Distributed Data Gathering with Buffer Constraints and Intermittent Communication

Meng Guo and Michael M. Zavlanos

***Abstract*— We consider a team of multiple dynamical and heterogeneous robots which are deployed for gathering different types of data within a common workspace. The robots have different roles due to different capabilities: some gather data from the workspace (Type-A robots) and others receive data from Type-A robots and upload them to a data center (Type-B robots). The data-gathering tasks are specified locally to each Type-A robot as high-level Linear Temporal Logic (LTL) formulas. All robots have a limited buffer to store the data. Thus the data gathered by Type-A robots should be transferred to Type-B robots before the buffers overflow, respecting at the same time limited communication range for all robots. The main contribution of this work is a distributed task coordination and intermittent meeting scheme that guarantees the satisfaction of all local tasks while obeying the above constraints. We present numerical simulations to demonstrate the advantages of the proposed method over most existing approaches that require all-time network connectivity.**

I. Introduction

Data-gathering robots can be deployed in a workspace to gather a variety of data, which are then uploaded to a data center for processing. For instance, a team of autonomous ground vehicles (AGV) can be deployed in a large forest to monitor the temperature, humidity and stand density, or a team of autonomous aerial vehicles (AAV) can be deployed over a farmland to monitor the behavior of animal flocks and growth of the crops [1]. Due to heterogeneous sensing and motion capabilities, the robots in these applications can gather different types of data over different regions of interest within the workspace. Thus the robots can be assigned local data-gathering tasks that vary across the team [1]. In this work, we rely on Linear Temporal Logic (LTL) as the formal language to describe complex high-level tasks beyond the classic point-to-point navigation. The LTL task formula is usually specified with respect to an abstraction of the robot motion [2], [3]. Then a high-level discrete plan is found using off-the-shelf model-checking algorithms [4], and is then executed through low-level continuous controllers [5]. This framework can be extended to allow for both robot motion and actions in the task specification [6]. Partially-known or dynamic workspaces are addressed in [7].

The above framework has also been applied to multi-robot systems either in a *top-down* approach where a global LTL task formula is assigned to the whole team of robots [3], [8], [9], or in a *bottom-up* manner where an individual LTL task formula is assigned locally to each robot [7], [10]. We favor the latter formalism for the data-gathering application as the number of deployed robots can be large, the robots can have heterogeneous capabilities, and each robot can have a specific data-gathering task assignment.

Besides accomplishing high-level temporal tasks, in this paper, we also require that the robots obey the following two additional constraints: (i) limited communication range; and (ii) limited buffer size to store the gathered data. Most existing work tackles the first constraint by either maintaining all initial communication links at all time based on potential fields [10]–[13], or by allowing the addition or removal of communication links while ensuring network connectivity at all time [6], [14]–[16]. However, these approaches are rather conservative and inefficient for the data-gathering application addressed in this paper, as all robots need to stay in close proximity to each other for all time. Our earlier work [17]–[19] proposes an intermittent connectivity control strategy for mobile robot networks, although without considering buffer constraints and local temporal tasks. The second constraint above is of practical importance for data-gathering applications, especially when the local tasks require an *infinite sequence* of data-gathering actions. In this case, the robots cannot keep gathering data indefinitely as their buffers have a limited size to store these data.

The main contribution of this work lies in the distributed on-line motion and task coordination scheme for multiple heterogeneous data-gathering robots, under limited communication and buffer size constraints. We show that all local data-gathering tasks specified as LTL formulas are satisfied while obeying both constraints. Intermittent communication and data transfer are coordinated during run time and are closely coupled with the execution of local plans. The proposed scheme does not require nor impose all-time connectivity of the communication network. Its overall efficiency over the centralized approach and two static approaches is demonstrated via a numerical case study.

II. Preliminaries of LTL

Atomic propositions are Boolean variables that can be either true or false. The ingredients of an LTL formula are a set of atomic propositions AP and several boolean and temporal operators, via the syntax [4]: $\varphi ::= \top \mid p \mid \varphi_1 \wedge \varphi_2 \mid \neg\varphi \mid \bigcirc \varphi \mid \varphi_1 \mathsf{U} \varphi_2$, where $\top \triangleq \texttt{True}$, $p \in AP$ and $\bigcirc$ (*next*), U (*until*). $\bot \triangleq \neg\top$. For brevity, we omit the derivations of other useful operators like $\Box$ (*always*), $\Diamond$ (*eventually*), $\Rightarrow$ (*implication*). We refer the readers to Chapter 5 of [4] for details of LTL syntax and semantics.

The authors are with the Department of Mechanical Engineering and Materials Science, Duke University, Durham, NC 27708 USA. Emails: meng.guo, michael.zavlanos@duke.edu. This work is supported in part by the NSF awards CNS #1261828 and CNS #1302284.

III. Problem Formulation

Consider a team of N dynamical robots where each robot $i \in \mathcal{N} \triangleq \{1, \cdots, N\}$ satisfies the unicycle dynamics: $\dot{x}_i = v_i \cos(\theta_i)$, $\dot{y}_i = v_i \sin(\theta_i)$, and $\dot{\theta}_i = \omega_i$, where $p_i(t) = (x_i(t), y_i(t)) \in \mathbb{R}^2$, $\theta_i(t) \in (-\mathbf{pi}, \mathbf{pi}]$ are robot i's position and orientation at time $t > 0$. The control inputs are given by $u_i(t) = (v_i(t), \omega_i(t))$ as the linear and angular velocities (with reference denoted by $v_i^{\texttt{ref}}$ and $\omega_i^{\texttt{ref}}$). The workspace is a bounded 2D area $\mathcal{W} \subset \mathbb{R}^2$, within which there are clusters of obstacles $\mathcal{O} \subset \mathcal{W}$. The free space is denoted by $\mathcal{F} = \mathcal{W} \backslash \mathcal{O}$. Note that all robots are assumed to be point masses and robot collision is not considered here.

The robots are categorized into two subgroups, denoted by $\mathcal{N}^l, \mathcal{N}^f \subset \mathcal{N}$ and $\mathcal{N}^l \cup \mathcal{N}^f = \mathcal{N}$. Robot $i \in \mathcal{N}^f$ can *only* send and receive data locally with other robots within its *proximity*, i.e., $\|p_i(t) - p_j(t)\| \leq r_i$, where $r_i > 0$ is the communication range. Robot $j \in \mathcal{N}^l$, besides the local communication ability, has an extra function to *upload* its stored data to the remote data center. In the sequel, we simply refer to robots in $\mathcal{N}^f$ as Type-A robots and robots in $\mathcal{N}^l$ as Type-B robots. Note that $|\mathcal{N}^f|, |\mathcal{N}^l| \geq 1$ holds.

