Dynamic Topology Construction in a Joint Deployment

Jianjun Wen and Waltenegus Dargie, *Senior Member, IEEE*

***Abstract*— Self-organizing protocols and algorithms in wireless sensor networks rely on knowledge of the topology of the network they manage. In most practical cases, establishing the topology prior to actual deployment is not possible, as the exact placement of nodes and the existence of a reliable link between any two individual nodes cannot be guaranteed. Therefore, this task has to be carried out post deployment. If the network is stand-alone and certain aspects are fixed (such as the identity and placement of the base station), then 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, self-organization becomes challenging. 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 base station. A node carried by a UAV initiates the topology construction process and the ground nodes direct the network traffic towards the initiator. Our approach is collision-tolerant and enables in-network processing, thus reducing the computation burden of the initiator. We demonstrate the scope and usefulness of our approach using both lab and field deployments.**

***Index Terms*— capture-effect, field deployment, self-organization, topology, Wireless Sensor Networks (WSN), Unmanned Aerial Vehicles (UAV)**

I. INTRODUCTION

On February 18, 2021 NASA successfully landed a rover and an Unmanned Aerial Vehicle (UAV) on Mars, at a location believed to have astro-biological relevance. The duo are tasked with determining whether the planet was habitable in the past and whether life had thrived on it. An essential component of their assignment is the search for biosignatures within accessible geological materials [1]. The joint deployment is intended to fulfill complementary objectives. The rover is equipped with several advanced instruments for extracting and analyzing samples, but its scope is limited due to its limited maneuverability. The UAV hovers above the rover or nearby, thus surveying a wider region in a shorter time; its activity, nevertheless, is limited due to its limited energy reserve. Currently, it occasionally leaves the rover to make short flights.

A joint deployment can achieve a higher degree of spatio-temporal sensing if it involves an intelligent wireless sensor network (WSN) on the ground. The sensor nodes can save the rover from traveling long distances, and the UAV, from making long flights. This type of deployment has several applications here on earth as well, such as monitoring remote, dangerous, inaccessible, or extensive areas [2]–[4]. At the Energy Lab (TU Dresden, Germany), we investigate the practical use of such deployments and develop the communication protocols and algorithms they require.

In a joint deployment, a well-coordinated communication is indispensable to mitigate Cross Technology Interference (CTI); and the impact of adverse conditions and harsh weather [5]. However, determining the precise topology of the ground network – essential 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 nodes. Consequently, the topology of the network and the gateway nodes with which the UAV interacts 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 UAV to identify one or more ground nodes with which it conveniently interacts. The remaining nodes on the ground aggregate neighborhood information in the form of a partially completed adjacency matrix and propagate packets towards the UAV. Finally, the UAV establishes the complete binary adjacency matrix and represents the underlying topology of the network with it. The adjacency matrix will be the basis for identifying cluster heads and assigning child nodes to them. The key aspect of our algorithm – unlike competing approaches in the literature (for example, [6], [7]; refer also to Sec. II) – is a gradual topology construction process (in-network processing) which reduces the computation burden of the initiator and the communication cost of intermediate nodes.

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

Manuscript first submitted on 29 March 2022.

This work has been partially funded by the German Research Foundation (DFG) in the context of the RoReyBan project (DA 1121/7-1).

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

II. Related work

Self-organization is an essential aspect of any resilient network [8], [9]. 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 [10]–[12]. Regardless of how it is implemented, it requires peer-to-peer communication, neighbor discovery, and in-network data aggregation [13], [14]. In this section, we review two aspects of self-organization and how they have been addressed in the past.

A. Concurrent Transmissions

A work closer to ours is the one proposed by Ma et al. [7]. The approach avoids the use of any routing protocol or a global data collection scheduling. Instead, nodes flood the network with packets whenever they wish to update their status or communicate some sensed data. Intermediate nodes transmit packets with the highest transmission power possible to take advantage of the "capture-effect" (to be explained in Sec. IV). It is assumed that short packets (a few bytes long) are adequate for many practical purposes. Packets are cascaded (packets are put inside packets) as they propagate towards the UAV. The authors claim to have achieved a superior collection efficiency – requiring 287 ms to complete data collection from a network of 26 ground nodes (ca. 1.8 times faster than state-of-the-art) and 965 ms from a network of 42 ground nodes (ca. 1.1 times faster than state-of-the-art). Similarly, Mohammad and Chan [6] propose an algorithm in which packet transmission is decentralized and takes place in time slots, the beginning and end of which is predetermined. In other words, nodes are implicitly synchronized and the "capture-effect" is exploited to recover corrupted packets. Furthermore, packet transmission is probabilistic (a node may or may not transmit a packet in a given time slot). Nodes spend much of the time listening. In order to reduce the network traffic, nodes aggregate data using network coding. Here the proposed approach is used for data collection whereas our goal is to establish the topology of a network (i.e., the algorithm runs only once, or occasionally, whenever an update is required). For the approach to be effective, an advanced computing platform is required to support network coding (for, otherwise, a simple network coding does not take advantage of spatial and temporal data correlations).

In [15], a UAV is tasked with collecting data from a ground wireless sensor network in a collision-tolerant manner. To exploit the "capture-effect", the UAV first transmits sample signals from multiple known locations; nodes receiving these signals respond simultaneously. The analysis of the received signal leads to the estimation of (1) the transmission timing and (2) the transmission power required to recover packets upon collision. Similarly, Crystal [16] 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 the 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.

