Temporal Logic Task Planning and Intermittent Connectivity Control of Mobile Robot Networks

Yiannis Kantaros, *Member, IEEE*, Meng Guo, *Student Member, IEEE*, and Michael M. Zavlanos, *Senior Member, IEEE*

***Abstract*—In this paper, we develop a distributed intermittent communication and task planning framework for mobile robot teams. The goal of the robots is to accomplish complex tasks, captured by local linear temporal logic formulas, and share the collected information with all other robots and possibly also with a user. Specifically, we consider situations where the robot communication capabilities are not sufficient to form reliable and connected networks, while the robots move to accomplish their tasks. In this case, intermittent communication protocols are necessary that allow the robots to temporarily disconnect from the network in order to accomplish their tasks free of communication constraints. We assume that the robots can only communicate with each other when they meet at common locations in space. Our distributed control framework jointly determines local plans that allow all robots to fulfill their assigned temporal tasks, sequences of communication events that guarantee information exchange infinitely often, and optimal communication locations that minimize a desired distance metric. Simulation results verify the efficacy of the proposed controllers.**

***Index Terms*—Distributed linear temporal logic (LTL)-based planning, intermittent communication, multirobot networks.**

I. Introduction

RECENTLY, there has been a large amount of work focused on designing controllers that ensure point-to-point or end-to-end network connectivity of mobile robot networks for all time. Such controllers either rely on graph theoretic approaches [1]–[5] or employ more realistic communication models that take into account path loss, shadowing, and multipath fading as well as optimal routing decisions for desired information rates [6]–[10]. However, due to the uncertainty in the wireless channel, it is often impossible to ensure all-time connectivity in practice. Moreover, such methods often prevent the robots from accomplishing their tasks, as motion planning is always restricted by connectivity constraints on the network. Therefore, a much preferred solution is to allow robots to communicate in an intermittent fashion and operate in disconnect mode the rest of the time.

Intermittent communication in multiagent systems has been studied in consensus problems [11], coverage problems [12], and in delay-tolerant networks [13], [14]. The common assumption in these works is that the communication network is connected over time, infinitely often. Relevant is also the work on event-based network control [15], [16], where although the network is assumed to be connected for all time, messages between the agents are exchanged intermittently when certain events take place. In this paper, we lift all connectivity assumptions and, instead, control the communication network itself so that it is guaranteed to be intermittently connected, infinitely often.

Specifically, we assume that robots can only communicate when they are physically close to each other. The intermittent connectivity requirement is captured by a global linear temporal logic (LTL) statement that forces small groups of robots, also called teams, to meet infinitely often at locations in space that are common for each team, but possibly different across teams. We assume that every robot belongs to at least one team and that there is a path, i.e., a sequence of teams where consecutive teams have nonempty intersections, connecting every two teams of robots, so that information can propagate in the network.

In addition to the intermittent communication requirement, we also assume that the robots are responsible for accomplishing independent tasks that are specified by local LTL formulas. These tasks can be, e.g., gathering of information in the environment that needs to reach all other robots and possibly a user through the proposed intermittently connected network. Given the global LTL statement comprised of the intermittent communication requirement and the local LTL tasks, existing control synthesis approaches for global LTL specifications [17]–[19] that rely on transition systems to abstract robot mobility can be used to obtain correct-by-construction controllers. Nevertheless, such approaches do not optimize task performance. Optimal control synthesis algorithms for mobile robot networks under global LTL specifications are proposed in [20]–[23]. Common in [20], [21] is that they rely on the construction of a synchronous product automaton among all robots and the application of graph search methods to synthesize optimal plans. Therefore, these approaches are resource demanding and scale poorly with the number of robots. Sampling-based optimal control synthesis methods under global LTL specifications have also

Manuscript received December 10, 2017; revised June 21, 2018; accepted December 29, 2018. Date of publication January 15, 2019; date of current version September 25, 2019. This work was supported in part by the ONR under Grant #N000141812374. Recommended by Associate Editor A. Girard. *(Corresponding author: Yiannis Kantaros.)*

The authors are with the Department of Mechanical Engineering and Materials Science, Duke University, Durham, NC 27708 USA (e-mail: yiannis.kantaros@duke.edu; meng.guo@duke.edu; michael.zavlanos@duke.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TAC.2019.2893161

been proposed by the authors in [22] that scale better than the methods in [20], [21]. The methods proposed in [20]–[22] are all centralized and offline and, therefore, not reactive to new tasks. Moreover, they require as an input the Bü chi automaton that corresponds to the global LTL formula, which is generated by a computationally expensive process. A distributed implementation of [22] that can optimize feasible motion plans online is presented in [23]. However, [23] requires an all-time connected communication network which is not the case here. A new logic, called counting LTL, is proposed in [24] that can be used for coordination of large collections of agents. However, this approach is centralized, offline, and assumes that the identity of the agents is not important for the successful accomplishment of the task, which is not the case here due to the intermittent connectivity requirement.

In this paper, our goal is to synthesize motion plans for all robots so that both the local LTL tasks and the global LTL formula capturing the intermittent connectivity requirement are satisfied while minimizing a desired distance metric. To achieve that, we avoid the construction of the product automaton altogether and instead propose an online and distributed framework to design correct-by-construction controllers for the robots. In particular, we first focus on the intermittent connectivity requirement and propose a new distributed framework to design sequences of communication events, also called communication schedules, for all teams of robots. Then, we develop discrete plans for the robots that satisfy the local LTL tasks while ensuring that teams can communicate according to the predetermined schedules. The locations of the communication events in these discrete plans are selected so that they optimize a desired distance metric. The proposed controllers are synthesized in a distributed and online fashion, and can be executed asynchronously, which is not the case in most relevant literature as, e.g., in [22]–[26].

To the best of our knowledge, the most relevant works to the one proposed here are recent works by the authors [27]–[30]. Specifically, [27] proposes an asynchronous distributed intermittent communication framework that is a special case of the one proposed here in that every robot belongs to exactly two teams and the robots in every team can only meet at a single predetermined location. This framework is extended in [28], where robots can belong to any number of teams and every team can select among multiple locations to meet, same as in the work considered here. Nevertheless, neither of the approaches in [27], [28] consider concurrent task planning. Intermittent communication control and task planning is considered in [29] that relies on the construction of a synchronous product automaton among all robots and, therefore, this approach is centralized and does not scale well with the number of robots. A distributed online approach to this problem is proposed in [30]. The method proposed here is more general in that it can handle the data gathering tasks and the star communication topology in [30] that considers information flow only to the root/user. In fact, in the proposed method, information can flow intermittently between any pair of robots and possibly a user in a multihop fashion. Another fundamental difference with [30] is that here the robots first decide *how* they want to communicate by constructing abstract schedules of communication events and then decide *where* they want to communicate by embedding online and optimally these schedules in the workspace so that the desired tasks are also satisfied. In fact, this is a unique feature of the proposed approach that differentiates it from existing literature on communication control where communication is always state dependent. Other relevant methods that do not rely on LTL for intermittent communication control are presented in [31], [32]. However, these methods impose strong restrictions on the communication pattern that can be achieved, while [31] also does not consider concurrent task planning. We provide theoretical guarantees supporting the proposed framework, as well as numerical simulations showing its ability to solve very large and complex planning problems that existing model checking techniques cannot solve. To the best of our knowledge, this is the first distributed, online, and asynchronous framework for temporal logic path planning and intermittent communication control that can be applied to large-scale multirobot systems.

The rest of this paper is organized as follows. In Section II, we present some preliminaries in LTL. The problem formulation is described in Section III. In Section IV, we design a distributed schedules of communication events that ensure intermittent connectivity. In Section V, we design discrete motion plans that satisfy the assigned local LTL tasks and the intermittent connectivity requirement as per the communication schedules while minimizing a distance metric. Theoretical guarantees of the proposed algorithm are presented in Section VI. Simulation results are included in Section VII.

II. Preliminaries

The basic ingredients of LTL are a set of atomic propositions $\mathcal{AP}$, the boolean operators, i.e., conjunction $\wedge$, and negation $\neg$, and two temporal operators, next $\bigcirc$ and until $\mathcal{U}$. LTL formulas over a set $\mathcal{AP}$ can be constructed based on the following grammar: $\phi ::= \text{true} \,|\, \pi \,|\, \phi_1 \wedge \phi_2 \,|\, \neg\phi \,|\, \bigcirc \phi \,|\, \phi_1 \,\mathcal{U}\, \phi_2$, where $\pi \in \mathcal{AP}$. For the sake of brevity, we abstain from presenting the derivations of other Boolean and temporal operators, e.g., *always* $\Box$, *eventually* $\Diamond$, *implication* $\Rightarrow$, which can be found in [33]. An infinite *word* σ over the alphabet $2^{\mathcal{AP}}$ is defined as an infinite sequence $\sigma = \pi_0\pi_1\pi_2\cdots \in (2^{\mathcal{AP}})^\omega$, where ω denotes infinite repetition and $\pi_k \in 2^{\mathcal{AP}}$, $\forall k \in \mathbb{N}$. The language $\texttt{Words}(\phi) = \left\{\sigma \in (2^{\mathcal{AP}})^\omega | \sigma \models \phi\right\}$ is defined as the set of words that satisfy the LTL formula ϕ, where $\models \subseteq (2^{\mathcal{AP}})^\omega \times \phi$ is the satisfaction relation.

Any LTL formula ϕ can be translated into a Nondeterministic Bü chi Automaton (NBA) over $2^{\mathcal{AP}}$ denoted by B, which is defined as follows [34].

Definition 2.1 (NBA): An NBA B over $2^{\mathcal{AP}}$ is defined as a tuple $B = \left(\mathcal{Q}_B, \mathcal{Q}_B^0, \Sigma, \rightarrow_B, \mathcal{F}_B\right)$, where $\mathcal{Q}_B$ is the set of states, $\mathcal{Q}_B^0 \subseteq \mathcal{Q}_B$ is a set of initial states, $\Sigma = 2^{\mathcal{AP}}$ is an alphabet, $\rightarrow_B \subseteq \mathcal{Q}_B \times \Sigma \times \mathcal{Q}_B$ is the transition relation, and $\mathcal{F}_B \subseteq \mathcal{Q}_B$ is a set of accepting/final states.

An *infinite run* ρ_B of B over an infinite word $\sigma = \pi_0\pi_1\pi_2\ldots$, $\pi_k \in \Sigma = 2^{\mathcal{AP}}$ $\forall k \in \mathbb{N}$ is a sequence $\rho_B = q_B^0 q_B^1 q_B^2 \ldots$ such that $q_B^0 \in \mathcal{Q}_B^0$ and $(q_B^k, \pi_k, q_B^{k+1}) \in \rightarrow_B$, $\forall k \in \mathbb{N}$. An infinite run ρ_B is called *accepting* if $\texttt{Inf}(\rho_B) \cap \mathcal{F}_B \neq \varnothing$, where $\texttt{Inf}(\rho_B)$ represents the set of states that appear in ρ_B infinitely often. The words σ that result in an accepting run of B

constitute the accepted language of B, denoted by $\mathcal{L}_B$. Then, it is proven [33] that the accepted language of a NBA B, associated with an LTL formula ϕ, is equivalent to the words of ϕ, i.e., $\mathcal{L}_B = \texttt{Words}(\phi)$.

III. Problem Formulation

Consider $N \geq 1$ mobile robots operating in a workspace $\mathcal{W} \subset \mathbb{R}^d$, $d \in \{2, 3\}$, containing $W > 0$ locations of interest denoted by $\mathbf{v}_j$, $j \in \mathcal{I} := \{1, \dots, W\}$. Mobility of robot $i \in \mathcal{N} := \{1, \dots, N\}$ in $\mathcal{W}$ is captured by a weighted Transition System (wTS) that is defined as follows:

Definition 3.1 (weighted Transition System): A wTS for robot i, denoted by wTS$_i$ is a tuple wTS$_i = (\mathcal{Q}_i, q_i^0, \rightarrow_i, w_i, \mathcal{AP}, L_i)$, where the following conditions hold:

1) $\mathcal{Q}_i = \{q_i^{\mathbf{v}_j}, j \in \mathcal{I}\}$ is the set of states, where a state $q_i^{\mathbf{v}_j}$ indicates that robot i is at location $\mathbf{v}_j \in \mathcal{W}$.
2) $q_i^0 \in \mathcal{Q}_i$ is the initial state of robot i.
3) $\rightarrow_i \subseteq \mathcal{Q}_i \times \mathcal{Q}_i$ is a given transition relation such that $(q_i^{\mathbf{v}_j}, q_i^{\mathbf{v}_e}) \in \rightarrow_i$ if there exists a controller that can drive robot i from location $\mathbf{v}_j$ to $\mathbf{v}_e$ in finite time without going through any other location $\mathbf{v}_c$.
4) $w_i : \mathcal{Q}_i \times \mathcal{Q}_i \rightarrow \mathbb{R}_+$ is a weight function that captures the distance that robot i needs to travel to move from $\mathbf{v}_j$ to $\mathbf{v}_e$.[1]
5) $\mathcal{AP} = \{\{\pi_i^{\mathbf{v}_j}\}_{i=1}^N\}_{j \in \mathcal{I}}$ is the set of atomic propositions associated with each state.
6) $L_i : \mathcal{Q}_i \rightarrow \mathcal{AP}$ is defined as $L_i(q_i^{\mathbf{v}_j}) = \pi_i^{\mathbf{v}_j}$, for all $i \in \mathcal{N}$ and $j \in \mathcal{I}$.

Every robot $i \in \mathcal{N}$ is responsible for accomplishing high-level tasks associated with some of the locations $\mathbf{v}_j$, $j \in \mathcal{I}$. Hereafter, we assume that the tasks assigned to the robots are independent from each other. Specifically, we assume that the task assigned to robot i is captured by a local LTL$_{-\bigcirc}$ formula ϕ_i [35] specified over the set of atomic propositions $\mathcal{AP} = \{\{\pi_i^{\mathbf{v}_j}\}_{i=1}^N\}_{j \in \mathcal{I}}$, where $\pi_i^{\mathbf{v}_j} = 1$ if $\|\mathbf{x}_i - \mathbf{v}_j\| \leq \epsilon$, for a sufficiently small $\epsilon > 0$, and 0 otherwise, for all $i \in \mathcal{N}$ and $j \in \mathcal{I}$.[2] Namely, the atomic proposition $\pi_i^{\mathbf{v}_j}$ is true if robot i is sufficiently close to location $\mathbf{v}_j$. For example, an LTL$_{-\bigcirc}$ task for robot i can be $\phi_i = (\square\lozenge\pi_i^{\mathbf{v}_4}) \wedge ((\neg\pi_i^{\mathbf{v}_4})\mathcal{U}\pi_i^{\mathbf{v}_8}) \wedge (\lozenge\pi_i^{\mathbf{v}_5}) \wedge (\square\neg\pi_i^{\mathbf{v}_3}) \wedge (\square\lozenge\pi_i^{\mathbf{v}_1})$, which requires robot i to the following conditions:

1) Visit location $\mathbf{v}_4$ infinitely often.
2) Never visit location $\mathbf{v}_4$ until location $\mathbf{v}_8$ is visited.
3) Eventually visit location $\mathbf{v}_5$.
4) Always avoid an obstacle located at $\mathbf{v}_3$.
5) Visit location at $\mathbf{v}_1$ infinitely often.

Together with accomplishing local tasks, robots are also responsible for communicating with each other so that any information that is collected as part of these tasks is propagated in the network and, possibly, eventually reaches a user.

[1]Note that alternative weights can be assigned to the transitions of the wTSs that can capture, e.g., the time or energy required for robot i to move from $\mathbf{v}_j$ to $\mathbf{v}_e$.

[2]The syntax of LTL$_{-\bigcirc}$ is the same as the syntax of LTL excluding the "next" operator. The choice of LTL$_{-\bigcirc}$ over LTL is motivated by the fact that we are interested in the continuous time execution of the synthesized plans, in which case the next operator is not meaningful. This choice is common in relevant works, see, e.g., [36] and the references therein.

To define a communication network among the robots, we first partition the robot team into $M \geq 1$ robot subgroups, called also teams, and require that every robot belongs to at least one subgroup. The indices i of the robots that belong to the mth subgroup are collected in a set denoted by $\mathcal{T}_m$, for all $m \in \mathcal{M} := \{1, 2, \dots, M\}$. We define the set that collects the indices of teams that robot i belongs to as $\mathcal{M}_i = \{m | i \in \mathcal{T}_m, m \in \mathcal{M}\}$. Also, for robot i, we define the set that collects the indices of all other robots that belong to common teams with robot i as $\mathcal{N}_i = \{j | j \in \mathcal{T}_m, \forall m \in \mathcal{M}_i\} \setminus \{i\}$, $\forall i \in \mathcal{N}$. Given the robot teams $\mathcal{T}_m$, for all $m \in \mathcal{M}$, we can define the graph over these teams as follows.

Definition 3.2 (Team membership graph $\mathcal{G}_\mathcal{T}$): The graph over the teams $\mathcal{T}_m, m \in \mathcal{M}$ is defined as $\mathcal{G}_\mathcal{T} = (\mathcal{V}_\mathcal{T}, \mathcal{E}_\mathcal{T})$, where the set of nodes $\mathcal{V}_\mathcal{T} = \mathcal{M}$ is indexed by the teams $\mathcal{T}_m$ and set of edges $\mathcal{E}_\mathcal{T}$ is defined as $\mathcal{E}_\mathcal{T} = \{(m, n) | \mathcal{T}_m \cap \mathcal{T}_n \neq \emptyset, \forall m, n \in \mathcal{M}, m \neq n\}$.