A. Data-gathering Tasks

Each Type-A robot $i \in \mathcal{N}^f$ has a local data-gathering task over its regions of interest within the freespace. Denote by $\Pi_i = \{\pi_{i,1}, \cdots, \pi_{i,M_i}\}$ these regions, where $\pi_{i,\ell} \subset \mathcal{F}$, $\forall \ell = 1, \cdots, M_i$ and $M_i > 0$. They contain information of interest. Moreover, there is a set of data-gathering actions that robot i can perform at these regions, denoted by $G_i = \{g_{i,0}, g_{i,1}, \cdots, g_{i,K_i}\}$, where $g_{i,k}$ means that "type-k data is gathered", $\forall k = 1, \cdots, K_i$ and $K_i \geq 1$. By default, $g_{i,0}$ means doing nothing. The time needed to perform each action is given by $Z_i : G_i \to \mathbb{R}^+$. With a slight abuse of notation, we denote the set of robot i's atomic propositions by $AP_i = \{\pi_{i,\ell} \wedge g_{i,k}, \forall \pi_{i,\ell} \in \Pi_i, \forall g_{i,k} \in G_i\}$, where $\pi_{i,\ell} \wedge g_{i,k}$ stands for "robot i gathers type-k data at region $\pi_{i,\ell}$". Then we can specify a high-level data-gathering task over AP_i, denoted by φ_i, following the LTL semantics in Section II. For instance, $\varphi_i = \square\diamond(\pi_{i,6} \wedge g_{i,7}) \wedge \square\diamond(\pi_{i,7} \wedge g_{i,2})$ means that "robot i should *infinitely often* gather type-7 data at region 6 and type-2 data at region 7".

B. Buffer Size and Communication Constraints

Each robot $i \in \mathcal{N}$ has a *limited buffer* to store data. Here we quantify the data size into *units*, i.e., robot i has a buffer to store maximum $\overline{B}_i > 0$ units of data, $\forall i \in \mathcal{N}$. Denote by $b_i(t) \in \mathbb{N}_{\geq 0}$ the number of data units stored in the buffer of any robot $i \in \mathcal{N}$ at time $t \geq 0$. It should hold that $b_i(t) \leq \overline{B}_i$, $\forall t > 0$. Whenever robot $i \in \mathcal{N}^f$ performs action $g_{i,k} \in G_i$ at time $t \geq 0$, $b_i(t)$ changes by $b_i(t^+) = b_i(t^-) + D_i(g_{i,k})$, where $D_i : G_i \to \mathbb{Z}^+$ is the data-gathering function for robot i's actions; $b_i(t^-)$ and $b_i(t^+)$ are the data units stored at robot i's buffer *before* and *after* action $g_{i,k}$ is performed. If $b_i(t^+) > \overline{B}_i$, then $g_{i,k}$ can not be performed as the buffer would overflow. We assume that $D_i(g_{i,k}) \leq \overline{B}_i$, $\forall g_{i,k} \in G_i$. Moreover, two robots can send and receive data within their proximity. Particularly, denote by $\texttt{c}_{ij} : \mathbb{R} \to \mathbb{Z}^+$ the data-transfer function from robot i to j at time $t \geq 0$. When robot i transfers $\texttt{c}_{ij}(t)$ units of data to robot j, their stored data change by $b_i(t^+) = b_i(t^-) - \texttt{c}_{ij}(t)$ and $b_j(t^+) = b_j(t^-) + \texttt{c}_{ij}(t)$, where the notations are similar as before. To allow this transfer, two conditions should hold: (i) $c_{ij}(t) \leq b_i(t^-)$; and (ii) $b_j(t^+) \leq \overline{B}_j$.

At last, any Type-B robot $j \in \mathcal{N}^l$ has an extra function to upload its stored data to the remote data center. Denote by $\texttt{d}_j : \mathbb{R} \to \mathbb{Z}^+$ the upload function of robot j at time $t > 0$. When robot j uploads $\texttt{d}_j(t)$ units of data to the data center, its stored data changes as $b_j(t^+) = b_j(t^-) - \texttt{d}_j(t)$, where $b_j(t^+)$ and $b_j(t^-)$ are defined similarly as before. Clearly, the uploaded data should not be more than the stored data, i.e., $\texttt{d}_j(t) \leq b_j(t^-)$ and $b_j(t^+) \geq 0$.

C. Problem Statement

Given the communication and buffer size constraints above, our goal is to synthesize the motion and action plan, and the associated data transfer protocol for all robots in $\mathcal{N}$ such that the local task φ_i is satisfied, $\forall i \in \mathcal{N}^f$.

IV. Dynamic Approach

The proposed dynamic approach consists of three main parts: (i) the workspace abstraction and the synthesis of the discrete plan; (ii) the coordination of meeting events between Type-A and Type-B robots; and (iii) the execution of the discrete plan and the data transfer protocol.

A. Local Discrete Plan Synthesis

Initially at time $t = 0$, each Type-A robot $i \in \mathcal{N}^f$ synthesizes its local discrete plan to satisfy its local task φ_i.

1) Road Map Construction: First, an abstraction of the allowed freespace $\mathcal{F}$ is constructed as the *roadmap* for all robots. In this work, we rely on the triangulation algorithm for polygons with holes; see Chapter 6 of [20]. Given the triangular decomposition, we can connect the center of each cell to the middle points of each facet (called waypoints).