B. Topology Construction

Topology construction is vital to structure (cluster) the network and to optimally configure routing protocols. The approaches closer to ours are the ones proposed by [17] and [18]. Both rely on the existence of an adjacency matrix describing physical connectivity. However, in both cases, a physical connectivity does not necessarily correspond to a logical connectivity. Instead, a logical topology is constructed on top of the physical topology. In the first, the aim is to prolong the network's lifetime by achieving energy efficiency; whereas in the second, the aim is to make the network robust against external attacks. Thus, in [17] a particle swarm optimization algorithm is employed to set up a cluster tree topology. To this end, the topology construction problem is translated into an energy consumption optimization problem. Similarly, in [18], the physical topology of the network is transformed into one which withstands external attacks. It is assumed that nodes with high degree are susceptible. The proposed approach first establishes the physical topology of the network and computes the rank (relative significance) of individual nodes. The topology is then fed to a genetic algorithm which performs crossover and mutation operations. The first refers to the process of pairing parent nodes to produce the next generation child nodes and the second, to the transformation of child nodes in a statistical sense, so that the resulting topology offers diversity of communication.

In [19], the authors propose an active neighbor discovery protocol to build single-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 awaits 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 exchange of NDM and NRM information takes place using a TDMA protocol. Similarly, in [20], the authors propose a low-latency, energy-efficient neighbor discovery protocol which is organized in time slots. In each slot, a node is in one of the three modes, namely, transmission, listening or sleeping. The transition to a mode is probabilistic. If a node is in a transmission mode in a slot, it transmits a message in which its ID is embedded, while other nodes which are in the listening mode in the same time slot can successfully receive the message and add the sender's ID to their neighbor list. The discovery process continues until no new neighbors are discovered.

In [21], the authors propose an expanded Borel-Cayley graph 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 responds

with a message containing their own neighbor lists in a fixed order, determined by the power index in the connection. After receiving these responses, 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 [22], the topology construction process is initiated by the root node which broadcasts a message whose hop-count increases as it propagates upward in the network (away from the root node). When intermediate nodes receive this message, they calculate their rank based on their relative distance to the root node and forward the message upward. Thus, the propagation of the message builds upward-connections since each node learns about its parent(s) from the message it receives. To construct the downward connections graph, each node in the network sends unicast advertisement messages to its parents. RPL, due to its simplicity, is a widely used data collection strategy in the community.

In [23] the flight route of a UAV is optimized for collecting data from a wireless sensor network. The task is formulated as a non-convex optimization problem. The optimal hovering locations are those interfacing the UAV with nodes whose residual energy is the least in the network. At each location of the UAV a Voronoi vertex [24] is formed so that the UAV can collect data from as many adjacent sensors as possible. Likewise, in [25], an optimal data collection strategy is proposed. Accordingly, a ground wireless sensor network is clustered into multiple clusters using an integer linear programming and a data aggregation/compression path is defined to establish an aggregate tree connecting child nodes with the cluster heads. The linear programming operates on an underlying binary matrix representing the topology of the network.

Our approach differs from existing work in three different ways. Firstly, the decision to transmit a packet during information exchange 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 knowledge about the size of the network or the topology of the network[1].

III. Background

Cluster-based network organization facilitates data collection and command dissemination [27]. Most existing clustering algorithms require some qualification criteria to identify potential cluster heads. In [28], we propose a model for quantifying 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 multi-hop links the nodes establish with their peers can be estimated as follows:

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

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

$$\mathbf{r} = \mathbf{u}\mathbf{T} \tag{3}$$

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. Quantifying the relative significance of the nodes thus enables the identification of strategic nodes within the network which are critical to disseminate commands and aggregate data.

In [29], we propose a clustering algorithm which dynamically identifies cluster heads based on their relative significance and relative hop-distance. Furthermore, the algorithm associates child nodes with the cluster heads, taking into account their placement relative to the cluster heads. The algorithm is useful for coordinating communication during joint deployments, as it enables a UAV to 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. The purpose of this paper is to address this concern. In Section IV, we propose an efficient and collision-tolerant topology construction algorithm to generate a binary adjacency matrix.

IV. Concept

We assume that (1) a single UAV interacts with a wireless sensor network and (2) the predominant traffic flow is directed towards the UAV. Since the specific hovering spots of the UAV and its flight route are determined by external conditions (the weather, the physical objects present in the deployment field, etc.), the ground network should be able to redirect the traffic flow to suit the present context of the UAV. However, the deployment lifetime is typically short, owing to the fact that the UAV operates with exhaustible batteries. Consequently, we also assume that while the UAV is flying, the topology of the ground network remains, by and large, stable.

In order to illustrate our approach, suppose we have a joint deployment consisting of five nodes (refer to Fig. 1). Node A represents the UAV and the others are ground nodes. By visually inspecting the topology of the network, it is possible to rank the nodes according to their relative connectivity. 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. The adjacency matrix in Equation 4 expresses the physical topology of the network. Applying Equation 3 on the adjacency matrix results in a

[1]But refer also to [26] where underwater robots with limited power self-organize without requiring a global knowledge.

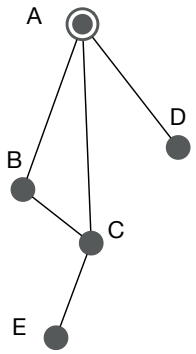


Fig. 1: A simple network topology consisting of five nodes.

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

TABLE I: Rank of nodes

quantitative rank of the nodes (refer to Tab. I). Accordingly, next to the UAV, the most important node in the network is node C.

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

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 or their transmission path 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 is initiated by the UAV (or a sensor node it carries). 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, update a local list of neighbors, and generate a partially complete adjacency matrix. In the second phase, the *report* phase, nodes propagate packets containing information about their neighbors towards the initiator. In order to ensure an efficient message dissemination, we define the following flags: Start (**S**), Update (**U**), Forward (**F**), and Report (**R**). Furthermore, 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 point in time.

