IEEE Robotics and Automation Letters (RAL) paper presented at the
2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 25-29, 2020, Las Vegas, NV, USA (Virtual)

Swarm Relays: Distributed Self-Healing Ground-and-Air Connectivity Chains

Vivek Shankar Varadharajan[1], David St-Onge[2], Bram Adams[1] and Giovanni Beltrame[1]

***Abstract*— The coordination of robot swarms – large decentralized teams of robots – generally relies on robust and efficient inter-robot communication. Maintaining communication between robots is particularly challenging in field deployments where robot motion, unstructured environments, limited computational resources, low bandwidth, and robot failures add to the complexity of the problem. In this paper we propose a novel lightweight algorithm that lets a heterogeneous group of robots navigate to a target in complex 3D environments while maintaining connectivity with a ground station by building a chain of robots. The fully decentralized algorithm is robust to robot failures, can heal broken communication links, and exploits heterogeneous swarms: when a target is unreachable by ground robots, the chain is extended with flying robots. We test the performance of our algorithm using up to 100 robots in a physics-based simulator with three mazes and several robot failure scenarios. We then validate the algorithm with physical platforms: 7 wheeled robots and 6 flying ones, in homogeneous and heterogeneous scenarios in the lab and on the field.**

I. INTRODUCTION

Swarm robotics is a field of engineering studying the use of large groups of simple robots to perform complex tasks [1]. Ideally, a single robot failure in a swarm should not compromise the overall mission because of the inherent redundancy of the swarm [1]. With robustness and scalability, robotic swarms are foreseen as cost effective solution for spatially distributed tasks.

In many such applications, the ability of the swarm to coordinate depends largely on its ability to communicate. A reliable communication infrastructure allows the robots to exchange information at any time. However, real deployments include many potential sources of failures (environmental factors, mobility, wear and tear, etc) that can break connectivity and compromise the mission. In this work, we address complete robot failures caused by a robot's inability to communicate or to be detected by its neighbors. In addition, most realistic applications require the robots to remain connected with an operator at a ground station, to either provide information (e.g. in a disaster response scenario) or to receive new commands (e.g. planetary exploration).

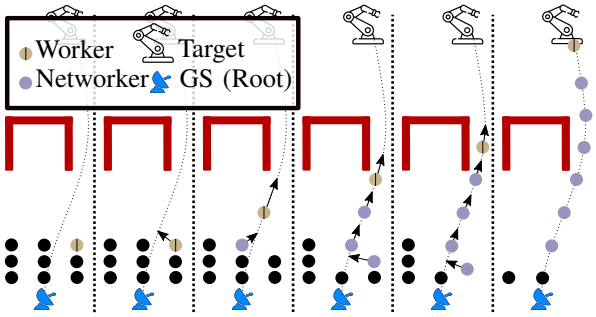

Fig. 1. Progressive formation of a chain to complete a task.

To maintain connectivity, we propose to progressively use the available robots of a swarm to form a self-healing communication chain from a ground station to a target (illustrated in Fig. 1). The operator sets the target and the desired number of redundant links. We target application scenarios including video relay and tele-operation for exploration tasks, where a consistent end-to-end link is required. Our algorithm uses a common path planner to generate a viable path, and builds a chain of robots towards the target. This work extends [2], which briefly presented the self healing aspects of this work without modeling and most experiments. The contributions of this work are: 1) a mathematical formulation of our self-healing chain formation algorithm with configurable redundancy; 2) the performance evaluation of our algorithm in simulated environments of various complexity and integrating robot failures; 3) the validation of the algorithm with a physical swarm of 7 wheeled robots and 6 flying robots, in homogeneous and heterogeneous configurations. This paper is organized as follows: we start with a brief summary of the related work in Section II, then we detail our model in Section III, and our proposed algorithm in Section IV. Finally, section V provides the experimental results.

II. RELATED WORK

There are two general approaches to connectivity maintenance in multi-robot systems: strict end-to-end connectivity [3], [4], or relaxed intermittent connectivity [5], [6], [7]. While the first demands to maintain a link from the source to the sinks (the ends of each branch of the network graph), the latter allows for momentary local breaks in the communication topology. When the mission requires to continuously relay information, like video, operators commands or

This work was supported by NSERC (Strategic Partnership under Grant 479149-2015). Cedar and Graham Compute Canada clusters were used for simulations.

[1]Vivek Shankar Varadharajan, Bram Adams, and Giovanni Beltrame are with the Computer and Software Engineering department, Polytechnique Montreal, Montreal, QC H3T 1J4, Canada (email: vivek-shankar.varadharajan@polymtl.ca, bram.adams@polymtl.ca, g.beltrame@polymtl.ca)

[2]David St-Onge is with the Department of Mechanical Engineering, École de Technologie Supérieure, Montreal, QC H3C 1K3, Canada (email: David.St-Onge@etsmtl.ca)

This letter has a supplementary video available at https://ieeexplore.ieee.org, supplementary material available at https://mistlab.ca/papers/SwarmRelays and source code can be downloaded from https://github.com/MISTLab/SwarmRelays, provided by the authors.

offloaded computation, strict end-to-end connectivity might be preferred; this is the approach we use in this work.

Algebraic Connectivity The problem of preserving end-to-end connectivity is widely discussed in recent literature [3], [4], [8]. Some works use continuous control models with algebraic connectivity [9] and implement mission-related control laws [10], considering robot failures. De Gennaro et al. [11] derive a gradient-based control law from Laplacian matrices' features such as the Fiedler vector to maximize connectivity. Stump et al. [12] also use Fiedler value and k-connectivity to create a bridge between a station and a robot moving towards a target in walled environments. However, the centralized computation of the Laplacian is hardly scalable, and distributed estimations require significant communication bandwidth and are sensitive to noise [13].

Hybrid approaches Connectivity can also be maintained by merging continuous motion controllers and discrete optimization for packet routing [14]. INSPIRE [15] uses a two layer control to preserve connectivity: the first layer applies a potential field local controller to maintain a connected configuration and the second layer optimizes the routing. These methods have long convergence time, and are generally not suitable for large robot groups. Ji and Egerstedt [16] integrate connectivity preservation in two control laws for rendezvous and pattern formation. Their controller does not explicitly take into account obstacles and hence cannot easily be adapted to cluttered environments.

