A Dynamic Topology Construction Algorithm for Self-Organizing Wireless Sensor Networks

Jianjun Wen and Waltenegus Dargie *Senior Member, IEEE*

***Abstract*—Self-organizing protocols and algorithms require knowledge of the underlying topology of the network. The topology can be represented by a graph or an adjacency matrix. In most practical cases, establishing the topology prior to a deployment is not possible because the exact placement of nodes and the existence of a reliable link between any two individual nodes cannot guaranteed. Therefore, this task has to be carried out after deployment. If the network is stand-alone and certain aspects are fixed (such as the identity of the base station, the size of the network, etc.), the task is achievable. If, however, the network has to interact with other systems – such as Unmanned Aerial Vehicles (UAVs) or mobile robots – whose operation is affected by environmental factors, the task can be difficult to achieve. In this paper we propose a dynamic topology construction algorithm, assuming that the network is a part of a joint deployment and does not have a fixed based station.**

I. INTRODUCTION

On February 18, 2021 NASA successfully landed a rover and a small robotic helicopter (UAV) on Mars at a location believed to have astro-biological relevance. The duo are tasked to determine whether the planet was habitable and life thrived on it in the past. An essential component of their assignment is the search for biosignatures within accessible geological materials [8]. The joint deployment is intended to fulfill complementary objectives. The rover is equipped with several advanced instruments for taking and analyzing samples, but its scope is limited due to its limited movement. The UAV hovers above the rover or nearby, thus surveying a wider region in a short time; its activities, nevertheless, are limited on account of its limited energy reserve. Currently, the UAV occasionally leaves the rover to make short flights.

The joint deployment can achieve a higher degree of spatio-temporal sensing if it includes intelligent wireless sensor networks (WSN). The nodes on the ground can save the rover from traveling long distances and the UAV, from making long flights. This type of deployment can have several applications here on earth as well, monitoring remote, dangerous, inaccessible, or extensive places [7], [24], [13]. At the Energy Lab (TU Dresden, Germany), we investigate the practical usefulness of such deployments and the type of communication protocols and algorithms they require.

In a joint deployment, a well-coordinated communication is indispensable to mitigate Cross Technology Interference (CTI) and to minimize the flight time of the UAVs [22]. However, determining the precise topology of the ground network – a prerequisite for establishing efficient routes and gateways – prior to the actual deployment may not be possible due to the difficulty of determining the precise physical placement of individual nodes. Consequently, the topology of the network and the gateway nodes with which the UAVs communicate should be determined after deployment, in a dynamic fashion.

In this paper we propose an algorithm for dynamically constructing the topology of a randomly deployed network. The algorithm enables a flying UAV to identify one or more ground nodes with which it conveniently interacts. Taking these nodes as a reference, the remaining nodes propagate and aggregate neighborhood information based on which a binary adjacency matrix signifying the underlying topology of the network is generated. Once the adjacency matrix is established, the UAV identifies cluster heads and associates child nodes to them. This is done according to the relative significance and distribution – both of which are determined from the adjacency matrix – of the nodes. The key aspects of the algorithm are that it is distributed and scalable.

The rest of this paper is organized as follows: In Section II, we establish the background of this work. In Section III, we present our concept. In Section IV, we discuss the implementation details and our evaluation based on a field deployment. In Section V, we review state-of-the-art and, finally, in Section VI, we give concluding remark and outline future work.

II. BACKGROUND

In [3], we proposed a model to quantify the relative significance of nodes in a wireless sensor network. The measure of significance takes into consideration the degree of connectivity of the nodes as well as the relative significance of their neighbors. The input for our model is a binary adjacency matrix encoding the physical topology of the network – 1 signifying the existence of a direct link and 0, the absence of a direct link. Hence, given an adjacency matrix $\mathbf{M}$, the normalized adjacency matrix $\mathbf{H}$ is given as:

$$\mathbf{H} = \frac{\mathbf{M}}{n-1} \tag{1}$$

where n is the number of nodes in the network. The normalized number of single- as well as multihop links the nodes establish with their peers can be expressed as:

$$\mathbf{T} = \sum_{k=1}^{\infty} (p\mathbf{H})^k = (p\mathbf{H})\,(\mathbf{I} - p\mathbf{H})^{-1} \tag{2}$$

This work has been partially funded by the Free State of Saxony under TG70 Research Funding program (Grant nummber: 100369691).

Jianjun Wen and Waltenegus Dargie are with the Faculty of Computer Science, Technische Universität Dresden, 01062 Dresden, Germany (e-mail: { jianjun.wen, waltenegus.dargie}@tu-dresden.de)

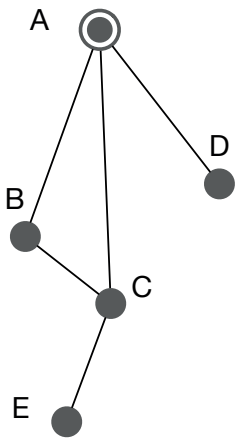


Fig. 1: A simple network topology consisting of five nodes.

p is a probability term expressing the quality of the one-hop wireless link. Given $\mathbf{T}$, the relative significance of the individual nodes can be computed as follows:

$$\mathbf{r} = \mathbf{uT} \tag{3}$$

where $\mathbf{u}$ is a column vector of n unit elements and $\mathbf{r}[i]$ encodes the relative significance of node i in the network. Thus quantifying the relative significance of nodes enables to identify strategic nodes which are critical to disseminate and aggregate data.

In [4], we proposed a self-organizing clustering algorithm which dynamically identifies cluster heads based on their relative significance and their relative hop-distance. Moreover, the algorithm associates child nodes with the cluster heads, taking into account their relative distance from the cluster heads. The algorithm is useful to coordinate and manage joint deployments because it enables a UAV to directly interact with the cluster heads. Nevertheless, both the algorithm and the metrics it relies on (Equation 3) presuppose the existence of a binary adjacency matrix. In Section III, we shall demonstrate how the binary adjacency matrix can be established in a dynamic fashion following a deployment.

