Intermittent Connectivity Maintenance with Heterogeneous Robots using a Beads-on-a-Ring Strategy

Rosario Aragues[1] and Dimos V. Dimarogonas[2]

***Abstract*— We consider a scenario of cooperative task servicing, with a team of heterogeneous robots with different maximum speeds and communication radii, in charge of keeping the network intermittently connected. We abstract the task locations into a $1D$ cycle graph that is traversed by the communicating robots, and we discuss intermittent communication strategies so that each task location is periodically visited, with a worst–case revisiting time. Robots move forward and backward along the cycle graph, exchanging data with their previous and next neighbors when they meet, and updating their region boundaries. Asymptotically, each robot is in charge of a region of the cycle graph, depending on its capabilities. The method is distributed, and robots only exchange data when they meet.**

I. Introduction

Servicing tasks is a core multi–robot application [1]. We consider a cooperative task servicing scenario, with a team of task–robots, visiting different task locations to service them, and a team of communicating–robots in charge of keeping the task locations intermittently connected. When a task–robot wants to propagate and get data updates, it waits at the current task place for a communicating–robot to show up, it exchanges data, and then moves to the next task location. The problem of visiting tasks located on the plane can be abstracted into a $1D$ scenario by building a tree connecting the task locations, and then building a cycle graph on it [2]. We focus on the coordination of the communicating–robots on this cycle graph, which are heterogeneous and have different maximum speeds and communication radii.

Connectivity [3] can be preserved at all times, by keeping the initial set of links with possible link additions [4], [1], or an underlying topology which is updated as robots move [5], [6], [7], [8], or by using global connectivity methods [9], [10], relying on the algebraic connectivity and Fiedler eigenvector. Keeping a network connected at all times may not be possible, depending on the environment size, the number of robots, and their communication capabilities. Intermittent connectivity methods [11] can cope with larger environments, at the cost of performance degradation. The network may be disconnected at every time instant, but it is jointly connected *over time* and infinitely often. One of the notable approaches is [11], where each robot moves forward and backward on one link of the environmental graph, and robots exchange data when they meet at the graph vertices.

We propose an intermittent connectivity strategy where the robots move forward and backward on the $1D$ cycle graph of the environment. Robots which are faster or with larger communicating radius, are in charge of larger regions in the cycle graph. Each robot has exactly two neighbors in the cycle graph, and robots exchange data when they meet at the boundaries of their assigned regions. Unlike [11], here we do not force the number of robots to equal the number of links in the graph, and we take advantage of the heterogeneous capabilities of the robots. In addition, [11] considers simultaneous data updates using several neighbors, whereas here the data exchange takes place between pairs of robots. Thus, we provide a higher flexibility, at the cost of a possible slower data exchange in some specific scenarios.

The proposed method is similar to a *beads–on–a–ring* strategy [12], [13], where each robot moves forward and backward on a specific segment of the ring, impacting with its previous and next neighbors, and exchanging data during the impacts. Whereas in [12], [13] robots synchronize to move at the same speed and to cover segments of equal length, here the aim is that robots are in charge of larger regions, if they are faster or have larger communication radii. In addition, [12], [13] let robots to speed up without restrictions, whereas here robots cannot move faster than their maximum speed.

The main contributions of our paper are: (i) a fully distributed method, which does not depend on a specific number of robots, which takes advantage of the heterogeneous nature of the robots, and which only requires data exchange during robot meetings; and (ii) the proof that, asymptotically, each robot is in charge of a region with a length depending on its maximum speed and communication radius, so that the time to traverse the region is the same for all the robots.

II. Cooperative Task Servicing and Intermittent Connectivity Maintenance

Assume a team of tasks–robots is in charge of servicing some tasks. Task–robots travel to the different locations of the l tasks placed in an environment as in Fig. 1. To provide task–robots with data exchange capabilities, we place in the area a dedicated team of communicating–robots, that communicate among them and arrive at the task locations periodically and infinitely often, as it is required by classical algorithms such as distributed averaging, $\max/\min$ consensus, or flooding. When a task–robot wants to propagate or get data updates, it just waits at its current task location for

*Supported by CAS18/00082, Min. Ciencia, Innovación y Universidades, JIUZ-2017-TEC-01 Univ. Zaragoza, DPI2015-69376-R Min. de Economía y Competitividad, Spain, group DGA_T45-17R, the Swedish Research Council (VR) and the Knut och Alice Wallenberg Foundation (KAW)

[1]R. Aragues is with DIIS Universidad de Zaragoza and Instituto de Investigación en Ingeniería de Aragón I3A, Spain raragues@unizar.es

[2]D. V. Dimarogonas is with the School of Electrical Engineering and Computer Science, KTH, Stockholm, Sweden dimos@kth.se

a communicating–robot to show up, and then, it exchanges data and moves to its next task location.

We build a task–tree (Fig. 1), e.g., a Minimum–distance Spanning Tree (MST), with $l-1$ edges connecting the l nodes with the task locations. There are n communicating–robots (*robots*), with different maximum motion speeds and communication radii, that move forward and backward through the edges of the graph, meeting and exchanging data with their neighbors. We do not make any assumptions on the relation between n and l, and also we do not restrict the robots to remain within one specific edge. Since every scenario with tasks located on a plane can be transformed into a MST and then into a cycle graph, from now on, we will no longer consider the underlying tree structure. We will focus instead on the behavior of the method on the associated cycle graph, using L as the total length of the cycle graph. In the simulations we will represent with a line the robot positions between 0 and L, where L will coincide with 0.

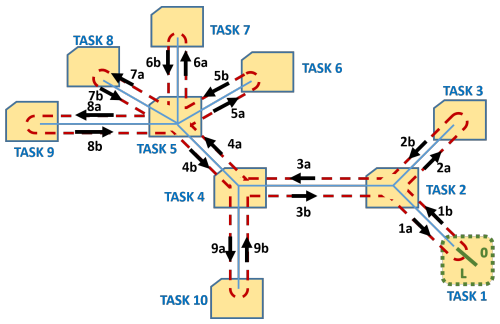

Fig. 1. A task–tree with 9 edges (blue lines) connects the locations of 10 tasks (orange regions). Each link can be traversed in two directions (black arrows), e.g., $2a$, $2b$ between Tasks 2 and 3. The *cycle graph* (red dashed) associated to the task–tree [2] starts at the location of Task 1 (green region), and traverses links $1b, 2a, 2b, \ldots$ and finally, $1a$, getting back to the initial position at the location of Task 1. The length L of the cycle graph equals twice the sum of the lengths of the individual links in the original tree.