Given the team membership graph $\mathcal{G}_\mathcal{T}$, we can also define the set $\mathcal{N}_{\mathcal{T}_m} := \{e \in \mathcal{V}_\mathcal{T} | (m, e) \in \mathcal{E}_\mathcal{T}\}$ that collects all neighboring teams of team $\mathcal{T}_m$ in $\mathcal{G}_\mathcal{T}$. Since the robots have limited communication capabilities, we assume that the robots in every subgroup $\mathcal{T}_m$ can only communicate if all of them are simultaneously present at a common location $\mathbf{v}_j \in \mathcal{W}$, hereafter called a communication point. We assume that there are $R \geq 1$ available communication points in the workspace at locations $\mathbf{v}_j \in \mathcal{W}$, where $j \in \mathcal{C} \subset \mathcal{I}$. Among those communication points, the ones that are specifically available to the robotic team $\mathcal{T}_m$ are collected in a finite set $\mathcal{C}_m \subseteq \mathcal{C}$, where the sets $\mathcal{C}_m$ are not necessarily disjoint. When all robots in a team $\mathcal{T}_m$ have arrived at a communication location, we assume that communication happens and the robots leave to accomplish their tasks or communicate with other teams. This way, a dynamic robot communication network is constructed, defined as follows:

Definition 3.3 (Communication network $\mathcal{G}_c(t)$): The communication network among the robots is defined as a dynamic undirected graph $\mathcal{G}_c(t) = (\mathcal{V}_c, \mathcal{E}_c(t))$, where the set of nodes $\mathcal{V}_c$ is indexed by the robots, i.e., $\mathcal{V}_c = \mathcal{N}$, and $\mathcal{E}_c(t) \subseteq \mathcal{V}_c \times \mathcal{V}_c$ is the set of communication links that emerge among robots in every team $\mathcal{T}_m$, when they all meet at a common communication point $\mathbf{v}_j$, for some $j \in \mathcal{C}_m$ simultaneously, i.e., $\mathcal{E}_c(t) = \{(e, i), \forall\, i, e \in \mathcal{T}_m,\ \forall m \in \mathcal{M} \,|\, \mathbf{x}_i(t) = \mathbf{x}_e(t) = \mathbf{v}_j, \text{ for some } j \in \mathcal{C}_m\}$.

To ensure that information is continuously transmitted across the network of robots, we require that the communication graph $\mathcal{G}_c(t)$ is *connected over time infinitely often*, i.e., that all robots in every team $\mathcal{T}_m$ meet infinitely often at a common communication point $\mathbf{v}_j, j \in \mathcal{C}_m$, that does not need to be fixed over time. For this, it is necessary to assume that the graph of teams $\mathcal{G}_\mathcal{T}$ is connected. Specifically, if $\mathcal{G}_\mathcal{T}$ is connected, then information can be propagated intermittently across teams through robots that are common to these teams and, in this way, information can reach all robots in the network. Connectivity of $\mathcal{G}_\mathcal{T}$ and the fact that robots can be members of only a few teams means that information can be transferred over long distances, possibly to reach a remote user, without requiring that the robots leave their assigned regions of interest defined by their assigned tasks and communication points corresponding to the teams they belong to. Moreover, we assume that the teams $\mathcal{T}_m$ are *a priori* known and can be selected arbitrarily as long as the graph of teams $\mathcal{G}_{\mathcal{T}}$ is connected.

Intermittent connectivity of the communication network $\mathcal{G}_c(t)$ can be captured by the global LTL formula

$$\phi_{\text{com}} = \wedge_{m\in\mathcal{M}} \left(\Box\Diamond \left(\vee_{j\in\mathcal{C}_m} \left(\wedge_{i\in\mathcal{T}_m} \pi_i^{\text{v}_j}\right)\right)\right) \tag{1}$$

specified over the set of atomic propositions $\{\{\pi_i^{\text{v}_j}\}_{i=1}^N\}_{j\in\mathcal{C}}$. Composing ϕ_{com} with the local LTL$_{-\bigcirc}$ formulas ϕ_i, yields the following global LTL statement:

$$\phi = (\wedge_{i\in\mathcal{N}}\phi_i) \wedge \phi_{\text{com}} \tag{2}$$

that captures the local tasks assigned to every robot and intermittent connectivity of the communication network $\mathcal{G}_c$.

Given the wTS$_i$, for all robots $i \in \mathcal{N}$, and the global LTL formula (2), the goal is to synthesize motion plans τ_i, for all $i\in\mathcal{N}$, whose execution satisfies the global LTL formula (2). Typically, such motion plans are infinite paths in wTS$_i$ [35], i.e., infinite sequences of states in wTS$_i$, such that $\tau_i(1) = q_i^0$, $\tau_i(\kappa) \in \mathcal{Q}_i$, and $(\tau_i(\kappa), \tau_i(\kappa+1)) \in\rightarrow_i$, $\forall\kappa\in\mathbb{N}_+$. In this form, they cannot be manipulated in practice. This issue can be resolved by representing these plans in a prefix–suffix form [34], i.e., $\tau_i = \tau_i^{\text{pre}}\left[\tau_i^{\text{suf}}\right]^\omega$, where the prefix part τ_i^{pre} and suffix part τ_i^{suf} are both finite paths in wTS$_i$, for all robots $i\in\mathcal{N}$. The prefix τ_i^{pre} is executed once and the suffix τ_i^{suf} is repeated indefinitely. The cost associated with a plan $\tau_i = \tau_i^{\text{pre}}\left[\tau_i^{\text{suf}}\right]^\omega$ is defined as

$$J_p(\tau_i) = \alpha J(\tau_i^{\text{pre}}) + (1-\alpha)J(\tau_i^{\text{suf}}) \tag{3}$$

where $J(\tau_i^{\text{pre}})$ and $J(\tau_i^{\text{suf}})$ represent the cost of the prefix and the suffix part, respectively, and $\alpha\in[0,1]$ is a user-specified parameter. The cost $J(\tau_i^{\text{suf}})$ of the suffix part is defined as

$$J(\tau_i^{\text{suf}}) = \sum_{\kappa=1}^{|\tau_i^{\text{suf}}|} w_i(\tau_i^{\text{suf}}(\kappa), \tau_i^{\text{suf}}(\kappa+1)) \tag{4}$$

where $|\tau_i^{\text{suf}}|$ stands for the number of states in the finite path τ_i^{suf}, $\tau_i^{\text{suf}}(\kappa)$ denotes the κth state in τ_i^{suf}, and w_i are the weights defined in Definition 3.1. The cost $J(\tau_i^{\text{pre}})$ of the prefix part is defined accordingly. In other words, $J_p(\tau_i)$ captures the distance that robot i needs to travel during a single execution of the prefix and suffix part weighted by a user-specified parameter $\alpha > 0$.

The problem that is addressed in this paper can be summarized as follows:

Problem 1: Consider any initial configuration of a network of N mobile robots in their respective wTSs, and any partition of the network in M subgroups $\mathcal{T}_m$, $m\in\mathcal{M}$ so that the associated graph $\mathcal{G}_{\mathcal{T}}$ is connected. Determine discrete motion plans τ_i, i.e., sequences of states $q_i^{\text{v}_j}\in\mathcal{Q}_i$, in prefix–suffix structure, for all robots such that the LTL specification ϕ defined in (2) is satisfied.

1) The local LTL$_{-\bigcirc}$ tasks ϕ_i are satisfied, for all $i\in\mathcal{N}$.
2) Intermittent communication among robots captured by ϕ_{com} is ensured infinitely often.
3) The distance metric $\sum_{i\in\mathcal{N}} J_p(\tau_i)$ is minimized.

To solve Problem 1, we propose a distributed algorithm that consists of two main parts. First, we design offline schedules of communication events for all robots, independently of their assigned tasks, that ensure intermittent communication among robots in every team infinitely often, see Section IV. These communication events depend on the structure of the graph $\mathcal{G}_{\mathcal{T}}$ and are not associated with specific locations in space. Then, in Section V, we design online discrete plans for the robots that satisfy their local tasks while ensuring that the robots in each team communicate as per the schedules defined in Section IV. The location of these communication events are selected so that the distance metric $\sum_{i\in\mathcal{N}} J_p(\tau_i)$ is minimized.

IV. Intermittent Communication Control

In this section, we construct infinite sequences of communication events (also called communication schedules) so that intermittent connectivity infinitely often as per (1) is guaranteed. Construction of the communication schedules occurs offline, i.e., before the robots are deployed in the workspace to satisfy the assigned LTL$_{-\bigcirc}$ tasks ϕ_i, and requires that the robots are connected so that they can share information with each other. Then, in Section V, these schedules are integrated online with task planning to synthesize discrete motion plans that ensure that the local tasks are satisfied, the network is intermittently connected as per the designed schedules, and the cost function defined in Section III is minimized.

Since every robot can be a member of more than one team, the objective in designing the proposed communication schedules is that no teams that share common robots communicate at the same time, as this would require that the shared robots are present at more than one possibly different communication points at the same time. We call such schedules conflict free. To construct such conflict-free schedules of communication events, we define a sequence S of teams that determines the order in which the robots construct their schedules.

Definition 4.1 (Sequence S): The finite sequence S is a sequence of teams defined as $S = \mathcal{T}_n, \mathcal{T}_m, \ldots$. The sequence S satisfies two requirements: 1) all teams $\mathcal{T}_m$, $m\in\mathcal{M}$ appear in S; and 2) consecutive teams $\mathcal{T}_n$ and $\mathcal{T}_m$ that appear in S are neighboring nodes in the graph $\mathcal{G}_{\mathcal{T}}$, i.e., $m\in\mathcal{N}_{\mathcal{T}_n} := \{e\in\mathcal{V}_{\mathcal{T}} | (n,e)\in\mathcal{E}_{\mathcal{T}}\}$.

In what follows, we assume that the sequence S is user defined and known by all robots. Moreover, we denote by $S(k)$ the kth team in S, $\forall k\in\{1,\ldots,|S|\}$, where $|S|$ stands for the length of S. Using the sequence S, we construct communication schedules $\texttt{sched}_i$ for all robots i that determine the order in which those robots participate in communication events for teams $\mathcal{T}_m$, $\forall m\in\mathcal{M}_i$ and are defined as follows.

Definition 4.2 (Schedule of communication events): The schedule $\texttt{sched}_i$ of communication events of robot i is defined as an infinite repetition of the finite sequence

$$s_i = X,\ldots,X,\mathcal{M}_i(1),X,\ldots,X,\mathcal{M}_i(2),X,\ldots,X,$$
$$\mathcal{M}_i(|\mathcal{M}_i|),X,\ldots,X \tag{5}$$

i.e., $\texttt{sched}_i = s_i, s_i, \cdots = s_i^\omega$, where ω stands for the infinite repetition of s_i.

In (5), $\mathcal{M}_i(e)$, $e\in\{1,\ldots,|\mathcal{M}_i|\}$ stands for the eth entry of $\mathcal{M}_i$ and represents a communication event for team with index $\mathcal{M}_i(e)$, and the discrete states X indicate that there is no communication event for robot i. The length of the sequence s_i is $\ell = \max\{d_{\mathcal{T}_m}\}_{m=1}^M + 1$ for all $i\in\mathcal{N}$, where $d_{\mathcal{T}_m}$ is the degree of node $m \in \mathcal{V}_{\mathcal{T}}$. It is shown in Proposition 4.4 that this length ℓ is sufficient for the construction of conflict-free communication schedules as per the algorithm described bellow. The schedule $\texttt{sched}_i$ defines the order in which robot i participates in the communication events for the teams $m \in \mathcal{M}_i$, for all robots $i \in \mathcal{N}$. Specifically, at a discrete time step $z \in \mathbb{N}_+$, robot i either communicates with all robots that belong to team $\mathcal{T}_m$, for $m \in \mathcal{M}_i$ if $\texttt{sched}_i(z) = m$, or does not need to participate in any communication event if $\texttt{sched}_i(z) = X$.

In what follows, we present a distributed process that relies on two rules that the robots execute in order to construct the schedules $\texttt{sched}_i$. These schedules are constructed sequentially across the teams $\mathcal{T}_m$, $m \in \mathcal{M}$, in an order that is determined by the sequence S. In other words, robots in team $S(k)$ will construct their respective schedules, only if all robots in team $S(k-1)$ have already designed their schedules. Assume that according to the sequence S, robots in team $S(k) = \mathcal{T}_m$, for some $k \geq 1$ are due to construct their schedules. By construction of the sequence S, consecutive teams in S are always neighboring teams, which means that there exists a team $\mathcal{T}_n$ with $n \in \mathcal{N}_{\mathcal{T}_m}$ such that $S(k-1) = \mathcal{T}_n$ and $\mathcal{T}_m \cap \mathcal{T}_n \neq \emptyset$. Consequently, there exist also robots $j \in \mathcal{T}_m \cap \mathcal{T}_n$ that previously constructed their sequences s_j. These robots j never reconstruct their schedules. Instead, one of the robots $j \in S(k) \cap S(k-1)$ is tasked with providing information to the other robots $i \in S(k) = \mathcal{T}_m$ that is necessary to construct their sequences s_i.

Specifically, this robot $j \in S(k) \cap S(k-1)$ first notifies the robots in team $S(k) = \mathcal{T}_m$ that it is their turn to construct their communication schedules.[3] Second, robot j transmits to robots $i \in \mathcal{T}_m$ all sequences s_b that have been constructed so far by the robots in teams $S(1), \ldots, S(k-1)$. Among all those sequences s_b, robots $i \in \mathcal{T}_m$ use only the sequences of robots $b \in \mathcal{N}_i$ to construct their sequences s_i.[4] As a result, all robots $i \in \mathcal{T}_m$ that have not constructed s_i yet, are aware of the indices $n_b^{\mathcal{T}_g}$ that point to entries in s_b associated with some communication events g. These indices satisfy $s_b(n_b^{\mathcal{T}_g}) = g$, $b \in \mathcal{N}_i$.[5] Notice that this means that robots $i \in \mathcal{T}_m$ are also aware of the indices $n_b^{\mathcal{T}_m}$. Using this information, every robot $i \in \mathcal{T}_m$ constructs the sequence s_i based on the following two rules that determine the indices $n_i^{\mathcal{T}_g}$ that point to entries in s_i where the communication event g will be placed, i.e., $s_i(n_i^{\mathcal{T}_g}) = g$, for all $g \in \mathcal{M}_i$.

1) *First rule:* Let $n_i^{\mathcal{T}_g}$ denote the index of the entry at which the communication event $g \in \mathcal{M}_i$ will be placed into s_i. If there exists a robot $b \in \mathcal{N}_i$ that has selected $n_b^{\mathcal{T}_g}$ so that $s_b(n_b^{\mathcal{T}_g}) = g$, then $n_i^{\mathcal{T}_g} = n_b^{\mathcal{T}_g}$. In this way, all robots $b \in \mathcal{T}_g$, including robot $i \in \mathcal{T}_m \cap \mathcal{T}_g$ will select the same index $n_b^{\mathcal{T}_g}$ and will participate in the same communication event g at the same discrete time instant, see line 3, Algorithm 1.
2) *Second rule:* If there do not exist robots $b \in \mathcal{N}_i$ that have selected indices $n_b^{\mathcal{T}_g}$, for communication event $g \in \mathcal{M}_i$, then the communication event g can be placed at any available entry $n_i^{\mathcal{T}_g}$ of s_i that satisfies the following requirement. The entry $n_i^{\mathcal{T}_g}$ in all sequences s_j of robots $j \in \mathcal{N}_i$ that have already been constructed should not contain communication events h such that $h \in \mathcal{N}_{\mathcal{T}_g}$, see line 2, Algorithm 1.

Algorithm 1: Distributed Construction of Sequence s_i, $i \in \mathcal{T}_m$.

```
  Input: Already constructed sequences $s_b$, $\forall b \in \mathcal{N}_i$.
  Output: Schedule of meeting events: $\texttt{sched}_i = [s_i]^\omega$
1 Construct an empty finite sequence $s_i$ of length $\ell$. ;
2 for $g \in \mathcal{M}_i$ do
3     if there exist constructed sequences $s_b$, $b \in \mathcal{T}_g$ then
4         $s_i(n_i^{\mathcal{T}_g}) := g$, where $n_i^{\mathcal{T}_g} := n_b^{\mathcal{T}_g}$, $\forall b \in \mathcal{T}_g$ ;
          ▷ First rule
5     else
6         Choose an available $n_i^{\mathcal{T}_g} \in \{1, \ldots, \ell\}$ such that it
            holds either $s_j(n_i^{\mathcal{T}_g}) := X$, or $s_j(n_i^{\mathcal{T}_g}) := h$ with
            $h \notin \mathcal{N}_{\mathcal{T}_g}$, $\forall j \in \mathcal{N}_i$. Then set $s_i(n_i^{\mathcal{T}_g}) := g$. ;
          ▷ Second rule
7 Put $X$ in the remaining entries;
```

Note that the index $n_i^{\mathcal{T}_m}$ will always be determined by the first rule, since robot $j \in S(k) \cap S(k-1)$ has already constructed its sequence s_j by placing the event m at an entry of s_j with index $n_j^{\mathcal{T}_m}$. To highlight the role of the second rule, assume that $h \in \mathcal{N}_{\mathcal{T}_g}$. Then, this means that there exists at least one robot $r \in \mathcal{T}_h \cap \mathcal{T}_g$. Notice that without the second rule, at a subsequent iteration of this procedure, robot $r \in \mathcal{T}_h \cap \mathcal{T}_g$ would have to place communication events for teams $\mathcal{T}_g$ and $\mathcal{T}_h$ at a common entry of s_r, i.e., $n_r^{\mathcal{T}_g} = n_r^{\mathcal{T}_h}$, due to the first rule and, therefore, a conflicting communication event in schedule $\texttt{sched}_r$ would occur. In all the remaining entries of s_i, X's are placed; see line 7, Algorithm 1. By construction of s_i, there are $\ell - |\mathcal{M}_i|$ X's in s_i.

Once all robots i in team $S(k)$ have constructed the sequences s_i, a robot $j \in S(k) \cap S(k+1)$ will notify all robots in team $S(k+1)$ that it is their turn to compute their respective schedules. The procedure is repeated sequentially over the teams in S until all robots have computed their respective schedules of meeting events. This process is summarized in Algorithm 1 and it is also illustrated in Example 4.3.