The discovery phase runs for k rounds. In each round there are exactly N number of 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 (the UAV) 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 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 simultaneously 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 further 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 do not rebroadcast these packets.

B. In-Network Data Aggregation

The adjacency matrix can be constructed in one of the following ways: either (1) 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, the 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 computational point of view, the whole process overwhelms the initiator, thereby resulting in a disproportionate amount of

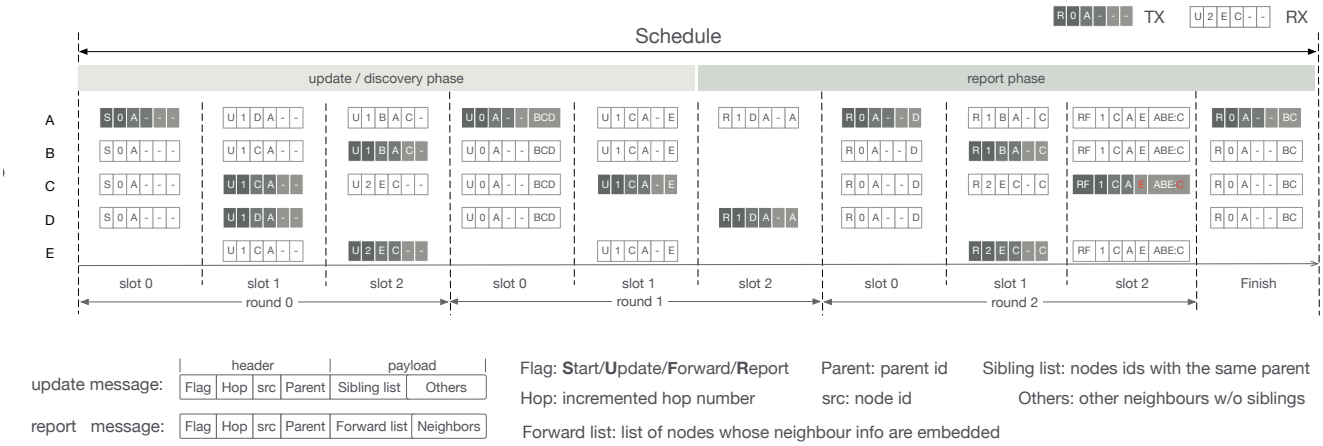


Fig. 2: An illustration of a topology discovery process for the network displayed in Fig. 1. Dark boxes imply packet transmission and light boxes imply packet reception.

energy being expended 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 constructed by Node E in 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 with two lists. The first list contains the node's ID and the second list contains whole numbers indicating the cell's address containing 1s. For the above example, the first list contains the IDs of Node C and Node D and the second list contains the numbers 2 and 3 as these cells contain 1s. Any adjacency matrix can be represented by these two lists, making the transmission and decoding of the adjacency matrix straightforward[2]. Likewise, the adjacency matrix Node C constructs resembles the following:

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

Accordingly, the first list contains the IDs of Node A, B, C, and E (so that the adjacency matrix is a 4×4 matrix) and the second list contains the numbers: 5, 7, 9, 10, 12, 15. Note that the matrix does not accurately reflect the topology yet. Node C may know that Node A is its neighbor (assuming it first received a discovery packet from it); this does not, however, necessarily mean that Node A knows that Node C is its neighbor. Node A may have to wait for the *report* phase to discover this. This is why $c_{13} = 0$ even though $c_{31} = 1$.

[2]Note that the size of the first list implicitly reveals the dimension of the adjacency matrix. Moreover, an adjacency matrix having $n \times n$ dimensions can be encoded with less than $n^2/2$ bytes. This is because we can choose to encode either the 0s or the 1s, whichever have the smallest count.

C. Implicit Time Synchronization

The condition to minimize and manage collision is a synchronized concurrent transmission. The nodes implicitly synchronize time by estimating the time at which a parent node started the current transmission. This time corresponds to the beginning of a current time slot the length of which is fixed. From the point in time 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 microcontroller unit (MCU) following the successful detection of a 1 byte *Start of Frame Delimiter* (**SFD**) at the physical layer (by the radio chip). A local timer registers this time (t_s). Before the SFD can be detected, a 4 byte preamble must be 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 the probability of collision, but also the longer the time needed to establish the topology of the network (i.e., the adjacency matrix). We exploit a phenomenon called the "capture-effect" [30]–[32], a well-established 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 specification [33]–[35], 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).

Algorithm 1: Transmission Policy

Input : received message: msg
Output: transmission state machine $State_{tx}$
Phase *Update*:
  if *msg.src not in LNS*[3] && *my_id not in msg.neighbor_set* **then**
    $State_{tx} = State_{tx} + 1$;
  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

The chance of two or more nodes transmitting simultaneously 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.

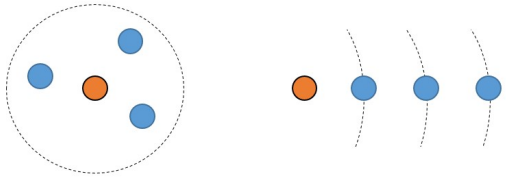


Fig. 3: An illustration of how the topology of a network affects the complexity of self-organization.

E. Complexity

The participation of nodes in self-organization incurs communication and computational burden on intermediate nodes. The size of the packet they receive and forward depends on both their relative location (depth or hop-distance) and the size of the network. The more hierarchically the nodes are structured, the higher is the complexity of the algorithm. Fig. 3 illustrates this feature. In the left, the network is flat and the nodes can directly send packets to the initiator (the golden node in the middle), whereas in the right, intermediate nodes have to be involved to forward packets from leaf nodes.