Tree-based approaches Majcherczyk et al. [4] deploy multiple robots towards different targets while preserving connectivity in a decentralized method based on tree construction. However, they carry all available robots along the path using a virtual communication force field. This approach can lead to unwanted redundant sub-structures and recurrent reconfiguration. Hung et al. [17] propose a decentralized global network integrity preservation strategy that performs strategic edge addition and removal to the network to maintain connectivity while reaching its targets. This approach considers coverage missions and decomposes the space into cells from which to select targets. However, distant and sparse targets increase convergence time significantly.

Planner Based Approaches Approaches leveraging a common path planner (centralized [18] or hybrid [3]) determine the optimal communication points for reliable connectivity. These works use a variant of RRT [19] that integrates a communication model to estimate connectivity levels. These methods have realistic communication models but they rely on a centralized solution (a mission planner) and are computationally heavy (they solve a second-order cone program – SOCP). Our approach is fully distributed: we estimate a path using an elected robot whenever a new target becomes available, and navigate while preserving connectivity (similar to [20]). Our method also allows for intermediate robot failures and changes in the environment, which are not considered in the above solutions.

Robustness to Failure Connectivity maintenance for multi-robot systems is a widely covered domain but very few works address robot failures. Some approaches [8] take into account a robustness factor to tackle robot failures but cannot recover a completely disconnected network. A few other approaches consider disconnection and recovery due to environmental mismatches [21] but do not consider complete robot failures. The Wireless Actor Networks domain has several works that address failure recovery [22], e.g. using dynamic programming [23] to reconnect a disjoint graph. These approaches disregard motion planning and have limited applicability in cluttered environments.

Task allocation Allocation of a fixed number of tasks to a set of robots is a well know combinatorial problem and requires heuristics [24] to approximate to a polynomial solution. Decentralized methods use local planners along with consensus algorithms to agree on the context of the task plan [25]. Other approaches compute a plan on each robot and use consensus algorithms to agree on the global assignment [26]. We use an approach similar to the latter to assign roles: a local bid on each robot serves to achieve consensus on the assignments.

Our work leverages the ideas in [17] and [4] to dynamically build structures towards the targets. We sequentially add edges to a tree in a distributed manner by using the path from a standard path planner, as in [3], and reactively enforce connectivity, as in [20]. The final contribution of our approach is the ability to recover from simultaneous robot failures while navigating complex environments with obstacles and preserving a network structure with a configurable number of redundant links. On top of this, our approach uses minimal computation and communication load on the robots, a key aspect for the deployment of other behaviors on top of connectivity maintenance. To the best of the authors' knowledge, this is the first approach that studies this problem in a holistic manner: a path planner, a failure recovery mechanism and a task allocation mechanism to dynamically assign tasks.

III. Preliminaries

Consider a team of N_r robots with their positions denoted by $X = \{x_1, x_2, ..., x_n\}, \forall x_i \in \mathbb{R}^3$. The evolving position of the robots at time t_i can be denoted as $X(t_i) \in \mathbb{R}^{3N}$. Given a set of target locations $\mathbb{T} = \{\tau_1, \tau_2, ..., \tau_n\}, \forall \tau_i \in \mathbb{R}^3$, our objective is to drive the robots to a formation that ensures (a) at least one communication path between any two robots; and (b) each target is within range of at least one robot $\|x_w - \tau_i\| \leq \delta_{tol} \forall \tau_i \in \mathbb{T}$. We consider a single integrator robot model ($\dot{x}_i(t) = u_i$) and assume that the robots are fully controllable with u_i. Taking into account that the robots' workspace, $\mathbb{X} \in \mathbb{R}^3$, is divided into obstacles $\mathbb{X}_{obs}$ and free space $\mathbb{X}_{free}$, we derive the control inputs $u_i(t)$ for all robots, avoiding obstacles and other robots.

A. Communication model

We assume the robots are equipped with a wireless communication device (e.g. 2GHz, 5GHz or 900MHz). The received signal strength is influenced by three main factors: path-loss, shadowing, and fading [27]. We approximate signal strength as a generic function of distance.

Inter-agent communication in a group can be then modeled as a weighted undirected graph $\mathcal{G} = (\nu, \epsilon, A)$, with the node set $\nu = \{r_1, ..r_N\}$ representing the robots, and the edge set $\epsilon = \{e_{ij} | i, j \in \nu, i \neq j\}$, representing communication links. A common approach to working with a communication graph is to use its adjacency matrix A, in which entries e_{ij} represent the probability of robot i decoding j's packets. We aim at maintaining $e_{ij} > e_{min}, \forall e_{ij} \in \epsilon$, that is to guarantee a minimum signal quality.

We consider the robots capable of broadcasting and relaying messages to their neighbors over a limited spherical communication range Z. The robots estimate e_{ij} for their neighbors using local information, if $d_{ij} > Z$ then $e_{ij} = 0$, whereas if $d_{ij} < \delta$ then $e_{ij} = 1$ and $e^{\frac{-5*d_{ij}}{Z}}$ otherwise. δ is a small constant, slightly larger than the radius of the robot. The approximation of connectivity using e_{ij} allows for modeling additive white noise in sensing. The surroundings of robot i are divided into communication zones:

- the safe zone Z_i^s, in which neighbors are considered to have a reliable network link ($e_{ij} > e_{min}$) up to the limit distance d_s with connectivity $e_{ij} = e_{min}$;
- the critical zone Z_i^c, in which neighbors are getting close to the limit of the communication range. In this zone, $e_c \triangleq d_c$ - $d_s > 0$ is defined as the *critical tolerance*. At the critical distance d_c, $e_{ij} < e_{min}$;
- the break-away zone Z_i^b, in which neighbors are expected to break their network link. In this zone, $e_b \triangleq d_b - d_c > 0$ is defined as the *break-away tolerance*. At the break-away distance d_b, $e_{ij} \approx 0$ and at Z, $e_{ij} = 0$ with $e_z \triangleq d_b - Z > 0$.

Let N_i be the neighbor set of robot i, which is divided into: $N_i = N_i^s \cup N_i^c \cup N_i^b$. Robot j is called a safe zone robot of robot i if $x_j \in Z_i^s$, with x_j its position vector. The set of all neighbors within the safe communication zone of robot i is: $N_i^s = \{j | x_j \in Z_i^s, \forall j \in N_i\}$. From this, we define the *safe connectivity set* as the entries e_{ij} of the adjacency matrix $\forall j \in N_i^s$. Similarly, we can define the *critical* and *break-away* connectivity sets with the corresponding entries of the adjacency matrix. These sets allow us to derive control inputs that guarantee the preservation of local connectivity.