Example 4.3 (Algorithm 1): To illustrate Algorithm 1, consider the network of $N = 3$ robots shown in Fig. 1, where the teams of robots are designed as $\mathcal{T}_1 = \{1, 2\}$, $\mathcal{T}_2 = \{2, 3\}$, and $\mathcal{T}_3 = \{3, 1\}$. Let the sequence S be $S = \mathcal{T}_1, \mathcal{T}_2, \mathcal{T}_3$. Hence, initially the robots 1 and 2 in team $\mathcal{T}_1$ coordinate to construct their respective sequences s_i. Assume that initially robot 1 constructs the sequence s_1 of length equal to $\ell = \max\{d_{\mathcal{T}_m}\}_{m=1}^{3} + 1 = 3$. Robot 1 belongs to teams $\mathcal{T}_1$ and $\mathcal{T}_2$ and it arbitrarily constructs s_1 as follows: $s_1 = 1, 3, X$. Then, the sequence s_1 is transmitted

[3]Note that if the teams in S were not necessarily neighboring teams, then robot $j \in S(k-1) = \mathcal{T}_n$ would have to know who the members of team $S(k) = \mathcal{T}_m$, $m \notin \mathcal{M}_j$, are in order to notify them that it is their turn to construct the communication schedules. Due to the fact that S connects neighboring teams, every robot j needs to know only the structure of teams $\mathcal{T}_m$, $m \in \mathcal{M}_j$.

[4]Note that robot j is not aware of the sets $\mathcal{N}_i$ and, therefore, it transmits all the sequences s_b that have already been constructed to robots $i \in \mathcal{T}_m$.

[5]Note that the discrete time instants at which the communication event $g \in \mathcal{M}_i$ will take place are $n_i^{\mathcal{T}_g} + z\ell$, where $z \in \mathbb{N}$, by definition of $\texttt{sched}_i$.

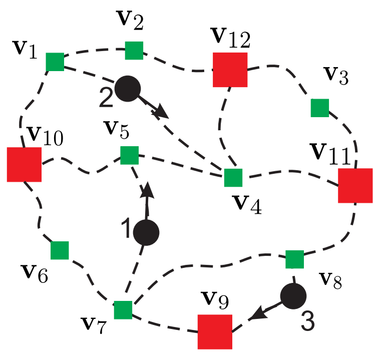


Fig. 1. Graphical illustration of the problem formulation. A network of $N = 3$ robots (black dots) divided into $M = 3$ teams is depicted. The robot teams are selected to be $\mathcal{T}_1 = \{1, 2\}$, $\mathcal{T}_2 = \{2, 3\}$, and $\mathcal{T}_3 = \{3, 1\}$. The set $\mathcal{I}$ consists of locations represented by red and green squares. Red squares comprise set $\mathcal{C}$ and represent communication points. Black dashed lines stand for paths in the workspace $\mathcal{W}$ that connect locations $\mathbf{v}_e$ and $\mathbf{v}_j$. The sets of communications points for each team are defined as $\mathcal{C}_1 = \{\mathbf{v}_9, \mathbf{v}_{10}\}$, $\mathcal{C}_2 = \{\mathbf{v}_{10}, \mathbf{v}_{11}\}$, and $\mathcal{C}_3 = \{\mathbf{v}_{12}\}$.

to robot 2 that belongs to teams $\mathcal{T}_1$ and $\mathcal{T}_2$. Now, robot 2 is responsible for constructing the sequence s_2. To construct s_2, according to the first rule, team $\mathcal{T}_1$ is placed at the first entry of s_2, i.e., $n_2^{\mathcal{T}_1} = n_1^{\mathcal{T}_1} = 1$. Next, the index $n_2^{\mathcal{T}_2}$ is determined by the second rule. Specifically, notice that among the two available entries in s_2 for team $\mathcal{T}_3$ the entry $n_2^{\mathcal{T}_3} = 2$ is invalid, since robot $1 \in \mathcal{T}_1$ has already constructed its sequence s_1 so that $n_1^{\mathcal{T}_3} = 2$ and for teams $\mathcal{T}_3$ and $\mathcal{T}_2$ it holds that $3 \in \mathcal{N}_{\mathcal{T}_2}$. Therefore, robot 2 selects $n_2^{\mathcal{T}_2} = 2$ and constructs the sequence $s_2 = 1, X, 2$. At the next iteration of Algorithm 1, the robots 2 and 3 in team $\mathcal{T}_2$ coordinate to construct their sequences s_i. Robot 2 has already constructed the sequence s_2 at the previous iteration and it transmits its constructed sequence s_2 and the previously constructed sequence s_1 to robot 3. Thus, robot 3 has now access to all already constructed sequences s_e, for $e \in \mathcal{N}_3 = \{1, 2\}$. Robot 3 constructs $s_3 = X, 3, 2$ using the first rule. Finally, the robots in the third team $\mathcal{T}_3 = \{3, 1\}$ have already constructed their finite paths at previous iterations.

In the following proposition, we show that Algorithm 1 can always construct sequences s_i if the length ℓ of s_i is selected as $\ell = \max\{d_{\mathcal{T}_m}\}_{m=1}^M + 1$.

Proposition 4.4: Algorithm 1 can always construct sequences s_i, for all $i \in \mathcal{N}$, if the length ℓ of s_i is selected as $\ell = \max\{d_{\mathcal{T}_m}\}_{m=1}^M + 1$.

Proof: The proof is based on contradiction. Assume that a robot i requires a sequence s_i of length greater than $\ell = \max\{d_{\mathcal{T}_e}\}_{e=1}^M + 1$ when Algorithm 1 is applied. This means that there is team $\mathcal{T}_m$, $m \in \mathcal{M}_i$, which cannot be placed at any of the first ℓ entries of s_i. By construction of Algorithm 1, this means that the team $\mathcal{T}_m$ has at least ℓ neighbors in graph $\mathcal{G}_{\mathcal{T}}$, i.e., $d_{\mathcal{T}_m} \geq \ell$, which can never happen. ■

Remark 4.5 (Repeated teams in S and initialization): Due to the requirement that consecutive teams in S need to be neighbors in $\mathcal{G}_{\mathcal{T}}$, it is possible that a team $\mathcal{T}_m$ may appear more than once in S, depending on the structure of the graph $\mathcal{G}_{\mathcal{T}}$. In this case, robots $i \in \mathcal{T}_m$ construct the sequences s_i only the first time that team $\mathcal{T}_m$ appears in S. Also, at the first iteration of Algorithm 1, robots of team $S(1)$ have to construct their sequences s_i, $i \in S(1)$. In this case, a randomly selected robot $j \in S(1)$ creates arbitrarily its sequence s_j by placing the teams $m \in \mathcal{M}_j$ at the $n_j^{\mathcal{T}_m}$ th entry of s_j. Then, the procedure described in Algorithm 1 follows.

Remark 4.6 (Discrete states X): In the schedules $\texttt{sched}_i$, defined in Definition 4.2 and constructed using Algorithm 1, the states X indicate that no communication events occur for robot i at the corresponding discrete time instants. These states are used to synchronize the communication events over the discrete time instants $c \in \mathbb{N}_+$, i.e., to ensure that the discrete time instant z at which communication happens for team $\mathcal{T}_m$, $m \in \mathcal{M}$, is the same for all robots $i \in \mathcal{T}_m$, see also Example 4.3. Nevertheless, as it will be shown in Theorem 6.5, in Section VI, it is the order of communication events in $\texttt{sched}_i$ that is critical to ensure intermittent communication, not the time instants that they take place. This is due to a communication policy proposed in Section V-C.

V. Integrated Task Planning and Intermittent Communication Control

In this section, we propose a distributed and online algorithm to synthesize motion plans for all robots i so that the global LTL formula (1) is satisfied, i.e., the assigned local LTL$_{-\bigcirc}$ tasks are accomplished, and the network is intermittently connected. These plans are generated iteratively and have the following prefix–suffix structure:

$$\tau_i^{n_i} = \texttt{path}_i^0|\texttt{path}_i^1|\dots|[\texttt{path}_i^{n_i}]^\omega \tag{6}$$

where $n_i \in \mathbb{N}$ is the iteration index associated with robot i, $\texttt{path}_i^{n_i}$ is a finite sequence of states in wTS$_i$, $|$ denotes the concatenation of discrete paths $\texttt{path}_i^{n_i}$, and ω denotes the infinite repetition. Each path $\texttt{path}_i^{n_i}$ is constructed so that 1) execution of $\texttt{path}_i^{n_i}$, for a every given n_i ensures that robot i will communicate exactly once with all teams $\mathcal{T}_m$, $m \in \mathcal{M}_i$ in an order that respects the schedules $\texttt{sched}_i$ designed in Section IV, and 2) execution of $\tau_i^{n_i}$ guarantees that the assigned local LTL$_{-\bigcirc}$ tasks ϕ_i are satisfied. In Section V-A, we discuss the distributed construction of the initial paths $\texttt{path}_i^0$ given the communication schedules $\texttt{sched}_i$. In Section V-B, we present the distributed construction of all subsequent paths $\texttt{path}_i^{n_i}$ that occurs online as the robots navigate the worskpace.

A. Construction of Initial Paths

Once robot i constructs its schedule $\texttt{sched}_i$, it locally designs the initial path $\texttt{path}_i^0$. To do this, feasible initial communication points for all teams $\mathcal{T}_m$, $m \in \mathcal{M}$, need to be selected first, that do not violate the local tasks ϕ_i. These can be found by exhaustively searching through the set of possible combinations of communication points for all teams. Specifically, let $\texttt{comb}_b$ denote any candidate combination of communication points that can be assigned to all teams $\mathcal{T}_m$, $m \in \mathcal{M}$, where $b \in \{1, \dots, \prod_{m\in\mathcal{M}} |\mathcal{C}_m|\}$. Given the communication points $\mathbf{v}_j$, $j \in \mathcal{C}_m$, in the candidate combination $\texttt{comb}_b$, every robot constructs the NBA B_i that corresponds to the following LTL formula:

$$\psi_i = \underbrace{\phi_i}_{\text{task}} \wedge \underbrace{\phi_{\text{com},i}}_{\text{communication}} \tag{7}$$

where

$$\phi_{\text{com},i} = \wedge_{m\in\mathcal{M}_i} (\square\Diamond \mathbf{v}_{j\in\mathcal{C}_m}). \tag{8}$$

In other words, the LTL formula $\phi_{\text{com},i}$ requires robot i to visit infinitely often the candidate communication points $\mathbf{v}_j, j \in \mathcal{C}_m$, of all teams $\mathcal{T}_m$, $m \in \mathcal{M}_i$, that are specified in $\texttt{comb}_b$. Then, given the wTS_i and the NBA B_i, every robot can synthesize a motion plan $\tilde{\tau}_i^0 \models \psi_i$, if it exists, which will be used to construct the initial path $\texttt{path}_i^0$. This process is repeated for all $b \in \{1,\ldots,\prod_{m\in\mathcal{M}} |\mathcal{C}_m|\}$ until feasible plans $\tilde{\tau}_i^0 \models \psi_i$ can be constructed for all robots $i \in \mathcal{N}$. Later, in Lemma 5.2, we show that the robots can search locally over the combinations $\texttt{comb}_b$ reducing in this way the computational cost of finding a feasible plan $\tilde{\tau}_i^0$.

Specifically, given candidate initial communication points for all teams $\mathcal{T}_m$, $m \in \mathcal{M}_i$, the motion plan $\tilde{\tau}_i^0$ can be constructed by checking the nonemptiness of the language of the *Product Bü chi Automaton* (PBA) $P_i = \text{wTS}_i \otimes B_i$, defined as follows [33]:

Definition 5.1 (Product Bü chi automaton): Given the $\text{wTS}_i = (\mathcal{Q}_i, q_i^0, \rightarrow_i, w_{P_i}, \mathcal{AP}, L_i)$ and the NBA $B_i = (\mathcal{Q}_{B_i}, \mathcal{Q}_{B_i}^0, 2^{\mathcal{AP}}, \rightarrow_{B_i}, \mathcal{F}_{B_i})$, the *Product Büchi Automaton* $P_i = \text{wTS}_i \otimes B_i$ is a tuple $(\mathcal{Q}_{P_i}, \mathcal{Q}_{P_i}^0, \longrightarrow_{P_i}, w_{P_i}, \mathcal{F}_{P_i})$, where the following conditions hold:

1) $\mathcal{Q}_{P_i} = \mathcal{Q}_i \times \mathcal{Q}_{B_i}$ is the set of states.
2) $\mathcal{Q}_{P_i}^0 = q_i^0 \times \mathcal{Q}_{B_i}^0$ is a set of initial states.
3) $\longrightarrow_{P_i} \subseteq \mathcal{Q}_{P_i} \times \mathcal{Q}_{P_i}$ is the transition relation. Transition $(q_P, q_P') \in \longrightarrow_{P_i}$, where $q_P = (q_i^{\mathbf{v}_j}, q_B) \in \mathcal{Q}_{P_i}$ and $q_P' = (q_i^{\mathbf{v}_e}, q_B') \in \mathcal{Q}_{P_i}$, exists if $(q_i^{\mathbf{v}_j}, q_i^{\mathbf{v}_e}) \in \rightarrow_i$ and $(q_B, L_i(q_i^{\mathbf{v}_j}), q_B') \in \rightarrow_B$.
4) $w_{P_i} : \mathcal{Q}_{P_i} \times \mathcal{Q}_{P_i} \rightarrow \mathbb{R}_+$ is the weight function, defined as $w_{P_i}((q_i^{\mathbf{v}_j}, q_B), (q_i^{\mathbf{v}_e}, q_B')) = w_i(q_i^{\mathbf{v}_j}, q_i^{\mathbf{v}_e})$.
5) $\mathcal{F}_{P_i} = \mathcal{Q}_i \times \mathcal{F}_{B_i}$ is a set of accepting/final states.

More precisely, a motion plan $\tilde{\tau}_i^0$ that satisfies ψ_i can be derived using graph search techniques on P_i, which can be viewed as a weighted graph $\mathcal{G}_{P_i} = \{\mathcal{V}_{P_i}, \mathcal{E}_{P_i}, w_{P_i}\}$, where $\mathcal{V}_{P_i} = \mathcal{Q}_{P_i}$, the set of edges $\mathcal{E}_{P_i}$ is determined by the transition relation $\longrightarrow_{P_i}$, and the weight function w_{P_i} is defined in Definition 5.1, see e.g., [20]–[23], [37], [38]. Then, a path from an initial state to an accepting state in $\mathcal{G}_{P_i}$ (the prefix path) followed by a cycle around this accepting state (the suffix path), which is repeated indefinitely, results in an accepting run of the PBA that has the following prefix–suffix structure:

$$\rho_{P_i}^0 = \rho_{P_i}^{\text{pre},0}\left[\rho_{P_i}^{\text{suf},0}\right]^\omega = \underbrace{(q_{\text{wTS}_i}^0, q_{B_i}^0)}_{\in\mathcal{Q}_{P_i}^0}(q_{\text{wTS}_i}^1, q_{B_i}^1)\ldots\underbrace{(q_{\text{wTS}_i}^F, q_{B_i}^F)}_{=q_{P_i}^F\in\mathcal{F}_{P_i}}$$
$$\times\left[(q_{\text{wTS}_i}^F, q_{B_i}^F)\ldots(q_{\text{wTS}_i}^L, q_{B_i}^L)\right]^\omega \tag{9}$$

where with slight abuse of notation, $q_{\text{wTS}_i}^\beta$ and $q_{B_i}^\beta$ denote a state of wTS_i and B_i, respectively, for all $\beta \in \{0,\ldots,F,\ldots,L\}$. The projection of $\rho_{P_i}^0$ onto the state space of wTS_i, denoted by $\Pi|_{\text{wTS}_i}\rho_{P_i}^0$, results in the desired prefix–suffix motion plan.

$$\tilde{\tau}_i^0 = \Pi|_{\text{wTS}_i}\rho_{P_i}^0 = \tilde{\tau}_i^{\text{pre},0}\left[\tilde{\tau}_i^{\text{suf},0}\right]^\omega$$
$$= \left[q_{\text{wTS}_i}^0 \ldots q_{\text{wTS}_i}^F\right]\left[q_{\text{wTS}_i}^F \ldots q_{\text{wTS}_i}^L\right]^\omega \tag{10}$$

that satisfies ψ_i provided feasible initial communication points have been selected [34]. To reduce the computational cost of synthesizing $\tilde{\tau}_i^0$, we only require a feasible plan $\tilde{\tau}_i^0$ and not the optimal one that minimizes (3), especially since subsequent paths $\texttt{path}_i^{n_i}$ will get optimized online.