PIP [7] encloses packet in packets. Thus, the size of a packet grows exponentially as it propagates towards the initiator. If a node is d-hop away from the initiator, the packet, p, it forwards will have a size of p^d by the time it reaches the initiator. For our case, the same intermediate node increases its packet size by ca. $2(d_{max} - d) + 1$ bytes, where d_{max} is the maximum hop count in the network. The reason is the following: A node $(d + 1)$-hop away from the initiator will construct a partially completed adjacency matrix having dimensions of ca. $(d_{max} - d) \times (d_{max} - d)$. The diagonal elements of this matrix are all zeros and need not be included in the coding process. In addition, the adjacency matrix will contain a significant number of zeros (unless the network is a fully-meshed network). Similarly, a node which is d-hop away will have an adjacency matrix of dimensions: $(d_{max} - (d - 1)) \times (d_{max} - (d-1))$. The dimensions of the adjacency matrix increase as the hop-count decreases. The computational burden a node in layer $(d + 1)$ introduces to a node in layer d is the difference of the two dimensions, which is $2(d_{max} - d) + 1$. In other words, the size of a packet increases linearly as it propagates towards the initiator. Generally, assuming the nodes are evenly distributed in the network, the complexity of our algorithm is in the order of $\mathcal{O}(d_{max} + n/d_{max})$.

V. Evaluation

Our algorithm was implemented for the Contiki operating system [36] and the Zolertia platform (RE-Motes[4]). The sensor platform integrates two IEEE 802.15.4-compliant transceivers

[3]LNS: Local Neighbor Set.

[4]https://zolertia.io/product/re-mote/

Fig. 4: Deployments of wireless sensor networks inside a lab and outdoors (in a field next to a modest forest).

TABLE II: Deployment parameters.

Parameters	Value
Network size	8, 21
Slot length (T_{slot})	10 ms
Slots in a round ($nslot$)	2, 4, 8
Schedule duration	30 s
Transmission power	7 dBm (field),-15 dBm (lab)

working in the 863-950 MHz and 2.4 GHz radio bands. In all our experiments, we employed the 2.4 GHz radio. Each sensor node was connected to a Raspberry PI board via a USB cable to establish a wireless local area network for controlling the experiments and for collecting performance related statistics. The sensor nodes were deployed in our lab as well as outdoors, in a field next to a modest forest (ref. to Fig. 4). The lab and outdoor deployments consisted of 20 and 8 ground nodes, respectively. The size of the outdoor deployment was limited by the coverage of a battery-powered wireless router we used to establish the backbone network. In both deployments, a DJI Mavic-2 Enterprise drone[5] hovered above the ground network carrying an additional sensor node. This node, acting as a gateway node to a remote server, initiated the topology reconstruction process, and established the final adjacency matrix. Hence, the network traffic was directed towards it. The UAV was maneuvered by a remote controller which employed a proprietary long range transmission, operating in 2.4 and 5.8 GHz radio bands (OcuSync 2.0 [37]). When operating in 2.4 GHz, it produced a considerable CTI. Because the location of the UAV was determined by several physical factors, the node on the UAV was able to establish connections with any of the ground nodes nearby.

We evaluated our algorithm in terms of:

- The average time it required to establish the network's topology.
- The portion of packets the nodes successfully transmitted.
- The average number of concurrent transmissions in a slot.
- The impact of the number of slots in the discovery and report phases on the job completion time.

A. Outdoor Deployment

In this deployment, the nodes on the ground established a grid topology and the UAV hovered approximately 30 m above the ground, approximately in the middle of the network. In this deployment, the aerial node was designated as Node 5. We repeated our experiment 100 times.

1) Job Completion Time: The job completion time refers to the time the algorithm needs to establish the topology of a network. When it refers to an individual node (except the initiator), it refers to the time the node requires to successfully construct and transfer a partially completed adjacency matrix. When it refers to the initiator, the job completion time is the same as the time the algorithm requires to establish the complete adjacency matrix).

