applied sciences

MDPI

Article

Enabling Autonomous Agents for Mobile Wireless Sensor Networks

José-Borja Castillo-Sánchez [1,*], José-Manuel Cano-García [1], Eva González-Parada [1] and Mirgita Frasheri [2]

[1] Department of Electronic Technology, Telecommunications Research Institute, University of Malaga, 29071 Malaga, Spain; jcgarcia@uma.es (J.-M.C.-G.); gonzalez@uma.es (E.G.-P.)
[2] Centre for Digitalisation, Big Data & Data Analytics (DIGIT), Department of Electrical and Computer Engineering, Aarhus University, 8200 Aarhus, Denmark; mirgita.frasheri@ece.au.dk
* Correspondence: jocasa@uma.es

Abstract: Wireless sensor networks (WSNs) play a pivotal role in monitoring and acting applications. However, suboptimal deployments and traffic imbalances lead to rapid network exhaustions. To address this, topology changes could be carried out by mobile robots. In this work, a software package to study different strategies and algorithms for the deployment, operation, and retrieval of mobile WSN is introduced. This package employs the globally known software ecosystem for robotics, ROS (Robot Operating System) 2, allowing to study the above-mentioned strategies and algorithms in simulation or in actual deployments. Two strategies concerning robot control are compared, the Social Potential Fields-only approach and an intelligent Agent layer. Each strategy is tested and optimized with different parameters. Results show that the Agents approach yields more consistent results and globally better metrics in terms of network lifetime and coverage.

Keywords: mobile wireless sensor networks; ROS 2; mobile robots; networking; network simulation; ad hoc networks

Academic Editor: Suchao Xie

Received: 11 April 2025
Revised: 26 May 2025
Accepted: 27 May 2025
Published: 30 May 2025

Citation: Castillo-Sánchez, J.-B.; Cano-García, J.-M.; González-Parada, E.; Frasheri, M. Enabling Autonomous Agents for Mobile Wireless Sensor Networks. *Appl. Sci.* **2025**, *15*, 6193. https://doi.org/10.3390/app15116193

1. Introduction

Wireless sensor networks (henceforth *WSN*) comprise a wide field of study. In a WSN, scattered battery-operated sensing elements transfer measured information wirelessly to data collection points called sinks. Traditionally, researchers have proposed several optimization techniques to enhance the weaknesses of the WSNs, that is, network lifespan and coverage [1–3]. Benefiting from the downfall in price and a considerable growth in popularity of the robotics systems, robots were introduced to cover the deficiencies present in the WSN [4,5]. The addition of mobile robots (Mobile WSNs) permits the replacement of sensing nodes that are about to run out of battery or are found to be damaged, resulting in an increased resiliency compared to static WSN. Moreover, the mobile robots can help to distribute the traffic load each node forwards by providing routing support or by redistributing nodes, which can lead to a more uniform battery depletion across the network [6]. Even recent trends, such as AI and ML, are merging with WSN [7–10]. However, despite the interest of the scientific community, several research challenges still need to be faced. These challenges are mostly related with practical implementations, mainly due to the lack of realistic studies or implementations that can actually be deployed [11]. This lack of field testing often leads to unrealistic assumptions regarding the collaboration methodologies employed in mobile WSNs, including the information required for cooperation and the algorithms that facilitate such collaboration. To address this, a software package is provided whose aim is to study different sensor deployment and maintenance strategies. This software package employs the globally

known robotics software ecosystem, ROS 2 [12,13]. ROS 2 software packages allow a seamlessly deployment of distributed applications across networked machines or to be run locally using simulators; therefore, algorithm and protocol validation can be achieved before deploying the software on actual hardware.

This paper is structured as follows: Section 2 provides a comprehensive review of the state of the art regarding mobile WSN and current deployments. In Section 3, our contribution is presented. Sections 4 and 5 provide a relation between the different phases a MWSN suffers and the strategies and implementation followed in this article. Next, in Section 5.5, an overview of the algorithms that are put to test is provided. Results are presented and discussed in Sections 6 and 7, respectively.

2. Related Work

In this section, a survey containing the state of the art is presented. First, the relevance of WSNs is highlighted with some key applications. Hereafter, a focus on the metrics in WSNs is given along with some of the techniques to improve those metrics. Following this, some studies regarding mobile WSNs are presented along with orchestrating software and methods.

2.1. Wireless Sensor Networks: Applications

Wireless sensor networks are becoming ubiquitous in the 21st century, exhibiting their presence in many current applications (industry, domestic, urban environments, etc.). Among these applications, those that tend to lack proper power and communications infrastructures are those that pose the biggest challenges, such as military zones, wildlife monitoring, and environmental disasters [14–16]. Given its transcendence for the economy and development, precision agriculture represents a topic where several of those challenges meet. Hence, several research lines converge WSN with precision agriculture [17–19].

2.2. Wireless Sensor Networks: Metrics

Accounting for the relevance that WSNs currently exhibit, optimizations related to metrics have been an ongoing subject of study for researchers. Due to their battery-operated nature (and thus limited lifetime), extending the lifetime of the network is therefore a critical issue. Since current consumption on wireless nodes can be mostly attributed to either CPU operation or transceiver (network) usage, it is clear that in order to maximize lifetime, those power-hungry components must be used as little as possible. This fact is easily verified by the datasheets from popular wireless transceivers [20,21].

However, there are a number of factors that can come into play when it is no longer possible to minimize either CPU idle or network sleep. As pointed out in [1], traffic load imbalances lead to faster battery depletions in the nodes that assume a higher network operation. This accelerated discharge is the factor that causes connectivity losses in the network as those nodes that have exhausted their batteries are now unable to route the traffic originating from their neighbors, thus forming gaps in the network with no connection.

Regarding network longevity, [22] highlights the problems with the different WSN longevity strategies in relation to their actual usefulness and presents some useful terms. *First node dies*, or the time until the first node runs out of power, may result too conservative because it assumes that the network dies when the first node runs out of power. *Last node dies*, on the other hand, measures the time until all nodes run out of power. Maximizing *last node dies* can be a misleading measure because it depends on the idle time of the least active nodes.

Alternative metrics include the so-called beta surviving nodes, which represent a minimum proportion of active nodes required for the network to be considered operational. However, determining this value presents a significant challenge. In addition to the aforementioned strategies, there are those based on coverage. 'k-coverage' measures the time during which each region is covered by at least k sensors. 'Alpha-coverage' quantifies the time during which the coverage ratio of a region is greater than an alpha threshold. It should be noted that it is necessary to not only cover a region, but also to be able to transmit these data (*connected coverage*). The authors' aimed contribution is to minimize redundant traffic and therefore consumption by assigning different roles in the network.

In order to enhance deployment, the contribution presented in [23] employs an evolutionary algorithm to improve the layout of the nodes in WSNs. In practice, the issue can be framed as an optimization problem, with the objective of minimizing energy consumption at the node with the highest consumption. This approach is entirely simulation-based, which presents a challenge in terms of its applicability to actual nodes. Similarly, ref. [24] focuses on reducing energy dissipation by proposing a relay-based clustering approach, thus reducing multi-hop communication. A simulation model of the LEACH routing protocol is used.

Other trends in the literature tend to consider network lifetime as a secondary aspect, emphasising instead the importance of network connectivity. In [25], authors present a mathematical model for WSN, which they entitle 'Probabilistic Network Connectivity Algorithm (PNCA)', considering the distribution of nodes as a Poisson process and the connectivity between them as a binomial process. MATLAB R2018a is used to simulate their proposed optimization algorithm for a number of 50 nodes. The problem to be solved is based on the search for the optimal deployment to maximize connectivity. This connectivity is a probability (network probability) as a function of the connectivity factor with a fixed communication radius and, conversely, the network probability for a fixed connectivity factor value equal to two and different coverage radius values. According to the simulations they present, the algorithm is more efficient (in terms of energy consumed) for the same coverage radii than DSR, ZTR, and LEACH protocols. Meanwhile, authors in [26] consider that sensor operation time is limited by energy storage, and the assumption of a regenerative power supply may not be true for most battery-operated implementations. In this case, authors state that the most important parameter is connectivity and, in order to obtain higher coverages, clustering of nodes and the correct choice of the gateways positions are crucial. To achieve this, a three-stage algorithm for the choice of clusters is proposed.

The contributions presented in this section demonstrate the validity and popularity that WSNs exhibit for the academia, not only presenting metrics but establishing the trade-offs between network lifetime and coverage. Despite research efforts, some of the proposals are planned as proprietary or simulation-only models, complicating the implementation of their work on current hardware. In the next subsection, the addition of robots as a measure to extend the capabilities of WSNs is introduced.

2.3. Mobile WSNs

As it was presented in the previous section, mobile elements can be introduced to dynamically modify network topology [27]. Controlled changes [28] in the network topology can improve network balancing and therefore network lifetime. However, not only were those mobile elements introduced to improve network's operation time, but mobile WSNs can extend the characteristics and resiliency in comparison with static WSNs.

With regard to current applications, several trends can be identified. As it was the case for static WSNs, agriculture and precision agriculture benefit from the addition of robots.

In [29], authors use an unmanned aerial vehicle (UAV) as a mobile sink to collect agricultural monitoring data. By following predefined paths, simulations results claim data can be collected at a low energy expense. Nevertheless, most of the use cases found in the literature employ robot clusters instead of a single robot to expand network capabilities, as presented in [30]. This claim is supported by [31], where robots collectively map and collect data in a greenhouse environment. Monitoring of city pollutants using UAVs is discussed in [32], but the limited number of UAVs raises some doubts about system scalability. Multirobot systems can additionally be found in hazardous and harsh environments as in [33], where authors propose a deployment strategy for coal mine monitoring that claims to optimize the number of nodes.