III. Concept

Consider Fig. 1 where we have five sensor nodes. A solid line represents the existence of a direct wireless link between two nodes. If a UAV is tasked to interact with this network, its best gateways are node A and C. We may establish this fact by simply inspecting the network. When the network's size is appreciably large and the topology is complex, however, visual inspection does not yield an objective ranking of the nodes' significance in the network. The adjacency matrix in Equation 4 expresses the physical topology of the network. Applying Equation 3 on the adjacency matrix results in a quantitative rank of the nodes (refer to Tab. I).

	1	2	3
Node ID	A, C	B	D, E

TABLE I: Rank of nodes

$$\mathbf{M} = \begin{array}{c} \\ A \\ B \\ C \\ D \\ E \end{array} \begin{array}{c} \begin{array}{ccccc} A & B & C & D & E \end{array} \\ \begin{bmatrix} 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix} \end{array} \tag{4}$$

In most practical cases, the actual topology of the network cannot be known prior to deployment. This is because some of the nodes may not be placed or function as intended and the transmission path of some of the nodes may be blocked by nearby physical objects. Therefore, the topology of the network should be determined dynamically, through local interaction. We combine four complementary features to achieve this goal:

- Random packet transmission.
- In-network data aggregation.
- Implicit time synchronization.
- Collision tolerant medium access.

A. Random Packet Transmission

In the beginning, the ground nodes do not have information about their neighbors or the size of the network. The process of establishing the topology of the network begins with one of the ground nodes taking the initiation. This node may or may not become a gateway eventually. To complete the task as swiftly as possible, we enable a random and collision tolerant interaction.

The process is carried out in two phases. In the first phase, the *discovery* phase, nodes exchange discovery packets and update their own local list of neighbors. In the second phase, the *report* phase, nodes propagate packets towards the initiator, informing it what they know about their neighbors. In order to ensure an efficient message dissemination, we define the following flags: Start (**S**), Update (**U**), Forward (**F**), and Report (**R**). Besides, the header of a packet contains the hop count, the source ID, and the parent ID. Its payload contains a partially completed adjacency matrix based on the knowledge of the node up to that time point (nodes update this matrix whenever they receive packets from their neighbors).

The discovery phase runs for k rounds. In each round there are exactly N slots, where N is a global parameter determined by the size of the network. In each round a node may transmit a packet only once. Which slot it chooses to transmit a packet is determined by a discrete random variable $\mathbf{x} \in \mathbb{W}, 1 \leq \mathbf{x} \leq N$. If a node has multiple packets to transmit, it has to do so in multiple rounds.

The discovery phase begins with the initiator broadcasting a discovery request in slot 0. This packet is flagged **S**; its hop-count is 0 and it contains no payload. This is illustrated in Fig. 2 where node A, the initiator, broadcasts in slot 0. All nodes receiving this packet for the first time list this node as

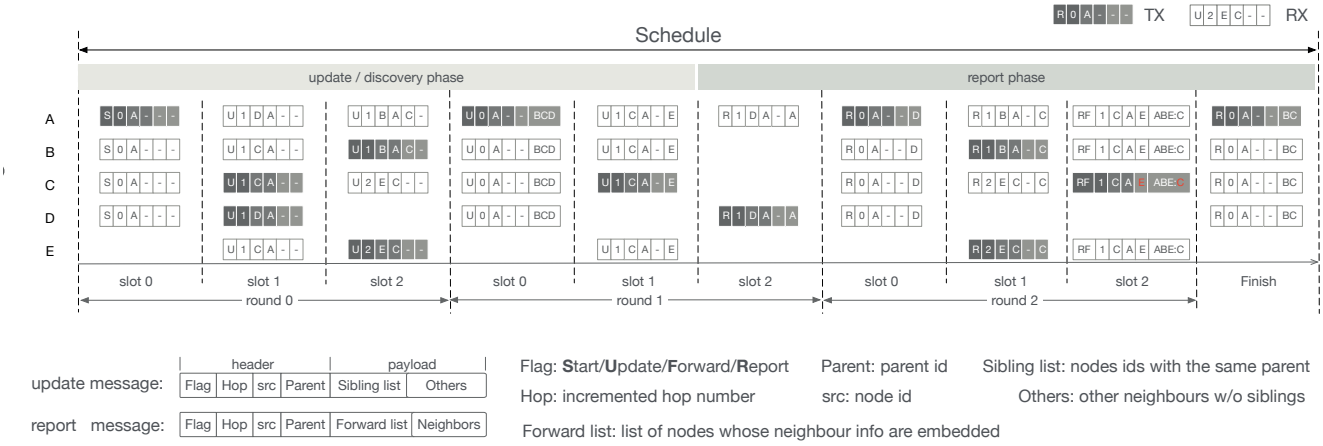


Fig. 2: A topology discovery process for the topology depicted in Fig. 1. Dark boxes signify transmitted packets and light boxes signify received packets.

their parent node. Meanwhile, all neighbor nodes randomly choose a whole number between 1 and N as the value of $\mathbf{x}$. If $\mathbf{x} = 1$, this corresponds to slot 1 and they are eligible to transmit in slot 1, otherwise, they wait until the value of $\mathbf{x}$ matches the slot number. More than one nodes may pick the same value for $\mathbf{x}$, thereby transmitting packets at the same time and causing a collision. In the next subsection, we will discuss how we resolve this concern.

A node eligible to transmit in slot 1 raises the flag **U**, sets the hop-count to 1, fills its parent ID and broadcasts the packet, so that both the initiator and the nodes farther away from the initiator discover it (refer to Fig. 2, slot 1). The update process continues likewise, nodes locally updating their neighbor list every time they receive new packets and increasing the hop count of the packets they rebroadcast. They also keep track of the slot sequence. The discovery phase for a given node comes to an end when the node does not receive any new packets for successive N slots (an entire round). Thereafter, it begins the reporting phase during which time it broadcasts a partially completed adjacency matrix encoding the neighbors of the nodes and those of its neighbors. Child nodes receiving report packets from their parents implicitly receive acknowledgment that they are recognized by their parent nodes, other than that they don't rebroadcast these packets.