B. Local connectivity preservation

We formulate the constraint for the preservation of connectivity among the robots using geometric arguments as in [17]. However, our solution continuously adds edges to the local graph until specific robots with specific roles reach the targets. Our approach reduces the graph construction time, computational load and communication rounds required to determine which edges to remove in [17].

Fig. 2(a) shows the initial position of robot i ($x_i(t)$) and robot j ($x_j(t)$) with their relative distance $d_{ij}(t)$, together with their new position and relative distance after a time Δt. To guarantee the preservation of the communication link, the control must ensure:

$$(\|\Delta x_i\| + \|\Delta x_j\|) \leq d_b - d_{ij}, \quad (1)$$

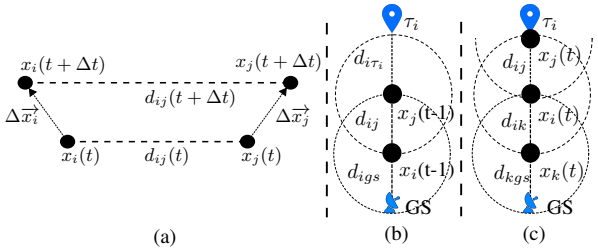

Fig. 2. Variables related to inter-robot distance and robot position before and after a time step Δt (a) illustration of combined distance of a connectivity chain before (b) and after (c) the addition of a new robot.

with d_b, the break-away distance where $e_{ij} = 0$, at which the connectivity breaks. In our implementation we choose to split the responsibility of respecting the available margin $d_b - d_{ij}$ equally among the two robots.

Proof: With $d_{ij}(t + \Delta t)$ the distance between two robots i and j after a time step Δt defined as $d_{ij}(t + \Delta t) = \|x_i(t + \Delta t) - x_j(t + \Delta t)\| = d_{ij}(t) + \|\Delta x_i + \Delta x_j\|$. When we apply (1) we get:

$$d_{ij}(t + \Delta t) \leq d_b. \quad (2)$$

proving that robots i and j stay connected.

We can extend the following remarks considering the three neighborhood sets (safe, critical, break-away):

1) A robot j can be in N_i^s if and only if, its position lies within the safe zone Z_i^s of robot i. This implies that $d_b - d_{ij} \geq e_c + e_b$. If we choose control inputs that allow Δx_i and Δx_j to satisfy the condition of (1), with d_s as the safe limit, the robots will always stay within the safe communication distance:
$$\Delta x_i \leq \frac{d_s - d_{ij}}{2} \quad and \quad \Delta x_j \leq \frac{d_s - d_{ij}}{2} \quad (3)$$
2) For robots $j \in N_i^c$, located in the critical communication zone Z_i^c, if we choose control inputs that allow Δx_i and Δx_j to satisfy the condition of (1), with d_s again as the safe limit, the robots tend to regain safe connectivity:
$$\Delta x_j \leq (d_s - d_{ij}) \quad and \quad \Delta x_i = 0 \quad (4)$$
3) We can apply an identical reasoning for robots $j \in N_i^b$ with their position within and break-away communication zone Z_i^b, have one robot stationary and the other apply a control input Δx_i to satisfy the condition of (1).

By applying control inputs satisfying the condition of Equ. (3) robots in critical/break-away connectivity move towards each other to regain safe connectivity. We use the critical tolerance e_c and break-away tolerance e_b to account for control errors when applying Equ. 4.

C. Global Connectivity Chains

Given a group of robots, we build a chain from a ground station to a target location. To build the chain, we assign roles and relationships to the robots. Robots in the chain are assigned parent-child relationships, to manage the construction of the chain towards a target while maintaining connectivity.

Definition 1: A connectivity chain is a tree C represented as a partially ordered set $(C, <) = \{c_1, c_2, ..., c_n\}$ with $|C| = n$. All vertices $c_i \in C$, have a parent c_{i-1} and a child c_{i+1}, except for c_1, the *root*, that is without parent and c_n, the *worker*, which is without children. All other c_i are *networkers*. The edge set $\epsilon = \{e_{c_i,c_{i+1}} | \forall c_i \in C\}$.

Proposition 1: Given a connectivity chain $(C, <)$ and the distance of the worker c_n to a target $d_{i\tau_i}$, the addition of a new robot between any two robots c_i and c_{i+1} in the chain decreases the distance $d_{i\tau_i}$. Let $d(t-1)$ denote the sum of the distances between all the robots in a chain before the addition of robot c_k, considering the distances of the robots $d_{n-1,n} = d_s \forall n \in (C, <)$. The distance d(t) after the addition of c_k into chain is $d(t) = d(t-1) + d_s$, adding a slack of d_s into the chain, which in turn allows the worker to move towards the target, decreasing $d_{i\tau_i}$, as shown in Fig. 2(b-c). The desired network structure from the root robot to a worker robot is specified as the minimum number of mandatory communication links C_n and the algorithm enforces C_n individual connectivity chains.

D. Task allocation problem

Let the set of free robots $N_f = \{N_r\}/N_c$, with N_r the set of all the available robots and N_c the set of robots in a connectivity chain. The goal of the task allocation algorithm is to assign the set of tasks $\mathbb{T}$ to robots in N_f. The problem of assigning each robot to a single task is commonly refereed to as Single Assignment (SA) problem [26]. The resulting task allocation problem takes the following form:

$$\min \quad \sum_{i=1}^{|N_f|} \sum_{j=1}^{|\mathbb{T}|} c_{ij}(x_i) a_{ij}$$

$$\text{subject to} \sum_{j=1}^{|\mathbb{T}|} a_{ij} \leq 1, \forall i \in N_f \text{ and } \sum_{i=1}^{|N_f|} a_{ij} \leq 1, \forall j \in \mathbb{T}$$

where $c_{ij}(x_i) = ||x_i - \tau_j||$ denotes the cost of robot i at position x_i performing task j and $a_{ij} \in \{0, 1\}$ is the assignment variable indicating the assignment of task j to robot i. The first constraint indicates that each robot can be assigned at most one task and second indicating no two robots get the same task. Consensus-based auction algorithm (CBAA) [26] uses an auction to obtain initial bids from robots and unifies the assignment using a consensus phase. CBAA guarantees convergence and provides a bound on optimality when the cost function follows a diminishing marginal gain [26]. In this work, we use Virtual Sigmergy [28] to share task assignments in a decentralized manner. Virtual Stigmergy is a decentralized information sharing mechanism: once a value is written in the Virtual Stigmergy by a robot, it can be accessed from all other robots.