2.4. Orchestration Methods and Software for Multi-Robot Systems

Due to the foreseen relevance of multi-robot systems, our literature review identifies several efforts to establish coordination mechanisms. This coordination is crucial for the success of joint operations in intelligent agents, as identified in [34]. In this aspect, intelligent agents must be able to carry out problem solving in a planned and rational way, but being flexible in their behavior according to the environmental characteristics of the surroundings.

Most orchestration methods rely on newly developed frameworks for popular already existing languages. Examples of this can be found in [35], a Python framework to manage robot teams at a high level of abstraction. This work also proposes several hierarchies, highlighting the primary–secondary or peer–peer organization. However, all tests remain in simulation and it looks like no further progress has been made to continue this project. Scala language also seems to be adequate for these frameworks. The contribution of [36] is based on aggregate computing, presenting an API for the Scala language that allows the description of high-level behaviors such as moving and performing group behaviors. Authors evaluate the use case of 'find-and-rescue' in order to show the ability to execute complex behaviors. While simulation results are favorable using the 'Alchemist' simulator, authors consider actual deployments for future work. A parallel can be drawn with [37], where the attribution of tasks to agents is described with a high degree of mathematical precision. Nevertheless, the feasibility of implementing this proposal for actual agents, or indeed its capacity for simulation, remains ambiguous.

The Resh programming language is proposed in [38]. Resh aims for the orchestration of robot swarm systems. It offers the orchestration of large concurrent systems and largely asynchronous tasks, allowing the developer to focus on the tasks and actions that the robots must perform, letting the language to handle all other scheduling tasks. However, Resh is not designed for tasks that require strict control of the robot swarm. Using the 'Resh Interpreter' and one 'Robot Agent' per robot, it seems possible to interact with ROS-based robots, although authors do not provide an extensive documentation about the list of platforms supported and whether Resh support for ROS-enabled platforms can be expanded anymore.

Perhaps the most complete solution in this area can be found in Aerostack 2 [39]. Built on top of ROS 2, Aerostack enables high-level and collective behaviors on diverse drone platforms. Despite the existence of contributions targeting ground robots such as [40] and [41] (among others), none of those have attained the same level of popularity as the initial one.

In conclusion, as identified in [11], there is a clear lack of implementations that could actually be deployed on real life. This is where our contribution lies, as we additionally consider both robots and sensor nodes together.

3. Contributions

The objective of this paper is to introduce and evaluate a set of algorithmic strategies for deployment and maintenance in mobile wireless sensor networks, focusing on maximizing both network lifetime and connected coverage. These algorithms build upon prior approaches [42,43] but introduce key conceptual advances: (i) agent negotiation protocols are redesigned for improved scalability and efficiency, (ii) energy consumption models are refined to better capture realistic operational costs, and (iii) the algorithms operate under more practical assumptions, such as relying only on local knowledge of neighboring agents' positions. All algorithmic evaluations are conducted in simulation under controlled and idealized conditions to ensure reproducibility and comparability.

Complementing these conceptual contributions, we present a modular software package implemented using the standard robotics middleware ROS 2. This package serves as a testbed for deploying, evaluating, and extending the proposed algorithms. It includes wrappers for controller classes, simulation support, and tools for logging and visualization. The package is publicly available at: https://github.com/DIANA-IoT/ROS-2-Agents, facilitating reproducibility and future development. We also discuss the practical feasibility of deploying the algorithms on real robots, bridging the gap between theoretical validation and real-world application.

4. Network Operation Phases

In the context of mobile WSN operations, which encompasses deployment, maintenance, and retrieval, the proposed system adheres to a logical sequence depicted in Figure 1.

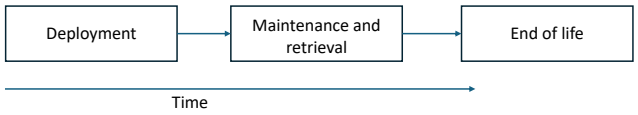

Figure 1. General phases for MWSN.

In mobile WSNs, the deployment phase is carried out by a set of mobile nodes: vehicles, specialized workers, or mobile robots. The mobile nodes aim to reach maximum coverage by determining optimal positions according to different algorithms. Once the network is deployed, the network operates routing traffic and during this phase, the necessary adjustments must be performed to keep the network operational for as long as possible while maintaining the connected coverage. However, over time, the number of degraded nodes increases, which can lead to such a large number of inoperable nodes that the network is considered to be depleted and reach the end of its operational life.

In this paper, we consider the mobile elements to be ground robots with embodied sensors. The movement of each robot is determined by the application of the concept of Social Potential Forces (SPFs) [44,45]. The concept of SPFs comprises two distinct types of virtual forces: repulsion forces and attractive forces. The aforementioned forces are combined to form an emergent vector, which is then decomposed along the 2D or 3D axes in order to determine the motion and direction to be followed by the robot. Potential Forces algorithms are well known in the literature due to their low computational requirements and modest bandwidth demands.

Figure 2 presents the operations performed during each phase for the two strategies that are developed in this article, SPF-only and Agents. The SPF-only strategy applies a different set of forces according to the network phase. The agents approach adds an intelligent layer when the network is in the maintenance phase, allowing to replace degrading robots and balancing traffic.

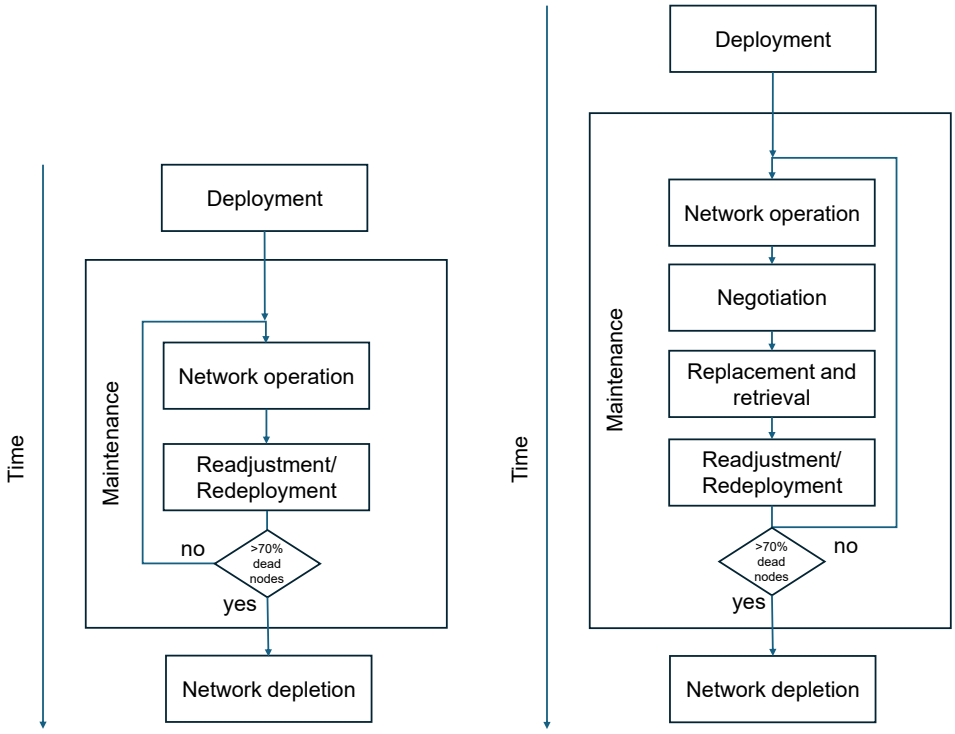

Figure 2. Operations for SPF-only (**left**) and Agents (**right**) strategies.

4.1. SPF-Only Strategy

At a high level, the SPF-only strategy consists of the following sequence of operations. During the deployment phase, robots move according to SPF forces, as previously described. A key contribution of this work is the inclusion of *link-local communication—meaning robots are only aware of the information broadcast by their immediate neighbors*—to share positional data. The system transitions into the maintenance phase once the network is deemed stationary; that is, when robots have reached their optimal deployment positions and the SPF forces have reached equilibrium. In the maintenance phase, the network operations occur. During network operations, the mobile elements are stationary and know the locations of all nodes in the network. At this moment, nodes route traffic to the sink, and a computation is effectuated to determine which node would deplete first. When enough time passes to deplete this node, the network requires a balancing, denoted in Figure 2 as 'Readjustment or redeployment'. This process carries on until at least 70 percent of the initial nodes have run out of battery. This represents the threshold needed to advance into the 'Network depletion' phase in which the network is too degraded to be considered operational.

4.2. Agents Strategy

Similarly to the SPF-only strategy, the deployment phase is carried out by the computation of SPF forces. In maintenance, the network operations are similar, but the Agents strategy needs to calculate the time until a node needs help instead of the time until a node dies. This allows robots with low battery levels to trigger a negotiation round, where they ask for help. Agents with higher battery levels show their disposition to help others (henceforth *willingness*) and the robot asking for help selects the most suitable candidate to replace them. The process continues with the replacement of the robot in need which retrieves to a collection point. After these two operations, the robots move again according to the SPF forces to adjust their positions. Just like in the SPF-only strategy, the maintenance is over when at least 70% of the robots have depleted.

5. Implementation

The algorithms are developed to be either tested on real robotic platforms (thanks to its ROS 2-based implementation) or to be tested on simulations. Robots move according to the SPF forces until the network is considered to be stationary. For a network to be considered stationary, at least 80% of alive robots must remain in with their forces at equilibrium or below a movement threshold.

During the maintenance phase, time is event-driven, allowing to steadily emulate battery draining across the network. This topic is introduced below.

5.1. Battery Simulation of Network Operations

Battery drainage in mobile WSNs is primarily contributed by two factors: robot movement and packet routing.