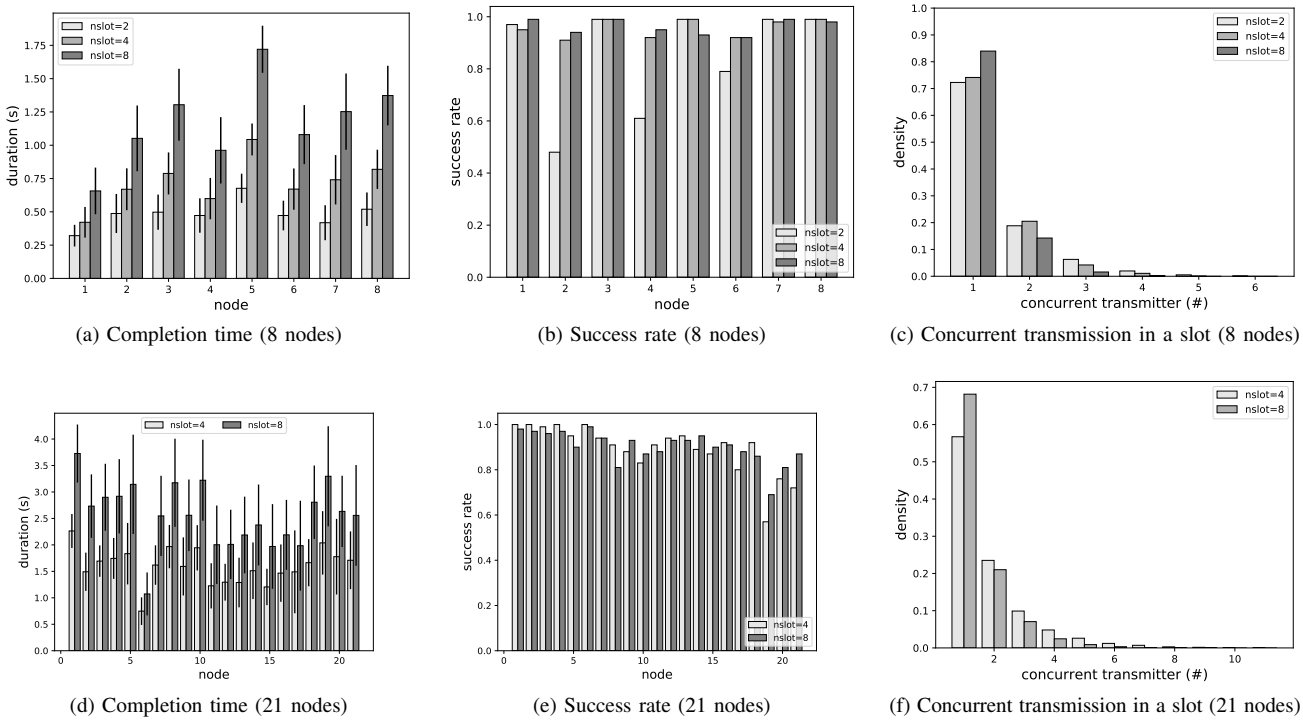
Fig. 5 (a) shows the average job completion time for the individual nodes. The error bar shows the standard deviation. The job completion time increased as the number of slots in a round increased. However, the increment was not linear. For example, when the number of slots in a round increased from 2 to 4, the job completion time did not double. The reason being that as the number of slots increased, the possibility of experiencing concurrent transmissions decreased as well. In other words, the probability of receiving a packet successfully increased. At the same time, when the number of slots in a round increased, the possibility of a node gathering neighbor information increased, as a result of which the duration of the update phase was shortened.

2) Packet Success Rate: The packet success rate refers to the portion of packets a node successfully transmits to its neighbors during the discovery and report phases. Fig. 5 (b) displays this metric for each sensor node when the number of time slots in a round was 2, 4, and 8, respectively. We observed that when the number of time slots was 2, the success ratio of three nodes (node 2, 4, 6) was below 0.8. When we increased the slot size to 4, the success rate exceeded 0.9. When, however, the number of slots increased to 8, there was no appreciable increase in the packet success rate.

3) Number of Concurrent Transmission: Fig. 5 (c) shows the distribution of concurrent transmissions in both the discovery and report phases. When the number of slots in a round increased, the number of concurrent transmissions decreased. Overall, more than 70% of the slots experienced a single transmission only. The factors which contributed to this figure were the following:

- The random selection of a transmission slot.
- The in-network strategy and the gradual aggregation of adjacency matrices, enabling nodes to indirectly learn about their neighbors.
- The concurrent transmissions occurred in the early stages of the algorithm. The probability of receiving new packets

[5]https://www.dji.com/mavic-2-enterprise



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

(d) Completion time (21 nodes) (e) Success rate (21 nodes) (f) Concurrent transmission in a slot (21 nodes)

Fig. 5: Performance evaluation based on different schedule parameters for two different network sizes.

from unknown neighbors decreased over time (exponentially, as can be seen in the figure).

B. Indoors Deployment

The deployment in our lab enabled us to increase the network size to 21, since we could use multiple access points for establishing the backbone WLAN. As previously, we conducted 100 independent experiments, but this time we limited the number of time slots in a round to 4 and 8.

As in the previous case, the job completion time increased as we increased the number of slots in a round. Furthermore, the increment was not linear. Whereas the network size was nearly tripled, the job completion time increased only by twofold in both settings (4 and 8), as shown in Fig. 5 (a) and (d). By comparison, the packet success rate deteriorated noticeably, as can be seen in Fig. 5 (b) and (e). Understandably, as the network size increased, the number of concurrent transmissions 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, ca. 25% of the time packet transmission occurred with at least 2 concurrent transmissions, whereas the number of concurrent transmissions increased to ca. 47% when the network size increased to 21.

C. Real-time Activity

Next we analyzed the activity of individual nodes during self-organization. The data trace for this section was obtained from the lab deployment, when the number of slots in a round was 4. The result is shown in Fig. 6. The time axis is represented in terms of the number of slots. For this particular experiment, the complete process lasted 165 slots (i.e., 1650 ms).

In Sec. V-A.3, we observed that intensive concurrent transmissions occur in the early stages of the discovery phase. This aspect can be seen in Fig. 7 (left) where significant concurrent transmissions occurred in the first 30 slots, during which nodes discovered new neighbors and were busy sharing this information. As time went by, the frequency of discovering new neighbors decreased and the nodes had little to share.

Fig. 7 (middle) shows the average duration the nodes spent in the two phases (discovery and report). We observed that more than half of the nodes (13 nodes, to be exact) completed the update phase within 60 slots (600 ms). Only 4 of the nodes required more than 100 slots for the update phase. By looking at the activity of these nodes in Fig. 6, one can observe that between slots 40 and 100 the four nodes rarely obtained new information from their neighbors even though they kept on receiving packets. As a result, the update phase was extended.

Fig. 7 (right) displays the number of slots the nodes spent in the discovery and update phases. A considerable amount of time was spent in a listening (discovering) state. An idle time in this context signifies a state in which the nodes neither received nor transmitted packets because they had no new information to share. Idle time slots slow down the topology construction process. We aim to minimize this duration in a future work.