B. In-Network Data Aggregation

The adjacency matrix can be constructed in one of the following ways: (1) Either nodes propagate lists containing information about their neighbors towards the initiator, so that the latter reconstructs the adjacency matrix by filtering and aggregating these lists or (2) the process can be carried out gradually, nodes constructing a partially completed adjacency matrix based on their local view of neighborhood and passing this matrix to their neighbors. From a communication point of view, the former is simpler, as intermediate nodes need only to relay the packets they receive from their neighbors towards the initiator. This, however, results in a significant amount of duplicate packets being retransmitted. From a computation point of view, the whole process overwhelms the initiator, thereby resulting in a disproportionate amount of energy being consumed by the initiator. Our model is based on (2). Hence, each node sets up an adjacency matrix based on its local information and forwards this information to its neighbors.

For example, the adjacency matrix node E constructs (Fig. 1) looks like:

$$\mathbf{E} = \begin{array}{c} \\ C \\ E \end{array} \begin{array}{c} \begin{array}{cc} C & E \end{array} \\ \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \end{array} \tag{5}$$

Since an adjacency matrix is binary in our context, a node encodes it into two lists to compress it. The first list contains the nodes' id and the second list contains whole numbers indicating the cells' address containing 1s. For the above example, the first list contains the IDs of nodes C and D[1] and the second list contains the numbers 1 and 4 because these cells contain 1s. Any adjacency matrix can be represented by these two lists, making the transmission and decoding of the adjacency matrix straightforward. Likewise, the adjacency matrix node C constructs resembles the following:

$$\mathbf{C} = \begin{array}{c} \\ A \\ B \\ C \\ E \end{array} \begin{array}{c} \begin{array}{cccc} A & B & C & E \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \end{array} \tag{6}$$

Accordingly, the first list contains the IDs of nodes A, B, C, and E (so that the adjacency matrix is a 4×4 matrix) and the second list contains the numbers: 7, 9, 10, 12, 15. Note that even if B is the neighbor of A, C may not have this information yet, which is why the adjacency matrix does not reflect their neighborhood. Similarly, C may know that A is its neighbor (because the propagation of the discovery packets always goes from a parent to a child); this does not, however, necessarily mean that A knows that C is its neighbor. A may have to wait

[1]Note that the size of the first list implicitly reveals the dimensions of the adjacency matrix.

for the *report* phase to discover this. Which is why $c_{13} = 0$ even though $c_{31} = 1$.

C. Implicit Time Synchronization

The condition to minimize and manage collision is a well-synchronized concurrent transmission. Nodes implicitly synchronize time by estimating the time at which a parent node started the current transmission, because this time corresponds to the beginning of a current slot the length of which is fixed. From the time point a transmitter issues a send command to the actual transmission of the packet a minimum of τ_m elapses. This time is the time the node switches from a receiving mode to a transmission mode, for the node's default mode is a receiving mode. At the receiver's side, the first sign indicating the reception of a packet is a hardware interrupt raised by the receiver's MCU following the successful detection of a 1 byte *Start of Frame Delimiter* (**SFD**) at the physical layer (by the radio chip). A local time registers this time (t_s). Before the SFD, a 4 byte preamble must have been transmitted to synchronize packet transmission between the transmitter and the receiver. Hence, the beginning of the current slot is estimated to be:

$$t_{cs} = t_s - (\tau_m + \tau_p) \tag{7}$$

where τ_p is the time needed to transmit the preamble and the SFD. Equation 7 neglects the propagation time of the electromagnetic signal. Consequently, the beginning of the next slot is simply:

$$t_{ns} = t_{cs} + T_s \tag{8}$$

where T_s is the duration of a single slot, which is a fixed parameter. The transmission schedule of a particular node is given as:

$$t_{tx} = \mathbf{x}\, t_{ns} \tag{9}$$

In general, the transmission schedule for the round $r + 1$ is given as:

$$t_{tx}^{r+1} = t_{tx}^{r} + T_s\,(N + 1 - i + \mathbf{x}) \tag{10}$$

where i is the current slot number.

D. Collision Tolerant Packet Detection

The random packet transmission strategy minimizes packet collision but does not avoid it altogether. The larger the number of slots in a round (N), the smaller is the probability of collision, but also the longer is the time needed to establish the topology of the network (i.e., the adjacency matrix). We exploit a phenomenon called "the capture effect" [18], [1], an old concept in wireless communication, to reduce the impact of packet collision. The idea is as follows: If two or more nodes transmit packets concurrently, a collision may occur at a particular receiver. The receiver may, however, be able to successfully decode one of the packets, provided that two conditions are fulfilled. In the context of the IEEE 802.15.4 [20], [10], [5] specification, these are:

- The time needed to transmit the preamble and the SFD is ca. 160 µs. If, during this time another strong signal arrives, the receiver may lock to this signal provided that:
- The signal's power is at least 3 dB higher than the superposition power of all the other surrounding signals (which it considers as background noise).

The chance of two or more nodes transmitting at the same time and their transmitted signals having the same power level at a receiving node is small. These features make our approach collision tolerant. In Fig. 2, concurrent transmission is illustrated in round 0, slot 1; round 0 slot 2; and round 2 slot 1.