Robot movement is independent from the network stage as it applies whenever the robot is in movement. The battery consumption (in percentage) due to robot movement is modelled as follows: $m(d_m) = 100.0 \times \frac{d_m \times K_m}{B_{cp}}$, where d_m represents the distance travelled in any axis, stated in meters, and K_m is a robot-model-dependant constant that represents a certain current motor drain at a constant velocity, in mAh/m. Lastly, B_{cp} is the battery capacity in mAh. The $m(d_m)$ percentage is calculated internally by each robot concurrently with its movement. This calculation is subsequently subtracted from the current battery percentage, denoted as b_c.

Concerning packet routing during network operations, sensing nodes employ an ad hoc 2.4 GHz-operated protocol that implements a controlled flooding to transmit messages from the sensing nodes towards the data sink. This model, although simple to simulate, is realistic based on actual implementations [20,21]. According to this routing mechanism, a packet from node i to node j is forwarded (i.e., broadcasted) again by any node receiveing it if, and only if, that node has not already received the same packet and it is closer to the destination than the previous hop. The controlled flooding mechanism is summarized in Listing 1, where $dist(n1, n2)$ is a function that returns the Euclidean distance between Nodes n1 and n2, $mark\_seen(n, msg)$ is a function that sets a flag indicating that packet msg has already been retransmitted by node n, and $seen(n, msg)$ is a function that returns the state of that flag (i.e., if node n has already forwarded message msg).

Listing 1. Pseudocode for the controlled flooding routing protocol.

```
msg = {id: msg_id, dest: j, sender: i}
forward(i,msg)

function forward(node,msg):
  if node==msg.dest:
    process(msg)
  else:
    for neighbor in neighbors(node):
      if not seen(neighbor,msg) and dist(neighbor,msg.dest) < dist(node,msg.dest):
        mark_seen(neighbor,msg)
        forward(neighbor,msg)
```

This model assumes that if a node n is within reach from a node i, there is a 100% percent likelihood of receiving a packet from node i in node n. This is purely determined by the chosen radio technology and its range.

Battery drainage due to network communications for each node n is proportional to the number of packets node n is transmitting and forwarding according to the traffic model and the controlled flooding routing mechanism.

The battery drainage, $U_p(n)$, for sending a single packet from node n that can reach all its neighbors within a certain range depends on (but is not proportional to) the transmission power necessary to cover that range. To model this, we define $U_p(n)$ as

$U_p(n) = R_{packet} \times (1 + R_{incdis} \times Range(n))$, where $Range(n)$ is the range covered by node n, R_{packet} represents the baseline battery drainage when transmitting a packet, and $R_{incdis}(mAh/m)$ models the increase in this drainage depending on the range covered by node n ($Range(n)$). R_{packet} and R_{incdis} depend on the transceiver model and the chosen radio technology. In the tests presented in this paper, all the nodes are considered to have the same range and therefore the same battery drainage per transmitted packet.

Once the battery drainage for sending a single packet from node n to all its neighbors within the range it covers is determined, it is necessary to compute the amount of packets each node is transmitting and forwarding according to the traffic and routing models. In order to estimate the battery drainage at each node due to network communication when the MWSN is stationary, we assume that during this phase all network nodes periodically transmit data to the AP using the controlled flooding algorithm, and then we compute $P(n)$, the number of packets each node n is transmitting or forwarding during a network cycle. Thus, we can compute the battery drained by node n during every single network cycle as

$$Drain(n) = K_{duty} + P(n) \times U_p(n) \qquad (mAh) \tag{1}$$

where K_{duty} is a constant that accounts for the battery drainage due to the waking up and duty-cycling of the sensing nodes.

Therefore, during a cycle and from a network perspective, all nodes send a packet to the AP, and intermediate nodes forward them according to the controlled flooding routing mechanism. As a result, nodes closer to the AP (which act as a data sink) experience higher energy consumption due to their higher number of packets forwards, and, consequently, they deplete first. The number of cycles each robot n lasts until depleted can be calculated by the difference between the current battery percentage and a critical level denoted by b_{l0} as

$$Cycles(n) = \frac{0.01 \times (b_c(n) - b_{l0}) \times B_{cp}}{Drain(n)} \tag{2}$$

This critical level b_{l0} is further explained in Section 5.2.

For efficiency's sake, our network simulations follow an event-drivent approach. Thus, network simulations advance in time until a node dies or when a node needs help in the Agents strategy, as stated in Equation (3). This is the minimum number of cycle value of N alive nodes.

$$sim_{Cycles} = \min\{Cyles(0), Cycles(1), \ldots, Cycles(N-1)\} \tag{3}$$

After determining the minimum number of simulation cycles for the most critical node, the corresponding battery subtraction is performed on all alive nodes according to their expected energy expenditure. Once this operation completes, the network simulation ends and the system behaves differently depending on the election of the strategy: SPF-only or Agents, according to Figure 2. So, the network simulation interval takes place between a topology change and the moment when Equation (3) is met; that is, when a robot requires assistance and consists of a certain number of cycles, as in Equation (2).

5.2. Agents: Negotiation and Replacement

Agents negotiation adds an intelligent negotiation protocol to maximize both coverage and lifetime by replacing robots (agents) that are about to be depleted and thus balancing network load.

Each robot estimates the absolute minimum battery level it requires to reach a certain depot point safely (denoted as bl_0). To figure out how close robots are to reach that critical level, two additional heuristically chosen thresholds are introduced, bl_h, bl_1. In the equation below, d_{dp} represents the distance between the current robot position and the depot point,

expressed in meters. K_m is a model of the current consumption due to motor movement, expressed in $mA \times h/m$, as introduced in the previous subsection. B_0 is the battery charge at simulation start expressed in $mA \times h$ and B_{cp} represents the battery capacity in $mA \times h$.

$$bl_0 = 100 \times (d_{dp} \times K_m + 5\% \times B_0)/B_{cp} \tag{4}$$

$$bl_h = 10\% \times bl_0 + bl_0 \tag{5}$$

$$bl_1 = 30\% \times bl_0 + bl_0 \tag{6}$$

The relation between b_c (or current battery level) and the above-mentioned parameters indicates a robot's urge to be replaced or, by contrast, its disposition to replace a robot in need. Those thresholds are combined in the *willingness to interact* or $w(t)$, as in Equation (7).

$$w(t) = \begin{cases} -1, & if b_c \leq bl_0, \\ \frac{b_c - bl_1}{b_c}, & if b_c > bl_0 \end{cases} \tag{7}$$

A negotiation round is triggered when an agent reaches a willingness value of -1, that is, critically close to depletion. This is accomplished by sending a message through the network asking for help (Figure 3a). Robots with positive willingness answer to this request by replying back with their disposition to help (Figure 3b). Other agents with negative willingness, but not higher than 20% on top of w_H, stated as w_{H20}, also send their request to be replaced. In this way, not only critical robots are replaced, but also a fraction of them that are becoming close to that level.

$$w_H = \frac{bl_h - bl_1}{bl_h} \tag{8}$$

$$bl_{h20} = bl_h + 0.2 \times (bl_1 - bl_h) \tag{9}$$

$$w_{H20} = \frac{bl_{h20} - bl_1}{bl_{h20}} \tag{10}$$

Agents inclined to help others provide their disposition by sending a combination of their willingness and the utility parameter $u(t)$ to form $W(t)$ (Figure 3b). The utility is the mathematical model showing the trade-off between current position and moving to a new one. In $u(t)$, b_m represents the battery expenditure to move to the new position and b_n is the battery level from the new position to the depot point; similarly, b_p is the battery level required to move from the current point to the depot point. Traffic routing is also considered, where p_A reflects the battery spent on forwarding at the current position and p_B is the battery that is necessary to spend on the updated point. From a physical perspective, the term $\frac{p_B - p_A}{p_B + p_A}$ enables the utility parameter to increase as the difference between the packets routed at the new position is greater than the routed packets at the current position. Similarly, $\frac{b_m + b_n - b_p}{b_c}$ ensures that agents farther away from the depot point are also participating in the negotiations as long as their new location is closer to the depot point than their current location. Thus, agents only provide a $W(t)$ greater than zero when they only have enough battery to move and operate in the new position, so that movement favors locations with higher traffic demands. In case of competing constraints after computing $W(t)$, the assignment is determined by the chosen algorithm. More details about the assignment algorithms are given in Section 5.5.

$$u(t) = \begin{cases} 1 - \dfrac{b_m + b_n}{b_c} + \dfrac{b_p}{b_c} + \dfrac{p_B - p_A}{p_B + p_A}, & if\, 1 - \dfrac{b_m + b_n - b_p}{b_c} >= 0.3, \\ -w(t), & otherwise \end{cases} \tag{11}$$

$$W(t) = w(t) + u(t) \tag{12}$$

Robots asking for help notify their replacements by sending a packet through the network (Figure 3c). Once this is achieved, robots with critical battery level and those who are close to it return to the collection point. Agents chosen to replace others now move to the former locations. This process represents the *Replacement and retrieval* operation in Figure 2.

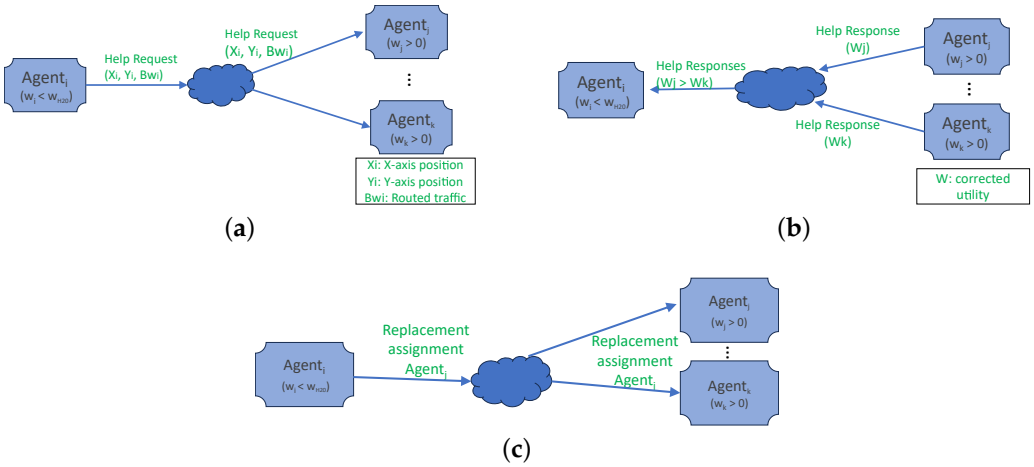

Figure 3. Messages exchanged during negotiation.