Given the motion plans $\tilde{\tau}_i^0 = \tilde{\tau}_i^{\text{pre},0}[\tilde{\tau}_i^{\text{suf},0}]^\omega$, we design the discrete paths $\texttt{path}_i^0$ as follows. First, we initialize $\texttt{path}_i^0$ as $\texttt{path}_i^0 = \tilde{\tau}_i^{\text{pre},0}|\tilde{\tau}_i^{\text{suf},0}$. Recall that all paths $\texttt{path}_i^0$ are designed so that if executed, then robot i will communicate once with all teams $\mathcal{T}_m$, $m \in \mathcal{M}_i$, in an order that respects the schedules $\texttt{sched}_i$. Therefore, the state $q_i^{\mathbf{v}_j}$ corresponding to the candidate communication point $\mathbf{v}_j$, $j \in \mathcal{C}_m$, appears at least once in the suffix part of $\tilde{\tau}_i^0$, by definition of ψ_i, for all $m \in \mathcal{M}_i$. However, these communication states may not appear in $\texttt{path}_i^0 = \tilde{\tau}_i^{\text{pre},0}|\tilde{\tau}_i^{\text{suf},0}$ in an order that respects the schedules $\texttt{sched}_i$, as this is not required by the LTL formula ψ_i in (7). Therefore, we append at the end of $\texttt{path}_i^0$ the suffix part $\tilde{\tau}_i^{\text{suf},0}$ enough times so that $\texttt{path}_i^0 = \tilde{\tau}_i^{\text{pre},0}|\tilde{\tau}_i^{\text{suf},0}|\ldots|\tilde{\tau}_i^{\text{suf},0}$ respects the schedule $\texttt{sched}_i$, i.e., there exists a sequence of indices κ_i^m that point to entries in $\texttt{path}_i^0$ corresponding to states $q_i^{\mathbf{v}_j}$ with $\mathbf{v}_j$, $j \in \mathcal{C}_m$, that satisfy $\kappa_i^m < \kappa_i^h$, if the communication event for team $\mathcal{T}_m$ appears before the communication event for team $\mathcal{T}_h$ in $\texttt{sched}_i$, for all teams $\mathcal{T}_m$, $\mathcal{T}_h$, $m, h \in \mathcal{M}_i$, see also Example 5.3. Note that since the state $q_i^{\mathbf{v}_j}$, $j \in \mathcal{C}_m$, appears at least once in the suffix part of $\tilde{\tau}_i^0$, for all $m \in \mathcal{M}_i$, the suffix part $\tilde{\tau}_i^{\text{suf},0}$ will be appended to $\texttt{path}_i^0$ at most $|\mathcal{M}_i| - 1$ times. With slight abuse of notation, the initial path τ_i^0 in (6) is defined using only $\texttt{path}_i^0$ as follows:

$$\tau_i^0 = \tilde{\tau}_i^{\text{pre},0}[\tilde{\tau}_i^{\text{suf},0}|\ldots|\tilde{\tau}_i^{\text{suf},0}]^\omega. \tag{11}$$

In what follows, we show that to find a feasible initial combination of communication points $\texttt{comb}_b$ that is needed to determine initial plans $\tilde{\tau}_i^0$, the robots can search locally in the set of $\prod_{m\in\mathcal{M}} |\mathcal{C}_m|$ possible combinations of communication points by solving at most $\prod_{m\in\mathcal{M}_i} |\mathcal{C}_m|$ control synthesis problems each, instead of $\prod_{m\in\mathcal{M}} |\mathcal{C}_m|$. To see this, observe that, for any robot $i \in \mathcal{N}$, there exist multiple combinations $\texttt{comb}_b$ that share the same communication points for all teams $\mathcal{T}_m$, $m \in \mathcal{M}_i$, and only differ in the communication points for teams $\mathcal{T}_m$, $m \in \mathcal{M} \setminus \mathcal{M}_i$. All these combinations, correspond to the same formula ψ_i, which means that that robot i needs to solve a single control synthesis problem to determine if they are feasible. Motivated by this observation, in the following lemma, we show that if every robot $i \in \mathcal{N}$ solves locally at most $\prod_{m\in\mathcal{M}_i} |\mathcal{C}_m|$ control synthesis problems, then all combinations $\texttt{comb}_b$ will be exhaustively explored. By combining the feasible local combinations of communication points $\texttt{comb}_{b_i}^i$ that can be assigned to teams $\mathcal{T}_m$, $m \in \mathcal{M}_i$, where $b_i \in \{1,\ldots,\prod_{m\in\mathcal{M}_i} |\mathcal{C}_m|\}$, that are identified by all robots i, it it easy to obtain feasible global combinations $\texttt{comb}_b$. Note that, in general, it holds that $\prod_{m\in\mathcal{M}_i} |\mathcal{C}_m| \leq \prod_{m\in\mathcal{M}} |\mathcal{C}_m|$, where the equality holds if $\mathcal{M}_i = \mathcal{M}$ or if $|\mathcal{C}_m| = 1$, for all $m \in \mathcal{M} \setminus \mathcal{M}_i$.

Moreover, $\prod_{m\in\mathcal{M}_i}|\mathcal{C}_m|$ is smaller for sparse graphs $\mathcal{G}_\mathcal{T}$, given a fixed number of teams and fixed sets $\mathcal{C}_m$.

Lemma 5.2 (Complexity of initialization): Let $\texttt{comb}^i_{b_i}$ with $b_i \in \{1,\ldots,\prod_{m\in\mathcal{M}_i}|\mathcal{C}_m|\}$ denote a combination of communication points that can be assigned to teams $\mathcal{T}_m$, $m \in \mathcal{M}_i$. Moreover, assume that every robot $i \in \mathcal{N}$ solves $\prod_{m\in\mathcal{M}_i}|\mathcal{C}_m|$ control synthesis problems using the LTL formula (7), one for every combination $\texttt{comb}^i_{b_i}$. Then, the robots can collectively detect any feasible combination of communication points $\texttt{comb}_b$, $b \in \{1,\ldots,\prod_{m\in\mathcal{M}}|\mathcal{C}_m|\}$, if it exists, that can be assigned to all teams $\mathcal{T}_m$, $m \in \mathcal{M}$.

Proof: In what follows, we show by contradiction that under this local construction of $\texttt{comb}_b$, the robots can detect all feasible combinations $\texttt{comb}_b$. Assume that there exists a feasible combination $\texttt{comb}_b$, that cannot be detected if all robots solve their respective $\prod_{m\in\mathcal{M}_i}|\mathcal{C}_m|$ control synthesis problems. Also, let $\Pi|_{\mathcal{M}_i}\texttt{comb}_b$ denote the combination of communication points in $\texttt{comb}_b$ that correspond to all teams $\mathcal{T}_m$, $m \in \mathcal{M}_i$. Since $\texttt{comb}_b$ cannot be detected by the robots, this means that there exists at least one robot i that either could not find a feasible solution to the control synthesis problem that corresponds to the combination $\Pi|_{\mathcal{M}_i}\texttt{comb}_b$ or did not consider the combination $\Pi|_{\mathcal{M}_i}\texttt{comb}_b$. The first case contradicts the assumption that $\texttt{comb}_b$ is a feasible combination of communication points that can be assigned to all teams $\mathcal{T}_m$, $m \in \mathcal{M}$, while the second case contradicts the assumption that every robot $i \in \mathcal{N}$ searches over all combinations $\texttt{comb}^i_{b_i}$. ■

Example 5.3 (Construction of $\texttt{path}^0_i$): Consider a robot i with $\mathcal{M}_i = \{2,3,4,5\}$ and communication schedule $\texttt{sched}_i = [2,3,X,4,5]^\omega$. Consider also the motion plan $\tilde{\tau}^0_i = \tilde{\tau}^{\text{pre},0}_i[\tilde{\tau}^{\text{suf},0}_i]^\omega = q_i^{\mathbf{v}_1}\, q_i^{\mathbf{v}_6}\, q_i^{\mathbf{v}_4}\, q_i^{\mathbf{v}_5}\, q_i^{\mathbf{v}_2}\, q_i^{\mathbf{v}_3}\, [q_i^{\mathbf{v}_3}\, q_i^{\mathbf{v}_5}\, q_i^{\mathbf{v}_4}\, q_i^{\mathbf{v}_6}\, q_i^{\mathbf{v}_2}]^\omega$, where $\mathbf{v}_2$, $\mathbf{v}_3$, $\mathbf{v}_4$ are the candidate communication points for teams $\mathcal{T}_2$, $\mathcal{T}_3$, $\mathcal{T}_4$, respectively. The path $\texttt{path}^0_i$ is initialized as $\texttt{path}^0_i = \tilde{\tau}^{\text{pre},0}_i|\tilde{\tau}^{\text{suf},0}_i$. To ensure the existence of indices κ^m_i in $\texttt{path}^0_i$ for all teams $\mathcal{T}_m$, $m \in \mathcal{M}_i$, that respect the schedule $\texttt{sched}_i$, the suffix part needs to be appended to $\texttt{path}^0_i$ once more, i.e., $\texttt{path}^0_i = q_i^{\mathbf{v}_1} q_i^{\mathbf{v}_6} q_i^{\mathbf{v}_4} q_i^{\mathbf{v}_5} q_i^{\mathbf{v}_2} q_i^{\mathbf{v}_3} [q_i^{\mathbf{v}_3} q_i^{\mathbf{v}_5} q_i^{\mathbf{v}_4} q_i^{\mathbf{v}_6} q_i^{\mathbf{v}_2}][q_i^{\mathbf{v}_3} q_i^{\mathbf{v}_5} q_i^{\mathbf{v}_4} q_i^{\mathbf{v}_6} q_i^{\mathbf{v}_2}]$, where the sequence of states in brackets stands for the suffix part $\tau^{\text{suf},0}_i$. Observe that in $\texttt{path}^0_i$, there exists indices $\kappa^2_i = 5$, $\kappa^3_i = 6$, $\kappa^4_i = 9$, and $\kappa^5_i = 13$, so that $\kappa^2_i < \kappa^3_i < \kappa^4_i < \kappa^5_i$ as dictated by $\texttt{sched}_i$.

Remark 5.4 (Initialization): Note that there are cases where feasible initial communication points can be easily identified by inspection, e.g., if there exists a communication point $\mathbf{v}_j$, $j \in \mathcal{C}_m$ that 1) does not appear in the atomic propositions $\pi_i^{\mathbf{v}_e}$ that capture the tasks ϕ_i assigned to robots $i \in \mathcal{T}_m$ and 2) is directly connected to all locations $\mathbf{v}_e$, $e \in \mathcal{I}$, that robots $i \in \mathcal{T}_m$ should visit to accomplish their tasks, i.e., the atomic propositions $\pi_i^{\mathbf{v}_e}$ appear in the tasks ϕ_i, $i \in \mathcal{T}_m$. Then, $\mathbf{v}_j$, $j \in \mathcal{C}_m$, is a feasible communication point for team $\mathcal{T}_m$, since it does not violate the tasks ϕ_i for all $i \in \mathcal{T}_m$ and it does not affect the communication points the other teams can select due to 1). Also, due to 2), robots $i \in \mathcal{T}_m$ can visit $\mathbf{v}_j$ directly from any location $\mathbf{v}_e$ without passing through locations that may violate ϕ_i. Finally, if the negation operator does not appear in the tasks ϕ_i of all robots $i \in \mathcal{T}_m$, then any communication point $\mathbf{v}_j$, $j \in \mathcal{C}_m$, assigned to team $\mathcal{T}_m$ is feasible.

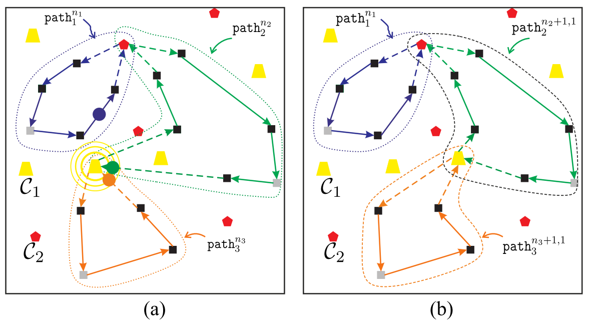


Fig. 2. Illustration of Algorithm 2 for network of $N = 3$ robots (colored dots) with schedules $\texttt{sched}_1 = [X,2]^\omega$, $\texttt{sched}_2 = [1,2]^\omega$, and $\texttt{sched}_3 = [1,X]^\omega$. All robots currently execute paths $\texttt{path}_i^{n\,i}$ constructed by Algorithm 2. (a) Communication events within team $\mathcal{T}_1$. (b) Corresponding paths $\texttt{path}_i^{n_i+1,c_i}$ constructed at this communication event. Observe in (b) that robot 3 has finalized the construction of the paths $\texttt{path}_3^{n_3+1}$ since $|\mathcal{M}_3| = 1$. The gray square denotes the state $\Pi|_{\text{wTS}_i} q^F_{P_i}$.

Remark 5.5 (Formula $\phi_{com,i}$): An alternative selection for $\phi_{\text{com},i}$, defined in (8), is $\phi'_{\text{com},i} = \square(\lozenge\mathbf{v}_{j\in\mathcal{C}_m} \wedge (\lozenge\mathbf{v}_{e\in\mathcal{C}_h} \wedge (\lozenge\mathbf{v}_{d\in\mathcal{C}_g} \wedge \ldots)))$ that requires robot i to visit communication points for all teams $\mathcal{T}_m$, $m \in \mathcal{M}_i$ in a given order that respects the schedules $\texttt{sched}_i$. However, using this formula, there is still no guarantee that all communication points will appear in the suffix part $\tilde{\tau}^{\text{suf},0}_i$ in an order that respects $\texttt{sched}_i$, as this depends on the structure of the LTL formula ϕ_i and the wTS_i. Therefore, we have chosen (8), instead of $\phi'_{\text{com},i}$, since (8) corresponds to a much smaller NBA that makes the proposed algorithm more computationally efficient.

B. Online Construction of Paths

The construction of the paths $\texttt{path}_i^{n_i}$ occurs online and in an iterative fashion, for all $n_i \in \mathbb{N}_+$, as the robots navigate the workspace. Specifically, $\texttt{path}_i^{n_i+1}$ is constructed and updated every time robot i participates at communication events, as it executes $\texttt{path}_i^{n_i}$. Hereafter, we denote by $\texttt{path}_i^{n_i+1,c_i}$ the path constructed when robot i participates at the c_ith communication event in $\texttt{path}_i^{n_i}$. The iteration index c_i is initialized as $c_i = 1$ at the beginning of execution of $\texttt{path}_i^{n_i}$ and is updated as $c_i = c_i + 1$ when the path $\texttt{path}_i^{n_i+1,c_i}$ is constructed. Once robot i has participated in $|\mathcal{M}_i|$ communication events, i.e., $c_i = |\mathcal{M}_i|$, then the next path $\texttt{path}_i^{n_i+1} = \texttt{path}_i^{n_i+1,|\mathcal{M}_i|}$ has been constructed and will be executed after the execution of $\texttt{path}_i^{n_i}$.

In what follows, we present the distributed construction of $\texttt{path}_i^{n_i+1}$, which is also summarized in Algorithm 2 and illustrated in Fig. 2. Also, in Algorithm 2, for simplicity of notations, we assume that the indices of the teams in the sets $\mathcal{M}_i$ are ordered as per the respective schedules $\texttt{sched}_i$. This means that if the robots in team $\mathcal{T}_m$, $m = \mathcal{M}_i(c_i)$, communicate then the next communication event that robot i needs to participate during the execution of $\texttt{path}_i^{n_i}$ is $\mathcal{M}_i(c_i + 1)$. Assume that the robots $i \in \mathcal{T}_m$, $m = \mathcal{M}_i(c_i)$, communicate during the

Algorithm 2: Distributed Construction of $\texttt{path}_i^{n_i+1}$, $\forall i \in \mathcal{T}_m$, $\forall n_i \in \mathbb{N}$.

```
Input: Set $\mathcal{C}_m$, wTS$_i$, $n_i$
Output: Paths: $\texttt{path}_i^{n_i+1}$, $\forall i \in \mathcal{T}_m$
1  Initialize $c_i = 1$;
2  while $c_i \leq |\mathcal{M}_i|$ do
3      if team $\mathcal{T}_m$ with $m = \mathcal{M}_i(c_i)$ communicates then
4          for $j \in \mathcal{C}_m$ do
5              Define $\psi_i$ by (7) given (i) $\mathbf{v}_j$ for team $\mathcal{T}_m$
                 and (ii) the selected communication points
                 for other teams $\mathcal{T}_h$, $h \in \mathcal{M}_i \setminus \{m\}$;
6              Construct $P_i$ and synthesize a suffix loop
                 $\rho_{P_i}^{\text{suf},j}$ (if it exists) around $q_{P_i}^F$ defined in (9)
                 that minimizes $J(\Pi|_{\text{wTS}_i}\rho_{P_i}^{\text{suf},j})$;
7              Compute $\tilde{\tau}_i^{\text{suf},j} = \Pi|_{\text{wTS}_i}\rho_{P_i}^{\text{suf},j}$;
8      Define $\texttt{Cost}_j = \sum_{r\in\mathcal{T}_m} J(\tilde{\tau}_r^{\text{suf},j})$, for all $j \in \mathcal{C}_m$;
9      Compute $j^* = \text{argmin}_{j\in\mathcal{C}_m}\{\texttt{Cost}_j\}_{j\in\mathcal{C}_m}$ ;
10     Initialize paths $\texttt{path}_i^{n_i+1,c_i} = \tilde{\tau}_i^{\text{suf},j^*}$, for all $i \in \mathcal{T}_m$;
11     while $\texttt{path}_i^{n_i+1,c_i}$ does not respect $\texttt{sched}_i$ do
12         Update $\texttt{path}_i^{n_i+1,c_i} = \texttt{path}_i^{n_i+1,c_i}|\tilde{\tau}_i^{\text{suf},j^*}$
13     Update $c_i = c_i + 1$;
14 Return path $\texttt{path}_i^{n_i+1} = \texttt{path}_i^{n_i+1,|\mathcal{M}_i|}$;
```

execution of the paths $\texttt{path}_i^{n_i}$. To design the paths $\texttt{path}_i^{n_i+1,c_i}$, the robots $i \in \mathcal{T}_m$ need to select a new communication point $\mathbf{v}_j$, $j \in \mathcal{C}_m$ and possibly update the waypoints $\mathbf{v}_j$, $j \in \mathcal{I}$ so that the LTL$_{-\bigcirc}$ tasks ϕ_i are satisfied. The paths $\texttt{path}_i^{n_i+1,c_i}$ are constructed in a similar way as the paths $\texttt{path}_i^0$ in Section V-A. The only difference lies in the definition of the LTL formula ψ_i in (7), since now the robots need to autonomously select a new optimal communication point for team $\mathcal{T}_m$ given the already selected communication points for all other teams. Specifically, all robots $i \in \mathcal{T}_m$ perform in parallel the following two steps for all candidate new communication points $\mathbf{v}_j$, $j \in \mathcal{C}_m$, for team $\mathcal{T}_m$ (lines 2–4, Algorithm 2). First, every robot $i \in \mathcal{T}_m$ constructs the LTL formula ψ_i, defined in (7), for every candidate new communication point $\mathbf{v}_j$, $j \in \mathcal{C}_m$ for team $\mathcal{T}_m$, and given the already selected communication points for all other teams $\mathcal{T}_h, h \in \mathcal{M}_i \setminus \{m\}$, see (7) (line 5, Algorithm 2). Second, given the wTS$_i$ and the NBA B_i that corresponds to ψ_i, every robot $i \in \mathcal{T}_m$ constructs the corresponding PBA $P_i = \text{wTS}_i \otimes B_i$ and computes the *optimal* suffix loop, denoted by $\rho_{P_i}^{\text{suf},j}$, around the same PBA final state $q_{P_i}^F = (q_{\text{wTS}_i}^F, q_B^F)$ that was used to construct the initial suffix loop of $\rho_{P_i}^0$ in (9). Note that by optimal suffix loop $\rho_{P_i}^{\text{suf},j}$, we refer to the path that minimizes the cost $J(\Pi|_{\text{wTS}_i}\rho_{P_i}^{\text{suf},j})$. The projection of this optimal suffix loop $\rho_{P_i}^{\text{suf},j}$ on the state space of wTS$_i$ is denoted by $\tilde{\tau}_i^{\text{suf},j}$ (lines 6–7, Algorithm 2).