Definition 1: The roadmap over $\mathcal{F}$ is a weighted undirected graph $\mathbf{M} = (M, H, W)$, where M is the set of waypoints $m \in \mathbb{R}^2$, $\forall m \in M$; $H \subseteq M \times M$ indicates whether two waypoints are connected; and $W : H \to \mathbb{R}^+$ is the Euclidean distance between two waypoints. ■

Using the roadmap $\mathbf{M}$, we can construct a finite transition system (FTS) to abstract the motion of each Type-A robot $i \in \mathcal{N}^f$ among its regions of interest. Denote this motion model by $\mathcal{T}_i = (\Pi_i, \to_i, \Pi_{i,0}, T_i)$, where $\to_i \subseteq \Pi_i \times \Pi_i$ is the transition relation, $\Pi_{i,0} \in \Pi_i$ is the initial region, $T_i : \text{“} \to_i \text{”} \to \mathbb{R}^+$ approximates the time of each transition. Particularly, consider two regions $\pi_{i,s}, \pi_{i,f} \in \Pi_i$, of which the closest waypoints to their center points are denoted by $m_{i,s}, m_{i,f} \in M$, respectively. Then $(\pi_{i,s}, \pi_{i,f}) \in \to_i$ if there exists a path in $\mathbf{M}$ starting from $m_{i,s}$ to $m_{i,f}$ *without* crossing any other waypoint $m_{i,\ell} \in M$ that belongs to any other region $\pi_{i,\ell} \in \Pi_i$ with $\ell \neq s, f$. Denote the shortest one by $\Gamma_{i,sf} = m_{i,s} m_{i,s+1} \cdots m_{i,f}$ over $\mathbf{M}$. Furthermore, the time for robot i to traverse $\Gamma_{i,sf}$ can be approximated easily given its reference linear and angular velocities.

The complete robot model [6] integrates the motion abstraction $\mathcal{T}_i$ above with the data-gathering actions in G_i:

Definition 2: The complete robot model is given by the FTS $\mathcal{R}_i = (\Pi_{i,\mathcal{R}}, \rightarrow_{i,\mathcal{R}}, AP_i, L_i, \Pi_{i,0,\mathcal{R}}, T_{i,\mathcal{R}})$, where $\Pi_{i,\mathcal{R}} = \Pi_i \times G_i$ is the full state; $\rightarrow_{i,\mathcal{R}} \subseteq \Pi_{i,\mathcal{R}} \times \Pi_{i,\mathcal{R}}$ is the allowed transitions via motion or actions so that $(\langle \pi_{i,s}, g_{i,\ell}\rangle, \langle \pi_{i,f}, g_{i,k}\rangle) \in \rightarrow_{i,\mathcal{R}}$ if (i) $\langle \pi_{i,s}, \pi_{i,f}\rangle \in \rightarrow_i$ and $g_{i,k} = g_{i,0}$, or (ii) $\pi_{i,s} = \pi_{i,f}$ and $g_{i,\ell}, g_{i,k} \in G_i$; AP_i are the atomic propositions; $L_i(\langle \pi_{i,s}, g_{i,\ell}\rangle) = \{\pi_{i,s}, g_{i,\ell}\}$, $\forall \langle \pi_{i,s}, g_{i,\ell}\rangle \in \Pi_{i,\mathcal{R}}$ is the labeling function; $\Pi_{i,0,\mathcal{R}} = \langle \Pi_{i,0}, g_{i,0}\rangle$ is the initial state; and $T_{i,\mathcal{R}}(\langle \pi_{i,s}, g_{i,\ell}\rangle, \langle \pi_{i,f}, g_{i,k}\rangle) = T_i(\pi_{i,s}, \pi_{i,f}) + Z_i(g_{i,k})$, $\forall (\langle \pi_{i,s}, g_{i,\ell}\rangle, \langle \pi_{i,f}, g_{i,k}\rangle) \in \rightarrow_{i,\mathcal{R}}$ is the time measure. ■

2) Local Plan Synthesis: The local plan of each robot $i \in \mathcal{N}$ is an infinite path of $\mathcal{R}_i$, denoted by $\tau_{\mathcal{R},i}$, such that φ_i is satisfied. The algorithm to compute $\tau_{\mathcal{R},i}$ with the prefix-suffix structure and minimal summation cost can be found in [7], [21]. We omit details here due to limited space. The initial plan is denoted by $\tau^0_{i,\mathcal{R}} = \pi^0_{i,\mathcal{R}} \cdots \pi^{L_i-1}_{i,\mathcal{R}} (\pi^{L_i}_{i,\mathcal{R}} \cdots \pi^{H_i}_{i,\mathcal{R}})^\omega$, where $\pi^\ell_{i,\mathcal{R}} \in \Pi_{i,\mathcal{R}}$, $\forall \ell = 0, \cdots, H_i$. Thus $\tau^0_{i,\mathcal{R}}$ provides an infinite sequence of motion and data-gathering actions to be performed to satisfy φ_i.

Remark 1: Note that each Type-A robot $i \in \mathcal{N}^f$ synthesizes its discrete plan $\tau^0_{i,\mathcal{R}}$ locally *without* coordination with other robots. Thus, robot i might not execute $\tau^0_{i,\mathcal{R}}$ successfully, due to its limited buffer size and the infinite sequence of data-gathering actions in $\tau^0_{i,\mathcal{R}}$. ■

B. Intermittent Meeting-events Coordination

To execute the plan of each Type-A robot $i \in \mathcal{N}^f$, we need to *ensure* that its stored data is transferred to any Type-B robot $j \in \mathcal{N}^l$ *before* its buffer overflows. The main difficulty lies in the limited communication range for both Type-A and Type-B robots, meaning that both data transfer and coordination are only possible when two robots are close enough. The key idea is to design a method that allows Type-A and Type-B robots each time they meet (i.e., connect to each other) to negotiate *when* and *where* they should meet the *next time*, while minimizing the waiting time at the new meeting location. Afterwards, they move independently without communication, until they meet again at the agreed location and time, and the same procedure repeats.