Algorithm 1: Transmission Policy

Input : received message: msg
Output: transmission state machine $State_{tx}$

Phase *Update*:
  if *msg.src not in LNS* && *my_id not in msg.neighbor_set* **then**
    $State_{tx} = State_{tx} + 1$;
  [2]**else**
    if $msg.src == parent\_id$ && *my_id not in msg.neighbor_set* **then**
      $State_{tx} = State_{tx} + 1$;
    end
    if $msg.parent == my\_id$ && *msg.src not in LNS* **then**
      $State_{tx} = State_{tx} + 1$;
    end
  end
end
Phase *Report*:
  if $msg.hop < my\_hop$ **then**
    if *I am not reported* && *my_id not in msg.forward_list* **then**
      $State_{tx} = State_{tx} + 1$;
    end
  else if $msg.hop > my\_hop$ **then**
    $State_{tx} = State_{tx} + 1$;
  else
    drop message;
  end
end

IV. Evaluation

We implemented the algorithm in Contiki-OS [6] on Zolertia RE-Mote revision B motes[3]. For the purpose of evaluation, we deployed the wireless sensor nodes on one of our faculty's corridor on both sides of which were many pillars (refer to Fig. 3). The transmission power of each node was set to -15 dBm (the maximum possible), but the dense concrete walls of the corridor as well as the pillars produced a significant amount of signal reflections and attentions. As we shall demonstrate later, this means the wireless links were not asymmetric. Nor did physical nearness correspond to the existence of a direct or a reliable wireless link.

[2]LNS: Local Neighbor Set.
[3]https://zolertia.io/product/re-mote/

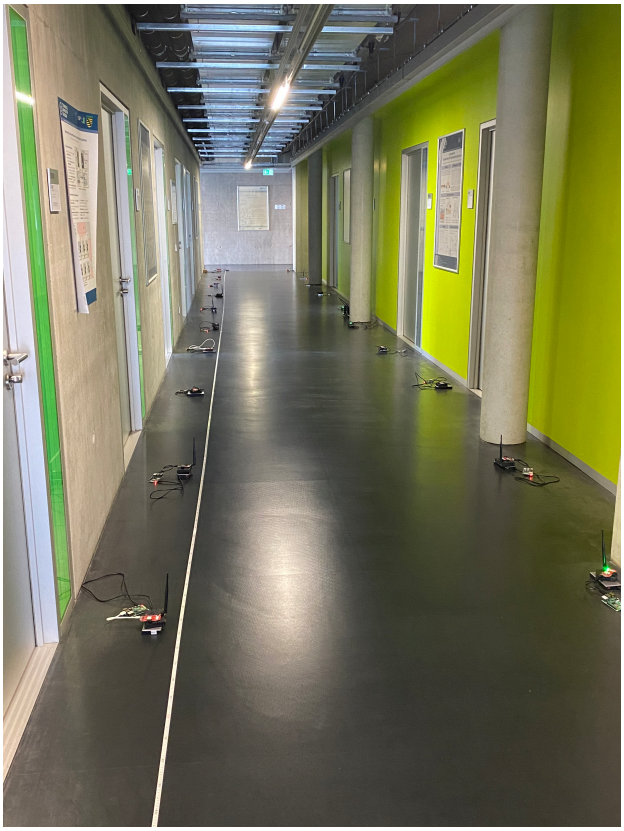


Fig. 3: Deployment of a wireless sensor network along a corridor.

TABLE II: Deployment parameters.

Parameters	Value
Network size	8, 23
TX power	-15
Slot length (T_{slot})	10 ms
Slots in a round ($nslot$)	2, 4, 8
Schedule duration	30 s

A. Number of Slots in a Round

We first evaluated how the algorithm's parameters affected its performance, considering three aspects:

- The average time the algorithm needed to establish the network's topology.
- The success ratio signifying the number of times a node successfully reported neighbor information to the initiator before a schedule was terminated.
- The average number of concurrent transmissions within a slot.

We evaluate these aspects first with a network of 8 nodes. Fig. 4 displays the resulting plots based on 100 independent experiments.

1) Completion Time: Fig. 4 (a) shows the average completion time for each node. The error bar shows the standard deviation. As expected, the completion time increased as the number of slots in a round increased. However, this relationship was not proportional. For example, when the number of slots in a round increased from 2 to 4, the completion time did not double. The reason is that the more slots were available in a round, the less was the probability of concurrent transmission, thereby increasing the probability of receiving a packet successfully. Similarly, as the number of slots increased, the chance of a node gathering neighbor information increased, thus facilitating the update phase. Nevertheless, this parameter should be adapted according to the density of the network. Notice that the node which required the longest time (Node 5) to collect neighbor information was the initiator.

2) Success Ratio: The success ratio refers to the number of times a node was able to successfully transmit a packet. Fig. 4 (b) displays this for each sensor node when the number of slots in a round was 2, 4, and 8, respectively. We observed that when the number of slots was 2, the success ratio of three nodes (node 2, 4, 6) was below 0.8. When we increased the number to 4, the success ratio exceeded 0.9. When, however, the number of slots increased to 8, there was no appreciable increase in the success ratio.

3) Concurrent Transmission: Fig. 4 (c) shows the distribution of concurrent transmissions. As expected, when the number of slots increased, the number of concurrent transmission reduced. For all the settings, over 70% of the slots experienced a single transmission. The factors which contributed to this are the following:

- The random selection of a transmission slot.
- The in-network strategy enabling nodes to learn and disseminate information about their neighbors in single slots.
- The concurrent transmissions occurred in the early stages of the algorithm. The probability of receiving new packets from unknown neighbors decreased over time (exponentially, as can be seen in the figure).