III. Notation and Problem Description

We consider n robots moving along the cycle graph. Each robot $i = 1, \ldots, n$ has a communication radius $r_i \in \mathbb{R}_{\geq 0}$ and a maximum motion speed $v_i \in \mathbb{R}_{>0}$, and it is assigned a scalar $p_i(t) \in \mathbb{R}$, which represents its position in the cycle graph, $p_i(t) \in [0, L]$, for $i = 1, \ldots, n$. We assume that the cycle graph cannot be covered by the robots at static positions using their radii r_i, and thus, the intermittent connectivity strategy is required. Robots cannot move faster than their maximum speed. At every time instant, each robot i can move forward, backward, or be stopped. This information is stored in $a_i(t) \in \{0, 1\}$ (a robot is respectively stopped or moving), and $o_i(t) \in \{-1, +1\}$ (it moves respectively backward or forward). Note that stopped robots have an orientation associated. Robots either move at their maximum speeds or remain stopped, so that

$$\dot{p}_i(t) = v_i a_i(t) o_i(t). \tag{1}$$

Each robot $i \in 1, \ldots, n$ has two neighbors, its left $(i-1)$ and right $(i+1)$ neighbor. For the clarity of the presentation, we assume the robot identifiers are sorted according to their position on the cycle graph, from left to right. From now on, $i+1 = 1$ for $i = n$, and $i-1 = n$ for $i = 1$. Between each pair of robot neighbors i and $i+1$, for $i = 1, \ldots, n$ there is a *boundary* $y_i(t)$ (its computation is explained in Section IV). Each robot i is responsible of the region in the cycle graph within its boundaries $y_{i-1}(t)$, $y_i(t)$. Robot i moves forward and backward within its region, until its communication zone reaches its boundaries (Fig. 2). When robot i meets its neighbor $i+1$ at the boundary $y_i(t)$ (or neighbor $i-1$ at boundary $y_{i-1}(t)$) at time t_e, they can exchange data. We let $d_i(t)$ be the *length* of the region associated to a robot i, that depends on its boundaries $y_{i-1}(t)$, $y_i(t)$,

$$d_1(t) = y_1(t), \;\; d_i(t) = y_i(t) - y_{i-1}(t), \;\; i = 2, \ldots, n, \tag{2}$$

and $e_i(t)$ be the *traversing time* that robot i needs to move between its boundaries at maximum speed,

$$e_i(t) = \frac{d_i(t) - 2r_i}{v_i}, \;\; \text{for } i = 1, \ldots, n. \tag{3}$$

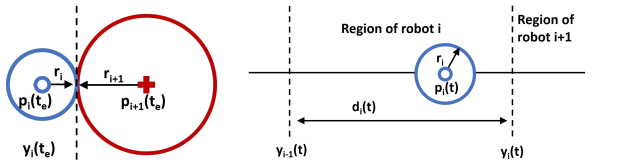

Fig. 2. **Left:** Events like arriving to a boundary and meeting, catching, or discovering a neighbor, take place when the communication regions of robots get in touch, or touch the boundary. **Right:** Region associated to robot i, length $d_i(t)$ of the region, and position of the boundaries $y_{i-1}(t), y_i(t)$.

The goal is to design a strategy that makes the robots meet infinitely often along the cycle graph, so that all the task locations are intermittently connected, and they are periodically revisited, where the worst case *revisiting time* is associated to the tasks placed at the leafs in the tree.

IV. Intermittent Connectivity Maintenance with Heterogeneous Robots

Robots execute the algorithm presented in this section for meeting intermittently, and for computing their boundaries $y_i(t)$. We discuss its properties in Section V. The method roughly consists of each robot i moving until it reaches or defines a boundary, waiting at this boundary until it meets with its neighbor, updating their data, and then moving to its other boundary and repeating the process. We distinguish between the following behaviors for the robots:

- Participating in an *event*: they affect at most two neighbors, and they modify their values of the activity $a_i(t)$, orientation $o_i(t)$, and boundary $y_i(t)$.
- *Between events*: robot positions $p_i(t)$ evolve according to (1). Robots may be in the following states:
  - (STmove) moving from their current position until they arrive to or define a boundary ($a_i(t) = 1$), or
  - (STwait) waiting at a boundary ($a_i(t) = 0$).

States (STmove) give rise to events, and states (STwait) define the type of event. Events have associated a time t_e.

Along the section, we define the types of events, and how robots react to them. We make the following assumptions:

***Assumption** 4.1 (A1, A2):* $(A1)$ Robots n and 1 have a fixed boundary $y_n(t) = L$ for all $t \geq 0$, placed at position L (equivalently, due to the cycle structure, at position 0). $(A2)$ $o_i(0) \neq o_j(0)$ for at least a pair of robots $i, j, i \neq j$.

The proposed algorithm works as follows.

***Algorithm** 4.1 (Initialization):* Robots $i = 1, \ldots, n$ start randomly placed $p_i(0)$ so that their communication zones do not overlap, with initial orientations $o_i(0)$ satisfying $(A2)$, and all are active $a_i(0) = 1$, for all $i = 1, \ldots, n$. Initially, robots do not know their boundaries $y_i(0), i = 1, \ldots, n-1$ but $y_n(0) = L$ from Assumption $(A1)$.