1) Initial Coordination: Initially at $t = 0$, each Type-A robot needs to coordinate its first meeting event with at least one Type-B robot. Denote by $\mathcal{N}_i(t) \subset \mathcal{N}$ the set of robots that robot $i \in \mathcal{N}$ can communicate with at time $t \geq 0$, i.e., $\mathcal{N}_i(t) = \{j \in \mathcal{N} \,|\, \|p_i(t) - p_j(t)\| \leq r\}$. Then denote by $\mathcal{N}^l_i(t) = \mathcal{N}_i(t) \cap \mathcal{N}^l$ the set of Type-B robots that a Type-A robot $i \in \mathcal{N}^f$ is connected to at time t. To begin with, we need the following assumption for the initial configuration:

Assumption 1: At time $t = 0$, each Type-A robot $i \in \mathcal{N}^f$ is connected to *at least* one Type-B robot $j \in \mathcal{N}^l$, i.e., $\mathcal{N}^l_i(0) \neq \emptyset$, $\forall i \in \mathcal{N}^f$. ■

Meeting requests by Type-A robots: Every Type-A robot $i \in \mathcal{N}^f$ needs to estimate where and when it needs to meet with a Type-B robot $j \in \mathcal{N}^l$, given its discrete plan $\tau^0_{i,\mathcal{R}}$ from Section IV-A. Particularly, it searches through the future sequence of states in $\tau^0_{i,\mathcal{R}}$ and determines the *first* state where the data stored in its buffer would exceed its size $\overline{B}_i$ if it has not met any Type-B robot to transfer its data in the meanwhile. Denote by $\pi^{k_e}_{i,\mathcal{R}} \in \tau^0_{i,\mathcal{R}}$ this state and by $\pi^{k_t}_{i,\mathcal{R}} \in \tau^0_{i,\mathcal{R}}$ the current state of robot i, where $k_e > k_t \geq 0$. Specifically, the index k_e is the index that

$$\textstyle\sum_{k=k_t}^{k_e} D_i(g_{i,\ell_k}) < \overline{B}_i,\ \sum_{k=k_t}^{k_e+1} D_i(g_{i,\ell_k}) \geq \overline{B}_i, \quad (1)$$

where $\pi^k_{i,\mathcal{R}} = \langle \pi_{i,s_k}, g_{i,\ell_k}\rangle$, $\forall k_t \leq k \leq k_e$ and $D_i(g_{i,\ell_k})$ is the number of data units gathered via action g_{i,ℓ_k}. Thus the buffer is not full after storing the gathered data up to $\pi^{k_e}_{i,\mathcal{R}}$, but it will overflow at $\pi^{k_e+1}_{i,\mathcal{R}}$ after performing $g_{i,\ell_{k_e+1}}$.

Then, robot i calculates the trajectory and the associated time to transition from $\pi^{k_e}_{i,\mathcal{R}}$ to $\pi^{k_e+1}_{i,\mathcal{R}}$. Let $\pi^{k_e}_{i,\mathcal{R}}|_{\Pi_i} = \pi_{i,s_i}$ and $\pi^{k_e+1}_{i,\mathcal{R}}|_{\Pi_i} = \pi_{i,f_i}$. From Section IV-A.1, we know that the shortest path from π_{i,s_i} to π_{i,f_i} is given by $\Gamma_{i,s_if_i} = m_{i,s_i} \cdots m_{i,f_i}$ and the associated time of reaching each waypoint $m_{i,s_i} \in \Gamma_{i,s_if_i}$ is denoted by $t_{i,k_i} \in T_{i,s_if_i}$, where $T_{i,s_if_i} = t_{i,s_i} \cdots t_{i,f_i}$, and $k_i \in \mathcal{I}^{sf}_i \triangleq \{s_i, \cdots, f_i\}$. Similar to $T_i(\cdot)$ in the FTS $\mathcal{T}_i$, the time sequence T_{i,s_if_i} is calculated using the reference velocities. At last, the *request* message from a Type-A robot $i \in \mathcal{N}^f$ to a Type-B robot $j \in \mathcal{N}^l_i(0)$, denoted by $\mathbf{Req}_{ij}(0)$, is given by $\mathbf{Req}_{ij}(0) = (\Gamma_{i,s_if_i}, T_{i,s_if_i})$, where Γ_{i,s_if_i} and T_{i,s_if_i} are defined above, $\forall j \in \mathcal{N}^l_i(0)$.

Replies by Type-B robots: Upon receiving the requests from all Type-A neighbors $i \in \mathcal{N}^f_j(0)$, where $\mathcal{N}^f_j(0) \triangleq \mathcal{N}_j(0) \cap \mathcal{N}^f$, each Type-B robot $j \in \mathcal{N}^l$ should decide the location and time to meet each of them and reply accordingly. Denote by $\mathbf{Rep}_{ji}(t)$ the reply message from robot j to robot i at time $t \geq 0$, which has the structure: $\mathbf{Rep}_{ji}(0) = (m_{ji}, t_{ji})$, where $m_{ji} \in M$ is the waypoint where the two robots would meet; $t_{ji} > 0$ is the meeting time, $\forall i \in \mathcal{N}^f_j(0)$. In the following we describe how the replies can be determined to satisfy all the requests.

Given the requests $\mathbf{Req}_{ij}(0) = (\Gamma_{i,s_if_i}, T_{i,s_if_i})$, $\forall i \in \mathcal{N}^f_j(0)$, we intend to find a path $\Gamma_j(0) = m_{j,1} \cdots m_{j,S_j}$, where $m_{j,s_j} \in M$, $\forall s_j = 1, \cdots, S_j$ and the associated time sequence $T_j(0) = t_{j,1} \cdots t_{j,S_j}$ such that the following *two* criteria hold. First, Γ_j intersects with Γ_{i,s_if_i} exactly once, i.e., there exists exactly one waypoint $m_{ji} \in \Gamma_{i,s_if_i}$ that $m_{ji} \in \Gamma_j$, $\forall i \in \mathcal{N}^f_j$. Without loss of generality, let $m_{ji} = m_{i,k_{ji}}$ where $s_i \leq k_{ji} \leq f_i$ and $m_{ji} = m_{j,s_{ji}}$ where $1 \leq s_{ji} \leq S_j$. Second, this path minimizes the sum of the differences in the expected meeting time between robot j and each $i \in \mathcal{N}^f_j(0)$, i.e., $\sum_{i \in \mathcal{N}^f_j} |t_{i,k_{ji}} - t_{j,s_{ji}}|$, where $t_{i,k_{ji}} \in T_{i,s_if_i}$ and $t_{j,s_{ji}} \in T_j$ are the corresponding time instances of reaching m_{ji} in $\Gamma_{i,s_if_i}(0)$ and $\Gamma_j(0)$.