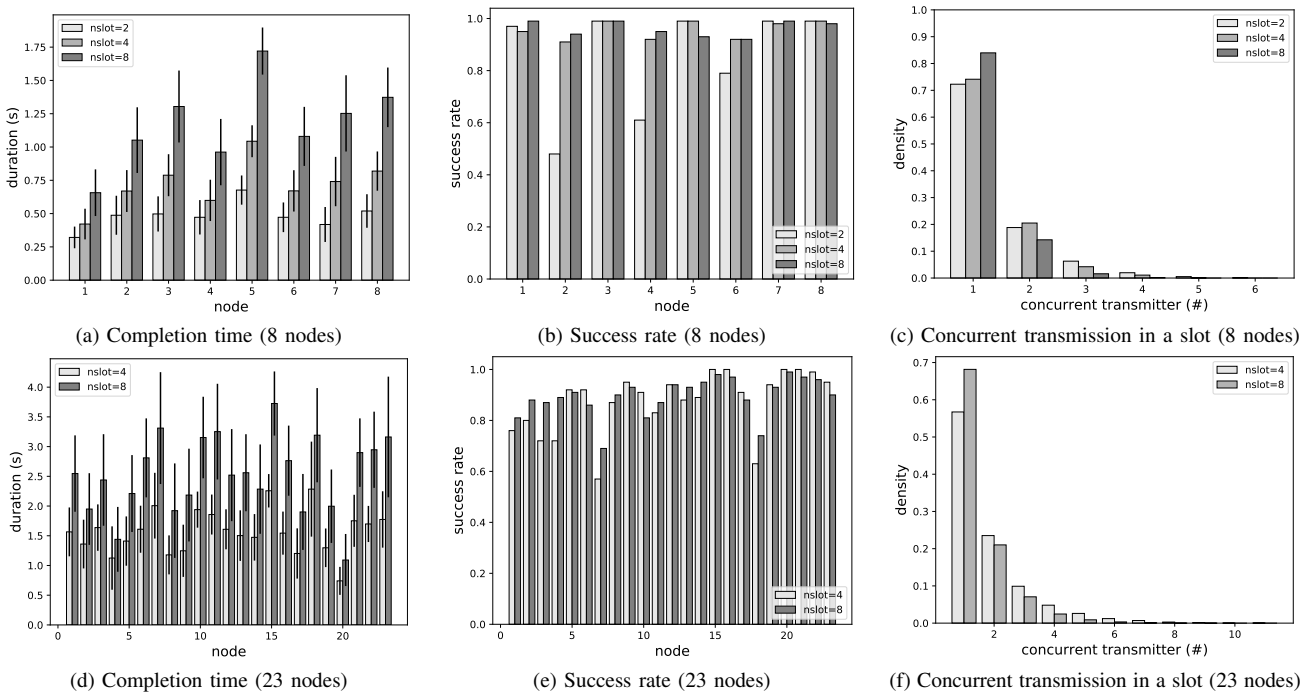
B. Impact of Network size

To investigate the impact of network size on the establishment of the network's topology, we increased the number of nodes to 23. In this case, however, we considered only two slot sizes in a round, namely, 4 and 8.

As in the previous case, the time needed to construct the topology of the network increased as we increased the number of slots in a round. However, the increment was not proportional once again. Whereas the network size was tripled (8 vs. 23), the completion time increased only by 2.1 fold in both settings (4 and 8) (as shown in Fig. 4 (a) and (d)). By comparison, the success ratio deteriorated noticeably, as can be seen in Fig. 4 (b) and (e). Obviously, as the network size increased, concurrent transmission increased, and with it the capture effect became less effective. When the network consisted of 8 nodes and the number of slots in a round was 4, 25% of the time packet transmissions occurred with at least 2 concurrent transmissions, whereas concurrent transmission increased to 45% when the network consisted of 23 nodes.

C. Activity of Nodes in Realtime

After evaluating the overall performance of our algorithm, we analyzed the activities of individual nodes during self-organization. The data trace for this section was obtained from the experiments involving the 23 nodes, when the number of



(a) Completion time (8 nodes) (b) Success rate (8 nodes) (c) Concurrent transmission in a slot (8 nodes)

(d) Completion time (23 nodes) (e) Success rate (23 nodes) (f) Concurrent transmission in a slot (23 nodes)

Fig. 4: Performance evaluation based on different schedule parameters for two different network sizes.

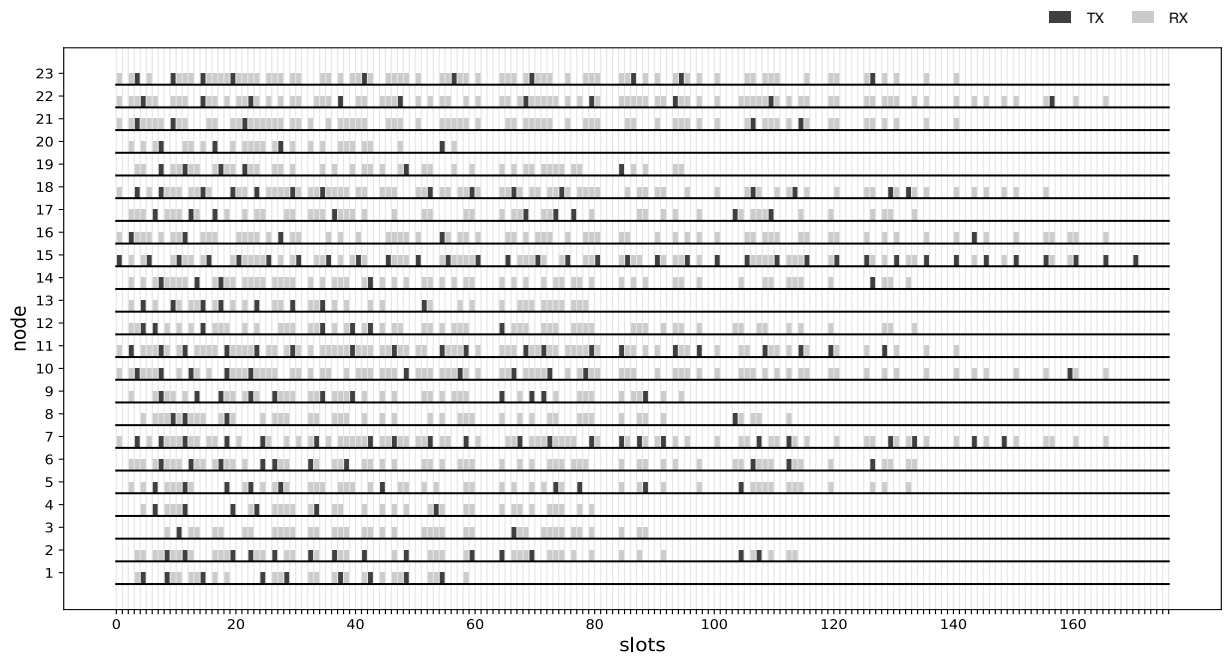


Fig. 5: A real trace of the activities of nodes during self-organization. Network size: 23. Node 15 was the initiator.

slots in a round was 4. The result is shown in Fig. 5. The time line is represented in terms of slots. Node 15 initiated the process of self-organization which lasted 176 slots.