***Algorithm** 4.2 (Discovery and Catch):* Initially, robots move to discover their neighbors and set an initial value for their boundaries. There are two events associated:

Discovery event: Robots i and $i+1$ are moving (STmove), ($a_i(t) = 1$ and $a_{i+1}(t) = 1$), with robot i moving forward and robot $i+1$ moving backward, $o_i(t) > 0, o_{i+1}(t) < 0$, and they do not know $y_i(t)$. The discovery happens when their communication regions get in touch for $t_e \geq t$ (Fig. 2),

$$p_i(t_e) + r_i = p_{i+1}(t_e) - r_{i+1}. \tag{4}$$

At the *discovery*, the involved robots i and $i+1$ initialize their common boundary $y_i(t)$ with this discovery position, they reverse their orientations and move in opposite directions,

$$y_i(t_e^+) = p_i(t_e) + r_i, \quad o_i(t_e^+) = -1, \; o_{i+1}(t_e^+) = +1. \tag{5}$$

Catch event: Robots i and $i+1$ do not know $y_i(t)$. The catcher is robot i when it is active $a_i(t) = 1$, both i and $i+1$ are oriented forward $o_i(t) > 0, o_{i+1}(t) > 0$, and robot $i+1$ is stopped $a_{i+1}(t) = 0$ or moving slower $a_{i+1}(t) = 1, v_{i+1} < v_i$. (Equivalently, the catcher is robot $i+1$ when $a_{i+1}(t) = 1, o_i(t) < 0, o_{i+1}(t) < 0$, and robot i is stopped $a_i(t) = 0$ or moving slower $a_i(t) = 1, v_i < v_{i+1}$). The catch happens when their communication regions get in touch for $t_e \geq t$ as in eq. (4) and Fig. 2. At the *catch*, robots i and $i+1$ initialize their common boundary $y_i(t)$, and the catcher robot (e.g., robot i) remains waiting (STwait) at this boundary,

$$y_i(t_e^+) = p_i(t_e) + r_i, \qquad a_i(t_e^+) = 0. \tag{6}$$

During the first time instants, some robots may be discovering and catching neighbors (Algorithm 4.2), and others may have already discovered them. Once robot i has discovered its two neighbors, it knows $y_{i-1}(t), y_i(t)$, and it moves within these boundaries, updating them, and acting from then on according to the following algorithm.

***Algorithm** 4.3 (Arrivals and meetings):* In this phase, the events that can take place are the following:

Arrival event: Robot i moving backward $a_i(t) = 1$, $o_i(t) < 0$ (or moving forward $a_i(t) = 1$, $o_i(t) > 0$) arrives at its $y_{i-1}(t)$ boundary (or at $y_i(t)$) at the time $t_e \geq t$ when its communication region touches the boundary (Fig. 2), i.e.,

$$y_{i-1}(t_e) = p_i(t_e) - r_i, \; (y_i(t_e) = p_i(t_e) + r_i) \tag{7}$$

After an *arrival* to a boundary, robot i waits at the boundary,

$$a_i(t_e^+) = 0. \tag{8}$$

Note that robot i must already know the boundary (otherwise, it moves to discover or catch its neighbor). The event is arrival if robot i arrives to the boundary and there is no neighbor waiting there. Otherwise, it is a *meeting* event.

Meeting event: Robots i and $i+1$ meet at time $t_e \geq t$ when both of them arrive at their common boundary $y_i(t)$,

$$y_i(t_e) = p_i(t_e) + r_i = p_{i+1}(t_e) - r_{i+1}. \tag{9}$$

At the *meeting*, robots i, $i+1$ update their common boundary:

$$y_i(t_e^+) = \frac{v_{i+1}(y_{i-1}(t_e) + 2r_i) + v_i(y_{i+1}(t_e) - 2r_{i+1})}{v_i + v_{i+1}}. \tag{10}$$

In case they still do not have an initial value for either $y_{i-1}(t)$ or $y_{i+1}(t)$, they keep $y_i(t_e^+)$ unchanged. Then, robots reverse their orientations and get active,

$$o_i(t_e^+) = -1, o_{i+1}(t_e^+) = +1, a_i(t_e^+) = 1, a_{i+1}(t_e^+) = 1. \tag{11}$$

In fact, only i or $i+1$ (the first one that arrived to the boundary) should be inactive. If both arrivals take place at the same time, we establish an order, e.g., to start with the left robot. After the *meeting* and the updates (10), (11), robots i, $i+1$ move away from each other, towards their opposite boundary. E.g., robot i moves towards the common boundary $y_{i-1}(t)$ with its neighbor $i-1$. Note that robots i and $i-1$ must have the same value for $y_{i-1}(t)$, since it can only be updated when both i and $i-1$ meet, and thus it cannot have been changed in the meantime. When robot i gets to the $y_{i-1}(t)$ boundary, a new *arrival* or *meeting* takes place.

V. Properties and Performance Metrics

We discuss some properties of the boundaries $y_i(t)$, $i = 1, \ldots, n$ computed by the robots and the values $y_i^\star$ they asymptotically achieve. Similar to eqs. (2) and (3), we let $d_i^\star$ and $t_\star$ be the length of the region associated to robot i and the traversing time required to move along it at maximum speed, when robots use the asymptotic boundaries $y_i^\star$,

$$\begin{aligned} &d_1^\star = y_1^\star, \quad d_i^\star = y_i^\star - y_{i-1}^\star, \text{ for } i = 2, \ldots, n, \\ &t_\star = (d_i^\star - 2r_i)/v_i, \text{ for } i = 1, \ldots, n. \end{aligned} \tag{12}$$

A. Common traversing time

Consider a cycle graph with length L, traversed by n robots with maximum speeds v_i and communication radii $r_i, i = 1, \ldots, n$. We want to assign regions to each robot i which are disjoint, with the only common point being the boundary, and whose union is the cycle graph $(0 \ldots L)$,

$$d_1^\star + d_2^\star + \cdots + d_n^\star = L. \tag{13}$$

We want the traversing times $t_\star$ employed by each robot i for moving between its boundaries, to be the same, i.e.,

$$t_\star = \frac{d_1^\star - 2r_1}{v_1} = \frac{d_2^\star - 2r_2}{v_2} = \cdots = \frac{d_n^\star - 2r_n}{v_n}. \tag{14}$$

Thus, the common traversing time is given by:

$$(v_1 + \cdots + v_n)t_\star = d_1^\star + \cdots + d_n^\star - 2r_1 - \cdots - 2r_n,$$

$$t_\star = (L - 2\sum_{i=1}^{n} r_i)/(v_1 + v_2 + \cdots + v_n). \tag{15}$$

Having smaller environments, more robots, faster, or with larger communication radii, produce lower common traversing times. From (15), (12), the length $d_i^\star$ of the region associated to robot i, for $i = 1, \ldots, n$, is:

$$d_i^\star = v_i t_\star + 2r_i = \frac{v_i(L - 2\sum_{j=1}^n r_j)}{v_1 + v_2 + \cdots + v_n} + 2r_i, \qquad (16)$$

and the position of each boundary $y_i^\star$, for $i = 1, \ldots, n$, obtained by summing up the region lengths $d_j^\star$, is:

$$y_i^\star = \sum_{j=1}^{i} \frac{v_j(L - 2\sum_{j'=1}^n r_{j'})}{v_1 + v_2 + \cdots + v_n} + 2r_j, \qquad (17)$$

where $y_n^\star$ gives L as expected. Next, we prove that, as robots execute the algorithm in Section IV, their boundaries $y_i(t)$ asymptotically converge to the boundaries $y_i^\star$ in eq. (17) associated to the common traversing time $t_\star$ in eq. (15).

