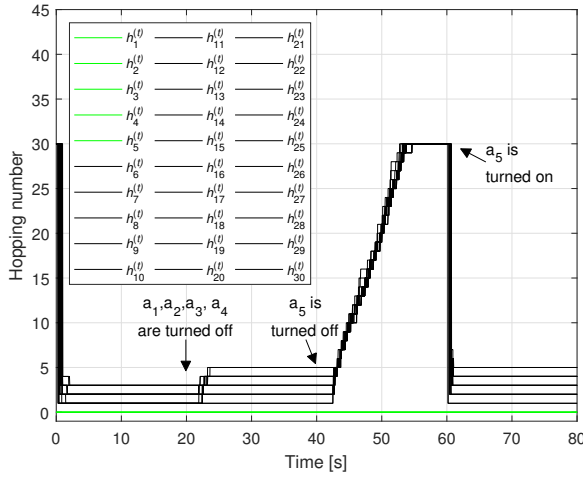
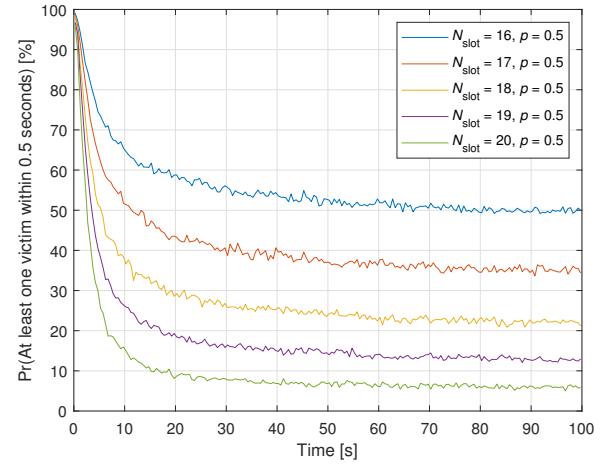


Fig. 5. In this example, the reference nodes are turned off and turned on and all nodes adapt themselves by re-calculating their hopping numbers (green: reference nodes, black: sensing nodes).

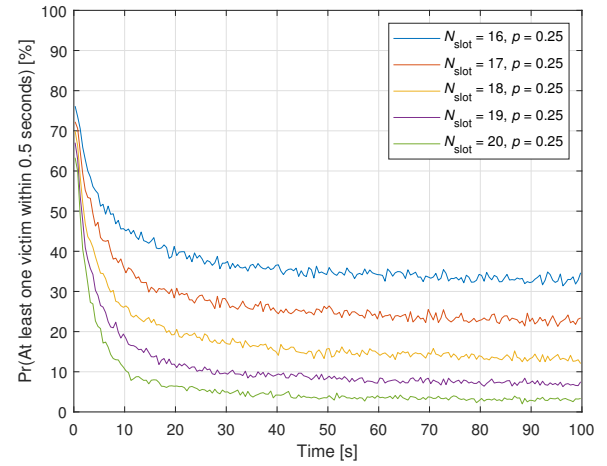
indicate the links between the nodes and mark the slots and the hopping numbers chosen by the nodes at $t = 100$ seconds for $N_{\text{slot}} = 12$. For this configuration, the nodes reach a consensus at $t = 17$ seconds and the number of victim nodes quickly reduces to 0. In Fig. 4(b), we consider random deployment for $N_{\text{slot}} = 16$. Initially, we observe a large number of victim nodes. However, the protocol effectively mitigates the collision over time and there is only 1 victim node after $t = 35$ seconds in this challenging graph. Note that for both scenarios, the protocol addresses the hidden node problems. For example, slot 11 is used by the (6, 9, 24)th nodes and none of their first and second degrees of neighbors use slot 11 after the resolution in Fig. 4(a). In Fig. 4(a) and Fig. 4(b), we also observe that each node obtains its hopping number based on its shortest hopping path to one of the reference nodes. For example, in Fig. 4(a), $h_{12}^{(t)} = 3$ as the minimum hopping number in $\mathcal{N}_{12}^{(t)} = \{a_{13}, a_{16}, a_{17}\}$ is 2.

In Fig. 5, we demonstrate the self-healing behavior of the network. We consider the same scenario in Fig. 4(a) and plot how the hopping numbers change over time. We turn off the first four reference nodes at $t = 20$ seconds. At $t = 22$ seconds, the nodes remove the turned-off nodes from their neighbor lists as they do not get any beacon from the turned-off nodes. Consequently, the sensing nodes re-adjust their hopping numbers within 2 seconds. At $t = 40$ seconds, we also turn off the last reference node. Hence, there is no reference node in the network. As a result, all the nodes gradually increase their hopping numbers till they reach $h_{N/A} = 30$. We then turn on the fifth node and the nodes quickly discover the reference node and learn their hopping numbers.