At last, by merging the partially completed adjacency matrices from the ground nodes, the initiator established the complete topology of the network as shown in Fig. 8. A shaded block in the figure represents the existence of a direct link between two nodes.

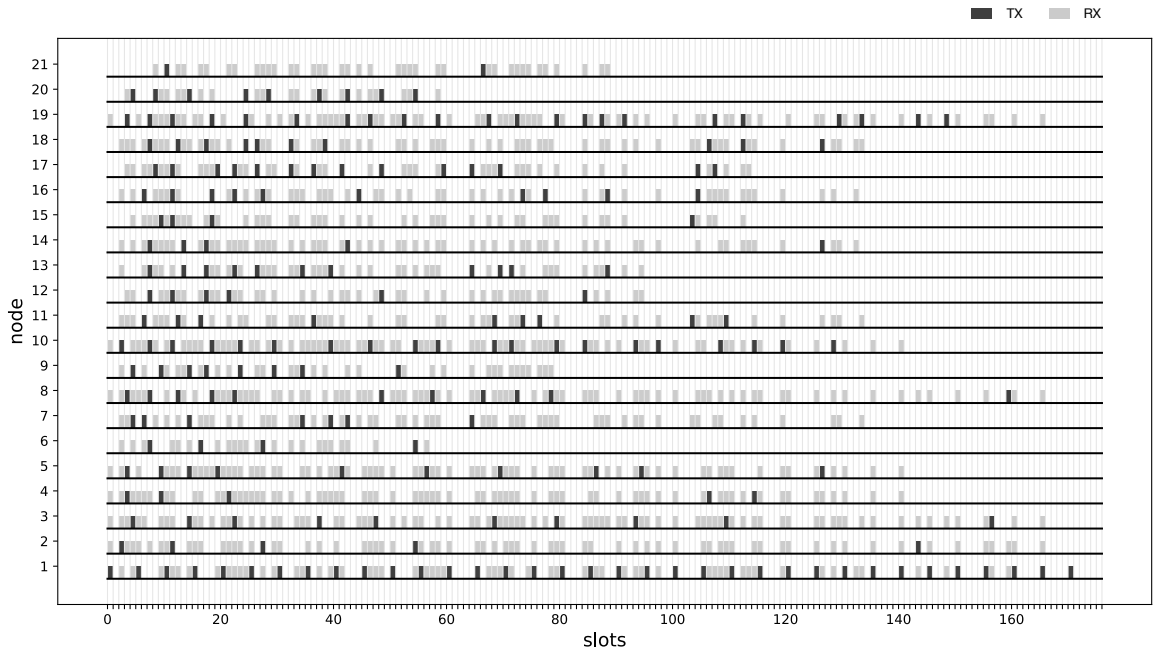
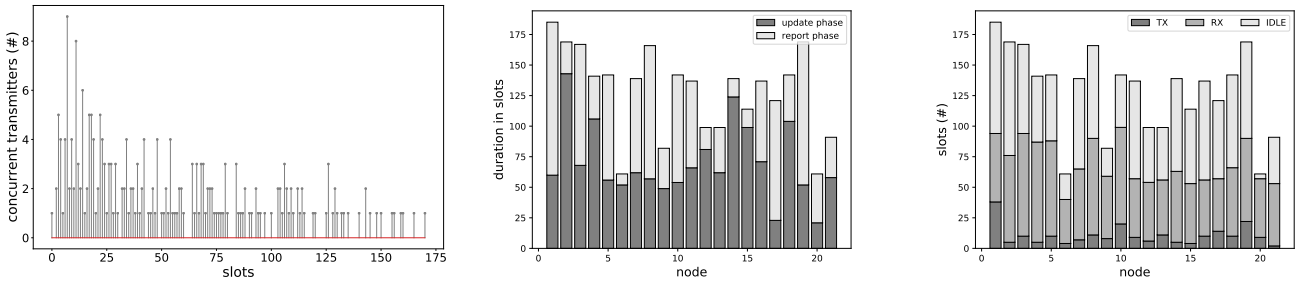


Fig. 6: The trace of the real-time activity of the nodes during self-organization. The network consisted of 21 nodes. Node 1 was the initiator.



(a) Pattern of concurrent transmission.

(b) Activity of nodes in discovery and report phases.

(c) Activity of nodes in each time slot.

Fig. 7: Performance evaluation of the indoor deployment.

D. Comparison

In [7] the authors report that their algorithm required 287 ms to complete data collection from a network of 26 ground nodes (according to their claim this is 1.8 times faster than state-of-the-art). By comparison, our algorithm required approximately 1.65 s to complete data collection from a network of 21 nodes. Our algorithm appears to be slower than PIP, but for our case, the packets are partially processed, whereas in PIP, intermediate nodes merely cascade packets, leaving it to the sink to unpack and process them. In this section, we shall analytically compare the communication cost of the two algorithms. For our analysis we adopt a deployment strategy resembling the distribution of electrodes around the nucleus of an atom. The UAV hovers in the middle of the deployment field. On the ground, there are $n = 2d^2$ nodes which are d-hop away from the UAV (we label these nodes as layer d nodes). For both algorithms the MAC header is 9 bytes. To simplify our analysis, we assume that a node in layer d does not receive duplicate messages from the nodes in layer $d + 1$.

To calculate the cost of communication, we referred to the technical specification of the RE-Mote. The node is based on Texas Instruments CC2538 System-On-Chip microcontroller. The 2.4 GHz radio consumes 24 mA when receiving at –100 dBm input power; and 34 mA when transmitting at 7 dBm output power. Its supply voltage can be varied between -0.3 and 3.9 V, but during all our experiments it was set to 3.9 V. Its nominal transmission rate is 250 Kbps.