B. Convergence to common traversing times

The proof relies on two facts. First, we rewrite eq. (10) in terms of the traversing times $e_i(t)$ (3) and show that it is an asynchronous weighted consensus method [14], [15]. We prove its convergence, assuming that the topology is jointly connected infinitely often. Then, in Section V-C, we prove that the topology is in fact jointly connected infinitely often.

***Proposition** 5.1 (Weighted consensus on traversing times):* Assume that algorithm (4.3) gives rise to a network which is jointly connected infinitely often. Then, the traversing times $e_i(t)$, region lengths $d_i(t)$, and boundaries $y_i(t)$ (eqs. (3), (2), (10)) asymptotically converge to the goal values $t_\star$, $d_i^\star$, $y_i^\star$ in eqs. (15), (16), (17), for $i = 1, \ldots, n$.

Proof: We first consider how the region lengths $d_i(t)$, $d_{i+1}(t)$ (eq. (2)) evolve when robots i, $i+1$, $i \in 1, \ldots, n-1$, update their boundary $y_i(t_e^+)$ with eq. (10) (the other region lengths are not affected by (10)). If several simultaneous events take place, we will consider any ordering, e.g., first the ones with lower identifiers. Note that (18) will be the same in the presence of simultaneous updates, since it depends on boundaries which require actions from robots i and $i+1$ and, since they are currently involved in their meeting, they cannot be simultaneously involved in other meetings.

$$\begin{aligned} d_i(t_e^+) &= y_i(t_e^+) - y_{i-1}(t_e^+) = y_i(t_e^+) - y_{i-1}(t_e), \quad (18)\\ d_{i+1}(t_e^+) &= y_{i+1}(t_e^+) - y_i(t_e^+) = y_{i+1}(t_e) - y_i(t_e^+). \end{aligned}$$

After some manipulation, (18) is equivalent to:

$$\begin{aligned} d_i(t_e^+) &= \frac{v_{i+1}(d_{i+1}(t_e) + d_i(t_e))}{v_i + v_{i+1}} + 2\frac{v_{i+1}r_i - v_i r_{i+1}}{v_i + v_{i+1}}, \quad (19)\\ d_{i+1}(t_e^+) &= \frac{v_{i+1}(d_{i+1}(t_e) + d_i(t_e))}{v_i + v_{i+1}} - 2\frac{v_{i+1}r_i - v_i r_{i+1}}{v_i + v_{i+1}}. \end{aligned}$$

The traversing times $e_i(t)$, $e_{i+1}(t)$ (eq. (3)) of robots i and $i+1$ are also affected by (10), due to (19) (the other traversing times are not affected by (10)):

$$\begin{aligned} e_i(t_e^+) &= e_i(t_e) + \frac{\epsilon_i}{v_i}(e_{i+1}(t_e) - e_i(t_e)), \ \epsilon_i = \frac{v_i v_{i+1}}{v_i + v_{i+1}},\\ e_{i+1}(t_e^+) &= e_{i+1}(t_e) + \frac{\epsilon_i}{v_{i+1}}(e_i(t_e) - e_{i+1}(t_e)). \qquad (20) \end{aligned}$$

In matrix form, eq. (20) is a discrete–time switching weighted consensus, with Perron matrix [14] P_i as in (24),

$$e(t_e^+) = P_{i(t_e)} e(t_e), \quad e(t) = [e_1(t), \ldots, e_{n-1}(t)]^T. \quad (21)$$

From Proposition 6.1 in the Appendix, if the network is jointly connected infinitely often, then $e(t)$ in (21) converges to $e_\star$ in (34). Thus, for all $i = 1, \ldots, n$, $e_i(t)$ defined by eq. (3) and evolving as in (20), converge to the weighted average of $e_j(0)$, with weighting vector given by $[v_1, \ldots, v_n]^T$,

$$\lim_{t\to\infty} e_i(t) = \frac{\sum_{j=1}^n v_j e_j(0)}{v_1 + \cdots + v_n} = \frac{\sum_{j=1}^n \frac{v_j(d_j(0) - 2r_j)}{v_j}}{v_1 + \cdots + v_n} = t_\star, \quad (22)$$

since $d_1(0) + \cdots + d_n(0) = y_n(t) = L$ (eq. (2)), with $t_\star$ as in (15), and thus, the region lengths $d_i(t)$, and the boundaries $y_i(t)$ converge to the values in (16), (17). ■

C. Joint connectivity

We give some intermediary results to prove that, under our algorithm, the topology is jointly connected infinitely often.

***Lemma** 5.1 ((STmove)):* Behavior (STmove) in Section IV has a bounded time associated, and after that, it always gives rise to an *arrival* or a *meeting* event.

Proof: Since L is fixed, and $y_n(t) = L$ is fixed then, for all $i = 1, \ldots, n$, $d_i(t) \leq L$ and thus $e_i(t) \leq L/v_i$ (3). In (STmove), robots move from a position inside their region to one of their boundaries, employing thus a time $\leq e_i(t)$, which as we saw, is bounded. After that, the event is an *arrival* if the boundary is empty, and a *meeting* if the neighbor is already waiting (STwait) at the boundary. ■

This observation allows us to focus on the behavior of the discrete asynchronous version of the method.

***Definition** 5.1 (Discrete asynchronous behavior):* The discrete asynchronous version of the method, includes only the event times $t_{e_1}, t_{e_2}, \ldots, t_{e_k}, \ldots$. Each robot i is always placed (STwait) at one of its boundaries, $p_i(t_{e_k}) \in \{y_{i-1}(t_{e_k}), y_i(t_{e_k})\}$. The states $y_i(t_{e_k})$, $o_i(t_{e_k})$, $a_i(t_{e_k})$, change due to *meeting* events (10), (11) (equivalently, $d_i(t_{e_k})$, $e_i(t_{e_k})$ (2), (3)). After a meeting between robots i, $i+1$ at time t_{e_k}, two arrival events take place in the future:

$$\begin{aligned} t_{e_{k'}} &= t_{e_k} + e_i(t_{e_k}^+), \qquad p_i(t_{e_{k'}}^+) = y_{i-1}(t_{e_k}), \text{ and}\\ t_{e_{k''}} &= t_{e_k} + e_{i+1}(t_{e_k}^+), \quad p_{i+1}(t_{e_{k''}}^+) = y_{i+1}(t_{e_k}). \quad (23) \end{aligned}$$

Since meeting and discovery events are equivalent, we consider only meetings in what follows.