Firstly, we analyze the temporal characteristics of the concurrent transmissions. In the previous subsection we claimed that intensive concurrent transmissions occurred in the early stages of self-organization. Fig. 6 confirms this claim. As shown in the figure, most of the dense concurrent transmissions occurred in the first 25 slots during which nodes discovered new neighbors and were busy sharing this knowledge. As time went by, the frequency of discovering new neighbors decreased and nodes had little to share.

Fig. 7 shows the durations nodes spent in the two phases (discovery and report). We observed that most of the nodes completed the update phase within 65 slots (15 nodes out of 23). Only 4 of the nodes required more than 100 slots for the update phase (node 6, 14, 16 and 21). By looking at the activities of these nodes in Fig. 5, one can discover that

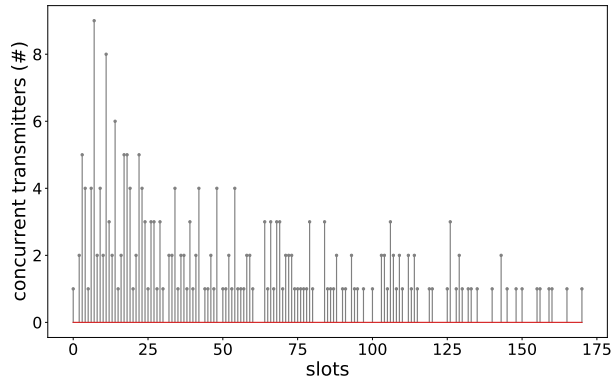


Fig. 6: Concurrent transmissions across the time line.

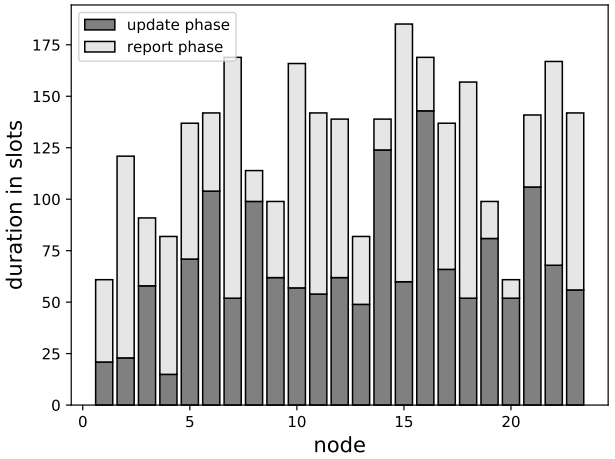


Fig. 7: Comparison of nodes' activities in the update and report phases.

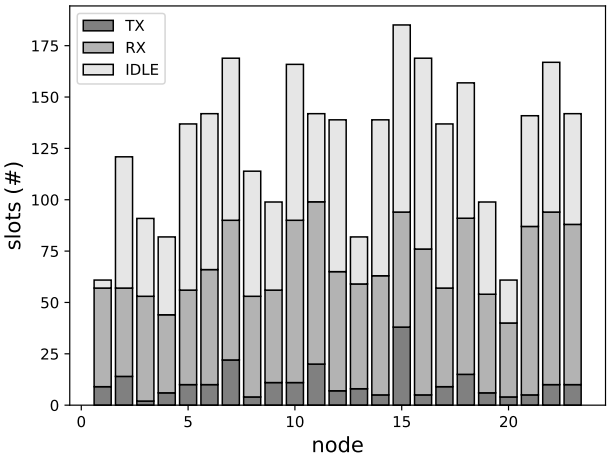


Fig. 8: The activities of individual nodes in individual time slots.

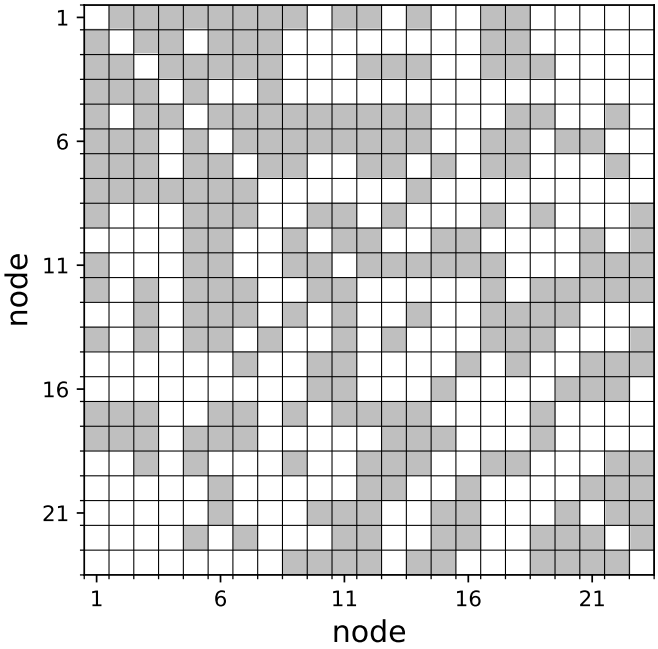


Fig. 9: The adjacency matrix generated after the completion of self-organization for the network of 23 nodes. The shaded area signify a direct connection.

between slot 40 and 100, these four nodes rarely obtained new information from their neighbors, even though they kept receiving packets. As a result, the update phase was extended.

Fig. 8 displays the number of slots the nodes utilized for each activities during self-organization. It can be seen that they spent much of the time listening (discovering). Idle time in this context signifies the time during which the nodes neither received nor transmitted packets because they had no new information to share. Idle slots slow down the process of establishing the topology of the network. We aim to minimize this effect in our future work.

At last, by merging the partially completed adjacency matrices from its neighbors, the initiator established the complete topology of the network as shown in Fig. 9. A shaded block in the figure represents the existence of a direct link between two nodes.