In Fig. 6, we analyze the probability of having at least one victim node within $[t - T/2, t + T/2]$ duration for a given t , i.e., $\Pr\left(\mathbb{I}\left[\left(\sum_{n=1}^{N_{\text{node}}} \int_{t-T/2}^{t+T/2} \mathbb{I}[\mathcal{M}_n(t') > 1] dt' > 0\right) = 1\right]\right)$ for the same scenario in Fig. 4(b) and set T to 0.5 seconds. We consider 2000 realizations and run the experiments for 100 seconds. As can be seen from Fig. 6, the proposed protocol



(a) $p = 0.5$.

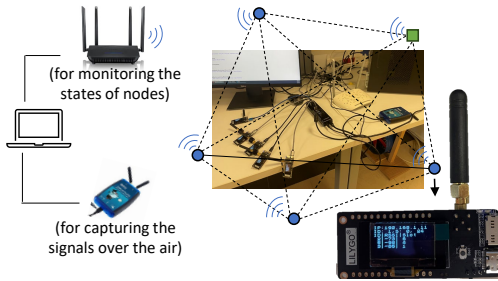


(b) $p = 0.25$.

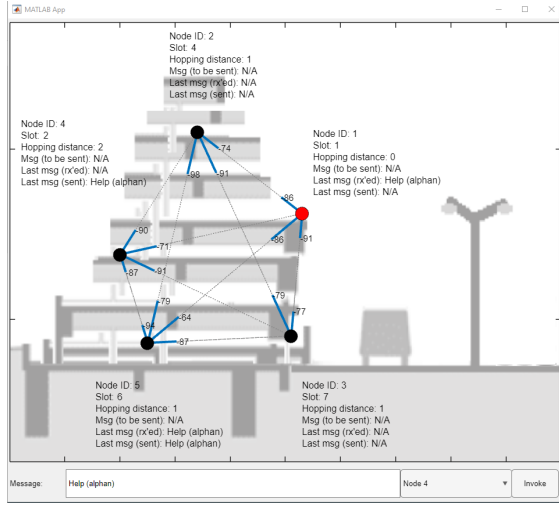
Fig. 6. The collision probability reduces over time with proposed protocol.

reduces the number of victim nodes over time. Also, as expected, increasing the number of slots reduces the collision events. In Fig. 6(a), we use $p = 0.5$. The collision probability is reduced from the range of 90%-100% to 10% within 20 seconds for $N_{\text{slot}} = 20$. In Fig. 6(b), we reduce p to 0.25 and the collision probability is lowered as compared to the first case. In this case, we observe that the probability is reduced from approximately 70% to 10% within 10 and 30 seconds for 20 and 19 slots, respectively.

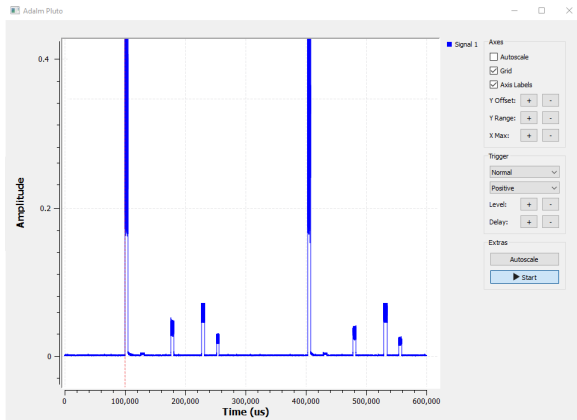
We also implement the proposed methods by using 5 LILYGO LoRa development boards using ESP32-S3 and Semtech SX1280 chipset as can be seen in Fig. 7 for $N_{\text{sen}} = 4$ and $N_{\text{ref}} = 1$ [12]. In this setup, we acquire the status of each node over a Wi-Fi network during the processing duration and monitor the activity in the spectrum with an Adalm-Pluto via GNU Radio, as shown in Fig. 7(a). We track the state of each node, its slot, ID, hopping number, and RSSI of their neighbors over a graphical user interface (GUI) in MATLAB as shown in Fig. 7(b). In the GUI, the bars indicate the RSSI in dBm. To create asymmetrical links, we intentionally add a 30 dB attenuator between the antenna and the RF port for Node 4



(a) The setup for the proof-of-concept demonstration.



(b) MATLAB GUI. The RSSI is shown by blue bars.



(c) An instant of the captured LoRa signals with GNU Radio.