IV. Proposed Method

Fig. 3 shows the modules of our controller: mobility, connectivity and control policy. The mobility module computes a viable path optimized for path length and stores the result

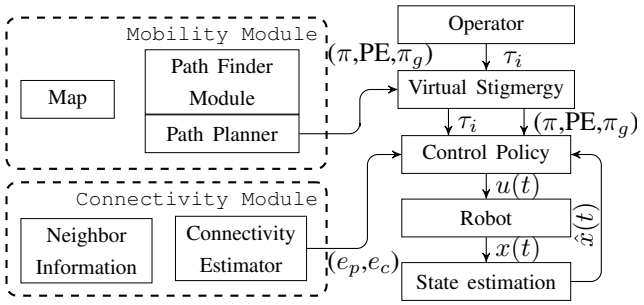

Fig. 3. Control architecture used within the chain construction algorithm.

in the Virtual Stigmergy. The connectivity module continuously estimates the current connectivity between the assigned parent (e_p) and child (e_c). The control policy computes the control inputs using path and connectivity information.

We consider two types of robots in the swarm: *ground robots* with state space $\mathbb{X} \in \mathbb{R}^2$, and *flying robots* with $\mathbb{X} \in \mathbb{R}^3$. We assume the map of the environment is known.

The mobility module consists of a path finder and an optimal path planner. The path finder scans for a viable path by splitting the workspace in cells at a given resolution R, and building a graph $\mathcal{G}_{\mathcal{PE}} = (N, E)$ of the cells. It then performs a depth-first-search in this graph to verify the reachability of target τ_i [29]. If it is found reachable, the module raises the *PE* flag. Ground robots and flying robots can then be used to build a chain on the resulting path π. If a path to τ_i is not found, it either means that the target is unreachable or that the check was done in $\mathbb{R}^2$, but a solution exists in $\mathbb{R}^3$. The path finder can be set to always check in $\mathbb{R}^3$, but that would create overhead for simple cases in $\mathbb{R}^2$. Nevertheless, the resulting path in $\mathbb{R}^3$ has a portion, $\pi_g \subset \pi$, that can be traveled by ground robots: the projection of the 3D path π on the ground plane.

The path planner module computes an optimized path (continuous vector-valued function) $\sigma : [0, T] \rightarrow \mathbb{X}_{free}$ such that $\sigma(0) = x_0$ and $\sigma(T) = \tau_i$. Similar path planning problems have led to several well-studied approaches [19], [30]. When optimizing for the shortest path length, sample-based approaches can quickly get an approximation that is then optimized. We selected RRT^* because it was shown to be faster in large environments and to perform well in cluttered spaces [19]. However, our control architecture is agnostic to the path planner algorithm, and we have tested it with several other planners (SST, BIT^* and PRM^*) with comparable results, as discussed in V.

The chain construction algorithm is implemented in Buzz [31], a programming language for robot swarms, which eases behavior design efforts by providing several swarm programming primitives. For the planners, we integrated classes of Open Motion Planning Library (OMPL) [32] in our architecture to be accessible as Buzz functions.

We assigned one of three roles to the robots: root, worker(s), and networkers. A detailed presentation of the task allocation strategy is given as supplementary material.

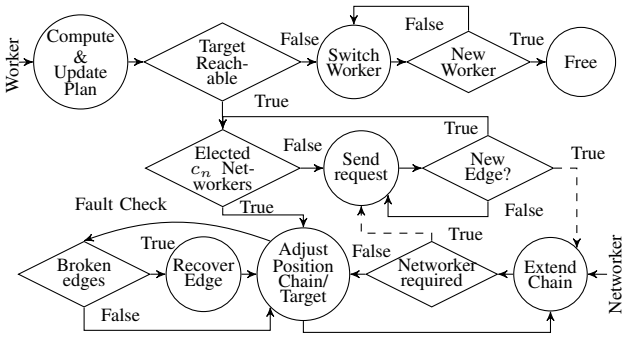

Fig. 4. Interplay between Networkers and Workers.

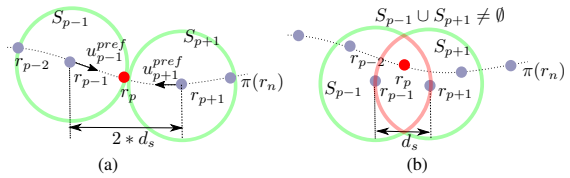

Fig. 5. Failure recovery: (a) at the time of failure and (b) after recovery.

At launch, the root is either pre-determined (ground station) or elected using gradient algorithm (detailed in the supplementary material). Each target also gets a worker robot elected with the same strategy. The gradient algorithm is implemented using Virtual Stigmergy [28].

Fig. 4 shows the interaction of the workers and networkers in the chain formation. The worker robot checks for the existence of a path tuple $\langle \pi, PE, \pi_g \rangle$ in the Virtual Stigmergy; if not available, it computes the path and shares it. The motivation behind using a single robot to compute the plan is twofold: 1) if multiple robots are computing a plan then all plans must be exchanged to reach consensus – a high load of messages/bandwidth; 2) assigning the task to the worker robot also leads to more efficient updates, as the worker will be the first to encounter any obstacles in the environment. In case of a mismatch between the computed path and the explored environment, the path can be adapted.

The path's tuple includes π, the path defined as a sequence of points (states), the *PE* flag and π_g, the projection of π on the ground plane. Following its locomotion type, the robot knows which sections of π it can reach. If the elected worker robot does not have the required locomotion type, the swarm elects a more suitable robot, as shown in Fig. 4. The worker robot determines the number of links C_n required to reach the target and elects the necessary networkers. If more edges are required than the number of robots surrounding the worker, the request is passed down the networkers towards the root, until a free robot (future edge) is found. An optional module, the Way-Point (WP) prediction, grants each robot in the chain to autonomously elect robots to expand the chain without a signal from their child. The number of robots required in a chain $n = ceil(\frac{d_{path}}{d_s})$ is estimated on each robot using the current plan length d_{path}.