V. Related work

Self-organization is an essential aspect of any intelligent network. Often it is realized as a cross-layer feature involving the MAC and the routing layers, but also as a separate feature, independent of any communication protocol. Regardless of how it is implemented, it requires peer-to-peer communication, neighbor discovery, and in-network data aggregation. In this section, we briefly review two aspects of self-organization and how they have been addressed in the past.

A. Concurrent Transmissions

Chaos [17] is an all-to-all communication protocol which supports in-network processing. The communication and data processing activities are organized in synchronized slots. To

ensure concurrent transmissions, Chaos fixes the execution time in terms of MCU clock cycles, between the reception interrupt and the next transmission. During interaction, nodes should know other participants and maintain a determined flag set (one bit for a node). All in-network processing and communication progress depend on these flags.

Crystal [15] is a data-collection strategy for supporting an aperiodic communication. The sink node starts an epoch by flooding the network with a synchronization request. When the synchronization phase is completed, nodes begin interacting with one another in a sequence of paired slots, one for transmission and another for acknowledgment. If nodes have data packets to transmit, they will flood concurrently in a transmission slot and wait for acknowledgments in the consecutive slot. Nodes without packets to transmit can receive one of the data packets in the transmission slot taking advantage of the "capture effect". When a node receives a data packet, it will broadcast an acknowledgment packet in the next slot. An epoch is terminated in a distributed manner, if a node does not receive packets for a number of consecutive slots.

Mixer [12] is another communication protocol based on synchronous transmission. It exploits random linear network coding (RLNC) [14] to maximize the utility of packets in data dissemination. Nodes combine multiple packets they received from their neighbors using RLNC and transmit them synchronously. Receiving nodes can decode the original messages by receiving sufficient coding vectors. Unlike Chaos and its variants which can only share information in a static network, Mixer accommodates even mobile nodes. Likewise, Codecast [19] combines network coding with synchronous transmission to achieve many-to-many data sharing.

B. Topology Construction

The dynamic topology construction is vital to structure (cluster) the network and to optimally configure routing protocols. A3 (a tree) [23] is a topology construction protocol which is based on a growing-tree technique. After deployment, a predefined node initiates a neighbor-discovery phase by sending a hello message to all the nodes within its communication range. The nodes reply to this packet by acknowledging the transmitter as their parent node. After a short period of time, the initial sender broadcasts a sorted list of nodes whose parent recognition packets have been successfully received. This process will be propagated down-ward until all nodes are connected in the tree.

In [16], the authors proposed an expanded Borel Cayley graphs topology construction (EBTC) algorithm to generate a collision avoidance communication topology. In the beginning, an initial sender broadcasts a "hello" message to its logical neighbors. To avoid collision, each logical neighbor replies with a response message with their own neighbor lists in a fixed order, which is determined by the power index in the connection. After receiving response messages, the initial sender broadcasts a connection request containing an updated logical neighbor list. This process is repeated until all nodes found at least two logical neighbors.

In RPL routing protocol [9], the topology construction process is initiated by the root node which broadcasts a message whose rank increases as it propagated upward in the network (away from the root node). When other nodes receive this message, they calculate their ranks based on their relative position in the topology and then forward the message upward. Thus, the propagation of the message builds upwards connections since each node learns about its parent(s) from the message it receives. To construct the downwards connection graph, each node in the network sends unicast advertisement messages to its parents.

In [11], the authors propose an active neighbor discovery protocol to build 1-hop neighbor tables at different transmission power levels. A node first broadcasts neighbor discovery messages (NDM) with its own neighbor list at a specific power level and waits for neighbor reply messages (NRM) for a period of time. The sender ID of NRM will be added to its 1-hop neighbor list at the power level. The NDM and NRM information exchange is supposed to use TDMA protocol.

In [2], Chou et al. propose a distributed dead-end free topology maintenance protocol (DFTM). Initially, a randomly selected node is nominated as initiator and starts broadcasting a message to discover neighbors. Subsequently, the initiator transits into a receiving mode for a period time to receive replies from active neighbor nodes. It adds all the replied neighbor nodes to the active neighbor set and then select the ones which fulfill the specific criteria as the next initiators. The selected nodes continue the neighbor discovery phase until all nodes are discovered in the network.

In [21], the authors proposed a low-latency, energy-efficient neighbor discovery protocol. The neighbor discovery process is organized in time slots. In each slot, a node is in one of the three states, namely, transmit, listen or sleep. The transition to a state is probabilistic. If a node is in a transmit state in a slot, it transmits a message in which its ID is embedded, while other nodes which are in the listen state in the same slot can successfully receive the message and add the sender's ID into their neighbor list. The discovery process continues till no new neighbors are discovered.

Our approach complements existing work in three different ways. Firstly, the decision to transmit a packet depends not merely on the last reception round but on the previous receptions in a round. Secondly, concurrent transmissions are randomized into different slots to leverage the "capture effect" without compromising on packet transmission reliability. Thirdly, and, most importantly, our approach does not rely on a global knowledge such as the network size and topology.

VI. Conclusion

In this paper we proposed an algorithm to dynamically construct the topology of a wireless sensor network. Our algorithm does not assume the existence of a dedicated base station nor does it assume that the size of the network is known prior to deployment.

We represent the topology of a network with a binary adjacency matrix. This matrix is gradually completed as nodes propagate knowledge of their neighbors towards an initiator. To facilitate this process, we introduced four aspects: (1) random packet transmission, (2) implicit time synchronization,

(3) concurrent transmission and the "capture effect", and (4) in-network processing.

We implemented the algorithm in a Contiki environment and tested its performance with networks consisting of 8 and 23 nodes, respectively. In each case 100 independent experiments were conducted to collect adequate statistics. Evaluation results indicate that, regardless of the network size, the number of concurrent transmissions decreases exponentially in time, clearly indicating that the algorithm is scalable. For the network of 23 nodes, the algorithm was able to establish the complete adjacency matrix in 176 time slots. However, as the network size increased, concurrent transmission increased, and with it, the capture effect became less successful. When the network consisted of 8 nodes and the number of slots in a round was 4, 25% of the time packet transmissions occurred with at least 2 concurrent transmissions. When the network consisted of 23 nodes, concurrent transmission increased to 45%.