Once all robots $i \in \mathcal{T}_m$ have constructed the suffix parts $\tilde{\tau}_i^{\text{suf},j}$ for all $j \in \mathcal{C}_m$, they compute the total cost $\texttt{Cost}_j = \sum_{i\in\mathcal{T}_m} J(\tilde{\tau}_i^{\text{suf},j})$ (line 8, Algorithm 2). This cost captures the distance that all robots $i \in \mathcal{T}_m$ need to travel during a single execution of the suffix parts $\tilde{\tau}_i^{\text{suf},j}$ if the new communication point for team $\mathcal{T}_m$ is $\mathbf{v}_j$, $j \in \mathcal{C}_m$. Among all the suffix parts $\tilde{\tau}_i^{\text{suf},j}$, all robots $i \in \mathcal{T}_m$ select the suffix part $\tilde{\tau}_i^{\text{suf},j^*}$, with $j^* = \text{argmin}_j\{\texttt{Cost}_j\}_{j\in\mathcal{C}_m}$ (line 9, Algorithm 2).

Given the optimal suffix part $\tilde{\tau}_i^{\text{suf},j^*}$, we construct $\texttt{path}_i^{n_i+1,c_i}$ exactly as the initial paths $\texttt{path}_i^0$. Specifically, first, the paths $\texttt{path}_i^{n_i+1,c_i}$ are initialized as $\texttt{path}_i^{n_i+1,c_i} = \tilde{\tau}_i^{\text{suf},j^*}$ (line 10, Algorithm 2]. Then, we append $\tilde{\tau}_i^{\text{suf},j^*}$ to $\texttt{path}_i^{n_i+1,c_i}$ as many times as needed to satisfy the schedules $\texttt{sched}_i$ (line 11, Algorithm 2). Note that since the state $q_i^{\mathbf{v}_j}$, $j \in \mathcal{C}_m$ appears at least once in the suffix part of $\tilde{\tau}_i^{\text{suf},j^*}$, for all $m \in \mathcal{M}_i$, the suffix part $\tilde{\tau}_i^{\text{suf},j^*}$ will be appended at most $|\mathcal{M}_i| - 1$ times to $\texttt{path}_i^{n_i+1,c_i}$. After the construction of $\texttt{path}_i^{n_i+1,c_i}$, the iteration index c_i is updated as $c_i = c_i + 1$ and points to the next path $\texttt{path}_i^{n_i+1,c_i}$ that will be constructed when robot i communicates with the robots in team $\mathcal{T}_h$, $h = \mathcal{M}_i(c_i)$ (line 11, Algorithm 2).[6] If $c_i = |\mathcal{M}_i|$, then this corresponds to the last communication event that robot i needs to participate during the execution of $\texttt{path}_i^{n_i}$ and, therefore, the construction of $\texttt{path}_i^{n_i+1}$ is finalized, i.e., $\texttt{path}_i^{n_i+1} = \texttt{path}_i^{n_i+1,|\mathcal{M}_i|}$ (line 14, Algorithm 2). In this case, c_i is reinitialized as $c_i = 1$ (line 1, Algorithm 2).

Remark 5.6 (Implicit synchronization across robots): While the robots transition from $\texttt{path}_i^{n_i}$ to $\texttt{path}_i^{n_i+1}$ asynchronously, there is an implicit synchronization in the system since, for any iteration $n \in \mathbb{N}_+$, the robots that finish the execution of $\texttt{path}_i^n$, first cannot finish the execution of $\texttt{path}_i^{n+1}$ until all other robots r have finished the execution of their paths $\texttt{path}_r^n$. The reason is that every robot i has to participate in $|\mathcal{M}_i|$ communication events during the execution of $\texttt{path}_i^n$ and the graph of teams $\mathcal{G}_\mathcal{T}$ is connected by construction of the teams. Therefore, if there exist robots i and r where robot i executes the path $\texttt{path}_i^{n+2}$ and robot r executes the path $\texttt{path}_r^n$, it must be the case that robot i has skipped at least one communication event during the execution of $\texttt{path}_i^{n+1}$, which cannot happen by construction of the proposed algorithm. Therefore, there exist time instants t_n so that $\texttt{path}_i^{n_i} = \texttt{path}_i^n$, for every $n \in \mathbb{N}_+$ and for all $i \in \mathcal{N}$.

Remark 5.7 (Computational cost): Note that to design the path $\texttt{path}_i^{n_i+1,c_i}$, every robot i needs to solve $|\mathcal{C}_m|$ optimal control synthesis problems. Therefore, the computational cost of Algorithm 2 increases with $|\mathcal{C}_m|$. To reduce the computational burden, Algorithm 2 can be executed over subsets $\bar{\mathcal{C}}_m \subseteq \mathcal{C}_m$ that can change with iterations n_i but always include the current communication point for team $\mathcal{T}_m$. The latter is required to ensure that paths $\texttt{path}_i^{n_i}$ can be synthesized for all $n_i > 0$, if a solution to Problem 1 exists, see Proposition 6.1. Moreover, sampling-based approaches can be used to synthesize the suffix parts $\tilde{\tau}_i^{\text{suf},j}$ that do not require the explicit construction of the PBA or the application of computationally expensive graph search methods [22]. Finally, in Proposition 6.8, we show that Algorithm 2 terminates after a finite number of iterations, i.e., a repetitive pattern in the paths $\texttt{path}_i^{n_i}$ is eventually detected, for all $i \in \mathcal{N}$. This means that the computational cost is bounded.

Remark 5.8 (Fixed final state $q_{P_i}^F$): Recall that the fixed PBA final state $q_{P_i}^F$, defined in (9), is used to construct the

[6]Note that the next communication event $\mathcal{M}_i(c_i)$ respects the schedules $\texttt{sched}_i$, by construction of $\mathcal{M}_i$.

Algorithm 3: Asynchronous Execution of $\texttt{path}_i^{n_i}$.

```
Input: Discrete path $\texttt{path}_i^0$ and set $\mathcal{K}_i^0$
1 $n_i = 0$;
2 for $\kappa_i = 1 : K_i^{n_i}$ do
3     Move towards the state $\texttt{path}_i^{n_i}(\kappa_i)$;
4     if $\kappa_i \in \mathcal{K}_i^{n_i}$ then
5         Wait at communication point $\mathbf{v}_j$, $j \in \mathcal{C}_m$ [Definition 5.9];
6         if all robots in $\mathcal{T}_m$ are present at node $\mathbf{v}_j$ then
7             Communication occurs within team $\mathcal{T}_m$ and execution of Algorithm 2 ;
8 Execute the next path $\texttt{path}_i^{n_i+1}$;
```

paths $\texttt{path}_i^{n_i+1}$, for all $n_i \in \mathbb{N}$ and for all $i \in \mathcal{N}$, This requirement can be relaxed by defining the paths $\texttt{path}_i^{n_i+1,c_i}$ as $\texttt{path}_i^{n_i+1,c_i} = \Pi_{\text{wTS}_i}\rho_{c_i}$, where $\rho_{c_i} = \rho_{c_i,1}|\rho_{c_i,2}|\ldots,|\rho_{c_i,K}$ is a feasible path in the state space of P_i, $\rho_{c_i,k}$ a feasible path in the state space of P_i that connects two possibly different PBA final states, for all $k \in \{1,\ldots,K\}$, and $K < |\mathcal{M}_i|$ is determined so that execution of $\texttt{path}_i^{n_i+1,c_i}$, for any c_i, ensures that robot i will communicate exactly once with all teams $\mathcal{T}_m$, $m \in \mathcal{M}_i$.[7] In this case, $\texttt{path}_i^{n_i+1}$ is not a periodic path that can be executed infinitely and, therefore, (6) cannot be used to model the solution of Algorithm 2, which will now be an infinite aperiodic sequence of states. Also, allowing the paths $\texttt{path}_i^{n_i+1}$ to be associated with multiple PBA final states would increase the computational burden of Algorithm 2, as it requires the computation of K paths in the PBA P_i.

C. Asynchronous Execution

In the majority of global LTL-based motion planning, robots are assumed to execute their assigned motion plans synchronously, i.e., all the robots pick synchronously their next states, see e.g., [25], [29]. However, assuming that robot motion is performed in a synchronous way is conservative due to, e.g., uncertainty and exogenous disturbances in the arrival times of the robots at their next locations as per the discrete path $\texttt{path}_i^{n_i}$. To the contrary, here the discrete plans $\texttt{path}_i^{n_i}$ are executed asynchronously across the robots, as per Algorithm 3.

In Algorithm 3, $\texttt{path}_i^{n_i}(\kappa_i)$ stands for the κ_ith state of the discrete path $\texttt{path}_i^{n_i}$. The different indices κ_i for the robots's states in the plans $\texttt{path}_i^{n_i}$ allow us to model the situation where the robots pick asynchronously their next states in wTS$_i$. Also, in Algorithm 3, the set $\mathcal{K}_i^{n_i}$ collects an index κ_i^m for all teams $\mathcal{T}_m$, $m \in \mathcal{M}_i$ that satisfy $\texttt{path}_i^{n_i}(\kappa_i^m) = q_i^{\mathbf{v}_j}$, where $q_i^{\mathbf{v}_j}$ is associated with a communication point $\mathbf{v}_j$, $j \in \mathcal{C}_m$, $m \in \mathcal{M}_i$ and respect the schedules as described in Section V-A. Note that such indices κ_i^m exist by construction of the paths $\texttt{path}_i^{n_i}$. According to Algorithm 3, when the state of robot i is $\texttt{path}_i^{n_i}(\kappa_i) = q_i^{\mathbf{v}_j}$, $j \in \mathcal{I}$ i.e., when robot i arrives at a location $\mathbf{v}_j$ in the workspace, it checks if $\kappa_i \in \mathcal{K}_i^{n_i}$ (lines 3–4, Algorithm 3). If so, then robot i performs the following control policy (line 5, Algorithm 3).

[7]Observe that if all paths $\rho_{c_i,k}$ are defined as the shortest loops around $q_{P_i}^F$, then $\rho_{c_i,k}$ coincides with the $\rho_{P_i}^{\text{suf},j}$, for all $k \in \{1,\ldots,K\}$.

Definition 5.9 (Control policy at communication locations): Every robot i that arrives at a communication location $\mathbf{v}_j$, $j \in \mathcal{C}_m$, $m \in \mathcal{M}_i$, selected by Algorithm 2 waits there indefinitely, or until all other robots in the team arrive.

When all the other robots of team $\mathcal{T}_m$ arrive at the communication location $\mathbf{v}_j$, $j \in \mathcal{C}_m$, communication for team $\mathcal{T}_m$ occurs and Algorithm 2 is executed to synthesize $\texttt{path}_i^{n_i+1,c_i}$ (lines 6–7, Algorithm 3). After that, robot i moves toward the next state $\texttt{path}_i^{n_i}(\kappa_i+1)$ (line 2, Algorithm 3). In line 2 of Algorithm 3, $K_i^{n_i}$ denotes the number of waypoints/states in $\texttt{path}_i^{n_i}$. This process is repeated until robot i visits all locations in $\texttt{path}_i^{n_i}$. Once robot i visit all waypoints of $\texttt{path}_i^{n_i}$, it starts executing the path $\texttt{path}_i^{n_i+1}$ (line 8, Algorithm 3). If n_i is the last iteration of Algorithm 2, then $\texttt{path}_i^{n_i}$ is executed indefinitely.

VI. Algorithm Analysis

In this section, we present results pertaining to completeness and optimality of the proposed distributed control framework. Specifically, in Section VI-A, we show that if there exists a solution to Problem 1, then the proposed distributed framework will generate prefix–suffix plans $\tau_i^{n_i}$, defined in (6), that can be executed asynchronously according to Algorithm 3, and satisfy the assigned LTL tasks and the intermittent connectivity requirement, for every iteration $n_i \geq 0$. Then, in Section VI-B, we show that the cost of the suffix part of the plans in (6) decreases with every iteration of Algorithm 2, while in Section VI-C we show that these plans converge in a finite number of iterations. Note that since the proposed algorithm is online, synthesis and execution take place concurrently and this is reflected in the subsequent results.

A. Completeness

First, we show that if there exists a feasible solution to Problem 1, then feasible paths $\texttt{path}_i^{n_i}$ i.e., feasible loops $\rho_{P_i}^{n_i}$ defined over the state space of the corresponding PBA P_i, can be designed, for all $n_i \in \mathbb{N}$. This implies that Algorithm 2 can generate plans $\tau_i^{n_i}$, for any $n_i \geq 0$ and that robots i in any team $\mathcal{T}_m$, for $m \in \mathcal{M}_i$, can stop executing Algorithm 2 at any iteration $n_i^m \geq 0$.

Proposition 6.1 (Feasibility): Assume that there exists a solution to Problem 1. Then, feasible plans $\texttt{path}_i^{n_i}$ can be constructed for all $n_i \geq 0$.

Proof: First, observe that if there exists a solution to Problem 1, then feasible initial paths $\tilde{\tau}_i^0$ that satisfy ψ_i in (7), for all robots $i \in \mathcal{N}$, will be detected since at initialization we exhaustively search through all available communication points assigned to the teams $\mathcal{T}_m$, $m \in \mathcal{M}$, as shown in Lemma 5.2. Therefore, initial feasible paths $\texttt{path}_i^0$ can be constructed. Then, to prove this result, it suffices to show that if there exists a feasible path $\texttt{path}_i^{n_i}$, then Algorithm 2 can construct a feasible path $\texttt{path}_i^{n_i+1}$ for all $n_i \geq 0$. This means that Algorithm 2 will not deadlock. Note that Algorithm 2 does not search over all combinations of communication points assigned to the teams.

In what follows, we show by induction that if there exists a feasible path $\texttt{path}_i^{n_i}$, then Algorithm 2 will construct feasible paths $\texttt{path}_i^{n_i+1,c_i}$ for all $c_i \in \{1,\ldots,|\mathcal{M}_i|\}$, and consequently, it will construct a feasible path $\texttt{path}_i^{n_i+1,|\mathcal{M}_i|} = \texttt{path}_i^{n_i+1}$ for all $n_i \geq 0$. To show this, we first define the sets $\mathcal{F}_{c_i}^{n_i+1}$ that collect the suffix parts $\tilde{\tau}_i^{\text{suf},j}$ constructed by Algorithm 2 during the construction of $\texttt{path}_i^{n_i+1,c_i}$, for all $c_i \in \{1,\dots,|\mathcal{M}_i|\}$. Now, assume that there exists a feasible path $\texttt{path}_i^{n_i}$. This means that $\mathcal{F}_0^{n_i+1} := \{\tilde{\tau}_i^{\text{suf},j^*,n_i}\} \neq \emptyset$, where $\tilde{\tau}_i^{\text{suf},j^*,n_i}$ is the suffix part used for the construction of the path $\texttt{path}_i^{n_i}$. First, we show that $\mathcal{F}_1^{n_i+1} \neq \emptyset$, i.e., that Algorithm 2 will construct a feasible plan $\texttt{path}_i^{n_i+1,1}$. Note that the only difference between the paths $\texttt{path}_i^{n_i+1,1}$ and $\texttt{path}_i^{n_i} = \texttt{path}_i^{n_i,|\mathcal{M}_i|}$, in terms of the selected communication points for teams $\mathcal{T}_m$, $m \in \mathcal{M}_i$, lies in the selected communication point of exactly one team $\mathcal{T}_m$, $m \in \mathcal{M}_i$. Also, recall that Algorithm 2 searches over all communication points $j \in \mathcal{C}_m$, including the current communication point of $\mathcal{T}_m$ that appears in $\texttt{path}_i^{n_i}$, to select the new communication point for team $\mathcal{T}_m$. Therefore, there exists an optimal control synthesis problem that is solved by Algorithm 2 during the computation of $\texttt{path}_i^{n_i+1,1}$ such that the LTL formula ψ_i is defined over the communication points selected in $\texttt{path}_i^{n_i,|\mathcal{M}_i|}$. Since this optimal control synthesis problem is feasible, by the assumption that $\texttt{path}_i^{n_i}$ is a feasible path, the generated suffix part, which was also used to construct $\texttt{path}_i^{n_i,|\mathcal{M}_i|}$, belongs to $\mathcal{F}_1^{n_i+1}$, i.e., $\mathcal{F}_1^{n_i+1} \neq \emptyset$. The inductive step follows. Assume that $\mathcal{F}_{c_i}^{n_i+1} \neq \emptyset$. Then, following the same logic as before we can show that the feasible suffix path used to construct $\texttt{path}_i^{n_i+1,c_i}$ belongs to $\mathcal{F}_{c_i+1}^{n_i+1}$, i.e., $\mathcal{F}_{c_i+1}^{n_i+1} \neq \emptyset$. By induction, we conclude that if $\mathcal{F}_0^{n_i+1} \neq \emptyset$, i.e., if there exists a feasible path $\texttt{path}_i^{n_i}$, then $\mathcal{F}_{c_i}^{n_i+1} \neq \emptyset$ for all $c_i \in \{1,\dots,|\mathcal{M}_i|\}$ and all $n_i \geq 0$. ■

To prove task satisfaction and intermittent communication, we also need to show that the network is *deadlock free* when the paths $\texttt{path}_i^{n_i}$ are executed according to Algorithm 3. Specifically, we assume that there is a *deadlock*, if there are robots of any team $\mathcal{T}_m$ that are waiting forever at a communication point, selected by Algorithm 2, for the arrival of all other robots of team $\mathcal{T}_m$ due to the control policy in Definition 5.9.