A. Motion Control

The motion commands are computed by networkers and workers (the root is fixed) using the available path to extend the chain towards the target(s). The robots that do not belong to any chain are called free robots, and they wait close to the root until they get elected as a worker or a networker. The robots compute the preferred velocity as:

$$u_i^{pref} = \begin{cases} f_c(d_i^C) u_i^{path}(\pi, F) & \text{if } d_i^P \leq d_s \\ u_i^{path}(\pi, B) & \text{otherwise} \end{cases} \quad (6)$$

$$f_c(d_i^C) = \begin{cases} 1 & \text{if } d_i^C \leq d_s \\ 0 & \text{otherwise} \end{cases} \quad (7)$$

where $u_i^{path}(\pi, F)$ computes velocity commands to move the robot towards the target and $u_i^{path}(\pi, B)$ computes velocity commands to move the robot towards the root, both based on the path π. $f_c(d_i^C)$ stops the movement when a child reached the critical communication distance from its parent. In other words, a chain retracts if a robot is in the critical communication zone and expands otherwise. The inputs u_i^{pref} are computed from set point control and altered by a collision avoidance algorithm. Most reactive decentralized collision avoidance can be used within our architecture. We used the Reciprocal Velocity Obstacle (RVO) algorithm [33], which selects the best velocity considering all future potential collisions. The resulting control law is:

$$u_i = \arg \min_{u_i' \in Au_i} \sum_{j \in N_i^{close}} \alpha \frac{1}{tvo_j(u_i')} + \left\| u_i^{pref} - u_i' \right\| \quad (8)$$

$$Au_i = \{u_i' \mid \left\| u_i' \right\| < u_i^{conn}\}, \; u_i^{conn} = \min_{j \in P \cup C} \frac{(d_s - d_{ij})}{2\Delta t}$$

Equation 8 computes the control input from the velocity within the admissible set Au_i that minimize the risk of collision while maximizing the fit to the path. The function $tvo_j(u_i')$ computes the penalty of future collision between i and j while the norm is a penalty for deviation from the preferred velocity on the path u_i^{pref}. α is a scaling factor to balance the penalty terms (set to 1 in our tests). $N_i^{close} = \{j \mid d_{ij} < r_{col} \forall \; j \; \in N_i\}$ is the set of neighbors close enough to be inside the collision radius r_{col}. u_i^{conn} is the maximum velocity allowed, a bound to maintain a safe communication distance according to Equation 1.

B. Self-Healing

We consider two types of robot failures in our work: due to sensing errors and complete robot failures. Both cases create two or more disconnected groups of robots that then need to be reconnected. Dealing with complete failures is harder than sensing errors since no information transfer is possible between each side of the disconnection. Without any means of communication, it is impossible to agree on the reconnection strategy between chain segments. In this work, we get the information required to reconnect the chain in both cases using a periodic broadcast message (detailed in the the supplementary material).

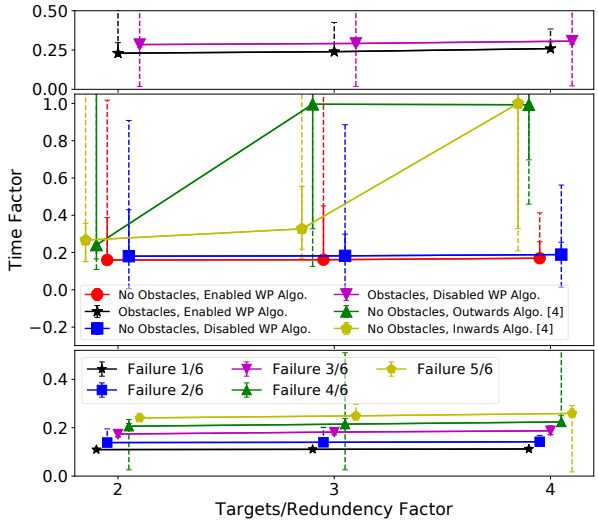

Fig. 6. Middle: comparison of convergence time with [4], convergence time of the chain construction algorithm by enabling and disabling the way-point algorithm that allows the robots in a chain to predict the number of robots required. Top: convergence time by introducing one "C" shaped obstacle. Bottom: comparison of failure recovery time by failing different number of consecutive robots in a chain and creating two subgraphs.

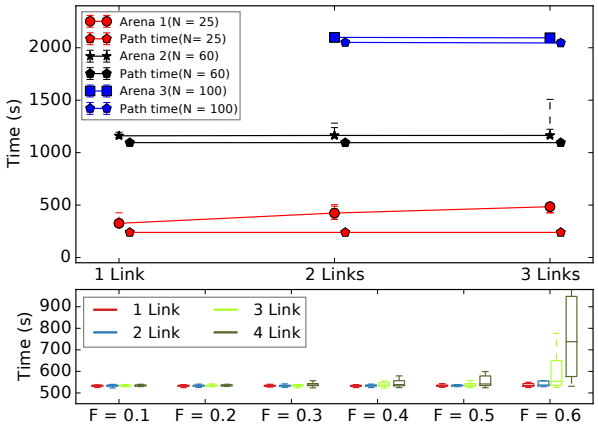

Fig. 7. Top: time taken by 25 (arena 1), 60 (arena 2) and 100 (arena 3) robots to reach the target with 1, 2, and, 3 communication chains. The traveling time on the bottom indicates how fast a robot can travel the path without considering the chain. Bottom: time taken to build the communication with different percentage of random robot failures.