Fig. 7. Implementation ($N_{\text{sen}} = 4$ and $N_{\text{ref}} = 1$). The message sent from Node 4 reaches Node 1 over Node 5. The synchronization is achieved and the slots are chosen as $s_1^{(t)} = 1$, $s_2^{(t)} = 4$, $s_3^{(t)} = 7$, $s_4^{(t)} = 2$, and $s_5^{(t)} = 6$.

and Node 5. Hence, they can decode the packets, but their signals cannot be decoded by all nodes. With this setup, we demonstrate the slot assignments, time adjustments, and multi-hop routing. For the implementation, we set $T_{\text{slot}} = 25$ ms, $T_{\text{proc}} = 100$ ms, $N_{\text{slot}} = 8$, $N_{\text{max}} = 50$, and $h_{N/A} = 127$. For the LoRa packets, we use a spreading factor of 10, a coding rate of $4/5$, and 812.5 kHz bandwidth, respectively. As can be seen from Fig. 7(b) and Fig. 7(c), the nodes choose their slots as $s_1^{(t)} = 1$, $s_2^{(t)} = 4$, $s_3^{(t)} = 7$, $s_4^{(t)} = 2$, and

$s_5^{(t)} = 6$ and synchronization in the entire network is achieved without any GPS signal and it is well-aligned in the time domain. In the demo, the hopping number of Node 4 is 2 because the minimum hopping number among the nodes in $\mathcal{N}_5^{(t)}$ (i.e., Node 5) is 1. Hence, if Node 4 is triggered to transmit a message, the corresponding message is forwarded to Node 5 first. Afterward, the Node 5 forwards the message to the reference node, i.e., Node 1.

V. CONCLUDING REMARKS

In this study, we present a self-healing mesh network that does not rely on global-time synchronization. In the network, all nodes adjust their slots and timings via the received beacon signals and synchronize themselves with their neighbors locally. Also, the nodes obtain the forwarding nodes on the optimal routes without knowing the communication graph. Through comprehensive simulations, we show that the proposed protocol effectively reduces collision events, resolves hidden node problems, and tracks the changes in the network. We also implement the proposed protocols by using LoRa SX1280 chipsets and provide proof-of-concept results. In particular, the proposed protocol can be useful for building a low-cost mesh network in challenging scenarios where the GPS signals cannot penetrate. Future work will analyze the data rate and energy efficiency of the proposed scheme while incorporating the sensing features of nodes.

REFERENCES

- [1] M. Jouhari, N. Saeed, M.-S. Alouini, and E. M. Amhoud, "A survey on scalable LoRaWAN for massive IoT: Recent advances, potentials, and challenges," *IEEE Commun. Surveys & Tutorials*, vol. 25, no. 3, pp. 1841–1876, 2023.
- [2] R. P. Centelles, F. Freitag, R. Meseguer, and L. Navarro, "Beyond the star of stars: An introduction to multihop and mesh for LoRa and LoRaWAN," *IEEE Pervasive Computing*, vol. 20, no. 2, pp. 63–72, 2021.
- [3] J. R. Cotrim and J. H. Kleinschmidt, "LoRaWAN mesh networks: A review and classification of multihop communication," *Sensors*, vol. 20, no. 15, 2020.
- [4] H. Huh and J. Y. Kim, "LoRa-based mesh network for IoT applications," in *Proc. IEEE World Forum on Internet of Things (WF-IoT)*, 2019, pp. 524–527.
- [5] G. Zhu, C.-H. Liao, T. Sakdejayont, I.-W. Lai, Y. Narusue, and H. Morikawa, "Improving the capacity of a mesh LoRa network by spreading-factor-based network clustering," *IEEE Access*, vol. 7, pp. 21 584–21 596, 2019.
- [6] R. Berto, P. Napoletano, and M. Savi, "A LoRa-based mesh network for peer-to-peer long-range communication," *Sensors*, vol. 21, no. 13, 2021.
- [7] C. Ebi, F. Schaltegger, A. Rüst, and F. Blumensaat, "Synchronous LoRa mesh network to monitor processes in underground infrastructure," *IEEE Access*, vol. 7, pp. 57 663–57 677, 2019.
- [8] P. Manzoni, S. E. Merzougui, C. E. Palazzi, and P. Pozzan, "A resilient LoRa-based solution to support pervasive sensing," *Electronics*, vol. 12, no. 13, 2023.
- [9] J. M. Solé, R. P. Centelles, F. Freitag, and R. Meseguer, "Implementation of a LoRa mesh library," *IEEE Access*, vol. 10, pp. 113 158–113 171, 2022.
- [10] D. Wu and J. Liebeherr, "A low-cost low-power LoRa mesh network for large-scale environmental sensing," *IEEE Internet of Things Journal*, vol. 10, no. 19, pp. 16 700–16 714, 2023.
- [11] Mestastic. (2023) Mestastic: An open source, off-grid, decentralized, mesh network built to run on affordable, low-power devices. [Online]. Available: <https://meshtastic.org/docs/overview/range-tests>
- [12] A. Şahin, "Project: LoRa Quake," <https://github.com/alphansahin/LoRa-Quake>, 2023.
- [13] D. Bertsekas and R. Gallager, *Data Networks (2nd Ed.)*. USA: Prentice-Hall, Inc., 1992.