In PIP, at each layer d we have a minimum transmission cost of:

$$Tx(d) = 2d^2 Tx \tag{11}$$

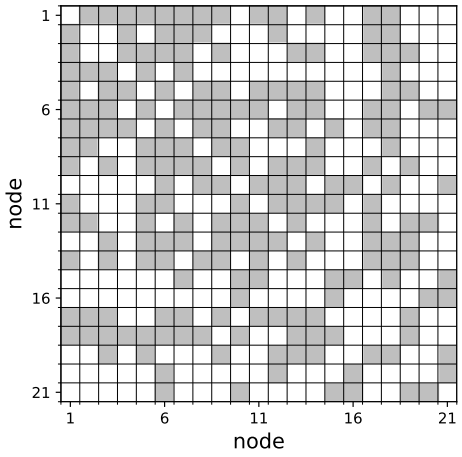


Fig. 8: An adjacency matrix generated after the completion of a self-organization phase. The shaded area signifies the presence of a direct link. a_{ii} in the matrix is by default set to zero.

TABLE III: Assumption about the distribution of nodes (**n**) in each layer (**d**) of the deployment field.

n	25	50	75	100	125	150	175	200
d_{max}	3	4	5	5	6	6	7	7

where Tx is the energy cost of transmitting a single packet. Moreover, the nodes in layer d receive packets from the nodes in layer $d+1$ in order to forward them towards the center. The corresponding cost is:

$$Rx(d) = 2(d+1)^2(d_{max} - d)Rx \tag{12}$$

where d_{max} is the maximum hop distance from the UAV and Rx is the cost of receiving a single packet. The cost of forwarding the packets is:

$$F(d) = 2(d+1)^2(d_{max} - d)Tx, \ d < d_{max} \tag{13}$$

The overall communication cost of the nodes in layer d is $Tx(d) + Rx(d) + F(d)$.

For our case, too, the size of the adjacency matrix encoding neighbor information increases as it advances towards the UAV. If we assume that at least half of the partially completed adjacency matrix consists of 0s, the cost of receiving packets from the nodes in layer $d+1$ is:

$$Rx(d) = (d+1)^2 \left((d_{max} - d)^2 \rho_x + 2Rx_m\right) \tag{14}$$

where ρ_x is the cost of receiving a single byte and Rx_m is the cost of receiving the MAC header. Similarly, the transmission cost of the nodes in layer d is:

$$Tx(d) = d^2 \left((d_{max} - d + 1)^2 \tau_x + 2Tx_m\right) \tag{15}$$

where τ_x is the cost of transmitting a single byte and Tx_m is the cost of transmitting the MAC header. The overall communication cost of the nodes in layer d is therefore $Rx(d) + Tx(d)$.

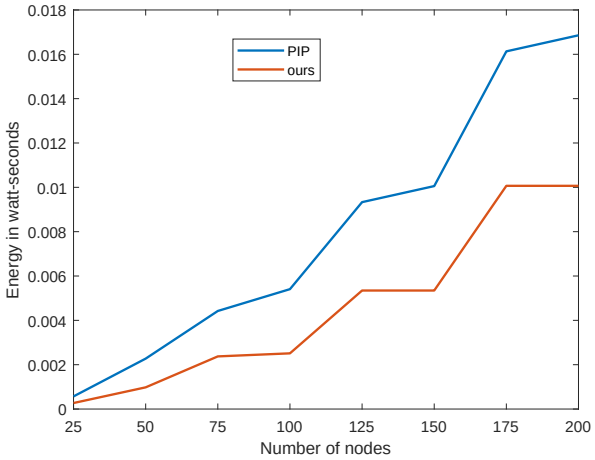


Fig. 9: Comparison between the overall energy cost of PIP and our algorithm.

Fig. 9 compares the communication cost of PIP and our algorithm for eight different network sizes. Tab. III shows the nodes' distribution. It can be seen from the figure that PIP's fast job completion time comes with a high energy cost.

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 (a UAV). To facilitate this process, we introduced four aspects: (1) random packet transmission, (2) implicit time synchronization, (3) concurrent transmission and the exploitation of the "capture-effect", and (4) in-network processing.

We implemented the algorithm for the Contiki environment and tested its performance with networks consisting of 8 and 21 sensor nodes. In each case 100 independent experiments were conducted to collect adequate statistics. Our evaluation indicates that, regardless of the network size, the number of concurrent transmissions decreases exponentially over time, clearly indicating that the algorithm is scalable. For the network of 21 nodes, the algorithm was able to establish the complete adjacency matrix in 1.65 s. 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, approximately 25% of the time packet transmissions occurred with at least 2 concurrent transmissions. When the network consisted of 21 nodes, concurrent transmission increased to 47%. Our future plan is to enlarge the network size and carry out more extensive field experiments.

References

[1] 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*, vol. 216, no. 8, pp. 1–41, 2020.

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

[3] C. Zhan, Y. Zeng, and R. Zhang, "Energy-efficient data collection in uav enabled wireless sensor network," *IEEE Wireless Communications Letters*, vol. 7, no. 3, pp. 328–331, 2017.

[4] 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*, vol. 78, no. 1, pp. 159–179, 2015.

[5] J. Wen and W. Dargie, "Evaluation of the quality of aerial links in low-power wireless sensor networks," *IEEE Sensors Journal*, vol. 21, no. 12, pp. 13 924–13 934, 2021.