Every robot r_i in a chain C uses a common plan $\pi(r_n)$ updated by worker r_n and shared through Virtual Stigmergy. They are aware of k local edges of their chain C and its current depth through the chain link messages (a sequence of IDs obtained through broadcasts). Every robot can reconstruct the $k*2$ immediate portion of the chain $C_{local} \subseteq C$ by concatenating parent and child chain messages. If a robot in the communication path fails, the communication chain is broken and the chain cannot be completed. In this case, the parent and the child of the failed robot attempt to bridge the broken link by navigating toward each other using the available chain information as illustrated in the Fig. 5. Without loss of generality, we consider an arbitrary robot r_p in a chain failing with a parent r_{p-1} and child r_{p+1}. The result is two disconnected chains $C_1 = \{r_1, ..., r_{p-1}\}$ and $C_2 = \{r_{p+1}, ..., r_n\}$. The robots r_{p-1} and r_{p+1} will attempt to bridge the gap. The safe connectivity workspace $S_i = \{y \in \mathbb{R}^n | \|y - x_i\| \leq d_s\}$ of robot i contains all the states within the safe communication distance d_s. At the time of failure $\{x_{p-1}\} \cap S_{p+1} = \emptyset$ and $\{x_{p+1}\} \cap S_{p-1} = \emptyset$, formally indicating the disconnection. Robots r_{p-1} and r_{p+1} already share consensus on the plan $\pi(r_n)$, when the robot is executing the current segment of the plan. This allows the robots $p-1$ and $p+1$ to compute control inputs $u^{pref}_{p+1} = u^{path}_{p+1}(\pi(r_n), B)$ and $u^{pref}_{p-1} = u^{path}_{p-1}(\pi(r_n), F)$ respectively. When using an identical plan $\pi(r_n)$ and applying u^{path}_i, it is guarantied that the robots will regain safe communication distance ($d_{ij} \leq d_s$) with $x_i \in S_j$. The same reasoning applies to n-consecutive robot failures, with the inter-chain distance between two disconnected chains being $(n+1)*d_s$ instead of $2*d_s$. The two bordering robots in the disconnected chain will compute and apply the control input u^{pref}_i based on $\pi(r_n)$. The boundary robot that lost a child's connection acts as a temporary worker until the chain is healed or it becomes the worker.

V. Experimental Evaluation

Performance comparison We use ARGoS3 [34] physics-based simulator to assess the performance of our method. We compare our results with [4], the closest approach to ours. We set up a simple open environment to compare our algorithm with [4] in 12 conditions: 2 to 4 targets uniformly distributed on a circle ($r = 10m$), with and without obstacles (C-shaped) between the robots and the targets, and with or without the WP prediction algorithm mentioned in Section IV. Each conditions was repeated 35 times with random initialization.

We set the algorithm's parameters to: $d_s = 1.4m$, $d_c = 1.6m$, $d_b = 1.8m$, $v_{max} = 50cm/s$, $\alpha = 1$ and $r_{col} = 25cm$, with a fixed planning time of 2s, a bidding time for the gradient algorithm of 10s, a forgetting time for status messages to 3s and a link failure declaration time of 5s.

Fig. 6 shows the resulting comparison of performance. The metric is *time factor*, i.e. the time taken by the algorithm divided by the traversal time [4]. Our method outperforms both the inwards and outwards algorithms proposed in [4], particularly when increasing the number of targets or redundancy factor. This can be attributed to the fact that all robots are part of the large virtual force field in [4], whereas we form a directed chain with a near-optimal number of robots.

Unlike [4], our chain construction scales well with the number of targets, showing the ability to reach separate targets in parallel. The time factor decreases by 2% without obstacles and 5% with obstacles when using WP prediction.

Robustness We then run the same configuration as above, but inducing up to 80% consecutive robot failures. Fig. 6 (bottom) shows the time factor required by the robots to recover from these broken links. There were in average 6 intermediate networker robots connecting the root and the worker. The failure factor denotes the number of consecutive robots disabled after the chain reached the targets,

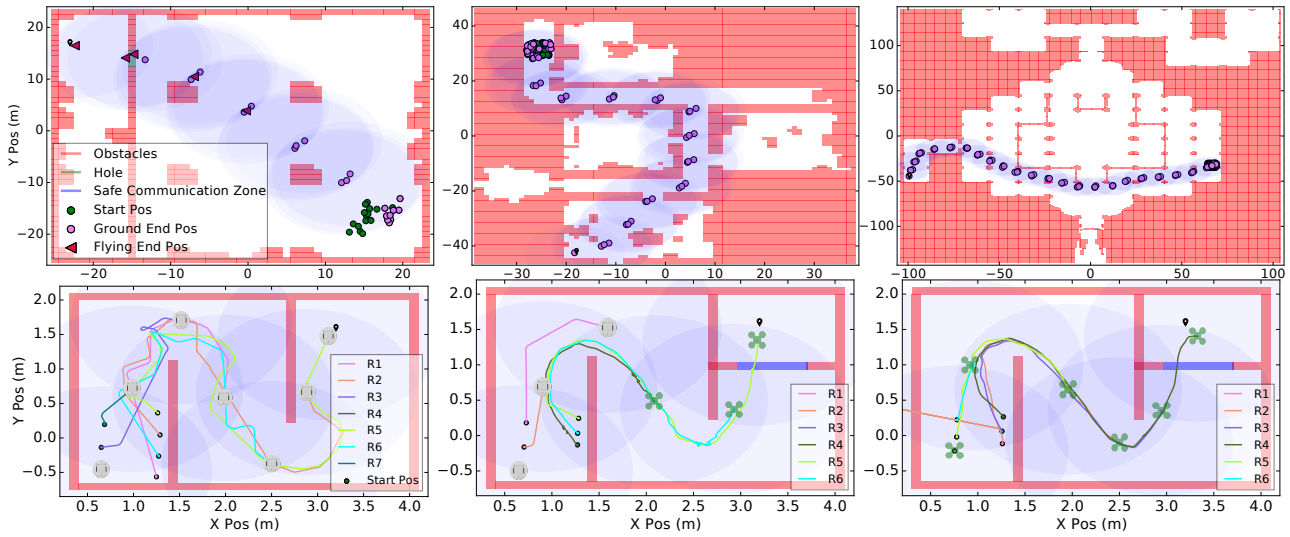

Fig. 8. Top: Illustration of simulated maps with each robot starting position, end position and the safe communication disk region: on the left the smallest arena with flying robots going over an obstructed zone, in the middle a medium-sized arena with ground robots only, and on the right our largest arena, also using only ground robots. Bottom: Real robot experimental arena with start, end, trajectory and end position: on the left 7 khepera IV robots in arena1, in the middle 3 khepera IV robots and 3 CrazyFlie in arena 2 and on the right 6 CrazyFlie in arena 2.

for instance a factor of $5/6$ denotes failing 5 out of 6 robots in a row. Experiments show an increase in 5% of the time ratio for every additional robot failing in the chain. Simultaneously failing a constant fraction of robots leads to almost constant recovery time, as reconnections occur in parallel. The supplementary material provides additional experiments that show our method's robustness to noise.