Proposition 6.2 (Deadlock free): The mobile robot network is deadlock free when the paths $\tau_i^{n_i}$ in (6) are executed according to Algorithm 3.

Proof: Let $\mathcal{W}_{\mathbf{v}_e} \subset \mathcal{T}_m$ denote the set of robots that are waiting at communication point $\mathbf{v}_e$, $e \in \mathcal{C}_m$, selected by Algorithm 2, for the arrival of the other robots that belong to team $\mathcal{T}_m$. Assume that the robots in $\mathcal{T}_m \backslash \mathcal{W}_{\mathbf{v}_e}$ never arrive at that node so that communication at node $\mathbf{v}_e$ for team $\mathcal{T}_m$ never occurs. This means that the robots in $\mathcal{T}_m \backslash \mathcal{W}_{\mathbf{v}_e}$ are waiting indefinitely at communication locations $\mathbf{v}_j \in \mathcal{C}_n$, $j \neq e$, $n \neq m$, $n \in \mathcal{N}_{\mathcal{T}_m}$, selected by Algorithm 2, to communicate with robots in team $\mathcal{T}_n$. The fact that there are robots that remain indefinitely at node $\mathbf{v}_j \in \mathcal{C}_n$ means that a communication within team $\mathcal{T}_n$ never occurs by construction of Algorithm 3. Following an argument similar to the above, we conclude that the robots in $\mathcal{T}_n \backslash \mathcal{W}_{\mathbf{v}_j}$ are waiting indefinitely at nodes $\mathbf{v}_{k\neq j} \in \mathcal{C}_f$ to communicate with robots that belong to a team $\mathcal{T}_f$, $f \in \mathcal{N}_{\mathcal{T}_n}$. Therefore, if a communication event never occurs for team $\mathcal{T}_m$, then all robots $i \in \mathcal{N}$ need to be waiting at communication locations selected by Algorithm 2 and, consequently, there is no communication location where all robots are present, i.e., there is no team within which communication will ever occur. Throughout the rest of the proof, we will refer to this network configuration as a *stationary configuration*.

In what follows, we show by contradiction that the network can never reach a stationary configuration when the paths in (6) are executed asynchronously as per Algorithm 3. To show this result, we first model the asynchronous execution of the schedules $\texttt{sched}_i$, constructed by Algorithm 1, as per Algorithm 3. Specifically, we introduce discrete time steps z_i that are initialized as $z_i = 1$ and are updated as $z_i = z_i + 1$ asynchronously across the robots as follows. If at the current discrete time step z_i robot i participates in the communication event $\texttt{sched}_i(z_i) = m$, for some $z_i \in \mathbb{N}_+$ and $m \in \mathcal{M}_i$, then robot $i \in \mathcal{T}_m$ waits until all the other robots in team $\mathcal{T}_m$ are available to communicate. Once all robots in $\mathcal{T}_m$ are available, the discrete time step z_i is updated as $z_i = z_i + 1$. If $\texttt{sched}_i(z_i) = X$, then robot i updates $z_i = z_i + 1$ without waiting.

Using this model to describe asynchronous execution of the schedules, we now show by contradiction that if the network gets trapped at a stationary configuration, then there exist robots of some team $\mathcal{T}_m$ that missed a communication event at node $\mathbf{v}_e$, $e \in \mathcal{C}_m$, at a previous time instant, which cannot happen by construction of Algorithm 3. Consider that there is an arbitrary time instant t_0 at which the network is at a stationary configuration and let the current communication event for all robots $i \in \mathcal{T}_m$ be $\texttt{sched}_i(n_i^{\mathcal{T}_m}(t_0)) = m$ for some $m \in \mathcal{M}_i$, where the indices $n_i^{\mathcal{T}_m}$ were defined in Algorithm 1. Define also the set $\mathcal{N}_{\text{min}}(t_0) = \left\{n_i^{\mathcal{T}_m}(t_0) | n_i^{\mathcal{T}_m}(t_0) = \min\{n_e^{\mathcal{T}_g}(t_0)\}_{e=1}^{N}, g \in \mathcal{M}_e\right\}$ that collects the smallest indices $n_i^{\mathcal{T}_m}(t_0)$ among all robots. Also, let $n_e^{\mathcal{T}_g}(t_0)$ be an index such that $n_e^{\mathcal{T}_g}(t_0) \in \mathcal{N}_{\text{min}}(t_0)$. By assumption, there are robots $e \in \mathcal{T}_g$ and $r \in \mathcal{T}_z$, $g \in \mathcal{N}_{\mathcal{T}_z}$, such that $e \in \mathcal{W}_{\mathbf{v}_f}(t_0)$, $\mathbf{v}_f \in \mathcal{T}_g$, and $r \in \mathcal{W}_{\mathbf{v}_d}(t_0)$, $\mathbf{v}_d \in \mathcal{T}_z$, and, therefore, the events that are taking place for these two robots according to their assigned schedules of meeting events are $\texttt{sched}_e(n_e^{\mathcal{T}_g}(t_0)) = g$ and $\texttt{sched}_r(n_r^{\mathcal{T}_z}(t_0)) = z$. Since $n_e^{\mathcal{T}_g}(t_0) \in \mathcal{N}_{\text{min}}(t_0)$, we have that $n_e^{\mathcal{T}_g}(t_0) \geq n_r^{\mathcal{T}_z}(t_0)$, which along with the fact that $g \in \mathcal{N}_{\mathcal{T}_z}$ results in $n_e^{\mathcal{T}_g}(t_0) > n_r^{\mathcal{T}_z}(t_0)$ by construction of Algorithm 1. This leads to the following contradiction. The fact that $n_e^{\mathcal{T}_g}(t_0) > n_r^{\mathcal{T}_z}(t_0)$ means that there exists a time instant $t < t_0$ at which the event that took place for robots $a \in \mathcal{T}_g \cap \mathcal{T}_z$ was $\texttt{sched}_a(n_r^{\mathcal{T}_g}(t)) = g$ and at least one of these robots did not wait for the arrival of all other robots in team $\mathcal{T}_g$, since at the current time instant t_0, there are still robots in team $\mathcal{T}_g$ waiting for the arrival of other robots. However, such a scenario is precluded by construction of Algorithm 3. Consequently, the asynchronous execution of the schedules $\texttt{sched}_i$ as per Algorithm 3 is deadlock free. Recall now that the paths (6) respect the schedules $\texttt{sched}_i$ and that it is not possible that there exist robots in any team $\mathcal{T}_m$ that wait for other robots in the same team at different communication points $\mathbf{v}_j$, $j \in \mathcal{C}_m$. Thus, we conclude that the network is deadlock free when the plans (6) are executed asynchronously, as per Algorithm 3. ■

Remark 6.3 (Bounded waiting times): Proposition 6.2 shows also that the waiting times introduced by Algorithm 3 are bounded.

In Theorems 6.4–6.5, we show that the assigned local tasks ϕ_i and the intermittent connectivity requirement captured by (1) are satisfied.

Theorem 6.4 (Task satisfaction): The asynchronous execution of the motion plans $\tau_i^{n_i}$ in (6) as per Algorithm 3 satisfies the LTL$_{-\bigcirc}$ statements ϕ_i, i.e., $\tau_i^{n_i} \models \phi_i$, for any $n_i \geq 0$ and all robots $i \in \mathcal{N}$.

Proof: First observe that Algorithm 2 can design feasible paths $\texttt{path}_i^{n_i}$, for any $n_i \geq 0$ as long as there exists a solution to Problem 1, due to Proposition 6.1. Moreover, the waiting times at the communication points in the plans $\tau_i^{n_i}$ are bounded by Proposition 6.2. Therefore, the infinite paths $\tau_i^{n_i}$ will be executed without any deadlocks. This is necessary to satisfy ϕ_i, as LTL formulas are satisfied by infinite sequences of states in wTS$_i$.

To prove this result, first we need to show that all transitions in wTS$_i$ that are generated by the plans in (6) respect the transition rule $\rightarrow_i$, see Definition 3.1. Next, we need to show that the infinite run ρ_{B_i} of the NBA B_i that corresponds to ϕ_i over the words $\sigma_i^{n_i}$ generated during the execution of $\tau_i^{n_i}$ is accepting, i.e.,[8]

$$\texttt{Inf}(\rho_{B_i}) \cap \mathcal{F}_{B_i} \neq \emptyset. \tag{12}$$

First, we show that all transitions in wTS$_i$ that are due to the plans in (6) respect the transition rule $\rightarrow_i$. Notice that all transitions incurred by the finite path $\texttt{path}_i^{n_i}$ respect the transition rule $\rightarrow_i$, for all $n_i \in \mathbb{N}$, by construction, see Algorithm 2. Next, we show that the transition from the last state in $\texttt{path}_i^{n_i}$ to the first state in $\texttt{path}_i^{n_i+1}$ also respects the transition rule $\rightarrow_i$, for all $n_i \in \mathbb{N}$. To show this, observe that the last state in $\texttt{path}_i^{n_i}$ is the last state in the suffix part $\tilde{\tau}_i^{\text{suf},j^*}$ used to construct $\texttt{path}_i^{n_i}$, for all $n_i \in \mathbb{N}$. Also, notice that the first state in $\texttt{path}_i^{n_i+1}$ is the state $\Pi|_{\text{wTS}_i} q_{P_i}^F$, for all $n_i \in \mathbb{N}$, which is also the first state in $\tilde{\tau}_i^{\text{suf},j^*}$. Therefore, by construction of $\tilde{\tau}_i^{\text{suf},j^*}$, the transition from the last state in $\texttt{path}_i^{n_i}$ to the first state in $\texttt{path}_i^{n_i+1}$ respects $\rightarrow_i$, for all $n_i \in \mathbb{N}$. Consequently, the plans in (6) respect $\rightarrow_i$.

Next, we show that (12) holds for the plans $\tau_i^{n_i}$ in (6), for all $n_i \geq 1$. The same logic also applies to the plans τ_i^0 in (11). To show this result, recall that the paths $\texttt{path}_i^{n_i}$, for all $n_i \geq 1$ are designed by 1) constructing a suffix path $\rho_{P_i}^{\text{suf},j^*}$ that lives in the state space $\mathcal{Q}_{P_i}$ around the fixed PBA final state $q_{P_i}^F$ defined in (9), and initializing $\texttt{path}_i^{n_i} = \Pi|_{\text{wTS}_i} \rho_{P_i}^{\text{suf},j^*}$, 2) appending the path $\Pi|_{\text{wTS}_i} \rho_{P_i}^{\text{suf},j^*}$ as many times as needed so that $\texttt{path}_i^{n_i}$ respects the schedule $\texttt{sched}_i$. Thus, $\texttt{path}_i^{n_i}$ can be written as the projection onto wTS$_i$ of the finite path $p_i^{n_i} = \rho_{P_i}^{\text{suf},j^*} | \rho_{P_i}^{\text{suf},j^*} | \ldots | \rho_{P_i}^{\text{suf},j^*}$, which means that $p_i^{n_i}$ visits the fixed PBA final state $q_{P_i}^F$ a finite number of times. Consequently, since the plans in (6) are defined as infinite sequences of paths $\texttt{path}_i^{n_i}$, we get that $q_{P_i}^F$ is visited infinitely often, i.e., (12) holds. ■

Theorem 6.5 (Intermittent communication): The asynchronous execution of the motion plans $\tau_i^{n_i}$ in (6) as per Algorithm 3, satisfies the intermittent communication requirement captured by the global LTL statement ϕ_{com}, for all $n_i \geq 0$.

Proof: By construction of the paths $\texttt{path}_i^{n_i}$, every robot i will communicate once with all teams $\mathcal{T}_m$, $m \in \mathcal{M}_i$, during a single execution of the path $\texttt{path}_i^{n_i}$. Moreover, by Proposition 6.2, there are no deadlocks during the execution of the plans $\tau_i^{n_i}$. Consequently, all robots i communicate infinitely often with all teams $\mathcal{T}_m$, $m \in \mathcal{M}_i$. ■

Combining the previous results, we can show that the proposed control scheme is complete.

Theorem 6.6 (Completeness): If there exists a solution to Problem 1, Algorithm 2 will find motion plans $\tau_i^{n_i}$ as in (6) that when executed asynchronously as per Algorithm 3, satisfy the local LTL$_{-\bigcirc}$ tasks ϕ_i and the global LTL intermittent connectivity requirement ϕ_{com}.

Proof: By Proposition 6.1, we get that if there exists a solution to Problem 1, then prefix–suffix motion plans as in (6) will be generated for any $n_i \geq 0$. Due to Theorems 6.4 and 6.5, the asynchronous execution of these plans as per Algorithm 3 satisfies the local LTL$_{-\bigcirc}$ tasks ϕ_i and the intermittent communication requirement captured by the global LTL statement ϕ_{com}. ■

B. Optimality

As discussed in Remark 5.6, execution of the plans in (6) is synchronized implicitly so that there exists a time instant t_n when all robots execute the path $\texttt{path}_i^n$. In the following proposition, we examine the optimality of the paths $\texttt{path}_i^n$ in terms of the total cost $\sum_{i \in \mathcal{N}} J(\texttt{path}_i^n)$, for any $n \in \mathbb{N}$.

Proposition 6.7 (Optimality): Algorithm 2 generates discrete paths $\texttt{path}_i^{n+1}$ so that

$$\sum_{i \in \mathcal{N}} J(\texttt{path}_i^n) \leq \sum_{i \in \mathcal{N}} J(\texttt{path}_i^{n+1}) \tag{13}$$

for all $n \geq 0$.

Proof: Consider the discrete paths $\texttt{path}_i^n$, for some fixed $n \geq 0$. Recall that the robots may start executing the paths $\texttt{path}_i^n$ asynchronously, i.e., at different time instants. Therefore, given a time instant t, we divide the robots $i \in \mathcal{N}$ in the following five disjoint sets. First, we collect in the set $\mathcal{R}^{n-1}(t)$ the robots that execute the paths $\texttt{path}_i^{n-1}$ at a time t. Next, we collect in the set $\mathcal{R}_{\text{new}}^n(t)$ the robots that are new to executing the path $\texttt{path}_i^n$ and have not participated in any communication event contained in $\texttt{path}_i^n$ yet. Notice that the robots in $\mathcal{R}^{n-1}(t)$ and $\mathcal{R}_{\text{new}}^n(t)$ have not constructed yet any path $\texttt{path}_i^{n+1,c_i}$. Also, we collect in the set $\mathcal{R}_{\text{com}}^n(t)$ the robots of all teams $\mathcal{T}_m$, $m \in \mathcal{M}$, that communicate at time t while executing the paths $\texttt{path}_i^n$. All other robots that at time t execute the path $\texttt{path}_i^n$ but they do not participate in any communication event are collected in the set $\mathcal{R}_{\overline{\text{com}}}^n(t)$. Finally, the robots that have already finished the execution of the paths $\texttt{path}_i^n$ at time t are collected in the set $\mathcal{R}^{n+1}(t)$. Observe that $\mathcal{N} = \mathcal{R}^{n-1}(t) \cup \mathcal{R}_{\text{new}}^n(t) \cup \mathcal{R}_{\text{com}}^n(t) \cup \mathcal{R}_{\overline{\text{com}}}^n(t) \cup \mathcal{R}^{n+1}(t)$, for all $t \geq 0$, for some $n \geq 0$. Also, observe that if $\mathcal{R}^{n+1}(t) \neq \emptyset$, then $\mathcal{R}^{n-1}(t) = \emptyset$, as discussed in Remark 5.6.

[8]The generated word $\sigma_i^{n_i}$, also called trace of τ_i [35] and denoted by $\texttt{trace}(\tau_i)$, is defined as $\sigma_i^{n_i} = \texttt{trace}(\tau_i^{n_i}) := L_i(\tau_i^{n_i}(1))L_i(\tau_i^{n_i}(2))\ldots$, where L_i is the labeling function defined in Definition 3.1.

To prove the inequality (13), we need to define the following cost function:

$$\begin{aligned}\texttt{cost}(t) = &\sum_{i\in\mathcal{R}^n_{\text{new}}(t)\cup\mathcal{R}^{n-1}(t)} J(\texttt{path}_i^n)\\ &+\sum_{i\in\mathcal{R}^n_{\text{com}}(t)} J(\texttt{path}_i^{n+1,c_i(t)})\\ &+\sum_{i\in\mathcal{R}^n_{\overline{\text{com}}}(t)} J(\texttt{path}_i^{n+1,c_i(t)})+\sum_{i\in\mathcal{R}^{n+1}(t)} J(\texttt{path}_i^{n+1})\end{aligned} \tag{14}$$

where $\texttt{path}_i^{n+1,c_i(t)}$ denotes the path that has been constructed by Algorithm 2 by the time instant t. Also, note that the robots $i\in\mathcal{R}^{n-1}(t)$ may not have completed the construction of the paths $\texttt{path}_i^n$ yet. Therefore, in the first summation in (14), the paths $\texttt{path}_i^n$ for $i\in\mathcal{R}^{n-1}(t)$, are the ones that these robots will create once they complete their construction.