5.3. Readjustment

According to the routing protocol previously explained, robots with higher energy demands become the first to run out of battery (due to their more intense traffic routing, usually those who are closer to the sink). If the SPF-only strategy is chosen, those nodes continue operating at their place until their depletion, possibly creating a hole in the network between the sink and the further nodes. To avoid this, after each network simulation round, a redeployment is effectuated in order to make the network more compact. In the case of Agents, redeployments are performed without necessarily making the network more compact. This is due to the intelligent negotiation algorithm, which aims to maintain maximum connectivity while extending network lifespan.

5.4. SPF Forces

As previously introduced, both SPF-only and Agents strategies are supported on top of the SPF navigation algorithm. In order to determine the heading and the velocity the robots should follow, a set of attraction and repulsion forces are defined.

Obstacle repulsion forces, just as the name implies, prevent robots from colliding with elements (other robots, obstacles, walls, etc.). The detection of obstacles can be achieved by several means such as bumper sensors, imaging, or LIDAR. In this work, we opt for LIDAR-based obstacle detection. As mentioned, obstacles can include both physical statically placed obstacles (e.g., walls or objects in the middle of the environment) and other robots. Robots can be seen as obstacles when they are physically close to each other during the deployment stage or when they are stationary. The reason robots can be stationary at other phases include the exhaustion of their batteries (thus they cannot move any more) or when they are not part of a replacement operation in the Agents strategy. The formula for obstacle repulsion forces is defined:

$$fr_1(d_{i,j}) = \frac{256}{(d_{i,j})^8} \tag{13}$$

Not only do obstacle repulsion forces exist in SPF navigation algorithms, properly tuned robot repulsion forces aid with network expansion, whose expression is described below.

$$fr_2(d_{i,j}) = \frac{8 \times 10^8}{(d_{i,j})^7} \tag{14}$$

In both cases, $d_{i,j}$ represents the Euclidean distance between a robot i and an obstacle or another robot j.

With respect to attraction forces, punctual attraction forces allow robots to reach a specific point (p) when an agent needs to be replaced.

$$fa_1(d_{i,p}) = -arctan(d_{i,p}) \times 2 \tag{15}$$

Lastly, a cohesion force to gradually contract the network is presented in Equation (16). The cohesion force depends on the parameter α, $n_{fail}(t)$ is the number of depleted nodes, n is the initial number of nodes and $n_{closeAP}(t)$ is the number of nodes within the radio coverage of the sink. This force acts by pulling robots to the origin of the coordinate frame (z), as indicated by the distance $d_{i,z}$.

$$f_{c_i}(t) = -\frac{\alpha \times n_{fail}(t)}{n \times n_{closeAP}(t)} \times \arctan(d_{i,z}) \tag{16}$$

Depending on the network phase and the chosen strategy, a set of forces commands the robots to achieve their goal (i.e., expanding forces -$fr1$- in the deployment or attractive forces when an agent needs to be replaced -fa_1-). A summary of the aforementioned forces is given in Table 1.

Table 1. Forces according to strategy and network phase.

	SPF		**Agents**		
	Deployment	**Adjustment**	**Deployment**	**Replacement and Retrieval**	**Adjustment**
Obstacles, $fr_1(d_{i,j})$	✓	✓	✓	✓	✓
Robots, $fr_2(d_{i,j})$	✓	✓	✓	✗	✓
Attraction, $fa_1(d_{i,p})$	✗	✗	✗	✓	✗
Cohesion, $f_c(t)$	✗	✓	✗	✗	✗/✓

5.5. Algorithms

This paper, as previously pointed out, improves our existing work (as in [42,43]). Apart from refactoring some formulations in the negotiation part and porting the software to the ROS 2 environment, we additionally propose several assignment algorithms to be used with the Agents approach. These assignment algorithms are focused on determining which agent is to take care of a robot in need during the negotiation operation. In that sense, every algorithm takes place when a negotiation round is triggered. While the negotiation round is active, an agent or potentially a set of them is asking for help. However, based on their willingness, each agent has a different urgency to be replaced.

The first algorithm, 'the algorithm 0' (*ALG0*), recreates a distributed scenario where every agent that needs help is randomly picked up and the most adequate replacement is chosen. This approach, although straightforward to distribute, presents an inconvenience due to randomness. The disadvantage in this case is that it cannot be guaranteed that the

agent with the most urgency is attended in the first place, which could lead to sub-optimal network utility.

In contrast, ALG1 sorts out the willingness of the agents in need and takes care of them in a logical order, prioritizing the ones with a more negative $w(t)$. Although this ensures the order in which help requests are attended, it represents a more challenging scenario to implement the ALG1 in a distributed manner.

The last algorithm proposed, ALG2, attempts to achieve optimal network utility. This is achieved by taking a slightly different approach. Contrary to the previous algorithms, ALG2 focuses on the robots that are willing to provide help. ALG2 sorts the agents with positive $w(t)$ in descending order and picks the robot in need in which the combined utility $W(t)$ is greater. Thus, robots with high willingness are maximizing network utility. The disadvantage this approach presents is the amount of information that needs to be exchanged, which complicates the implementation in a distributed form.

In order to conduct empirical comparison between ALG0, ALG1, and ALG2 against our base approach, the SPF-only strategy is effectuated in Section 6.

5.6. Software Architecture

The software that enables the implementation of the above-mentioned strategies and operations is based on ROS 2. The usage of ROS 2 permits the execution of the algorithms in a distributed manner in simulation or in actual deployments. The proposed software architecture is shown at Figure 4.

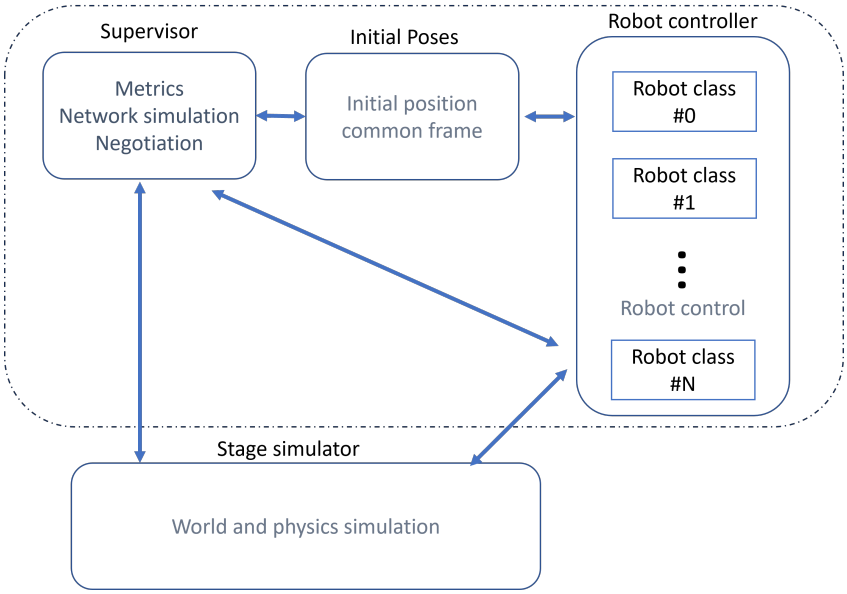

Figure 4. Software architecture for the proposed package (contained in dots) plus their functionality.

- **Stage simulator plus ROS 2 node binding**. This ROS 2 node (executable) carries out physics simulation for the robot models within a determined simulation world.
- **Initial pose node**. The ROS 2 node in charge of publishing the initial poses for each robot in the world reference frame [46]. The existence of this node can be justified due to the differences in the local robot position estimations (odometry) at 'zero time' and the real position in a common spatial reference.
- **Robot controller**. As the name implies, this software component manages the execution state of each robot. It is designed to run on both simulated and real robots. Essentially, the robot controller acts as a wrapper for one or more controller instances, referred to as the *Robot class*. By encapsulating multiple *Robot class* instances within a single *Robot Controller* ROS 2 node, the computational load is reduced compared

to running a dedicated node for each robot. This makes it possible to run large-scale multi-robot simulations more efficiently. Without this wrapping, simulations involving 100 to 200 robots have proven too demanding for our test machines. The advantages of grouping multiple Robot class instances within one *Robot Controller* are demonstrated in Table 2, which highlights how computational requirements grow with swarm size. In real robots, however, this optimization is not feasible: each *Robot Controller* can only manage a single *Robot class* instance.

- **Supervisor**. The supervisor is a simulation mechanism the purposes of which comprise the launching of the deployment phase, the networking simulation, and the negotiation protocol (as right now it is centralized, and only when it applies). Additional tasks the supervisor carries out are those related to metrics collection and visualizing robot positions.

The software architecture presented above allows to not only take advantage of the already developed ROS 2 tools like simulators, packages, and utilities, but, given its modularity, to modify the number of robots either in simulation or in actual deployments only requiring minor tweaks in the Supervisor configuration files and spawning the correct set of Robot Controllers. Moreover, the recently mentioned ROS 2 configuration files permit a straightforward method to set various simulation parameters, like the forces or the algorithms and strategies to be used. More details about the different parameters used in this work are given in the next section.

Table 2. CPU and memory usage for 100 and 200 robots. Test machine specifications: Intel Core i7 12700K, 32 GB RAM, Ubuntu 22.04. Ten Robot controllers with 10/20 robots, respectively.