***Lemma** 5.2 (Properties):* Consider n robots executing algorithm 4.3. The method satisfies the following facts:

- (i) $\sum_{i=1}^n o_i(t)$ remains constant for all t.
- (ii) The regions associated to each robot are disjoint, with the only common point being the boundary.
- (iii) In the discrete asynchronous behavior (Def. 5.1) the order of the robots is preserved.

Proof: (i) *About the orientations*: Orientations $o_i(t)$ only change during *discovery* and *meeting* events and, in both cases (eqs. (5), (11)), the orientations of the two involved agents i and $i+1$ are simultaneously changed.

(ii) *About the regions*: This is true during the *discovery/ catch*, where robots i, $i+1$ define their common boundary at the same time. After, at meetings (eq. (10)) robots i, $i+1$ change simultaneously their common boundary $y_i(t_e^+)$, and the update rule makes $y_i(t_e^+)$ remain strictly between $y_{i-1}(t_e^+) = y_{i-1}(t_e)$ and $y_{i+1}(t_e^+) = y_{i+1}(t_e)$.

(iii) *About robots not exchanging positions*: The region associated to each robot i is defined by the boundaries $y_{i-1}(t)$ and $y_i(t)$. After meeting with robot $i+1$, the position of $y_i(t_e^+)$ changes (eq. (10)), with $y_i(t_e^+) \in [y_{i-1}(t_e), y_{i+1}(t_e)]$. Then, robot i goes to its other boundary $y_{i-1}(t) \leq y_i(t)$, and when later it comes back to boundary $y_i(t)$, it holds $y_i(t) \leq y_{i+1}(t)$, regardless the fact that robot $i+1$ may have updated or not $y_{i+1}(t)$. Thus, robot i will reach $y_i(t)$ and will never reach $y_{i+1}(t)$. Thus, in the discrete asynchronous behavior (Def. 5.1), robots do not exchange the positions. ■

Depending on the relative speeds of robot i and $i+1$, it may be the case that, during (STmove) they exchange positions. E.g., if $y_i(t_e^+) > y_i(t_e)$, and $v_i >> v_{i+1}$, robot i may get to $y_{i-1}(t_e)$ and get back to $y_i(t_e^+)$ before robot $i+1$ has reached $y_i(t_e^+)$. This is temporary: robot i will stop at $y_i(t_e^+)$, but robot $i+1$ will continue (STmove) to $y_{i+1}(t_e) \geq y_i(t_e^+)$. Thus, in the discrete asynchronous behavior (Def. 5.1), the order of the robots is preserved, and robots do not need to e.g., exchange identifiers.

Now, we discuss the joint connectivity of the network. We prove that each robot $i = 1, \ldots, n$ meets its neighbors $i-1$ and $i+1$ after some bounded amount of time.

Proposition *5.2 (Joint connectivity):* Algorithm (4.3) under Assumptions $(A1)$, $(A2)$, gives rise to a network which is jointly connected infinitely often.

Proof: Considering the discrete asynchronous behavior (Def. 5.1) of the algorithm, we represent the system as n boundaries and n robots placed at the boundaries. As we show next, the system cannot experience blocking, and the meetings propagate through the network.

Blocking: The only possible blocking situation is one with each robot waiting (STwait) at a different boundary since, as long as two robots fall in a common boundary, a *meeting* event takes place, making the system evolve. At the blocking, robots with $o_i(t) > 0$ would be at their right boundary, and robots with $o_i(t) < 0$ at their left boundary, with these boundaries being the unique common points between the disjoint regions of each robot (Lemma 5.2(ii)). From Assumption $(A2)$, at least one robot has a different initial orientation than the others, and by Lemma 5.2 (i) this remains like this during all the execution of the method. Thus, at least two robots will fall at the same boundary, giving rise to a *meeting* event.

Propagation: After robots i, $i+1$ meet, they move to their opposite boundaries, thus propagating the process to the preceding and following robots $i-1$ and $i+2$, since the order of the robots is preserved (Lemma 5.2 (iii)). Repeating the same reasoning with robots $i-1$, $i+2$, we conclude that meetings are propagated through the network. The only reasons not to propagate would be either a blocking (we proved this is not possible), or that the same subset of robots would be meeting each other, without involving the others. But in order for $i, i+1$ to meet again, there must have been a meeting between $i-1, i$ and $i+1, i+2$, so this case is discarded as well.

Bounded times: The discrete asynchronous behavior (Def. 5.1) does not exhibit blocking and ensures propagation of the meetings, and behavior (STmove) has a fixed time associated to it (Lemma 5.1). Therefore, the time required for the network to be jointly connected is bounded. ■

D. Traversing, Revisiting, and Inter–meeting times

Up to now, we have presented a method that allows the robots to converge to a configuration with common *traversing times* $t_\star$ (15), i.e., the time each robot i employs to move at its maximum speed between its two boundaries.

Definition *5.2 (Revisiting and Inter–meeting times):* : We define the *revisiting time* as the time required for a robot to visit a particular point in the cycle graph, arriving back at the point with the same orientation as the first time. As a metric for the revisiting time, in our simulations we use the *inter–meeting time* $f_i(t)$, which is the time elapsed between consecutive meetings of robots i and $i+1$, and which is in fact the revisiting time of the $y_i(t)$ boundary.

The asymptotic *revisiting time* t_{rev} includes:

- $2t_\star$ to traverse the robot region in both directions and getting back to the original point with the same orientation, plus
- the time robot waits (STwait) at the boundaries.

In all our simulations, we have observed that, when the number of robots with positive $o_i(t) > 0$ and negative $o_i(t) < 0$ orientations is the same (balanced situation), robots asymptotically achieve a configuration where they never wait at the boundaries, and thus their inter–meeting times $f_i(t)$ converge to the asymptotic revisiting time $t_{\text{rev}} = 2t_\star + 0$. We are currently researching on a formal characterization of these balanced and unbalanced situations and a proof of convergence to these scenarios.