To further assess the robustness of our method, we used a motion planning benchmarking dataset [35]: 1) small (dragon age/arena), 2) medium (dragon age/den203d), and, 3) large (dragon age/arena2). , using the same parameters as above (except for $d_s = 9.5m$, $d_c = 9.7m$, $d_b = 10m$ to fit the environment) but injecting random robot failures at every control step with an occurrence probably of $p = 0.0005$ and bounded to a maximum percentage of robots failures F from the set $F \in \{0.1, 0.2, 0.3, 0.4, 0.5, 0.6\}$, with $0 \leq F \leq 1$. Fig. 7 (bottom) shows the resulting time taken to build the chain. $F \leq 0.3$ does not have significant impact on the algorithm performance. However, with more failures, completion time increases as well as the performance variability. As more robots fail, the links have to be repaired before proceeding to the target, increasing the convergence time with F.

Scalability We use again the benchmarking dataset [35], but adding a wall with a small window above the ground to the first arena to force the use of flying robots (top leftmost map of Fig. 8). We used 20 ground robots and 5 flying robots for arena 1, 60 ground robots for arena 2, and 100 ground robots for arena 3. For each of the arenas, we enforced 1, 2, and, 3 communication links, except for the 100 robot runs (arena 3) covering only 2 and 3 links. Fig 8 (top) shows that the robots stayed within the safe communication zone even when navigating in narrow corridors. Fig. 7 (top) plots the time taken by the robots to build 1, 2, and, 3 links in each of the arenas, along with the reference traveling times computed using the path length and the maximum velocity of the robots (set again to 10 cm/s). As expected, the time taken to build a chain increases with the size of the arena, but the average time to build different number of links is similar across different arenas. In particular, when computing the time factor, the median time ratio for N=60 is 0.106 and 0.102 for N=100, for all types of links. This proves our claim that the proposed approach spends on average equal amount of time reaching multiple targets with single or multiple links. However, in arena 1 the chain construction time increases with the number of links: this is due to the need of a specific type of robot (flying) to cross a section of the arena. On average, traversal time increases by 2% for every additional heterogeneous robot chain.

Real robot experiments We validated the performance of our implementation using 6 Crazyflie flying robots and 7 Khepera IV ground robots. We performed experiments in three different settings with two different arena configurations as shown in Fig. 8: 1) 7 ground robots in arena 1, 2) 3 ground robots and 3 flying robots in arena 2; arena 2 has a wall containing a hole before the target and can be only reached by a flying robot, 3) 6 flying robots in arena 2). The ground robots run an instance of the Buzz Virtual Machine (BVM) on-board to perform both control and path planning, whereas the flying robots use PyBuzz, a decentralized infrastructure detailed in [36]. We use a motion capture camera system emulating situated communication for state estimation. For these experiments, we set the design parameters to $d_s = 1.2m$, $d_c = 1.4m$, $d_b = 1.5m$ and $v_{max} = 1.5m/s$ to match the size of our robots and arena.

Fig. 8 (bottom) shows one of the 10 test runs we performed in each of the three different settings. The figure shows the starting point, the trajectory, and the end position of the robots. On all runs the robots were able to build a

communication chain and reach the target. Ground robots, heterogeneous team and flying robots on an average took 100s, 50s and 40s respectively to reach the targets (as discussed in the supplementary material). Finally, we ported the algorithm to our ROS based infrastructure [37] for an outdoor test on a group of 4 larger quadcopters. The outdoor robots used GPS and communicated through an Xbee mesh network. This deployment was made to show the practical use of our algorithm on commercial hardware.

VI. Conclusions

We present a communication chain construction algorithm for a heterogeneous swarm of robots. Our approach is completely decentralized: it requires only relative and local information from neighbors. To tackle with robot failures, we exchange information and bridge the chain as soon as it is broken. We assess the algorithm performance with extensive simulations on up to 100 robots in five different arenas. Real robot experiments with flying and ground robots demonstrated the usability and robustness of the approach.

References

[1] M. Brambilla, E. Ferrante, M. Birattari, and M. Dorigo, "Swarm robotics: a review from the swarm engineering perspective," *Swarm Intelligence*, vol. 7, no. 1, pp. 1–41, Mar 2013.

[2] V. S. Varadharajan, B. Adams, and G. Beltrame, "The unbroken telephone game: keeping swarms connected," in *International Conference on Autonomous Agents and MultiAgent Systems*, 2019, pp. 2241–2243.

[3] J. Stephan, J. Fink, V. Kumar, and A. Ribeiro, "Concurrent Control of Mobility and Communication in Multirobot Systems," *IEEE Transactions on Robotics*, vol. 33, no. 5, pp. 1248–1254, 2017.

[4] N. Majcherczyk, A. Jayabalan, G. Beltrame, and C. Pinciroli, "Decentralized connectivity-preserving deployment of large-scale robot swarms," *arXiv preprint arXiv:1806.00150*, 2018.

[5] Y. Kantaros, M. Guo, and M. M. Zavlanos, "Temporal Logic Task Planning and Intermittent Connectivity Control of Mobile Robot Networks," *IEEE Transactions on Automatic Control*, vol. 64, no. 10, pp. 4105–4120, 2019.

[6] M. Guo and M. M. Zavlanos, "Distributed data gathering with buffer constraints and intermittent communication," in *International Conference on Robotics and Automation*. IEEE, 2017, pp. 279–284.

[7] G. Hollinger and S. Singh, "Multi-robot coordination with periodic connectivity," *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 4457–4462, 2010.

[8] J. Panerati, M. Minelli, C. Ghedini, L. Meyer, M. Kaufmann, L. Sabattini, and G. Beltrame, "Robust connectivity maintenance for fallible robots," *Autonomous Robots*, pp. 1–19, 2018.

[9] L. Sabattini, N. Chopra, and C. Secchi, "On decentralized connectivity maintenance for mobile robotic systems," in *Decision and Control and European Control Conference (CDC-ECC), 2011 50th IEEE Conference on*. IEEE, 2011, pp. 988–993.