Scenario	Mean CPU Usage (%)	Peak CPU Usage at Startup (%) [1]	Memory Consumption (GB)
Idle	0.5	NaN	1.7
100 robots	24.5	39.1	4.5
200 robots	38.1	91.2	9.4

[1] We attribute such high CPU utilization to the ROS 2 node discovery process and the launching of 'static transform publisher' nodes. The static transform nodes are necessary to visualize the positions of the nodes and to avoid overhead after the transient state.

6. Tests and Results

In this section, the results for comparing the SPF-only and Agents strategies are provided. Since the different strategies and algorithms aim to reach maximum lifespan while maintaining an optimal connected coverage, the two main metrics therefore consist of the connected coverage and the network lifetime.

As explained in Section 2, the connected coverage metric represents the fraction of the area that can be monitored by the robots that maintain network connectivity with the access point and the quality of the sensing they can perform in that area. In order to estimate this, we use the probabilistic sensing model proposed in [3], according to which the quality of sensing (sensitivity) that each robot can achieve gradually attenuates with increasing distance. Following this approach, in order to compute this metric, we divide the area into a probabilistic grid of M cells and we compute the quality of the sensing at each cell i as the probability of detecting an event that occurs at that cell (p_i). Since a given cell can be monitored by more than one robot, this in turn depends on the number of robots that can monitor that cell and the distance to each of them. Thus, if cell i is covered by N robots with connectivity to the AP, we compute p_i as

$$p_i = 1 - \overline{p_i} = 1 - \prod_{j=1}^{N}(1 - p_{ij}) \tag{17}$$

where p_{ij} is the probability of robot j detecting an event that occurs at cell i, which depends on Euclidean distance from robot j to cell i, d_{ij}. To estimate the quality of the sensing of robot j at cell i, we define several sensing ranges $R_0 < R_1 < R_2$ and compute p_{ij} as

$$p_{ij} = \begin{cases} 1 & if\ d_{ij} \leq R_0 \\ 0.6 & if\ R_0 < d_{ij} \leq R_1 \\ 0.2 & if\ R_1 < d_{ij} \leq R_2 \\ 0 & otherwise \end{cases} \tag{18}$$

Therefore, the quality of the sensing at each cell represented by p_i lies in the interval [0, 1], and p_i is 0 if cell i is not being covered by any connected robot. The total coverage of the area achieved by the network of robots is subsequently computed as the average of the quality of the sensing at each of the M cells into which we divide the area:

$$C = \frac{1}{M}\sum_{i=1}^{M} p_i \tag{19}$$

In the case of network lifetime, this metric is defined as the duration it takes the simulation to reach a certain threshold. This threshold, for example, represents the condition for ending the simulation, which is a number of 70 dead nodes.

6.1. Test Methodology

The tests are carried out using the Stage simulator. The simulated environment consists of a wall-delimited obstacle-free 150 by 150 m area. In this area, 100 differential-driven robots are placed, as a direct comparison with our previous work [43], although different scenarios can be set up in terms of robot density or robot models, as explained in Section 5.6.

Robots solely use LiDAR signals within a range of 3.5 m and the required sensors for odometry estimation. The characteristics present in these robots make them comparable with actual robotic platforms like Turtlebot 3. This fact, combined with the usage of ROS 2, can enable large-scale deployments of multirobot systems like this one in real scenarios.

The set of tests performed is stated in Table 3. With respect to SPF-only tests, our baseline values, an analysis of the α parameter is carried out by testing different values such as 4, 5, 6, 7, and 10, with lower numerical values representing weaker cohesion forces. Regarding the agents' strategy, the three previously described algorithms (Section 5.5) are tested. In addition, we examine whether the agents' negotiation algorithm could replace the cohesion forces involved in the readjustment operation. Avoiding the need to set a specific value is a complex and time-consuming process that can significantly impact the results. To assess this, tests without cohesion forces are conducted using the latest algorithm, and each algorithm is labelled accordingly as 'alg0', 'alg1', 'alg2', and 'alg2sc' for Algorithms 0, 1, 2, and 2 without cohesion force, respectively. As in the SPF-only test, several alpha values are considered, although for clarity's sake, only the results for the optimal alpha value (1.6) are presented. To ensure statistical significance of the results, 30 runs for every test configuration are performed.

Table 3. A summary of the different tests carried out.

Algorithm	Nomenclature	Alpha	Simulation Runs
SPF	SPF4	4	30
	SPF5	5	30
	SPF6	6	30
	SPF7	7	30
	SPF10	10	30
Agents	alg0	1.6	30
	alg1	1.6	30
	alg2	1.6	30
	alg2sc	0	30

6.2. Results

An overview of the results is presented in Figures 5–10.

Figure 5 presents the final state in the network for two SPF tests—Subplots a and b—with the extreme values regarding cohesion forces (SPF4 and SPF10, with SPF4 being the weakest and SPF10 the strongest). The two subplots below, Subplots c and d, show the effect of the cohesion forces in the Agents ALG2 algorithm. On the left side, the lack of cohesion forces leads to the isolation of some nodes, whereas the subplot on the right includes a cohesion force, showing a uniform node distribution across the network without isolating nodes. A matter of interest is represented in the battery state (color gradients) that the nodes exhibit between the different approaches. A step imbalance can be perceived at the battery depletion in the SPF tests, where some of the nodes, especially those at the boundaries, comparably spend way less battery than the others. On the contrary, the Agents balance the battery expenditure uniformly.

A graphical representation related to the evolution of the different tests is given in Figure 6. In this figure, the evolution in coverage is compared with the elapsed time, also highlighting the relevance of the crucial parameter α. All of the SPF-only tests demonstrate temporary certain coverage blackouts around 40 days of simulation time, regardless of whether a stronger cohesion force is employed (*spf10*) or a weaker cohesion force (*spf4*). The reason for the blackouts is due to how the cohesion forces operate, as they are directly proportional to the number of depleted nodes, among other parameters; Equation (16). Since the SPF strategy relies only on the cohesion forces as attractors to balance the network, there are moments where the distance between nodes is greater than the coverage radius, causing partial disconnections. In the Agents' case, no communication outages can be perceived, showing a better overall stability (except for the test without cohesion forces, where a step loss in coverage can be seen towards the simulation end). The presence of dropouts in coverage can provide misleading conclusions if only the final lifespan is regarded. This is due to the routing algorithm and the battery simulation related to it. As unconnected nodes do not forward neighbors' traffic, unconnected nodes spend less battery during a network simulation cycle, making the network artificially last longer. In this case, although the network has a greater lifespan, it is less usable than more stable alternatives. For this reason, different checkpoints are established to evaluate the merits of each approach at different simulation times. These values are obtained through ANOVA-1 analysis of the 30 simulations that take place for every test, ensuring the statistical significance of our results.

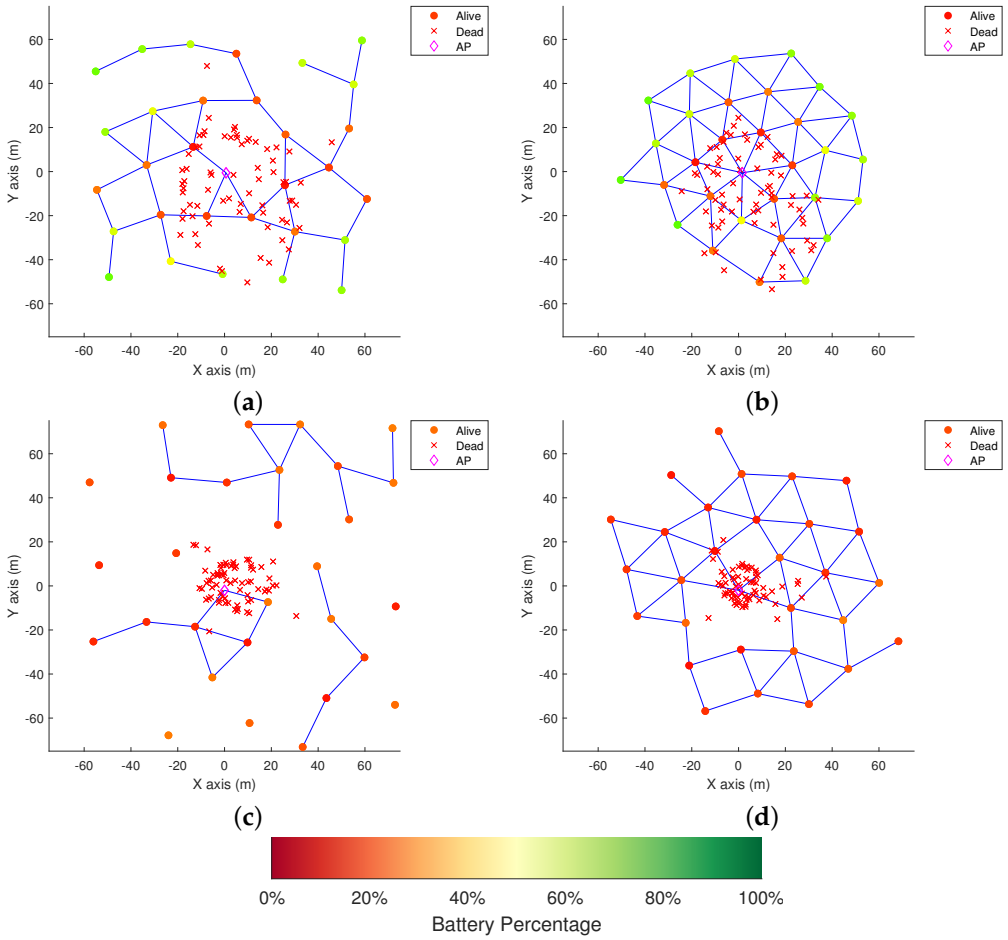

Figure 5. Battery levels (color gradient from green to red), network liveliness and connection diagram for the final state. Tests: (**a**) SPF4, (**b**) SPF10, (**c**) ALG2sc, (**d**) ALG2 .