Moreover, we define the finite sequence of time instants $\{t_0^n,t_1^n,\ldots,t_{F-1}^n,t_F^n\}$, where $t_0^n<\cdots<t_F^n$, t_0^n is an arbitrarily selected time instant such that $\mathcal{R}^n_{\text{new}}(t)\cup\mathcal{R}^{n-1}(t)=\mathcal{N}$, t_F^n is the time instant when all robots have completed construction of the paths $\texttt{path}_i^{n+1}$, i.e., $\mathcal{R}^{n+1}(t_F^n)=\mathcal{N}$, and $t_1^n<\cdots<t_{F-1}^n$ are the time instants corresponding to communication events during the execution of any of the paths $\texttt{path}_i^n$.[9] To prove (13), we need to show that

$$\texttt{cost}(t_{k+1}^n)\leq\texttt{cost}(t_k^n) \tag{15}$$

for all $k\in\{0,\ldots,F\}$.

Since the robots $i\in\mathcal{R}^n_{\text{new}}(t_{k+1}^n)\cup\mathcal{R}^{n-1}(t_{k+1}^n)$ have not constructed yet any path $\texttt{path}_i^{n_i+1,c_i}$, these robots cannot affect the cost $\texttt{cost}(t_k^n)$. Also, notice that $\texttt{path}_i^{n+1,c_i(t_{k+1}^n)}=\texttt{path}_i^{n+1,c_i(t_k^n)}$, for all robots $i\in\mathcal{R}^n_{\overline{\text{com}}}(t_{k+1}^n)$, since these robots do not communicate and, therefore, they do not execute Algorithm 2 at t_{k+1}^n. Thus, the robots $i\in\mathcal{R}^n_{\overline{\text{com}}}(t_{k+1}^n)$ cannot affect the cost $\texttt{cost}(t_k^n)$ either. The same holds for the robots $i\in\mathcal{R}^{n+1}(t_{k+1}^n)$. Therefore, for all robots that do not communicate at time t_{k+1}^n, it holds that $\sum_{i\in\mathcal{N}\setminus\mathcal{R}^n_{\text{com}}(t_{k+1}^n)}J(\texttt{path}_i^{n+1,c_i(t_{k+1}^n)})=\sum_{i\in\mathcal{N}\setminus\mathcal{R}^n_{\text{com}}(t_{k+1}^n)}J(\texttt{path}_i^{n+1,c_i(t_k^n)})$. In fact, only the robots $i\in\mathcal{R}^n_{\text{com}}(t_{k+1}^n)$ that communicate at time t_{k+1}^n design new paths such that $\texttt{path}_i^{n+1,c_i(t_{k+1}^n)}\neq\texttt{path}_i^{n+1,c_i(t_k^n)}$. Since $\mathcal{R}^n_{\text{com}}(t_{k+1}^n)$ contains all robots that communicate at t_{k+1}^n, the expression $\sum_{i\in\mathcal{R}^n_{\text{com}}(t_{k+1}^n)}J(\texttt{path}_i^{n+1,c_i(t_{k+1}^n)})$ can be rewritten as follows:[10]

$$\begin{aligned}&\sum_{i\in\mathcal{R}^n_{\text{com}}(t_{k+1}^n)}J(\texttt{path}_i^{n+1,c_i(t_{k+1}^n)})\\ &=\sum_{m\in\mathcal{A}(t_{k+1}^n)}\sum_{i\in\mathcal{T}_m}J(\texttt{path}_i^{n+1,c_i(t_{k+1}^n)})\end{aligned} \tag{16}$$

where $\mathcal{A}(t)\subseteq\mathcal{M}$ is the set of the teams that communicate at time t. By the proof of Proposition 6.1, we get that $\texttt{path}_i^{n+1,c_i(t_k^n)}$ is a feasible path returned by Algorithm 2 as a candidate path for $\texttt{path}_i^{n+1,c_i(t_{k+1}^n)}$; it will become $\texttt{path}_i^{n+1,c_i(t_{k+1}^n)}$ if it also the optimal one.

Therefore, we get that $\sum_{i\in\mathcal{T}_m}J(\texttt{path}_i^{n+1,c_i(t_{k+1}^n)})\leq\sum_{i\in\mathcal{T}_m}J(\texttt{path}_i^{n+1,c_i(t_k^n)})$, for all $m\in\mathcal{A}(t_{k+1}^n)$, which implies $\sum_{i\in\mathcal{R}^n_{\text{com}}(t_{k+1}^n)}J(\texttt{path}_i^{n+1,c_i(t_{k+1}^n)})\leq\sum_{i\in\mathcal{R}^n_{\text{com}}(t_{k+1}^n)}J(\texttt{path}_i^{n+1,c_i(t_k^n)})$, due to (16). Therefore, we get that (15) holds. ■

C. Complexity

In the following proposition, we show that Algorithm 2 terminates after a finite number of iterations and, therefore, the computational cost is bounded.

Proposition 6.8 (Convergence): There exist iterations $P\leq C$ in Algorithm 2 so that the sequence $\texttt{path}_i^P,\texttt{path}_i^{P+1},\ldots,\texttt{path}_i^C$ is repeated indefinitely for all $n_i\geq C$ and all $i\in\mathcal{N}$.

Proof: To show this result, notice that the sets of communication points $\mathcal{C}_m$ are finite, for all $m\in\mathcal{M}$ and, therefore, the number of possible combinations of communication points that can be assigned to the teams is finite. Therefore, there exists an index n where the paths $\texttt{path}_i^n$ contain communication points that have appeared in a previous path $n'\leq n$, as well, for all $i\in\mathcal{N}$. Let C be the first index n when it holds that the communication points that appear in the paths $\texttt{path}_i^C$ have already appeared in a previous path $\texttt{path}_i^{P-1}$, for some $P\leq C$ and for all $i\in\mathcal{N}$. Since the selected communication points in the paths $\texttt{path}_i^{P-1}$ and $\texttt{path}_i^C$ are the same, we have that Algorithm 2 generates the same optimal suffix path $\tilde{\tau}_i^{\text{suf},j^*}$ to synthesize both $\texttt{path}_i^{P-1}$ and $\texttt{path}_i^C$. Therefore, we get that $\texttt{path}_i^{P-1}=\texttt{path}_i^C$. Consequently, the path $\texttt{path}_i^{C+1}$ will be the same as the path constructed at iteration P, i.e., $\texttt{path}_i^{C+1}=\texttt{path}_i^P$, since the optimal control synthesis problems that are solved to construct the path $\texttt{path}_i^{C+1}$ and $\texttt{path}_i^P$ are the same, for all robots $i\in\mathcal{N}$. Similarly, we have that $\texttt{path}_i^{C+2}=\texttt{path}_i^{P+1}$. By inspection of the repetitive pattern, we conclude that for any $n\in\mathbb{N}$, it holds that $\texttt{path}_i^{C+n}=\texttt{path}_i^{C+n-(\lfloor(C+n)/(C-P+1)\rfloor-1)(C-P+1)}$, where $\lfloor\cdot\rfloor$ stands

[9]Note that the time instant t_0^n exists, since it corresponds to a time when the robots either execute paths $\texttt{path}_i^{n_i-1}$ or paths $\texttt{path}_i^{n_i}$ without having participated in any communication events yet, see also Remark 5.6. Also, the sequence $\{t_1^n,\ldots,t_{F-1}^n,t_F^n\}$ for any $n\geq0$ exists because the network is deadlock free, as shown in Proposition 6.2.

[10]Note that it is possible that two teams $\mathcal{T}_m$ and $\mathcal{T}_h$ that share at least a robot may be present simultaneously at the same communication point. This can happen, e.g., if the schedule of robot $i\in\mathcal{T}_m\cap\mathcal{T}_h$ has a schedule has the form $\texttt{sched}_i=[m,h,X]^\omega$ and $\mathcal{C}_m\cap\mathcal{C}_h\neq\emptyset$. In this case, we assume that communication at the common communication point will happen sequentially across the teams according to the schedules. This ensures that in the second summation in (16), we never double count the cost of the paths $\texttt{path}_i^{n+1,c_i(t)}$.

for the floor function. We conclude that the sequence $\texttt{path}_i^P, \texttt{path}_i^{P+1}, \ldots, \texttt{path}_i^C$ is repeated indefinitely for all iterations $n_i \geq C$ of Algorithm 2 and for all robots $i \in \mathcal{N}$ completing the proof. ■

Remark 6.9 (Optimality of Algorithm 2): Notice that Propositions 6.7–6.8 do not guarantee that Algorithm 2 will find the optimal prefix–suffix plan that minimizes the cost $J_p(\tau_i) = \alpha \sum_{i\in\mathcal{N}} J(\tau_i^{\text{pre}}) + (1-\alpha)\sum_{i\in\mathcal{N}} J(\tau_i^{\text{suf}})$. Instead they only ensure that the total cost $\sum_{i\in\mathcal{N}} J(\texttt{path}_i^n)$ decreases with every iteration n until $n = P$, when $\sum_{i\in\mathcal{N}} J(\texttt{path}_i^P) = \sum_{i\in\mathcal{N}} J(\texttt{path}_i^{P+1}) = \cdots = \sum_{i\in\mathcal{N}} J(\texttt{path}_i^C)$, while for all iterations $n_i \geq C$ the sequence of paths $\texttt{path}_i^P, \texttt{path}_i^{P+1}, \ldots, \texttt{path}_i^C$ is repeated indefinitely. Therefore, the best plans $\tau_i^{n_i}$ (6) are obtained for any $n_i \geq P$, for all robots $i \in \mathcal{N}$. Suboptimality is due to the decomposition of Problem 1 into intermittent communication control (Section IV) and task planning (Section V) that are solved independently. The optimal plan can be found by translating the global LTL formula (2) into an NBA, constructing a product automaton across all robots in the network as, e.g., in [20], [21], and using graph search methods to find the optimal plan. However, such centralized methods are computationally expensive and resource demanding as it is also discussed in the Introduction. Moreover, recall that in this paper we assume that the teams $\mathcal{T}_m$ are fixed and never change. Note that the total cost of the plans $\tau_i^{n_i}$ can be further minimized if the robots in every team $\mathcal{T}_m$ update not only the communication point $\mathbf{v}_j$, $j \in \mathcal{C}_m$, but also the teams they belong to. Optimal design of the teams is part of our future work.

VII. Simulation Studies

In this section, we present a simulation study, implemented using MATLAB R2015b on a computer with Intel Core i7 2.2 GHz and 4 Gb RAM that illustrates our approach for a network of $N = 12$ robots. Robots are categorized into $M = 12$ teams as follows: $\mathcal{T}_1 = \{1,2,9\}$, $\mathcal{T}_2 = \{3,4,5\}$, $\mathcal{T}_3 = \{3,6\}$, $\mathcal{T}_4 = \{1,3\}$, $\mathcal{T}_5 = \{2,5,6,11\}$, $\mathcal{T}_6 = \{4,12\}$, $\mathcal{T}_7 = \{5,9\}$, $\mathcal{T}_8 = \{4,9,12\}$, $\mathcal{T}_9 = \{6,7,10\}$, $\mathcal{T}_{10} = \{7,8,11\}$, $\mathcal{T}_{11} = \{8,10,11,12\}$, and $\mathcal{T}_{12} = \{7,10\}$. Notice that the construction of teams $\mathcal{T}_m$ results in a connected graph $\mathcal{G}_\mathcal{T}$ with $\max\{d_{\mathcal{T}_m}\}_{m=1}^M = 7$, as discussed in Section III. Mobility of each robot in the workspace is captured by a wTS with $|\mathcal{Q}_i| = 300$ states that represent $W = 300$ locations of interest and weights w_i that capture the distance between its states. Among the $W = 300$ locations of interest, $R = 70$ locations correspond to possible communication points. Also, every team has $4 \leq |\mathcal{C}_m| \leq 6$ communication points, while $\mathcal{C}_m \cap \mathcal{C}_n = \varnothing$, for all $m, n \in \mathcal{M}$. Also, the parameter α in (3) is selected as $\alpha = 0.5$. To model uncertainty in robot mobility, caused by exogenous disturbances that may affect the arrival times of the robots at the communication locations, we assume that the time required for robot i to travel from location $\mathbf{v}_e$ to $\mathbf{v}_j$, with $(q_i^{\mathbf{v}_e}, q_i^{\mathbf{v}_j}) \in \rightarrow_i$, is generated by a uniform distribution on [1,2], at the moment when robot i arrives at location $\mathbf{v}_e$.

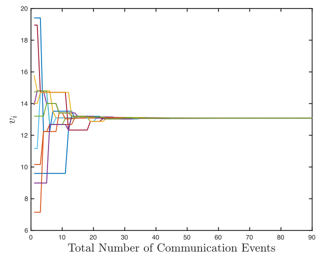
The LTL$_{-\bigcirc}$ tasks for robots 1 and 3 are $\phi_1 = \square\lozenge(\pi_1^{\mathbf{v}_{20}} \vee \pi_1^{\mathbf{v}_{10}} \vee \pi_1^{\mathbf{v}_{11}}) \wedge \square\lozenge(\pi_1^{\mathbf{v}_{61}}) \wedge \square\lozenge(\pi_1^{\mathbf{v}_{91}} \vee \pi_1^{\mathbf{v}_{100}} \vee \pi_1^{\mathbf{v}_{5}} \vee \pi_1^{\mathbf{v}_{60}}) \wedge \square(\neg\pi_1^{\mathbf{v}_{44}}) \wedge \lozenge(\pi_1^{\mathbf{v}_{6}} \vee \pi_1^{\mathbf{v}_{7}} \vee \pi_1^{\mathbf{v}_{133}})$ and $\phi_3 = \square\lozenge(\xi_3^1 \vee \xi_3^2) \wedge [\square\lozenge(\xi_3^3)] \wedge \lozenge[\xi_3^2 \rightarrow \square(\neg\xi_3^1)] \wedge (\neg\xi_3^3 \mathcal{U} \xi_3^1)$, respectively, where $\xi_3^1 = \pi_3^{\mathbf{v}_{81}} \vee \pi_3^{\mathbf{v}_{91}}$, $\xi_3^2 = \pi_3^{\mathbf{v}_{120}} \vee \pi_3^{\mathbf{v}_{91}} \vee \pi_3^{\mathbf{v}_{31}}$, and $\xi_3^3 = \pi_3^{\mathbf{v}_{91}} \vee \pi_3^{\mathbf{v}_{110}} \vee \pi_3^{\mathbf{v}_{15}} \vee \pi_3^{\mathbf{v}_{130}}$. All other robots are responsible for similar LTL tasks. For instance, the LTL formula in ϕ_3 requires robot 3 to satisfy infinitely often either the Boolean formula ξ_3^1 or ξ_3^2, satisfy infinitely often the Boolean formula ξ_3^3, never satisfy ξ_3^1 if ξ_3^2 is ever satisfied, and never satisfy ξ_3^3 until ξ_3^1 is satisfied. The Boolean formula ξ_3^1 is satisfied if robot 3 visits either $\mathbf{v}_{81}$ or $\mathbf{v}_{91}$. The Boolean formulas ξ_3^2 and ξ_3^3 are interpreted similarly. Also, note that robot 1 is responsible for visiting a user located at $\mathbf{v}_{61}$ infinitely often to transmit all collected information.

The schedules of communication events constructed as per Algorithm 1 have the following form with length $\ell = 4 \leq \max\{d_{\mathcal{T}_m}\}_{m=1}^{12} + 1 = 8$:

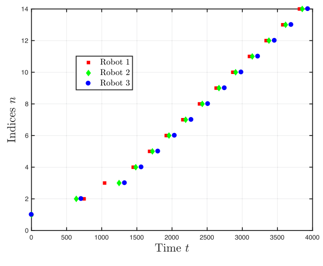
$$\begin{aligned}
&\texttt{sched}_1 = [1,\ 4,\ X,\ X]^\omega, \ \texttt{sched}_7 = [9,\ 12,\ 10,\ X]^\omega\\
&\texttt{sched}_2 = [1,\ 5,\ X,\ X]^\omega, \ \texttt{sched}_8 = [X,\ X,\ 10,\ 11]^\omega\\
&\texttt{sched}_3 = [2,\ 4,\ 3,\ X]^\omega, \ \texttt{sched}_9 = [1,\ X,\ 8,\ 7]^\omega\\
&\texttt{sched}_4 = [2,\ 6,\ 8,\ X]^\omega, \ \texttt{sched}_{10} = [9,\ 12,\ X,\ 11]^\omega\\
&\texttt{sched}_5 = [2,\ 5,\ X,\ 7]^\omega, \ \texttt{sched}_{11} = [X,\ 5,\ 10,\ 11]^\omega\\
&\texttt{sched}_6 = [9,\ 5,\ 3,\ X]^\omega, \ \texttt{sched}_{12} = [X,\ 6,\ 8,\ 11]^\omega.
\end{aligned}$$

Then, given the above schedules, feasible initial paths $\texttt{path}_i^0$ are constructed for all robots in 3 s approximately using [22]. Specifically, given communication points for all teams $\mathcal{T}_m$, $m \in \mathcal{M}_i$, [22] can synthesize a feasible plan $\tilde{\tau}_i^0$ that satisfies ψ_i in 0.35 s on average for all $i \in \mathcal{N}$. Similar runtimes are reported if off-the-shelf model checkers, such as NuSMV [39], are employed for initialization. Moreover, Algorithm 2 constructs online paths $\texttt{path}_i^{n_i}$ with $P = C = 5$. The size of the NBA B_i that corresponds to ψ_i in (7) satisfies $7 \leq |\mathcal{Q}_{B_i}| \leq 16$, for all $i \in \mathcal{N}$, while the average runtime to solve a single optimal control synthesis problem to generate the optimal suffix path $\tilde{\tau}_i^{\text{suf},j}$ was 45 s. Since $4 \leq |\mathcal{C}_m| \leq 6$, for all $m \in \mathcal{C}_m$, the average runtime of Algorithm 2 per iteration c_i is between $4 \times 45 = 180$ s and $6 \times 45 = 270$ s. Note that this runtime depends only on the size of the sets $\mathcal{C}_m$ and not on the size of the teams $\mathcal{T}_m$. Note also that this runtime is higher than the initialization runtime, since during initialization only feasible plans are required, while for the online construction of $\texttt{path}_i^{n_i}$ optimal suffix paths are created. More computational efficient methods are discussed in Remark 5.7 that can decrease the corresponding runtime.