The above problem is closely related to the well-known traveling salesman problem (TSP) and the generalized TSP [22]. This problem is NP-hard [20] as it contains the TSP as a special case. To find the exact solution, we formulate the following integer linear program (ILP):

$$\textbf{min} \sum_{k_i,\, k_h \in \mathcal{I}^{sf}_{i,h};\, i,h \in \mathcal{N}^f_{j,+}} c_{k_ik_h} \cdot \beta_{k_ik_h} \quad (2)$$

$$\textbf{s.t.}\quad \text{(I)} \sum_{k_h\in\mathcal{I}_h^{sf};\, h\in\mathcal{N}_{j,+}^f} \beta_{k_hk_i} = \sum_{h\in\mathcal{N}_{j,+}^f;\, k_h\in\mathcal{I}_h^{sf}} \beta_{k_ik_h},$$
$$\forall k_i\in\mathcal{I}_i^{sf},\ \forall i\in\mathcal{N}_{j,+}^f;$$
$$\text{(II)} \sum_{k_i,\, k_h\in\mathcal{I}_{i,h}^{sf};\, h\in\mathcal{N}_{j,+}^f} \beta_{k_ik_h} = 1, \quad \forall i\in\mathcal{N}_{j,+}^f;$$
$$\text{(III)}\quad \alpha_{k_i} - \alpha_{k_h} + (N_j^f+1)\cdot\beta_{k_ik_h} \le N_j^f,$$
$$\forall k_i,k_h\in\mathcal{I}_{i,h}^{sf};\ \forall i,\ h\in\mathcal{N}_j^f\cup\{\nu\};$$

where $\mathcal{N}_{j,+}^f = \mathcal{N}_j^f\cup\{j\}\cup\{\nu\}$ and ν is an artificial node; $N_j^f = |\mathcal{N}_j^f|$; $k_i,k_h\in\mathcal{I}_{i,h}^{sf}$ implies $k_i\in\mathcal{I}_i^{sf}$ and $k_h\in\mathcal{I}_h^{sf}$, similar definition for $k_i,k_j\in\mathcal{I}_{i,j}^{sf}$; $\Gamma_{j,s_jf_j} = \Gamma_{\nu,s_\nu f_\nu} = m_{\nu,0}$ and $T_{j,s_jf_j} = T_{\nu,s_\nu f_\nu} = 0$ to unify the notation; $c_{k_ik_h} = |t_{i,k_i} + T_i(m_{i,k_i},m_{h,k_h}) - t_{h,k_h}|$, $\forall k_i,\ k_h\in\mathcal{I}_{i,h}^{sf}$ and $\forall i,h\in\mathcal{N}_j^f$; $c_{k_hk_\nu}=0$, $\forall k_h\in\mathcal{I}_h^{sf}$ and $\forall h\in\mathcal{N}_j^f\cup\{j\}$; $c_{k_\nu k_h}=\infty$, $\forall k_h\in\mathcal{I}_h^{sf}$ and $\forall h\in\mathcal{N}_j^f$ but $c_{k_\nu k_j}=0$; $\beta_{k_ik_h}\in\mathbb{B}$ is 1 if Γ_j contains a segment from m_{i,k_i} to m_{h,k_h} and 0 otherwise, $\forall k_i,\ k_h\in\mathcal{I}_{i,h}^{sf}$ and $\forall i,\ h\in\mathcal{N}_{j,+}^f$. The first two constraints (I)-(II) require that exactly one element of Γ_{i,s_if_i} is intersected by Γ_j, $\forall i\in\mathcal{N}_{j,+}^f$ and the last constraint (III) requires that all these elements belong to one big cycle where ν is the last node and connected to j. More detailed explanations can be found in [22]. The above problem has $\binom{\hat{N}}{2}$ Boolean variables and $\hat{N}$ Integer variables, where $\hat{N} = \sum_{i\in\mathcal{N}_{j,+}^f}|\Gamma_{i,s_if_i}|$. Lastly, given the solutions Γ_j and T_j, the replies $\textbf{Rep}_{ji}(t)$ can be determined easily.

Confirmation by Type-A robots: Upon receiving the replies $\textbf{Rep}_{ji}(0)$ from all Type-B robots $j\in\mathcal{N}_i^l(0)$, each Type-A robot $i\in\mathcal{N}^f$ evaluates these replies and sends confirmations back. Denote by $\textbf{Conf}_{ij}(0)$ the confirmation message from robot i to $j\in\mathcal{N}_i^l(0)$ so that $\textbf{Conf}_{ij}(0)=\top$ if robot i confirms the meeting location and time with robot j, while $\textbf{Conf}_{ij}(0)=\bot$ if robot i refuses the reply and thus is *not* committed to the meeting event with robot j.

Specifically, given the replies $\textbf{Rep}_{ji}(0)=(m_{ji},t_{ji})$, $\forall j\in\mathcal{N}_i^l(0)$, robot i chooses the Type-B robot $j_i^\star\in\mathcal{N}_i^l(0)$ that yields the *minimum* waiting time for itself, i.e., $j_i^\star = \textbf{argmin}_{j\in\mathcal{N}_i^l(0)}|t_{ji}-t_{i,k_{ji}}|$, where $s_i\le k_{ji}<f_i$ satisfies that $m_{i,k_{ji}} = m_{ji}$. Then the confirmation message is $\textbf{Conf}_{ij_i^\star}(0)=\top$ for the $j_i^\star$ obtained above, while $\textbf{Conf}_{ij_i^\star}(0)=\bot$, $\forall j\in\mathcal{N}_i^l(0)$ and $j\neq j_i^\star$.

On the other hand, after receiving the confirmation messages $\textbf{Conf}_{ij}(0)$ back from all $i\in\mathcal{N}_j^f(0)$, each Type-B robot $j\in\mathcal{N}^l$ marks the confirmed meeting events in its path $\Gamma_j(0)$ from (2) such that it is only committed to meet the Type-A robot i that $\textbf{Conf}_{ij}(0)=\top$, $\forall i\in\mathcal{N}_j^f(0)$.

2) Coordination for Next Meeting Event: After the initial coordination at $t=0$, robots i and $j_i^\star$ will meet at the waypoint $m_{j_i^\star i}$ at time $t_{j_i^\star i}$, $\forall i\in\mathcal{N}^f$. To simplify notation, we replace $j_i^\star$ by j in this section. At the meeting, after the stored data at robot i is transferred to robot j, the two robots coordinate for their *next* meeting event.