[10] L. Siligardi, J. Panerati, M. Kaufmann, M. Minelli, C. Ghedini, G. Beltrame, and L. Sabattini, "Robust area coverage with connectivity maintenance," in *International Conference on Robotics and Automation*, 2019, pp. 2202–2208.

[11] M. C. De Gennaro and A. Jadbabaie, "Decentralized control of connectivity for multi-agent systems," in *Decision and Control, 2006 45th IEEE Conference on*. IEEE, 2006, pp. 3628–3633.

[12] E. Stump, A. Jadbabaie, and V. Kumar, "Connectivity management in mobile robot teams," in *2008 IEEE International Conference on Robotics and Automation*, 2008, pp. 1525–1530.

[13] J. Panerati, L. Gianoli, C. Pinciroli, A. Shabah, G. Nicolescu, and G. Beltrame, "From swarms to stars: Task coverage in robot swarms with connectivity constraints," in *2018 IEEE International Conference on Robotics and Automation*, 2018, pp. 7674–7681.

[14] M. M. Zavlanos, A. Ribeiro, and G. J. Pappas, "Network integrity in mobile robotic networks," *IEEE Transactions on Automatic Control*, vol. 58, no. 1, pp. 3–18, 2013.

[15] R. K. Williams, A. Gasparri, and B. Krishnamachari, "Route swarm: Wireless network optimization through mobility," in *International Conference on Intelligent Robots and Systems*, 2014, pp. 3775–3781.

[16] M. Ji and M. Egerstedt, "Distributed coordination control of multiagent systems while preserving connectedness," *IEEE Transactions on Robotics*, vol. 23, no. 4, pp. 693–703, 2007.

[17] P. D. Hung, T. Q. Vinh, and T. D. Ngo, "Hierarchical distributed control for global network integrity preservation in multirobot systems," *IEEE Transactions on Cybernetics*, pp. 1–14, 2019.

[18] J. Fink, A. Ribeiro, and V. Kumar, "Robust control of mobility and communications in autonomous robot teams," *IEEE Access*, vol. 1, pp. 290–309, 2013.

[19] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *International Journal of Robotics Research*, vol. 30, no. 7, pp. 846–894, 2011.

[20] M. M. Zavlanos and G. J. Pappas, "Distributed connectivity control of mobile networks," *Proceedings of the IEEE Conference on Decision and Control*, vol. 24, no. 6, pp. 3591–3596, 2007.

[21] Y. Marchukov and L. Montano, "Multi-robot coordination for connectivity recovery after unpredictable environment changes," *IFAC-PapersOnLine*, vol. 52, no. 8, pp. 446–451, 2019.

[22] A. A. Abbasi, K. Akkaya, and M. Younis, "A distributed connectivity restoration algorithm in wireless sensor and actor networks," in *32nd IEEE Conference on Local Computer Networks (LCN 2007)*. IEEE, 2007, pp. 496–503.

[23] K. Akkaya, F. Senel, A. Thimmapuram, and S. Uludag, "Distributed recovery from network partitioning in movable sensor/actor networks via controlled mobility," *IEEE Transactions on Computers*, vol. 59, no. 2, pp. 258–271, 2009.

[24] V. Pillac, M. Gendreau, C. Guéret, and A. L. Medaglia, "A review of dynamic vehicle routing problems," *European Journal of Operational Research*, vol. 225, no. 1, pp. 1–11, 2013.

[25] T. Shima, S. J. Rasmussen, and P. Chandler, "UAV Team Decision and Control Using Efficient Collaborative Estimation," *Journal of Dynamic Systems, Measurement, and Control*, vol. 129, no. 5, pp. 609–619, 04 2007. [Online]. Available: https://doi.org/10.1115/1.2764504

[26] H. Choi, L. Brunet, and J. P. How, "Consensus-based decentralized auctions for robust task allocation," *IEEE Transactions on Robotics*, vol. 25, no. 4, pp. 912–926, 2009.

[27] Y. Mostofi, A. Gonzalez-Ruiz, A. Gaffarkhah, and D. Li, "Characterization and modeling of wireless channels for networked robotic and control systems-a comprehensive overview," in *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2009, pp. 4849–4854.

[28] C. Pinciroli, A. Lee-Brown, and G. Beltrame, "A tuple space for data sharing in robot swarms," in *Proceedings of the 9th EAI International Conference on Bio-inspired Information and Communications Technologies*, ser. BICT'15, 2016, pp. 287–294.

[29] L. Zhang, Y. Kim, and D. Manocha, "A simple path non-existence algorithm using c-obstacle query for low dof robots," *Proc. Int. Workshop Alg. Found. Robot.(WAFR)*, pp. 1–16, 2006.

[30] Y. Li, Z. Littlefield, and K. E. Bekris, "Sparse methods for efficient asymptotically optimal kinodynamic planning," *Springer Tracts in Advanced Robotics*, vol. 107, pp. 263–282, 2015.

[31] C. Pinciroli and G. Beltrame, "Buzz: An extensible programming language for heterogeneous swarm robotics," in *International Conference on Intelligent Robots and Systems*, October 2016, pp. 3794–3800.

[32] I. A. Sucan, M. Moll, and L. E. Kavraki, "The open motion planning library," *IEEE Robotics & Automation Magazine*, vol. 19, no. 4, pp. 72–82, 2012.

[33] J. Van den Berg, M. Lin, and D. Manocha, "Reciprocal velocity obstacles for real-time multi-agent navigation," in *2008 IEEE International Conference on Robotics and Automation*. IEEE, 2008, pp. 1928–1935.

[34] C. Pinciroli and et. al., "ARGoS: a modular, parallel, multi-engine simulator for multi-robot systems," *Swarm Intelligence*, vol. 6, no. 4, pp. 271–295, 2012.

[35] N. Sturtevant, "Benchmarks for grid-based pathfinding," *Transactions on Computational Intelligence and AI in Games*, vol. 4, no. 2, pp. 144 – 148, 2012.

[36] R. Cotsakis, D. St-Onge, and G. Beltrame, "Decentralized collaborative transport of fabrics using micro-uavs," in *IEEE/RSJ International Conference on Robotics and Automation (ICRA)*, 2019.

[37] D. St-Onge, V. S. Varadharajan, I. Švogor, and G. Beltrame, "From design to deployment: Decentralized coordination of heterogeneous robotic teams," *Frontiers in Robotics and AI*, vol. 7, p. 51, 2020.