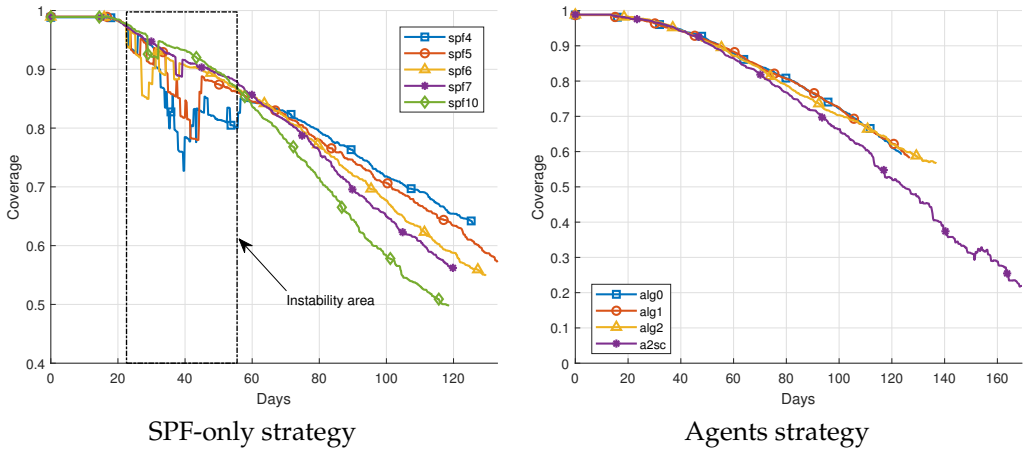

Figure 6. Average results (for 30 runs) in coverage vs days for the SPF-only and Agents strategies. The lack of traffic routing when there are outages and low coverage involve artificial extensions in lifetime.

A proposed milestone covers the event when half of the initial nodes have depleted. Because the simulations comprise a number of 100 robots, this event is evaluated when 50 nodes have run out of battery. Figures 7 and 8 represent the time needed to reach that event and the coverage with this number of nodes. In the Agents approach, a clear advantage in efficiency is shown with respect to the SPF approach. Every algorithm in the Agents strategy takes at least a median value for 80 days to deplete 50 nodes, whereas the SPF-only strategy takes, in the best scenarios, less than 75 days, even approaching

65 days when greater cohesion forces are applied. The efficiency of the Agents strategy does not come with a great loss in coverage, showing similar results to the SPF one, with all tests (except for the last algorithm without cohesion force) maintaining a coverage around 80 percent within less than 2% of standard deviation.

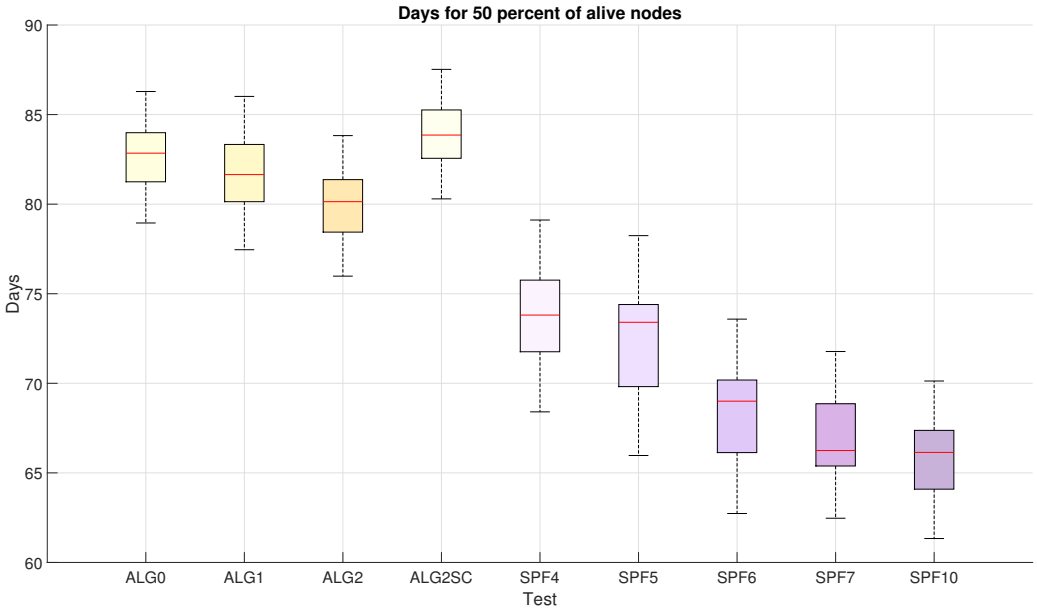

Figure 7. Time until 50% of nodes have depleted.

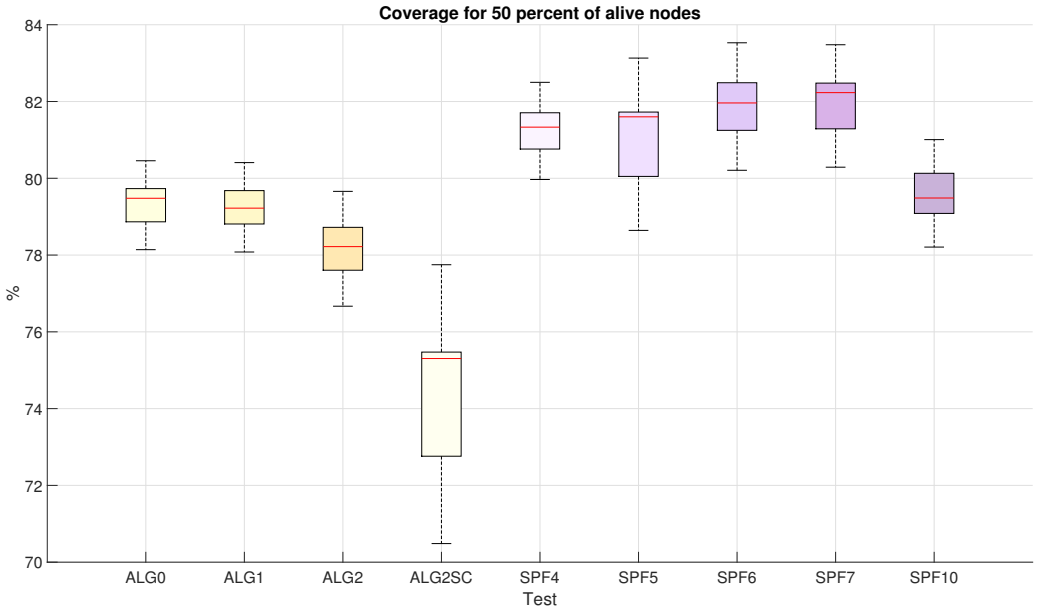

Figure 8. Coverage with 50% of depleted nodes.

Similarly, another approach involves evaluating the network in the event that the coverage falls below a threshold of the 60%. In this case, the time required to reach this point is shown at Figure 9. As in previous figures, SPF-only strategies represent greater uncertainties, which are specially notable with *spf4*, *spf5*, and *spf6*. At the values when the SPF is more stable, a clear disadvantage can be perceived with respect to the Agents approach both in terms of stability and the network lifetime, even compared to the test without cohesion force.

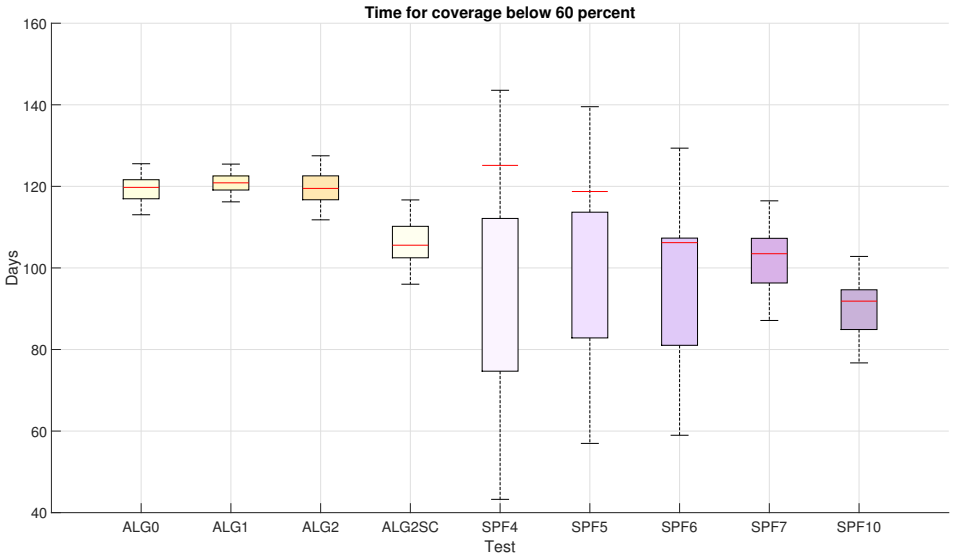

Figure 9. Days until a coverage below 60% is reached.