First, robot i needs to determine again the segment of path during which it should meet with a Type-B robot. The same equation by (1) can be applied given that its *current* buffer size is zero and $\pi_{i,\mathcal{R}}^{k_t}$ is the current state. Denote the new request message by $\textbf{Req}_{ij}(t) = (\Gamma_{i,s_if_i},T_{i,s_if_i})$ where $\Gamma_{i,s_if_i} = m_{i,s_i}\cdots m_{i,f_i}$ and $T_{i,s_if_i} = t_{i,s_i}\cdots t_{i,f_i}$ are defined analogously as before. Then, after receiving the request, robot j needs to reply with its preferred next meeting *location* and *time* with robot i, denoted by m_{ji}^+ and t_{ji}^+, respectively. Let m_{j,f_j} and t_{j,f_j} be the last committed meeting location and time of robot j with another Type-A robot, based on $\Gamma_j(t)$ and $T_j(t)$ after time t. Then choosing the index s_{ji}^+ of Γ_{i,s_if_i} in order to move from m_{j,f_j} to m_{i,s_{ji}^+} yields the minimum waiting time, i.e., $s_{ji}^+ = \textbf{argmin}_{s_i\le s_{ji}\le f_i}|t_{j,f_j} - t_{i,s_{ji}} + T_j(m_{j,f_j},m_{i,s_{ji}})|$, where $T_j(m_{j,f_j},m_{i,s_{ji}})$ is the time robot j would take to navigate from waypoint m_{j,f_j} to m_{i,s_{ji}^+}. This optimization problem can be solved by iterating through all waypoints in Γ_{i,s_if_i} to find the minimum relative waiting time. Thus the reply message from robot j to i is given by $\textbf{Rep}_{ji} = (m_{ji}^+,t_{ji}^+)$, where $m_{ji}^+ = m_{i,s_{ji}^+}$ and $t_{ji}^+ = t_{i,s_{ji}^+}$. After receiving the reply message, robot i sends back the confirmation as $\textbf{Conf}_{ij}=\top$ and marks m_{ji} as the next meeting location with robot j. On the other hand, after receiving the confirmation, robot j *concatenates* its path Γ_j with the shortest path $\Gamma_j(m_{j,f_j},m_{ji}^+)$ and marks m_{ji}^+ as the next meeting location with robot i.

3) Spontaneous Meeting Events: Lastly, when there are more than one Type-B robots in the team, it is possible that robot $i\in\mathcal{N}_i^f$ meets with another Type-B robot $j'\in\mathcal{N}^l$ on its way to meet the confirmed Type-B robot $j_i^\star$. We call this situation a *spontaneous* meeting event.

In this case, robot i transfers the stored data in its buffer to robot j', and coordinates with j' for the next meeting event in a similar way as described in Section IV-B.2, but now robot i takes into account the fact that it will meet with $j_i^\star$ at $m_{j_i^\star i}^+$ as previously confirmed. Thus the next path segment of Γ_i where robot i needs to meet with a Type-B robot should be calculated as in (1) by setting $\pi_{i,\mathcal{R}}^{k_t} = (m_{j_i^\star i}^+,g_0)$, i.e., robot i's buffer is zero after meeting robot $j^\star$ at $m_{j_i^\star i}^+$. After the coordination with robot j', robot i continues to meet robot $j_i^\star$. In this way, a Type-A robot can meet and transfer data through *all* Type-B robots it has met, instead of being restricted to the Type-B robot it was connected to initially.

Remark 2: It is crucial that the Type-A robot i *still* meets its initially confirmed Type-B robot $j_i^\star$ (even with an empty buffer), after a spontaneous meeting with robot $j'\in\mathcal{N}^l$. Otherwise, robot $j_i^\star$ will wait for robot i at the confirmed region indefinitely, which leads to a deadlock. ■

C. Integrated System

1) Plan Execution: At $t=0$, after the initial coordination, each Type-A robot $i\in\mathcal{N}^f$ starts executing its discrete plan $\tau_{i,\mathcal{R}}^0 = \pi_{i,\mathcal{R}}^0\cdots\pi_{i,\mathcal{R}}^{k_i-1}(\pi_{i,\mathcal{R}}^{k_i}\cdots\pi_{i,\mathcal{R}}^{K_i})^\omega$, where $\pi_{i,\mathcal{R}}^k = \langle\pi_{i,s_k},g_{i,\ell_k}\rangle\in\Pi_{i,\mathcal{R}}$, $\forall k=0,\cdots,K_i$. Starting from the initial position π_{i,s_0}, robot i first navigates to region π_{i,s_1} through path Γ_{i,s_0s_1}. The control inputs follow the turn-and-forward switching control: turn by $\omega_i = \omega_i^{\text{ref}}$ and then forward by $v_i = v_i^{\text{ref}}$. Once robot i reaches π_{i,s_1}, it

performs action g_{i,ℓ_1} there. Afterwards, robot i continues to state $\langle \pi_{i,s_2}, g_{i,\ell_2} \rangle$. This procedure repeats itself until robot i reaches the $(k_e)_{\text{th}}$ state $\pi_{i,\mathcal{R}}^{k_e}$ according to (1). During this period of time, the amount of data units stored in robot i's buffer is increased incrementally by $D_i(g_{i,\ell_k})$, $\forall k = 0, \cdots, k_e$. Then on its way from state $\pi_{i,\mathcal{R}}^{k_e}$ to $\pi_{i,\mathcal{R}}^{k_e+1}$, robot i meets with robot $j_i^\star$ at waypoint $m_{j_i^\star i}$. It is ensured by (1) that the buffer is never overflowed and all data-gathering actions can be performed before reaching $\pi_{i,\mathcal{R}}^{k_e+1}$. After the meeting, robot i continues executing the rest of its plan until the next meeting event with $j_i^\star$ or another Type-B robot. On the other hand, each Type-B robot $j \in \mathcal{N}^l$ executes the path Γ_j derived from (2) in a similar way.

2) Meeting Event Execution: Assume that $\Gamma_{i,s_i f_i} = m_{i,s_i} \cdots m_{i,f_i}$ is the route that robot i follows to navigate from m_{i,s_i} to m_{i,f_i}, and assume also that its confirmed meeting waypoint with a Type-B robot $j \in \mathcal{N}^l$ is $m_{i,s}$. Starting from m_{i,s_i}, robot i moves towards $m_{i,s}$. Then *two* cases are possible: (i) if robot j is already waiting at $m_{i,s}$, then robot i continues moving towards $m_{i,s}$ until robot j is in its communication range. When this happens, robot i transfers *all* its data to robot j. As a result, the stored data in the buffers of robots i and j are changed to $b_i(t^+) = 0$ and $b_j(t^+) = b_j(t^-) + b_i(t^-)$. When the data transfer is completed, robot j uploads all its data to the data station. Thus its stored data is changed to $b_j(t^+) = 0$. If the stored data at robot i is more than robot j's buffer size, it needs to be divided into smaller batches, which are then transferred to robot j sequentially; (ii) if robot j has not arrived at m_{ji} yet, then robot i waits until robot j enters its communication range and then follows the same procedure as in (i).