E. Simulations

Figure 3 shows a simulation with $n = 8$ robots traversing a cycle graph with length $L = 1000m$. The black line between position 0 and L represents the position of robots in the cycle graph (Fig. 1). Note that n is not equal to the number of links in the task–tree. Robots have maximum speeds $\{v_1, \ldots, v_n\} = \{0.6, 0.1, 0.5, 0.3, 0.7, 0.2, 0.8, 0.4\}m/s$, and communication radii equal to $20m$ apart from $r_3 = 50m$ and $r_7 = 100m$. They start randomly placed in the cycle graph, with their communication regions non overlapping, and with initial orientations $o_1, \ldots, o_4 = -1$, and $o_5, \ldots, o_8 = +1$. Asymptotically, robots reach a configuration where their regions have common traversing times $t_\star$ which, in this balanced configuration, equals $t_{\text{rev}}/2$. A video can be found at `https://youtu.be/RPggY5A_DOc`.

VI. CONCLUSIONS

We presented a method to ensure a robot team keeps the network intermittently connected. Robots move forward

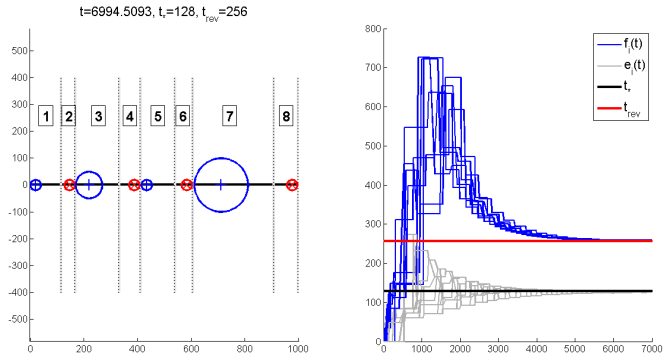

Fig. 3. Robots running the method in Section IV with different maximum speeds and communication radii (circles around the robots). They move forward (red) and backward (blue) at their maximum speeds between their boundaries $y_i(t)$ (black dashed, in vertical), which converge to the boundaries $y_i^\star$ (17)(gray solid, in vertical) associated to the common traversing time $t_\star$ (15). **Left:** Final configuration. **Right:** Evolution of $e_i(t)$ (gray) and $f_i(t)$ (blue) compared to $t_\star$ (black) and $t_{\rm rev}$ (red).

and backward on their regions, meeting intermittently with their previous and next neighbors. Simultaneously, they run a weighted consensus method to update their boundaries, so that the final regions associated to each robot can be traversed by them in a common time, that depends on the robots maximum speeds and communication radii. Future extensions include methods to balance the orientations, formal proofs of synchronization to balanced and unbalanced scenarios, and the consideration of more realistic robot dynamics and communication models.

APPENDIX

In this section, we let matrix V, scalar ϵ_i, and matrices P_i, $\tilde{P}_i$, $\mathcal{L}_i$, $\tilde{\mathcal{L}}_i$ associated to the link $(i, i+1)$, be

$$\begin{aligned} &V = \text{diag}(v_1, \dots, v_n), \quad \epsilon_i = (v_i v_{i+1})/(v_i + v_{i+1}), \\ &P_i = \mathbf{I} - (\text{diag}(v_1, \dots, v_n))^{-1} \epsilon_i \mathcal{L}_i, \\ &\tilde{P}_i = V^{1/2} P_i V^{-1/2} = \mathbf{I} - \tilde{\mathcal{L}}_i, \quad \tilde{\mathcal{L}}_i = V^{-1/2} \epsilon_i \mathcal{L}_i V^{-1/2}, \end{aligned} \tag{24}$$

where the entries in P_i, $\tilde{P}_i$, $\mathcal{L}_i$, and $\tilde{\mathcal{L}}_i$ are given by

$$\begin{aligned} &[P_i]_{j,j'} = 0, \text{ except for} \\ &[P_i]_{i,i} = 1 - (\epsilon_i / v_i), \quad [P_i]_{i+1,i+1} = 1 - (\epsilon_i / v_{i+1}), \\ &[P_i]_{i,i+1} = \epsilon_i / v_i, \quad [P_i]_{i+1,i} = \epsilon_i / v_{i+1}, \\ &[P_i]_{j,j} = 1, \text{ for all } j \neq i, j \neq i+1, \end{aligned} \tag{25}$$

$$\begin{aligned} &[\mathcal{L}_i]_{j,j'} = 0, \text{ except for} \\ &[\mathcal{L}_i]_{i,i} = 1, [\mathcal{L}_i]_{i+1,i+1} = 1, [\mathcal{L}_i]_{i,i+1} = -1, [\mathcal{L}_i]_{i+1,i} = -1, \end{aligned} \tag{26}$$

$$\begin{aligned} &[\tilde{\mathcal{L}}_i]_{j,j'} = 0, \text{ except for} \\ &[\tilde{\mathcal{L}}_i]_{i,i} = v_{i+1}/(v_i + v_{i+1}), \quad [\tilde{\mathcal{L}}_i]_{i+1,i+1} = v_i/(v_i + v_{i+1}), \\ &[\tilde{\mathcal{L}}_i]_{i,i+1} = [\tilde{\mathcal{L}}_i]_{i+1,i} = -\sqrt{v_i}\sqrt{v_{i+1}}/(v_i + v_{i+1}), \end{aligned} \tag{27}$$

$$\begin{aligned} &[\tilde{P}_i]_{j,j'} = 0, \text{ except for} \\ &[\tilde{P}_i]_{i,i} = 1 - \frac{v_{i+1}}{v_i + v_{i+1}}, \quad [\tilde{P}_i]_{i+1,i+1} = 1 - \frac{v_i}{v_i + v_{i+1}}, \\ &[\tilde{P}_i]_{i,i+1} = [\tilde{P}_i]_{i+1,i} = \sqrt{v_i}\sqrt{v_{i+1}}/(v_i + v_{i+1}), \\ &[\tilde{P}_i]_{j,j} = 1, \text{ for all } j \neq i, j \neq i+1. \end{aligned} \tag{28}$$