To illustrate that the designed motion plans ensure intermittent communication among the robots infinitely often, we implement a consensus algorithm over the dynamic network $\mathcal{G}_c$. Specifically, we assume that initially all robots generate a random number $v_i(t_0)$ and when all robots $i \in \mathcal{T}_m$ meet at a communication point $j \in \mathcal{C}_m$ they perform the following consensus update $v_i(t) = \frac{1}{|\mathcal{T}_m|}\sum_{e\in\mathcal{T}_m} v_e(t)$. Fig. 3(a) shows that eventually all robots reach a consensus on the numbers $v_i(t)$, which means that communication among robots takes place infinitely often, as proven in Theorem 6.5. Moreover, Fig. 3(b) shows the time instants when robots 1, 2, and 3 started executing the paths $\texttt{path}_i^n$, for all $n \in \{1, \ldots, 14\}$. Observe in Fig. 3(b) that there exist time instants t_n when all three robots are executing their

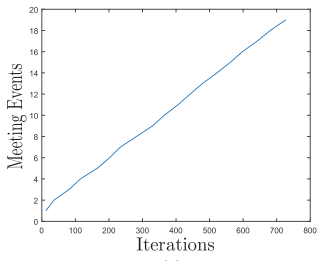


(a)

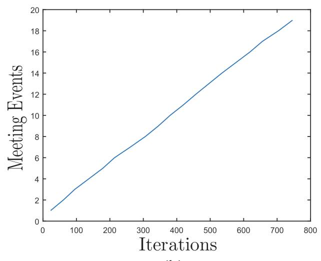


(b)

Fig. 3. (a) Consensus of numbers $v_i(t)$. (b) Time instants when the robots 1, 2, and 3 started executing the paths $\texttt{path}_i^n$. For instance, the time between the second and the third red square denotes the time required by robot 1 to travel along the path $\texttt{path}_1^2$.



(a)



(b)

Fig. 4. Graphical depiction of communication events for team $\mathcal{T}_1$ (a) and $\mathcal{T}_4$ (b) with respect to time. (a) Team 1, (b) Team 4.

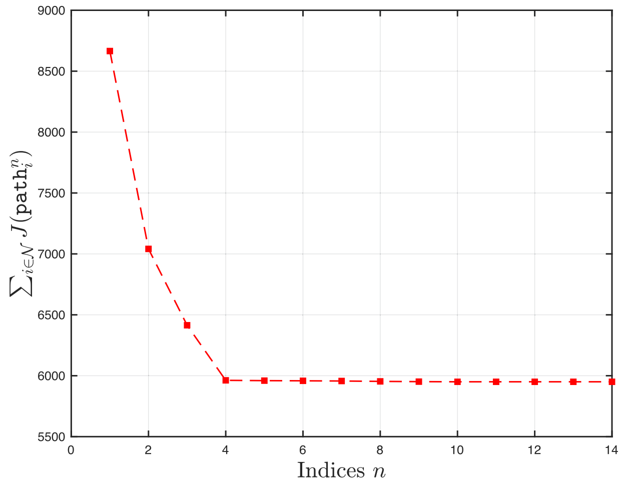


Fig. 5. Evolution of the total cost $\sum_{i=1}^{12} J(\texttt{path}_i^n)$ with respect to iterations n. Note that there is also a slight decrease in $\sum_{i=1}^{12} J(\texttt{path}_i^n)$ from $n = 4$ to $n = 5$. After $n = 5$, a repetitive pattern in $\texttt{path}_i^n$ is detected giving rise to motion plans τ_i in a prefix–suffix form.

respective paths $\texttt{path}_i^n$ for a common n, for all $n \in \{1, \ldots, 14\}$, as discussed in Remark 5.6. The communication events over time for teams $\mathcal{T}_1$ and $\mathcal{T}_5$ are depicted in Fig. 4. Observe in Fig. 4 that the communication time instances do not depend linearly on time, which means that communication within these teams is aperiodic. Fig. 5 shows that the total traveled distance $\sum_{i=1}^{N} J(\texttt{path}_i^n)$ with respect to $n \in \mathbb{N}$ which decreases as expected due to Proposition 6.7. The corresponding simulation video can be found in [40].

Note also that due to excessive memory requirements it would be impossible to generate optimal motion plans τ_i by using either the optimal control synthesis methods presented in [20], [21], [29] that rely on the construction of a synchronous product automaton or off-the-shelf model checkers [39], [41] that can construct feasible but not optimal paths. Specifically, [20], [21], [29] rely on the construction of a product transition system (PTS), whose state space has dimension $|\mathcal{Q}_{\text{PTS}}| = \times_{\forall i}|\mathcal{Q}_i| = W^{|N|} = 300^{12} = 5.3144 \times 10^{29}$. This PTS is combined with the Bü chi Automaton B that corresponds to the LTL statement $\phi = (\wedge_{\forall i \in \mathcal{N}} \phi_i) \wedge \phi_{\text{com}}$ to construct a Product Bü chi Automaton whose state space has dimension $|\mathcal{Q}_{\text{PBA}}| = |\mathcal{Q}_{\text{PTS}}| \times |\mathcal{Q}_B| = 5.3144 \times 10^{29} \times |\mathcal{Q}_B|$ which is too large to manipulate in practice let alone searching for an optimal accepting infinite run. Finally, we validated the efficacy of the proposed distributed algorithm by experimental results that are omitted due to space limitations. The video showing the conducted experiment along with its description can be found in [42].

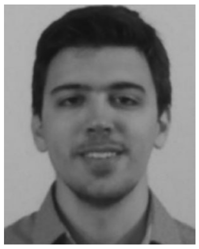
VIII. Conclusion

In this paper, we developed the first distributed and online intermittent communication framework for networks of mobile robots with limited communication capabilities that are responsible for accomplishing temporal logic tasks. Our proposed distributed online control framework jointly determines local plans that allow all robots to fulfill their assigned LTL$_{-\bigcirc}$ tasks, schedules of communication events that guarantee information exchange infinitely often, and optimal communication locations that minimize a desired distance metric. We showed that the proposed method can solve optimally very large-scale problems that are impossible to solve using current off-the-shelf model checkers.

References

[1] M. M. Zavlanos and G. J. Pappas, "Potential fields for maintaining connectivity of mobile networks," *IEEE Trans. Robot.*, vol. 23, no. 4, pp. 812–816, Aug. 2007.
[2] M. Ji and M. B. Egerstedt, "Distributed coordination control of multi-agent systems while preserving connectedness," *IEEE Trans. Robot.*, vol. 23, no. 4, pp. 693–703, Aug. 2007.
[3] M. M. Zavlanos and G. J. Pappas, "Distributed connectivity control of mobile networks," *IEEE Trans. Robot.*, vol. 24, no. 6, pp. 1416–1428, Dec. 2008.
[4] L. Sabattini, N. Chopra, and C. Secchi, "Decentralized connectivity maintenance for cooperative control of mobile robotic systems," *Int. J. Robot. Res.*, vol. 32, no. 12, pp. 1411–1423, 2013.
[5] M. M. Zavlanos, M. B. Egerstedt, and G. J. Pappas, "Graph theoretic connectivity control of mobile robot networks," *Proc. IEEE*, vol. 99, no. 9, pp. 1525–1540, Sep. 2011.
[6] M. M. Zavlanos, A. Ribeiro, and G. J. Pappas, "Network integrity in mobile robotic networks," *IEEE Trans. Autom. Control*, vol. 58, no. 1, pp. 3–18, Jan. 2013.
[7] Y. Yan and Y. Mostofi, "Robotic router formation in realistic communication environments," *IEEE Trans. Robot.*, vol. 28, no. 4, pp. 810–827, Aug. 2012.
[8] Y. Kantaros and M. M. Zavlanos, "Distributed communication-aware coverage control by mobile sensor networks," *Automatica*, vol. 63, pp. 209–220, 2016.
[9] Y. Kantaros and M. M. Zavlanos, "Global planning for multi-robot communication networks in complex environments," *IEEE Trans. Robot.*, vol. 32, no. 5, pp. 1045–1061, Oct. 2016.
[10] J. Stephan, J. Fink, V. Kumar, and A. Ribeiro, "Concurrent control of mobility and communication in multirobot systems," *IEEE Trans. Robot.*, vol. 33, no. 5, pp. 1248–1254, Oct. 2017.

[11] G. Wen, Z. Duan, W. Ren, and G. Chen, "Distributed consensus of multi-agent systems with general linear node dynamics and intermittent communications," *Int. J. Robust Nonlinear Control*, vol. 24, no. 16, pp. 2438–2457, 2014.
[12] Y. Wang and I. I. Hussein, "Awareness coverage control over large-scale domains with intermittent communications," *IEEE Trans. Autom. Control*, vol. 55, no. 8, pp. 1850–1859, Aug. 2010.
[13] A. Lindgren, A. Doria, and O. Schelén, "Probabilistic routing in intermittently connected networks," *ACM SIGMOBILE Mobile Comput. Commun. Rev.*, vol. 7, no. 3, pp. 19–20, 2003.
[14] E. P. Jones, L. Li, J. K. Schmidtke, and P. A. Ward, "Practical routing in delay-tolerant networks," *IEEE Trans. Mobile Comput.*, vol. 6, no. 8, pp. 943–959, Aug. 2007.
[15] D. V. Dimarogonas, E. Frazzoli, and K. H. Johansson, "Distributed event-triggered control for multi-agent systems," *IEEE Trans. Autom. Control*, vol. 57, no. 5, pp. 1291–1297, May 2012.
[16] P. Tabuada, "Event-triggered real-time scheduling of stabilizing control tasks," *IEEE Trans. Autom. Control*, vol. 52, no. 9, pp. 1680–1685, Sep. 2007.
[17] M. Kloetzer and C. Belta, "Distributed implementations of global temporal logic motion specifications," in *Proc. IEEE Int. Conf. Robot. Automat.*, Pasadena, CA, USA, May 2008, pp. 393–398.
[18] Y. Chen, X. C. Ding, and C. Belta, "Synthesis of distributed control and communication schemes from global LTL specifications," in *Proc. 50th IEEE Conf. Decis. Control Eur. Control Con.*, Orlando, FL, USA, Dec. 2011, pp. 2718–2723.
[19] Y. Chen, X. C. Ding, A. Stefanescu, and C. Belta, "Formal approach to the deployment of distributed robotic teams," *IEEE Trans. Robot.*, vol. 28, no. 1, pp. 158–171, Feb. 2012.
[20] A. Ulusoy, S. L. Smith, X. C. Ding, C. Belta, and D. Rus, "Optimality and robustness in multi-robot path planning with temporal logic constraints," *Int. J. Robot. Res.*, vol. 32, no. 8, pp. 889–911, 2013.
[21] A. Ulusoy, S. L. Smith, and C. Belta, "Optimal multi-robot path planning with LTL constraints: Guaranteeing correctness through synchronization," in *Distributed Autonomous Robotic Systems*. Berlin, Germany: Springer, 2014, pp. 337–351.
[22] Y. Kantaros and M. M. Zavlanos, "Sampling-based optimal control synthesis for multi-robot systems under global temporal tasks," *IEEE Trans. Autom. Control*, to be published, doi:10.1109/TAC.2018.2853558.
[23] Y. Kantaros and M. M. Zavlanos, "Distributed optimal control synthesis for multi-robot systems under global temporal tasks," in *Proc. 9th ACM/IEEE Int. Conf. Cyber-Phys. Syst.*, Porto, Portugal, Apr. 2018, pp. 162–173.
[24] Y. E. Sahin, P. Nilsson, and N. Ozay, "Provably-correct coordination of large collections of agents with counting temporal logic constraints," in *Proc. 8th Int. Conf. Cyber-Phys. Syst.*, Pittsburgh, PA, USA, 2017, pp. 249–258.
[25] M. Kloetzer and C. Belta, "Automatic deployment of distributed teams of robots from temporal logic motion specifications," *IEEE Trans. Robot.*, vol. 26, no. 1, pp. 48–61, Feb. 2010.
[26] G. Pola, P. Pepe, and M. D. Di Benedetto, "Decentralized supervisory control of networks of nonlinear control systems," *IEEE Trans. Autom. Control*, vol. 63, no. 9, pp. 2803–2817, Jan. 2018
[27] Y. Kantaros and M. M. Zavlanos, "Distributed intermittent connectivity control of mobile robot networks," *Trans. Autom. Control*, vol. 62, no. 7, pp. 3109–3121, Jul. 2017.
[28] Y. Kantaros and M. M. Zavlanos, "Simultaneous intermittent communication control and path optimization in networks of mobile robots," in *Proc. IEEE Conf. Decis. Control*, Las Vegas, NV, USA, Dec. 2016, pp. 1794–1795.
[29] Y. Kantaros and M. M. Zavlanos, "Intermittent connectivity control in mobile robot networks," in *Proc. 49th Asilomar Conf. Signals, Syst. Comput.*, Pacific Grove, CA, USA, Nov. 2015, pp. 1125–1129.
[30] M. Guo and M. M. Zavlanos, "Distributed data gathering with buffer constraints and intermittent communication," in *Proc. IEEE Int. Conf. Robot. Automat.*, Singapore, May/Jun. 2017, pp. 279–284.
[31] M. M. Zavlanos, "Synchronous rendezvous of very-low-range wireless agents," in *Proc. 49th IEEE Conf. Decis. Control*, Atlanta, GA, USA, Dec. 2010, pp. 4740–4745.
[32] G. Hollinger and S. Singh, "Multi-robot coordination with periodic connectivity," in *Proc. IEEE Int. Conf. Robot. Automat.*, Anchorage, AK, USA, May 2010, pp. 4457–4462.
[33] C. Baier and J.-P. Katoen, *Principles of Model Checking*, vol. 26202649. Cambridge, MA, USA: MIT Press, 2008.
[34] M. Y. Vardi and P. Wolper, "An automata-theoretic approach to automatic program verification," in *Proc. 1st Symp. Logic Comput. Sci.*, 1986, pp. 322–331.
[35] E. M. Clarke, O. Grumberg, and D. Peled, *Model Checking*. Cambridge, MA, USA: MIT Press, 1999.
[36] M. Kloetzer and C. Belta, "A fully automated framework for control of linear systems from temporal logic specifications," *IEEE Trans. Autom. Control*, vol. 53, no. 1, pp. 287–297, Feb. 2008.
[37] S. L. Smith, J. Tumova, C. Belta, and D. Rus, "Optimal path planning for surveillance with temporal-logic constraints," *Int. J. Robot. Res.*, vol. 30, no. 14, pp. 1695–1708, 2011.
[38] M. Guo and D. V. Dimarogonas, "Multi-agent plan reconfiguration under local LTL specifications," *Int. J. Robot. Res.*, vol. 34, no. 2, pp. 218–235, 2015.
[39] A. Cimatti *et al.*, "Nusmv 2: An opensource tool for symbolic model checking," in *Proc. Int. Conf. Comput. Aided Verification*, 2002, pp. 359–364.
[40] SimulationVideo, 2018. [Online]. Available: https://vimeo.com/274366915.
[41] G. J. Holzmann, *The SPIN Model Checker: Primer And Reference Manual*, vol. 1003. Reading, MA, USA: Addison-Wesley, 2004.
[42] ExperimentVideo, 2017. [Online]. Available: https://vimeo.com/239508876.



Yiannis Kantaros (S'14–M'19) received the Diploma in electrical and computer engineering from the University of Patras, Patras, Greece, in 2012, and the M.Sc. and Ph.D. degrees in mechanical engineering from Duke University, Durham, NC, USA, in 2017 and 2018, respectively.

He is currently a Postdoctoral Researcher with the University of Pennsylvania, Philadelphia, PA USA. His research interests include distributed control, multiagent systems, formal methods, and control synthesis with applications in robotics.

Meng Guo (S'14) received the M.Sc. degree in system, control and robotics, and the Ph.D. degree in electrical engineering from KTH Royal Institute of Technology, Sweden, in 2011 and 2016, respectively.

He is currently a Research Scientist with Bosch Center for Artificial Intelligence (BCAI), Renningen, Germany. Prior to joining BCAI, he was a Postdoc Associate with the Department of Mechanical Engineering and Materials Science, Duke University, Durham, NC, USA and later KTH Royal Institute of Technology, Stockholm, Sweden. His research interest includes distributed motion and task planning of multiagent systems and formal control synthesis.

Michael M. Zavlanos (S'05–M'09–SM'19) received the Diploma in mechanical engineering from the National Technical University of Athens, Athens, Greece, in 2002, and the M.S.E. and Ph.D. degrees in electrical and systems engineering from the University of Pennsylvania, Philadelphia, PA, USA, in 2005 and 2008, respectively.

He is currently an Associate Professor with the Department of Mechanical Engineering and Materials Science, Duke University, Durham, NC, USA. He also holds a secondary appointment in the Department of Electrical and Computer Engineering and the Department of Computer Science. Prior to joining Duke University, he was an Assistant Professor with the Department of Mechanical Engineering, Stevens Institute of Technology, Hoboken, NJ, USA, and a Postdoctoral Researcher with the GRASP Lab, University of Pennsylvania, Philadelphia, PA, USA. His research interests include control theory and robotics and, in particular, networked and distributed control systems, cyber-physical systems, and distributed robotics.

Dr. Zavlanos is a recipient of various awards including the 2014 Naval Research Young Investigator Program Award and the 2011 National Science Foundation Faculty Early Career Development (CAREER) Award.