V. Case Study

This section presents the simulation results for a team of 12 data-gathering robots. All algorithms are implemented in Python 2.7. "`Gurobi`" [23] and "`poly2tri`" are external packages. All simulations are carried out on a laptop (3.06GHz Duo CPU and 8GB of RAM).

A. System Description

All 12 robots satisfy the unicycle dynamics. There are 9 Type-A robots (denoted by $a_0, a_2, \cdots, a_8$) and 3 Type-B robots (denoted by l_1, l_2, l_3). Their common workspace has size $10m \times 10m$ with three polygonal obstacles. All robots' communication ranges are set to $1m$. The reference linear and angular velocities are chosen randomly between $[0.5, 0.8]m/s$ and $[0.1, 0.3]rad/s$. The buffer size of all Type-A robots is chosen randomly between $[3, 5]$ data units, while all Type-B robots have buffer size of 5 data units.

To simplify the task description, we divide the Type-A robots into three categories. The first category (a_0, a_1, a_2) will gather type-1 data at region r_1, type-2 data at region r_2 and type-3 data at region r_3 (in any order), infinitely often. This specification can be expressed by the LTL formula $\varphi_{c_1} = \square\diamond(r_2 \wedge g_2) \wedge \square\diamond(r_1 \wedge g_1) \wedge \square\diamond(r_3 \wedge g_3)$. The second category (a_3, a_4, a_5) will gather type-4 and type-5 data at regions r_4, then type-4 data at region r_6 (in this order) and also type-5 data at region r_5, infinitely often, i.e., $\varphi_{c_2} = \square\diamond(((r_4 \wedge g_4) \wedge \bigcirc(r_4 \wedge g_5)) \wedge \diamond(r_6 \wedge g_4)) \wedge \square\diamond(r_5 \wedge g_5)$. Finally, the third category (a_6, a_7, a_8) will gather type-6 data at regions r_7, r_9 and type-7 data at region r_8, infinitely often, i.e., $\varphi_{c_3} = \square\diamond(r_8 \wedge g_7) \wedge \square\diamond(r_7 \wedge g_6) \wedge \square\diamond(r_9 \wedge g_6)$. The actions g_2, g_3, g_4, g_6 gather 2 units of data, while actions g_1, g_5, g_7 gather 1 unit.

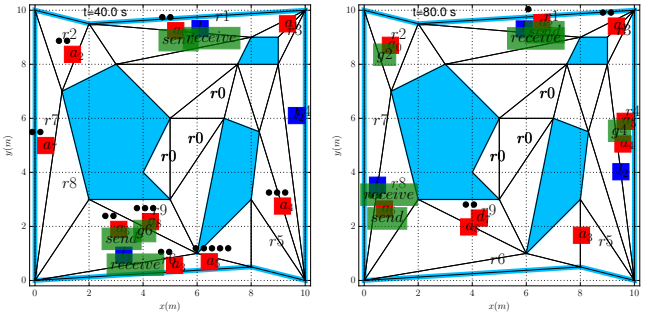

Fig. 1: Snapshots of simulation at $40s$ and $80s$. Type-A and Type-B robots are red and green squares, while the stored data units are indicated by black circles. The data-gathering actions, data transfer and upload actions are shown by filled green text boxes.

B. Simulation Results

First, the roadmap of each robot is constructed using the triangular partition as described in Section IV-A.1. For instance, for robots a_0, a_1, a_2, the FTS $\mathcal{R}_i$ has 16 nodes and 112 edges, the NBA $\mathcal{A}_{\varphi_i}$ has 4 nodes and 13 edges.

Then each Type-A robot synthesizes its discrete plan using the algorithm in [7], [21]. For instance, the first category has the discrete plan $\tau_{c_1} = r_0(r_1 g_1 r_2 g_2 r_3 g_3)^\omega$. The discrete plans are executed according to Section IV-C.1, while the data are transferred and uploaded during the meeting events described in Section IV-C.2. The coordination for the next meeting event and spontaneous meetings follow Sections IV-B.2 and IV-B.3. We simulate the system for $100s$ and 115 units of data are uploaded, as shown in Figure 3. Simulation snapshots at $40s$ and $80s$ are shown in Figure 1, where we show the number of data units stored at each robot's buffer and the data gathering or transfer actions taken by each robot. The evolution of stored data units at each robot's buffer along with time is shown in Figure 2. The complete simulation video can be found online [24]. Notice that the communication network among the robots is almost *never* connected, given the communication range of $1m$. In particular, the maximum number of connected robots remains below 5 during most of the simulation.

C. Comparisons to Other Approaches

1) Centralized Approach: A naive and centralized solution involves composing the motion models and task specifications of all robots into a complete system, which can then be model-checked to find a centralized plan, subject to the data constraints. This plan is then executed in a *fully-synchronized* way by all robots [4]. If this approach is used, the composed motion model of the whole system would have around 1.6×10^{14} nodes and 7.3×10^{22} edges. The composed

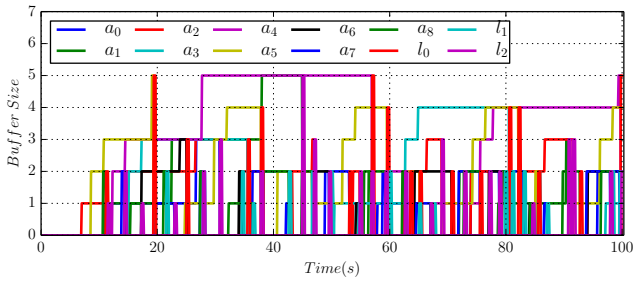

Fig. 2: Stored data in buffer for each robot during $[0s, 100s]$. The buffer sizes of robots $a_0, a_1 \cdots, a_8$ and l_0, l_1, l_0 are set to $[4, 5, 3, 4, 5, 5, 4, 5, 3, 5, 5, 5]$, which are respected for all time.