Lemma 6.1: The eigenvalues matrices P_i, $\tilde{P}_i$ defined in (24), for all $(i, i+1)$ links, with $i = 1, \dots, n-1$, satisfy:

$$\lambda(P_i) \in (-1, 1] \qquad \lambda(\tilde{P}_i) \in (-1, 1]. \tag{29}$$

Proof: The eigenvalues of $\tilde{P}_i$ are

$$\lambda(\tilde{P}_i) = 1 - \lambda(V^{-1/2} \epsilon_i \mathcal{L}_i V^{-1/2}) = 1 - \lambda(\tilde{\mathcal{L}}_i). \tag{30}$$

Note that $\mathcal{L}_i$ (24), (26) is the unweighted symmetric Laplacian matrix associated to the link $(i, i+1)$, and thus it is positive semidefinite [14]. Since $\epsilon_i > 0$, and matrix $V^{-1/2}$ is positive definite and symmetric, then $\tilde{\mathcal{L}}_i = \lambda(V^{-1/2} \epsilon_i \mathcal{L}_i V^{-1/2})$ (24) (27) is positive semidefinite [16, Chapter 7.1], with eigenvalues larger than or equal to 0, and with its largest eigenvalue begin smaller than or equal to the infinite matrix norm $\|\tilde{\mathcal{L}}_i\|_\infty = \max_j(|[\tilde{\mathcal{L}}_i]_{j1}| + \dots + |[\tilde{\mathcal{L}}_i]_{jn}|)$,

$$\begin{aligned} \lambda_{\max}(\tilde{\mathcal{L}}_i) \leq \|\tilde{\mathcal{L}}_i\|_\infty = \frac{\max(v_i, v_{i+1}) + \sqrt{v_i}\sqrt{v_{i+1}}}{v_i + v_{i+1}} \\ \leq (\max(v_i, v_{i+1}) + \max(v_i, v_{i+1}))/(v_i + v_{i+1}) < 2. \end{aligned} \tag{31}$$

From eqs. (30),(31), for all $i = 1, \dots, n-1$,

$$-1 = (1 - 2) < \lambda(\tilde{P}_i) \leq (1 - 0) = 1, \tag{32}$$

and since P_i is *similar* to $\tilde{P}_i$ (24), then P_i and $\tilde{P}_i$ have the same eigenvalues, and we conclude (29). ■

Proposition 6.1: Let matrices $P_{i(t)}$, $\tilde{P}_{i(t)}$ be as in (24). If the sequence of matrices that appear infinitely often are jointly connected, then, for all $z(0)$, $e(0)$, the iterations $z(t^+) = \tilde{P}_{i(t)} z(t)$, $e(t^+) = P_{i(t)} e(t)$, with

$$z(t) = V^{1/2} e(t), \qquad e(t) = V^{-1/2} z(t), \tag{33}$$

and V as in (24), converge respectively to

$$\begin{aligned} z_\star &= (V^{1/2} \mathbf{1}\mathbf{1}^T V^{1/2} / \mathbf{1}^T V \mathbf{1}) z(0), \\ e_\star &= (\mathbf{1}\mathbf{1}^T / \mathbf{1}^T V \mathbf{1}) V e(0). \end{aligned} \tag{34}$$

Proof: Consider the matrix $\tilde{P}_{i_1:i_j}$ associated to a particular jointly connected sequence $i_j \dots i_1$ (the sequence takes place in the opposite order to matrix multiplication),

$$\tilde{P}_{i_1:i_j} = \tilde{P}_{i_1} \tilde{P}_{i_2} \dots \tilde{P}_{i_j}. \tag{35}$$

For this matrix,

$$\begin{aligned} \rho(\tilde{P}_{i_1:i_j}) \leq \|\tilde{P}_{i_1:i_j}\|_2 \leq \|\tilde{P}_{i_1}\|_2 \|\tilde{P}_{i_2}\|_2 \dots \|\tilde{P}_{i_j}\|_2 \\ = \rho(\tilde{P}_{i_1}) \rho(\tilde{P}_{i_2}) \dots \rho(\tilde{P}_{i_j}) = 1, \end{aligned} \tag{36}$$

where we have used Lemma 6.1 ($\lambda(\tilde{P}_i) \in (-1, 1]$ for all $i = 1, \dots, n-1$), and the fact that $\tilde{P}_{i_1}, \tilde{P}_{i_2}, \dots, \tilde{P}_{i_j}$ are symmetric, and their spectral norms equal their spectral radius, $\|\tilde{P}_i\|_2 = \rho(\tilde{P}_i) = \max(|\lambda(\tilde{P}_i)|)$. Thus, all the eigenvalues of matrix $\tilde{P}_{i_1:i_j}$ are between $[-1, +1]$.

Now we pay attention to the structure of matrix $\tilde{P}_{i_1:i_j}$. Every matrix $\tilde{P}_i$ (28) has all the entries equal to zero, but for

the diagonal terms $(1,1)\dots(n,n)$, and the entries $(i,i+1)$, $(i+1,i)$, which are strictly positive. After multiplying matrices $\tilde{P}_{i_1},\dots,\tilde{P}_{i_j}$, we get a nonnegative matrix $\tilde{P}_{i_1:i_j}$ that has *at least* the following elements strictly positive (the remaining entries may be zero, or positive elements): the diagonal terms $(1,1)\dots(n,n)$, and all the (i_1,i_1+1) and (i_1+1,i_1) entries associated to all the i_1,i_1+1 links that appear in each associated matrix P_{i_1}. Since matrix $\tilde{P}_{i_1:i_j}$ contains at least all matrices associated to the $n-1$ different boundaries, then its structure contains *at least* positive elements in all the entries (i,i) for $i=1,\dots,n$, and $(i,i+1)$, $(i+1,i)$ for $i=1,\dots,n-1$. Thus, matrix $\tilde{P}_{i_1:i_j}$ is *primitive* and [17], [16] among its n eigenvalues, there is exactly one with the largest magnitude, and this eigenvalue is the only one possessing an eigenvector with all positive entries, and the remaining $n-1$ eigenvalues are all strictly smaller in magnitude than the largest one. From (36), this eigenvalue has modulus smaller than or equal to 1.