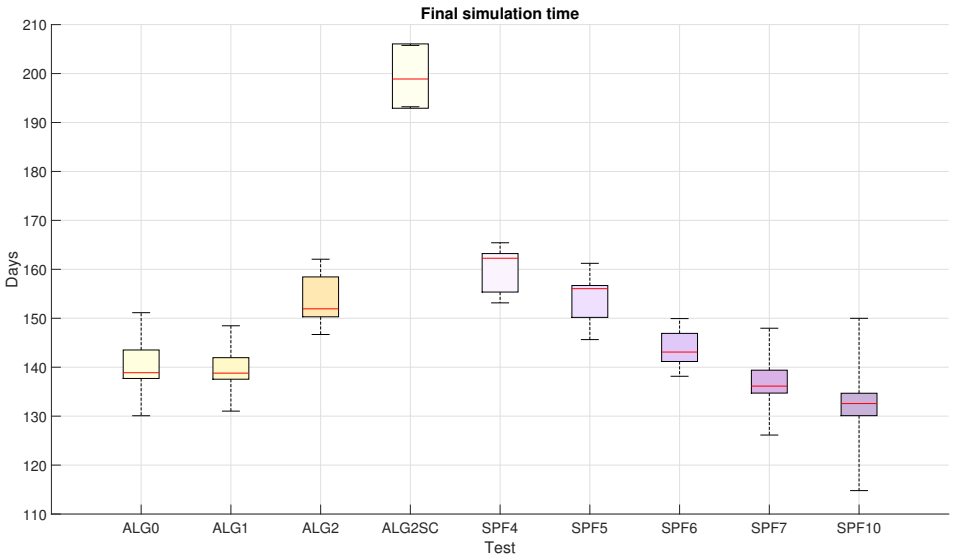

Figure 10. Final simulation time for all configurations.

Lastly, if the network lifetime is to be decided between the tests that are proven to be stable, a comparison is made in Figure 10. The blackouts in coverage and the instability present in *spf4*, *spf5*, *spf6*, and *alg2sc*, as shown in Figure 6, might discard this metric as a valid result (because of the reasons explained at the beginning of this section). This fact sets Algorithms 0, 1, 2, and the SPF-only strategy with cohesion α 7 and 10 as the remaining tests to be compared. The latter approach, even with temporary blackouts and greater uncertainties than the Agents, represent lower lifetimes with median values below 140 days. The similarities among the Agents result in a distinction favoring Algorithm 2 regarding final lifespan. This value is expected due to the optimality that this negotiation algorithm represents.

7. Conclusions and Future Work

In this work, we present a software package for the ROS 2 ecosystem aimed at providing a flexible test environment for evaluating deployment, operation, and retrieval strategies in mobile wireless sensor networks (MWSNs). We compare a standard SPF-only approach with an agent-based strategy, incorporating modifications such as cohesion force tuning and new negotiation algorithms. Our experiments demonstrate that the

Agent-based approach achieves greater stability and system lifetime while maintaining coverage comparable to SPF-based methods. Additionally, the Agent algorithms are more effective in distributing energy consumption across the network, as shown in Figure 5.

Among the three algorithms evaluated, the similarity between Algorithms 0 and 1 is primarily due to the limited number of robots operating concurrently—typically between one and three—making it difficult to distinguish random behavior from deliberate decisions. In contrast, Algorithm 2 demonstrates a markedly longer system lifetime by leveraging more comprehensive information, resulting in more efficient replacements.

This platform provides a structured foundation for ongoing research, enabling both the validation of current strategies and the exploration of improvements in replacement logic, communication constraints, and integration with real-world systems.

Looking ahead, we recognize the potential of the Agent-based approach, particularly when enhanced with richer information. However, the complexity of distributed implementation motivates us to pursue an intermediate step in our future work. Specifically, we aim to augment SPF-based strategies with role-based or hierarchical mechanisms, which may offer more straightforward and scalable deployment in real-world distributed environments.

A second line of work focuses on improving our network simulation capabilities. We are currently developing a contribution that evaluates link–local and mesh–local broadcast communications using standard radio technologies. This includes experiments on a physical testbed where we assess additional parameters such as latency and energy consumption. These insights will feed back into our simulation models and help guide the integration of a more feature-rich network simulator in the near future. This refinement will also allow us to reassess the assumptions made in this study and evaluate whether the required communication for each MWSN phase should be further constrained.

Finally, the last step in our roadmap involves the integration of Hardware-in-the-Loop (HIL) scenarios. In this setup, one or more physical robots interact with simulated agents, enabling real-world data to dynamically refine simulation parameters. As outlined in [47], this hybrid approach will further improve the realism and applicability of our models, bridging the gap between simulation and deployment.

Author Contributions: Conceptualization, M.F. and E.G.-P.; methodology, J.-B.C.-S. and J.-M.C.-G.; software, J.-B.C.-S.; validation, J.-M.C.-G., E.G.-P. and M.F.; formal analysis, J.-B.C.-S. and M.F.; investigation, J.-B.C.-S.; data curation, J.-B.C.-S.; writing—original draft preparation, J.-B.C.-S.; writing—review and editing, E.G.-P., J.-M.C.-G. and M.F.; visualization, J.-B.C.-S. and E.G.-P.; supervision, E.G.-P., M.F. and J.-M.C.-G. All authors have read and agreed to the published version of the manuscript.

Funding: This research and the APC were funded by MCIN/AEI/10.13039/501100011033 under Project TED2021-130456B-I00 by "NextGenerationEU/PRTR" Program.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The original data presented in the study are openly available in https://github.com/DIANA-IoT/.github.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

Acronym	Meaning
WSN	Wireless sensor network
ROS	Robot operating system
SPF	Social potential forces
MWSN	Mobile wireless sensor Network
CPU	Central processing Unit
$w(t)$	Willingness to interact
$m(d_m)$	Battery consumption due to robot movement (%)
d_m	Distance travelled (m)
K_m	Motor drain at constant velocity (mAh/m)
B_{cp}	Battery capacity (mAh)
b_c	Current battery percentage
$U_p(n)$	Energy expenditure to send a packet from n (mAh)
R_{packet}	Battery drain to send a single packet (mAh)
R_{incdis}	Increase in battery drainage according to distance (mAh/m)
$Range(n)$	Radio range (m)
$P(n)$	Number of packets sent during a network cycle
K_{duty}	Constant for battery drainage due to transceiver waking-up (mAh)
bl_0	Critical battery level (%)
bl_h, bl_1	Negotiation battery thresholds (%)
w_H, w_{H20}	Negotiation willingness thresholds
$u(t)$	Negotiation replacement utility
$W(t)$	Corrected utility
b_n, b_p, b_m	Battery expenditure to reach certain points (%)
d_{ij}, d_{ip}, d_{iz}	Distances from i to j, to a point p or to centre point (z)
n_{fail}, $n_{closeAP}$	Number of depleted robots, number of alive robots within AP range
p_i	Probability of detecting a connectivity event at cell i
p_{ij}	Probability of j detecting a connectivity event at cell i
C	Total coverage (%)

References

1. Yetgin, H.; Cheung, K.T.K.; El-Hajjar, M.; Hanzo, L. A Survey of Network Lifetime Maximization Techniques in Wireless Sensor Networks. *IEEE Commun. Surv. Tutor.* **2017**, *19*, 828–854. [CrossRef]
2. Akyildiz, I.F.; Wang, X.; Wang, W. Wireless mesh networks: A survey. *Comput. Netw.* **2005**, *47*, 445–487. [CrossRef]
3. Ghosh, A.; Das, S.K. Coverage and connectivity issues in wireless sensor networks: A survey. *Pervasive Mob. Comput.* **2008**, *4*, 303–334. [CrossRef]
4. Yang, Y.; Fonoage, M.I.; Cardei, M. Improving network lifetime with mobile wireless sensor networks. *Comput. Commun.* **2010**, *33*, 409–419. [CrossRef]
5. Huang, H.; Savkin, A.V.; Ding, M.; Huang, C. Mobile robots in wireless sensor networks: A survey on tasks. *Comput. Netw.* **2019**, *148*, 1–19. [CrossRef]
6. Chen, M.; Kwon, T.; Yuan, Y.; Leung, V.C. Mobile Agent Based Wireless Sensor Networks. *J. Comput.* **2006**, *1*, 14–21. [CrossRef]
7. Ghadi, Y.Y.; Mazhar, T.; Shloul, T.A.; Shahzad, T.; Salaria, U.A.; Ahmed, A.; Hamam, H. Machine Learning Solutions for the Security of Wireless Sensor Networks: A Review. *IEEE Access* **2024**, *12*, 12699–12719. [CrossRef]
8. Behiry, M.H.; Aly, M. Cyberattack detection in wireless sensor networks using a hybrid feature reduction technique with AI and machine learning methods. *J. Big Data* **2024**, *11*, 16. [CrossRef]
9. Saranya, M.; Srevarshine, S.; Sujitha, V.; Sobiya, T. Securing Wireless Sensor Networks from Intrusions Using Machine Learning-Based Detection and Response. In Proceedings of the 2025 International Conference on Multi-Agent Systems for Collaborative Intelligence (ICMSCI), Erode, India, 20–22 January 2025; IEEE: New York, NY, USA, 2025; pp. 236–241. [CrossRef]
10. Zhou, Z. Distributed WSN Vulnerability Remediation System Based on Mobile-N Policy. *Int. J. Inf. Secur. Priv.* **2025**, *19*, 1–25. [CrossRef]
11. Temene, N.; Sergiou, C.; Georgiou, C.; Vassiliou, V. A Survey on Mobility in Wireless Sensor Networks. *Ad Hoc Netw.* **2022**, *125*, 102726. [CrossRef]