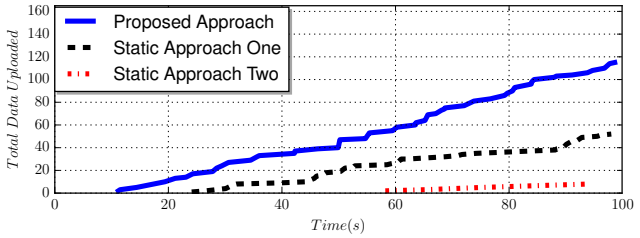

Fig. 3: The total amount of data uploaded under the proposed approach and two static approaches discussed in Section V-C. Simulation videos for all three cases are online [24]

Büchi automaton would have around 1.4×10^6 nodes and 1.5×10^{11} edges. Thus to construct the product automaton for the whole system would be computationally infeasible, not to mention the plan synthesis.

2) Static Approaches: One static approach is to require that all Type-B robots remain still at where they initial start *at all time*. As long as each Type-A robot is informed about the location of at least one Type-B robot, a Type-A robot can simply navigate to the closest Type-B robot once it has gathered some data that needs to be transferred. This solution is always *feasible* for the given formulation, but can be very inefficient if the workspace is large and many Type-B robots are located close to each other initially. We implement the above approach and simulate the system for $100s$ under the same settings. As a result, 58 units of data are uploaded in total, as shown in Figure 3, compared with 115 units via the proposed dynamic approach.

Alternatively, all Type-A and Type-B robots can converge to one location and move as a group *at all time*. Then the Type-A robots could follow a predefined order to execute their local plans. Since all Type-B robots are within the communication range, the data gathered by any Type-A robot can be transferred immediately to any Type-B robot and uploaded directly. This approach imposes all-time connectivity of the network. It can be inefficient since Type-A robots can not execute their local plans simultaneously, while Type-B robots are not fully utilized. We implement the above approach and simulate the system for $100s$ under the same settings. The Type-A robots rotate to execute their local plans according to the order of their indexes. As shown in Figure 3, only 8 units of data are uploaded in total.

VI. Conclusion and Future Work

In this work we proposed an online coordination scheme for multiple robots under local data-gathering tasks, under both communication and buffer size constraints. Future work includes realistic communication models.

References

[1] M. Dunbabin and L. Marques, "Robots for environmental monitoring: Significant advancements and applications," *Robotics & Automation Magazine, IEEE*, vol. 19, no. 1, pp. 24–39, 2012.

[2] A. Bhatia, L. E. Kavraki, and M. Y. Vardi, "Sampling-based motion planning with temporal goals," in *Robotics and Automation (ICRA), IEEE International Conference on*, 2010, pp. 2689–2696.

[3] 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," *The International Journal of Robotics Research*, vol. 32, no. 8, pp. 889–911, 2013.

[4] C. Baier and J.-P. Katoen, *Principles of model checking*. MIT press Cambridge, 2008.

[5] G. E. Fainekos, A. Girard, H. Kress-Gazit, and G. J. Pappas, "Temporal logic motion planning for dynamic robots," *Automatica*, vol. 45, no. 2, pp. 343–352, 2009.

[6] M. Guo, M. Egerstedt, and D. V. Dimarogonas, "Hybrid control of multi-robot systems using embedded graph grammars," in *Robotics and Automation (ICRA), IEEE International Conference on*, 2016.

[7] M. Guo and D. V. Dimarogonas, "Multi-agent plan reconfiguration under local ltl specifications," *The International Journal of Robotics Research*, vol. 34, no. 2, pp. 218–235, 2015.

[8] Y. Chen, X. C. Ding, A. Stefanescu, and C. Belta, "Formal approach to the deployment of distributed robotic teams," *Robotics, IEEE Transactions on*, vol. 28, no. 1, pp. 158–171, 2012.

[9] G. E. Fainekos, S. G. Loizou, and G. J. Pappas, "Translating temporal logic to controller specifications," in *Decision and Control, IEEE Conference on*, 2006, pp. 899–904.

[10] M. Guo, J. Tumova, and D. V. Dimarogonas, "Communication-free multi-agent control under local tasks and relative-distance constraints," *Automatic Control, IEEE Transactions on.*, 2016, To appear.

[11] M. Ji and M. B. Egerstedt, "Distributed coordination control of multi-agent systems while preserving connectedness." *Georgia Institute of Technology*, 2007.

[12] M. M. Zavlanos and G. J. Pappas, "Potential fields for maintaining connectivity of mobile networks," *IEEE Transactions on robotics*, vol. 23, no. 4, pp. 812–816, 2007.

[13] M. Guo, M. M. Zavlanos, and D. V. Dimarogonas, "Controlling the relative agent motion in multi-agent formation stabilization," *Automatic Control, IEEE Transactions on*, vol. 59, no. 3, pp. 820–826, 2014.

[14] M. M. Zavlanos and G. J. Pappas, "Distributed connectivity control of mobile networks," *IEEE Transactions on Robotics*, vol. 24, no. 6, pp. 1416–1428, 2008.

[15] A. Derbakova, N. Correll, and D. Rus, "Decentralized self-repair to maintain connectivity and coverage in networked multi-robot systems," in *Robotics and Automation (ICRA), IEEE International Conference on*, 2011, pp. 3863–3868.

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

[17] Y. Kantaros and M. M. Zavlanos, "Distributed intermittent connectivity control of mobile robot networks," *IEEE Transactions on Automatic Control*, 2016. To appear.

[18] ——, "A distributed ltl-based approach for intermittent communication in mobile robot networks," in *American Control Conference*, 2016.

[19] ——, "Simultaneous intermittent communication control and path optimization in networks of mobile robots," in *Decision and Control (CDC), IEEE Conference on*. IEEE, 2016, pp. 1794–1799.

[20] S. M. LaValle, *Planning algorithms*. Cambridge university press, 2006.

[21] P_MAS_TG, `https://github.com/MengGuo/P_MAS_TG`.

[22] E. L. Lawler, J. K. Lenstra, and D. B. Shmoys, *The traveling salesman problem: a guided tour of combinatorial optimization*. Wiley, 1985.

[23] Gurobi, `https://www.gurobi.com/`.

[24] Videos, `https://vimeo.com/157940708`, `157940918` and `180530665`.