[6] 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)*, 2018, pp. 72–83.

[7] X. Ma, P. Zhang, O. Theel, and J. Wei, "Gathering data with packet-in-packet in wireless sensor networks," *Computer Networks*, vol. 170, p. 107124, 2020.

[8] S. Qu, L. Zhao, and Z. Xiong, "Cross-layer congestion control of wireless sensor networks based on fuzzy sliding mode control," *Neural Computing and Applications*, vol. 32, no. 17, pp. 13 505–13 520, 2020.

[9] X. Fu, P. Pace, G. Aloi, L. Yang, and G. Fortino, "Topology optimization against cascading failures on wireless sensor networks using a memetic algorithm," *Computer Networks*, vol. 177, p. 107327, 2020.

[10] S. Hu and G. Li, "Fault-tolerant clustering topology evolution mechanism of wireless sensor networks," *IEEE Access*, vol. 6, pp. 28 085–28 096, 2018.

[11] B. Bhushan and G. Sahoo, "Routing protocols in wireless sensor networks," in *Computational intelligence in sensor networks*. Springer, 2019, pp. 215–248.

[12] H. Oh and C. T. Ngo, "A slotted sense multiple access protocol for timely and reliable data transmission in dynamic wireless sensor networks," *IEEE Sensors Journal*, vol. 18, no. 5, pp. 2184–2194, 2018.

[13] Y. Chang, X. Yuan, B. Li, D. Niyato, and N. Al-Dhahir, "A joint unsupervised learning and genetic algorithm approach for topology control in energy-efficient ultra-dense wireless sensor networks," *IEEE Communications Letters*, vol. 22, no. 11, pp. 2370–2373, 2018.

[14] T. Qiu, J. Liu, W. Si, and D. O. Wu, "Robustness optimization scheme with multi-population co-evolution for scale-free wireless sensor networks," *IEEE/ACM Transactions on Networking*, vol. 27, no. 3, pp. 1028–1042, 2019.

[15] A. Farajzadeh, O. Ercetin, and H. Yanikomeroglu, "Mobility-assisted over-the-air computation for backscatter sensor networks," *IEEE Wireless Communications Letters*, vol. 9, no. 5, pp. 675–678, 2020.

[16] 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*, ser. SenSys '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 83–95. [Online]. Available: https://doi.org/10.1145/2994551.2994558

[17] Y. Yu, B. Xue, Z. Chen, and Z. Qian, "Cluster tree topology construction method based on pso algorithm to prolong the lifetime of zigbee wireless sensor networks," *EURASIP Journal on Wireless Communications and Networking*, vol. 2019, no. 1, pp. 1–13, 2019.

[18] T. Qiu, J. Liu, W. Si, and D. O. Wu, "Robustness optimization scheme with multi-population co-evolution for scale-free wireless sensor networks," *IEEE/ACM Transactions on Networking*, vol. 27, no. 3, pp. 1028–1042, 2019.

[19] J. Gui and J. Deng, "A topology control approach reducing construction cost for lossy wireless sensor networks," *Wireless Personal Communications*, vol. 95, no. 3, pp. 2173–2202, 2017.

[20] 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)*. IEEE, 2021, pp. 173–180.

[21] 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)*. IEEE, 2014, pp. 398–404.

[22] O. Gaddour and A. Koubâa, "Rpl in a nutshell: A survey," *Computer Networks*, vol. 56, no. 14, pp. 3163–3178, 2012.

[23] J. Baek, S. I. Han, and Y. Han, "Energy-efficient uav routing for wireless sensor networks," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 2, pp. 1741–1750, 2020.

[24] X. Li, A. Krishnamurthy, I. Hanniel, and S. McMains, "Edge topology construction of voronoi diagrams of spheres in non-general position," *Computers & Graphics*, vol. 82, pp. 332–342, 2019.

[25] D. Ebrahimi, S. Sharafeddine, P.-H. Ho, and C. Assi, "Uav-aided projection-based compressive data gathering in wireless sensor networks," *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 1893–1905, 2019.

[26] J. Kim, "Three dimensional distributed rendezvous in spherical underwater robots considering power consumption," *Ocean Engineering*, vol. 199, p. 107050, 2020.

[27] E. E. Tsiropoulou, S. T. Paruchuri, and J. S. Baras, "Interest, energy and physical-aware coalition formation and resource allocation in smart iot applications," in *2017 51st Annual Conference on Information Sciences and Systems (CISS)*. IEEE, 2017, pp. 1–6.

[28] W. Dargie, "A quantitative measure of reliability for wireless sensor networks," *IEEE Sensors Letters*, vol. 3, no. 8, pp. 1–4, 2019.

[29] W. Dargie and J. Wen, "A simple clustering strategy for wireless sensor networks," *IEEE Sensors Letters*, vol. 4, no. 6, pp. 1–4, 2020.

[30] W. Wang, T. Xie, X. Liu, and T. Zhu, "Ect: Exploiting cross-technology concurrent transmission for reducing packet delivery delay in iot networks," in *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications*, 2018, pp. 369–377.

[31] 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*, 2007, pp. 19–26.

[32] 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)*. IEEE, 2017, pp. 1–5.

[33] 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*, vol. 15, no. 04, p. 0662, 2004.

[34] 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*. IEEE, 2010, pp. 204–209.

[35] 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*, vol. 13, no. 10, pp. 5366–5381, 2014.

[36] 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*, 2004, pp. 455–462.

[37] B. Kyle, "Dji ocusync vs. dji lightbridge – what's the difference?" 2018. [Online]. Available: https://dronedj.com/2018/07/27/dji-ocusync-vs-lightbridge/