Our future plan is to enlarge the network size and to carry out field deployments. Our current deployments were limited by the underlying wireless local area network we established to collect statistical data from each sensor node. The range of the wireless router was limited, thereby limiting the size of the wireless sensor network. Work is in progress to address this concern. We are also working to integrate multiple UAVs with the wireless sensor network.

REFERENCES

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

[2] C.-H. Chou, K.-F. Ssu, H. C. Jiau, W.-T. Wang, and C. Wang. A dead-end free topology maintenance protocol for geographic forwarding in wireless sensor networks. *IEEE Transactions on computers*, 60(11):1610–1621, 2010.

[3] W. Dargie. A quantitative measure of reliability for wireless sensor networks. *IEEE Sensors Letters*, 3(8):1–4, 2019.

[4] W. Dargie and J. Wen. A simple clustering strategy for wireless sensor networks. *IEEE Sensors Letters*, 4(6):1–4, 2020.

[5] P. Di Marco, C. Fischione, F. Santucci, and K. H. Johansson. Modeling ieee 802.15. 4 networks over fading channels. *IEEE Transactions on Wireless Communications*, 13(10):5366–5381, 2014.

[6] A. Dunkels, B. Gronvall, and T. Voigt. Contiki - a lightweight and flexible operating system for tiny networked sensors. In *29th Annual IEEE International Conference on Local Computer Networks*, pages 455–462, 2004.

[7] M. Erdelj, M. Król, and E. Natalizio. Wireless sensor networks and multi-uav systems for natural disaster management. *Computer Networks*, 124:72–86, 2017.

[8] K. A. Farley, K. H. Williford, K. M. Stack, R. Bhartia, A. Chen, M. de la Torre, K. Hand, Y. Goreva, C. D. Herd, R. Hueso, et al. Mars 2020 mission overview. *Space Science Reviews*, 216(8):1–41, 2020.

[9] O. Gaddour and A. Koubâa. Rpl in a nutshell: A survey. *Computer Networks*, 56(14):3163–3178, 2012.

[10] C. Gezer, C. Buratti, and R. Verdone. Capture effect in ieee 802.15. 4 networks: Modelling and experimentation. In *IEEE 5th International Symposium on Wireless Pervasive Computing 2010*, pages 204–209. IEEE, 2010.

[11] J. Gui and J. Deng. A topology control approach reducing construction cost for lossy wireless sensor networks. *Wireless Personal Communications*, 95(3):2173–2202, 2017.

[12] C. Herrmann, F. Mager, and M. Zimmerling. Mixer: Efficient many-to-all broadcast in dynamic wireless mesh networks. In *Proceedings of the 16th ACM Conference on Embedded Networked Sensor Systems*, SenSys '18, page 145–158, New York, NY, USA, 2018. Association for Computing Machinery.

[13] D.-T. Ho, E. I. Grøtli, P. Sujit, T. A. Johansen, and J. B. Sousa. Optimization of wireless sensor network and uav data acquisition. *Journal of Intelligent & Robotic Systems*, 78(1):159–179, 2015.

[14] T. Ho, M. Médard, R. Koetter, D. R. Karger, M. Effros, J. Shi, and B. Leong. A random linear network coding approach to multicast. *IEEE Transactions on Information Theory*, 52(10):4413–4430, 2006.

[15] T. Istomin, A. L. Murphy, G. P. Picco, and U. Raza. Data prediction + synchronous transmissions = ultra-low power wireless sensor networks. In *Proceedings of the 14th ACM Conference on Embedded Network Sensor Systems CD-ROM*, SenSys '16, page 83–95, New York, NY, USA, 2016. Association for Computing Machinery.

[16] D. Kim, E. Noel, and K. W. Tang. Wsn communication topology construction with collision avoidance and energy saving. In *2014 IEEE 11th Consumer Communications and Networking Conference (CCNC)*, pages 398–404. IEEE, 2014.

[17] O. Landsiedel, F. Ferrari, and M. Zimmerling. Chaos: Versatile and efficient all-to-all data sharing and in-network processing at scale. In *Proceedings of the 11th ACM Conference on Embedded Networked Sensor Systems*, SenSys '13, New York, NY, USA, 2013. Association for Computing Machinery.

[18] J. Lee, W. Kim, S.-J. Lee, D. Jo, J. Ryu, T. Kwon, and Y. Choi. An experimental study on the capture effect in 802.11 a networks. In *Proceedings of the second ACM international workshop on Wireless network testbeds, experimental evaluation and characterization*, pages 19–26, 2007.

[19] M. Mohammad and M. C. Chan. Codecast: Supporting data driven in-network processing for low-power wireless sensor networks. In *2018 17th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*, pages 72–83, 2018.

[20] A. F. Molisch, K. Balakrishnan, C.-C. Chong, S. Emami, A. Fort, J. Karedal, J. Kunisch, H. Schantz, U. Schuster, and K. Siwiak. Ieee 802.15. 4a channel model-final report. *IEEE P802*, 15(04):0662, 2004.

[21] S. Pandey, P. Shukla, and A. Tripathi. An efficient group-based neighbor discovery for wireless sensor networks. In *2021 Third International Conference on Intelligent Communication Technologies and Virtual Mobile Networks (ICICV)*, pages 173–180. IEEE, 2021.

[22] J. Wen and W. Dargie. Evaluation of the quality of aerial links in low-power wireless sensor networks. *IEEE Sensors Journal*, 21(12):13924–13934, 2021.

[23] P. M. Wightman and M. A. Labrador. A3: A topology construction algorithm for wireless sensor networks. In *IEEE GLOBECOM 2008-2008 IEEE Global Telecommunications Conference*, pages 1–6. IEEE, 2008.

[24] C. Zhan, Y. Zeng, and R. Zhang. Energy-efficient data collection in uav enabled wireless sensor network. *IEEE Wireless Communications Letters*, 7(3):328–331, 2017.