12. Oitzman, M. 2022 ROS 2 Metrics Report. 2023. Available online: https://www.therobotreport.com/2022-ros-2-metrics-report/ (accessed on 10 April 2025).
13. ROS Discourse. 2022 ROS Metrics Report. 2023. Available online: http://download.ros.org/downloads/metrics/metrics-report-2022-07.pdf (accessed on 10 April 2025).
14. Jamshed, M.A.; Ali, K.; Abbasi, Q.H.; Imran, M.A.; Ur-Rehman, M. Challenges, Applications, and Future of Wireless Sensors in Internet of Things: A Review. *IEEE Sens. J.* **2022**, *22*, 5482–5494. [CrossRef]
15. Majid, M.; Habib, S.; Javed, A.R.; Rizwan, M.; Srivastava, G.; Gadekallu, T.R.; Lin, J.C.W. Applications of Wireless Sensor Networks and Internet of Things Frameworks in the Industry Revolution 4.0: A Systematic Literature Review. *Sensors* **2022**, *22*, 2087. [CrossRef] [PubMed]
16. Rashid, B.; Rehmani, M.H. Applications of wireless sensor networks for urban areas: A survey. *J. Netw. Comput. Appl.* **2016**, *60*, 192–219. [CrossRef]
17. Saha, P.; Kumar, V.; Kathuria, S.; Gehlot, A.; Pachouri, V.; Duggal, A.S. Precision Agriculture Using Internet of Things and Wireless Sensor Networks. In Proceedings of the 2023 International Conference on Disruptive Technologies (ICDT), Greater Noida, India, 11–12 May 2023; IEEE: New York, NY, USA, 2023; pp. 519–522. [CrossRef]
18. Videgain, M.; Martínez-Casasnovas, J.A.; Vigo-Morancho, A.; Vidal, M.; García-Ramos, F.J. On-farm experimentation of precision agriculture for differential seed and fertilizer management in semi-arid rainfed zones. *Precis. Agric.* **2024**, *25*, 3048–3069. [CrossRef]
19. Musa, P.; Sugeru, H.; Wibowo, E.P. Wireless Sensor Networks for Precision Agriculture: A Review of NPK Sensor Implementations. *Sensors* **2023**, *24*, 51. [CrossRef]
20. EFR32MG24 Series 2 Multiprotocol Wireless SoC—Silicon Labs. Available online: https://www.silabs.com/wireless/zigbee/efr32mg24-series-2-socs (accessed on 10 April 2025).
21. nRF5340 Product Specification. Available online: https://mm.digikey.com/Volume0/opasdata/d220001/medias/docus/6470/NRF5340-DK.pdf (accessed on 10 April 2025).
22. Nurcan-Atceken, D.; Altin-Kayhan, A.; Tavli, B. A novel differentiated coverage-based lifetime metric for wireless sensor networks. *Ad Hoc Netw.* **2024**, *164*, 103636. [CrossRef]
23. Dao, T.C.; Tam, N.T.; Binh, H.T.T. Node depth Representation-based Evolutionary Multitasking Optimization for Maximizing the Network Lifetime of Wireless Sensor Networks. *Eng. Appl. Artif. Intell.* **2024**, *128*, 107463. [CrossRef]
24. Anand, R.; Singh, J.; Pandey, D.; Pandey, B.K.; Nassa, V.K.; Pramanik, S. Modern Technique for Interactive Communication in LEACH-Based Ad Hoc Wireless Sensor Network. In *Software Defined Networking for Ad Hoc Networks* ; EAI/Springer Innovations in Communication and Computing; Springer: Cham, Switzerland, 2022; pp. 55–73. [CrossRef]
25. Sachan, S.; Sharma, R.; Sehgal, A. Energy efficient scheme for better connectivity in sustainable mobile wireless sensor networks. *Sustain. Comput. Inform. Syst.* **2021**, *30*, 100504. [CrossRef]
26. Piltyay, S.; Bulashenko, A.; Demchenko, I. Wireless Sensor Network Connectivity in Heterogeneous 5G Mobile Systems. In Proceedings of the 2020 IEEE International Conference on Problems of Infocommunications Science and Technology, PIC S and T 2020, Kharkiv, Ukraine, 6–9 October 2020 ; pp. 625–630. [CrossRef]
27. Di Francesco, M.; Das, S.K.; Anastasi, G. Data Collection in Wireless Sensor Networks with Mobile Elements. *ACM Trans. Sens. Netw. (TOSN)* **2011**, *8*, 1–31. [CrossRef]
28. Natalizio, E.; Loscrí, V. Controlled mobility in mobile sensor networks: Advantages, issues and challenges. *Telecommun. Syst.* **2013**, *52*, 2411–2418. [CrossRef]
29. Lin, C.; Han, G.; Qi, X.; Du, J.; Xu, T.; Martinez-Garcia, M. Energy-Optimal Data Collection for Unmanned Aerial Vehicle-Aided Industrial Wireless Sensor Network-Based Agricultural Monitoring System: A Clustering Compressed Sampling Approach. *IEEE Trans. Ind. Inform.* **2021**, *17*, 4411–4420. [CrossRef]
30. Heinrich, M.K.; Wahby, M.; Dorigo, M.; Hamann, H. Swarm Robotics. In *Cognitive Robotics*; MIT Press: Cambridge, MA, USA, 2022; pp. 77–98. [CrossRef]
31. Roldán, J.J.; Garcia-Aunon, P.; Garzón, M.; de León, J.; del Cerro, J.; Barrientos, A. Heterogeneous Multi-Robot System for Mapping Environmental Variables of Greenhouses. *Sensors* **2016**, *16*, 1018. [CrossRef] [PubMed]
32. Landolsi, T.; Sagahyroon, A.; Mirza, M.; Aref, O.; Maki, F.; Maki, S. Pollution monitoring system using position-aware drones with 802.11 Ad-Hoc networks. In Proceedings of the 2018 IEEE Conference on Wireless Sensors, ICWiSe 2018, Langkawi, Malaysia, 21–22 November 2018; pp. 40–43. [CrossRef]
33. Tang, H.; Zhou, G.; Bai, D.; Li, M.; Zhang, X.; Tang, C. Deployment Method of Wireless Sensor Networks for Coal Mine Safety Monitoring Based on Multi Robot Systems. In Proceedings of the 2023 IEEE International Conference on Robotics and Biomimetics, ROBIO 2023, Koh Samui, Thailand, 4–9 December 2023. [CrossRef]
34. Weiss, G. *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*; MIT Press: Cambridge, MA, USA, 1999.

35. Koutsoubelias, M.; Lalis, S. TeCoLa: A programming framework for dynamic and heterogeneous robotic teams. In Proceedings of the 13th International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services, Hiroshima Japan , 28 November–1 December 2016; ACM International Conference Proceeding Series; pp. 115–124. [CrossRef]
36. Aguzzi, G.; Casadei, R.; Viroli, M. MacroSwarm: A Field-Based Compositional Framework for Swarm Programming. In *Coordination Models and Languages* ; Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics); Springer: Cham, Switzerland, 2023; Volume 13908, pp. 31–51. [CrossRef]
37. Kosak, O.; Huhn, L.; Bohn, F.; Wanninger, C.; Hoffmann, A.; Reif, W. Maple-Swarm: Programming Collective Behavior for Ensembles by Extending HTN-Planning. In *Leveraging Applications of Formal Methods, Verification and Validation: Engineering Principles* ; Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics); Springer: Cham, Switzerland, 2020; Volume 12477, pp. 507–524. [CrossRef]
38. Carroll, M.; Namjoshi, K.S.; Segall, I. The Resh Programming Language for Multirobot Orchestration. In Proceedings of the 2021 IEEE International Conference on Robotics and Automation (ICRA), Xi'an, China, 30 May–5 June 2021.
39. Fernandez-Cortizas, M.; Molina, M.; Arias-Perez, P.; Perez-Segui, R.; Perez-Saura, D.; Campoy, P. Aerostack2: A Software Framework for Developing Multi-robot Aerial Systems. *arXiv* **2023**, arXiv:2303.18237.
40. Kaiser, T.K.; Begemann, M.J.; Plattenteich, T.; Schilling, L.; Schildbach, G.; Hamann, H. ROS2SWARM—A ROS 2 Package for Swarm Robot Behaviors. In Proceedings of the IEEE International Conference on Robotics and Automation, Philadelphia, PA, USA, 23–27 May 2022; pp. 6875–6881. [CrossRef]
41. Bareis, A.; Bareis, M.; Bettstetter, C. Robots that Sync and Swarm: A Proof of Concept in ROS 2. In Proceedings of the International Symposium on Multi-Robot and Multi-Agent Systems, MRS 2019, New Brunswick, NJ, USA, 22–23 August 2019; pp. 98–104. [CrossRef]
42. Urdiales, C.; Aguilera, F.; González-Parada, E.; Cano-García, J.; Sandoval, F. Rule-Based vs. Behavior-Based Self-Deployment for Mobile Wireless Sensor Networks. *Sensors* **2016**, *16*, 1047. [CrossRef]
43. Frasheri, M.; Cano-García, J.; González-Parada, E.; Çürüklü, B.; Ekström, M.; Papadopoulos, A.V.; Urdiales, C. Adaptive autonomy in wireless sensor networks. In Proceedings of the 19th International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS'20), Auckland, New Zealand, 9–13 May 2020; pp. 375–383.
44. Sadeghi Ghahroudi, M.; Shahrabi, A.; Ghoreyshi, S.M.; Alfouzan, F.A. Distributed Node Deployment Algorithms in Mobile Wireless Sensor Networks: Survey and Challenges. *ACM Trans. Sens. Netw.* **2023**, *19*, 91. [CrossRef]
45. Konovalov, P.; Matveev, A. Efficient algorithms for distributed virtual force-based self-spreading of robotic sensor/actuator networks. In Proceedings of the 2024 7th International Conference on Robotics, Control and Automation Engineering (RCAE), Wuhu, China, 25–27 October 2024; IEEE: New York, NY, USA, 2024; pp. 127–132. [CrossRef]
46. Martín-Rico, F. Chapter: The TF Subsystem. In A Concise Introduction to Robot Programming with ROS 2; CRC Press: Boca Raton, FL, USA, 2023.
47. Zhang, D.; Wu, Z.; Chen, J.; Zhu, R.; Munawar, A.; Xiao, B.; Guan, Y.; Su, H.; Hong, W.; Guo, Y.; et al. Human-Robot Shared Control for Surgical Robot Based on Context-Aware Sim-to-Real Adaptation. In Proceedings of the IEEE International Conference on Robotics and Automation, Philadelphia, PA, USA, 23–27 May 2022; pp. 7694–7700. [CrossRef]