Now note that for each matrix P_i (24),

$$P_i\mathbf{1}=\mathbf{1}, \qquad \text{and that}$$
$$\tilde{P}_{i_1:i_j}=\tilde{P}_{i_1}\tilde{P}_{i_2}\dots\tilde{P}_{i_j}=V^{1/2}P_{i_1}P_{i_2}\dots P_{i_j}V^{-1/2}. \quad (37)$$

From (37), we conclude that $V^{1/2}\mathbf{1}$ is the eigenvector of $\tilde{P}_{i_1:i_j}$ associated to the eigenvalue 1,

$$\tilde{P}_{i_1:i_j}V^{1/2}\mathbf{1}=V^{1/2}\mathbf{1}. \quad (38)$$

This eigenvector has all its entries positive, and it is associated to the largest modulus eigenvalue, which has to be 1 and not -1. Matrix $\tilde{P}_{i_1:i_j}$ is also paracontractive (e.g., [18, Corollary 2], using $\mathrm{span}(V^{1/2}\mathbf{1})$ instead of $\mathrm{span}(\mathbf{1})$).

From [19, Theorem 1] [20, Theorem 2]: suppose that a finite set of square matrices $\{W_1,\dots,W_j\}$ are paracontractive, and denote $\mathcal{J}$ the set of integers that appear infinitely often in the sequence. Then, for all $\tilde{z}(0)$, the sequence of vectors $\tilde{z}(k+1)=W_{i(k)}\tilde{z}(k)$ has a limit $z_\star\in\cap_{i\in\mathcal{J}}\mathcal{H}(W_i)$, with $\mathcal{H}(W_i)=\{z|W_iz=z\}$. In our case, we use the fact that all our possible jointly connected matrices have the common eigenvector $V^{1/2}\mathbf{1}$ associated to the eigenvalue 1 and it is the only one. Thus, $\tilde{z}(k+1)=\tilde{P}_{i_1:i_j}(k)\tilde{z}(k)$, and thus $z(t^+)=\tilde{P}_{i(t)}z(t)$, converge to $z_\star$ in (34). Due to (33), $e(t^+)=P_{i(t)}e(t)$ converges to $e_\star$ (34). ■

REFERENCES

[1] M. Guo, J. Tumova, and D. V. Dimarogonas, "Communication-free multi-agent control under local temporal tasks and relative-distance constraints," *IEEE Transactions on Automatic Control*, vol. 61, no. 12, pp. 3948–3962, 2017.

[2] Y. Gabriely and E. Rimon, "Spanning-tree based coverage of continuous areas by a mobile robot," *Annals of mathematics and artificial intelligence*, vol. 31, no. 1-4, pp. 77–98, 2001.

[3] M. M. Zavlanos, M. B. Egerstedt, and G. J. Pappas, "Graph-theoretic connectivity control of mobile robot networks," *Proceedings of the IEEE*, vol. 99, no. 9, pp. 1525–1540, 2011.

[4] D. Boskos and D. V. Dimarogonas, "Robustness and invariance of connectivity maintenance control for multiagent systems," *SIAM Journal on Control and Optimization*, vol. 55, no. 3, pp. 1887–1914, 2017.

[5] T. Soleymani, E. Garone, and M. Dorigo, "Distributed constrained connectivity control for proximity networks based on a receding horizon scheme," in *American Control Conference*, Chicago, IL, USA, Jul. 2015, pp. 1369–1374.

[6] M. Schuresko and J. Cortes, "Distributed tree rearrangements for reachability and robust connectivity," *SIAM Journal on Control and Optimization*, vol. 50, no. 5, pp. 2588 – 2620, 2012.

[7] M. Aranda, R. Aragues, G. Lopez-Nicolas, and C. Sagues, "Connectivity-preserving formation stabilization of unicycles in local coordinates using minimum spanning tree," in *American Control Conference*, Boston, USA, Jun. 2016, pp. 1968–1974.

[8] H. Poonawala and M. W. Spong, "Preserving strong connectivity in directed proximity graphs," *IEEE Transactions on Automatic Control*, vol. 62, no. 9, pp. 4392–4404, 2017.

[9] A. Gasparri, L. Sabattini, and G. Ulivi, "Bounded control law for global connectivity maintenance in cooperative multirobot systems," *IEEE Transactions on Robotics*, vol. 33, no. 3, pp. 700–717, 2017.

[10] T. Nestmeyer, P. R. Giordano, H. H. Bulthoff, and A. Franchi, "Decentralized simultaneous multi-target exploration using a connected network of multiple robots," *Autonomous Robots*, vol. 41, no. 1, pp. 989–1011, 2017.

[11] Y. Kantaros and M. Zavlanos, "Distributed intermittent connectivity control of mobile robot networks," *IEEE Transactions on Automatic Control*, vol. 62, no. 7, pp. 3109–3121, 2017.

[12] H. Wang and Y. Guo, "Synchronization on a segment without localization: Algorithm, applications, and robot experiments," *Int. Journal of Intelligent Control and Systems*, vol. 15, no. 1, pp. 9–17, 2010.

[13] S. Susca, P. Agharkar, S. Martínez, and F. Bullo, "Synchronization of beads on a ring by feedback control," *SIAM Journal on Control and Optimization*, vol. 52, no. 2, pp. 914–938, 2014.

[14] R. Olfati-Saber, J. A. Fax, and R. M. Murray, "Consensus and cooperation in networked multi-agent systems," *Proceedings of the IEEE*, vol. 95, no. 1, pp. 215–233, 2007.

[15] W. Zhang, Z. Wang, Y. Guo, H. Liu, Y. Chen, and J. Mitola III, "Distributed cooperative spectrum sensing based on weighted average consensus," in *IEEE Global Telecommunications Conf.*, 2011, pp. 1–6.

[16] R. A. Horn, R. A. Horn, and C. R. Johnson, *Matrix analysis*. Cambridge university press, 1990.

[17] A. Jadbabaie, J. Lin, and A. S. Morse, "Coordination of groups of mobile autonomous agents using nearest neighbor rules," *IEEE Transactions on automatic control*, vol. 48, no. 6, pp. 988–1001, 2003.

[18] G. C. Calafiore, "Distributed randomized algorithms for probabilistic performance analysis," *Systems & Control Letters*, vol. 58, no. 3, pp. 202–212, 2009.

[19] L. Elsner, I. Koltracht, and M. Neumann, "On the convergence of asynchronous paracontractions with application to tomographic reconstruction from incomplete data," *Linear Algebra and its Applications*, vol. 130, pp. 65–82, 1990.

[20] L. Xiao, S. Boyd, and S. Lall, "A scheme for robust distributed sensor fusion based on average consensus," in *Information Processing in Sensor Networks, 2005. IPSN 2005. Fourth International Symposium on*. IEEE, 2005, pp. 63–70.