Hybrid Control of Multi-agent Systems with Contingent Temporal Tasks and Prescribed Formation Constraints

Meng Guo, Charalampos P. Bechlioulis, Kostas J. Kyriakopoulos and Dimos V. Dimarogonas

Abstract—In this paper, we present a distributed hybrid control strategy for multi-agent systems with contingent temporal tasks and prescribed formation constraints. Each agent is assigned a local task given as a Linear Temporal Logic (LTL) formula. Additionally, two commonly-seen kinds of cooperative robotic tasks, namely service and formation, are requested and exchanged among the agents in real-time. The service request is a short-term task provided by one agent to another. On the other hand, the formation request is a relative deployment requirement with predefined transient response imposed by an associated performance function. The proposed hybrid control strategy consists of four major components: (a) the contingent requests handling module, (b) the real-time events monitoring module, (c) the local discrete plan synthesis module, and (d) the continuous control switching module, and it is shown that all local tasks and contingent service/formation requests are fulfilled. Finally, a simulated paradigm demonstrates the proposed control strategy.

I. INTRODUCTION

In general, cooperative control of multi-agent systems focuses on designing local control protocols to achieve a single predefined global control objective, such as reference-tracking [1], consensus [2], or formation [3]. This work is motivated by the necessity to specify and achieve more structured and complex team behaviors than the aforementioned. Particularly, we adopt LTL formulas as appropriate descriptions of desired high-level goals. LTL allows the designer to rigorously specify various temporal tasks, such as periodic surveillance, sequencing, request-response and their combinations. Afterwards, formal verification-inspired methods may be used to synthesize a discrete plan over an abstraction of the state space, that guarantees the satisfaction of the specifications. Such a generic hierarchical approach has been formulated and widely employed during the last decades in single-agent as well as multi-agent settings. For instance, [4] proposes an automated framework to translate task specifications, expressed as LTL formulas, directly into a hybrid controller that drives a dynamical system towards achieving a task goal. Regarding multi-agent systems, LTL formulas have been used to specify complex high-level global tasks [5]–[9] and local tasks [10], [11]. In the first case, a global task that captures requirements on the overall team behavior is predefined. Then the motion of all agents is synthesized and coordinated in a centralized fashion. This top-down approach is exploited in [5], [9], where the global task is decomposed into smaller local tasks to be executed by the agents in a synchronized [5] or partially synchronized manner [9]. In the latter case, each agent is assigned a local task that favors a bottom-up approach. We adopt the second formulation here that agent has a locally-assigned task. However, in contrast to [10], these tasks are dependent due to contingent service and formation requests.

Service requests are of particular interest as they encapsulate the scenario where one agent needs another agent’s assistance on a short-term task. The service requests can only be sent and confirmed among the agents online and are given as temporal formulas, which need to be satisfied by the confirmed recipient. This means that each agent has to react to the received service requests in a contingent way. Similar work on real-time reactivity for dynamical systems under temporal tasks can be found in [12], where sensory inputs are included in the General Reactivity GR(1) formulas to take into account possibly dynamical environments, as well as in [13], where similar techniques are applied for various single-robot and multi-robot applications. The local plans of each agent are synthesized off-line to handle all modeled changes in the environment, which is however not feasible in our formulation as the service and formation requests are exchanged in real-time under a non-predefined manner (i.e., the agents should adapt their local plans on-line and according to the actually confirmed service or formation requests).

Another commonly-seen multi-robot scenario appears when an agent requests from another one to form a collaborative task. We denote such a requirement as a formation request, i.e., the confirmed pair of agents should converge to a desired relative configuration until the corresponding formation task is accomplished. Similar ideas of imposing formation constraints for multi-agent systems appeared in [7], where however a global formation task for the whole team was embodied. Although the formation control has been extensively studied for multi-agent systems, e.g., [3], [14], in this work, we enforce prescribed performance constraints on the transient response of the formation process. The prescribed performance control problem was originally studied for high-order MIMO.
nonlinear single-agent systems [15] and was recently extended to multi-agent systems in our earlier work [16]. However, we enhance this control technique here by: (i) considering transient performance specifications on the overall formation error, in contrast to the relative coordinates approach [16], thus simplifying the control design; and (ii) combining it with the high-level discrete planning such that various control modes are activated according to the real-time execution of the discrete plan.

This work is clearly motivated by applications where multirobot systems are requested to exhibit complex collaborative behavior in cluttered environments. For instance, consider three robots denoted by $R_1, R_2, R_3$. Robot $R_1$ has limited motion and sensing capabilities. Hence, each time it meets $R_2$ it requests $R_2$ to perform a short-term coverage service task, e.g., visit two regions of interest in sequence. Moreover, whenever $R_1$ meets $R_3$, it requests $R_3$ to converge sufficiently fast to a relative formation to accomplish a cooperative task together, e.g., move a heavy object. Therefore, both service and formation requests are advantageous for specifying collaborative tasks in multi-agent systems. Towards this direction, we propose a hybrid control scheme that guarantees the fulfillment of high-level temporal tasks involving contingent service and formation requests. It consists of four major components: (i) the communication module that sends, receives and replies service and formation requests; (ii) the events monitoring module that detects real-time events; (iii) the discrete plan synthesis and adaptation module that incorporates contingent requests into local tasks, and (iv) the continuous controller switching module. The main contribution is twofold: (i) we present a novel way to incorporate contingent service and formation tasks in the local task specification, with predefined transient responses; (ii) the proposed plan adaptation algorithms that handle online updates of the task specifications and ensure both the current and past requests are satisfied in finite time.

The rest of the paper is organized as follows. Section II introduces the preliminaries. The problem is formulated in Section III. Section IV presents the solution. Section V demonstrates the simulation study. We conclude in Section VI.

II. PRELIMINARIES

A. Linear Temporal Logic

The basic ingredients of an LTL formula are several boolean and temporal operators, and a set of Atomic Propositions ($AP$). In particular, LTL formulas are formed according to the following syntax [18]: $\varphi ::= \top \mid a \mid \neg \varphi \mid \varphi_1 \lor \varphi_2 \mid \varphi_1 \Uparrow \varphi_2$ where $a \in AP$, $\top \triangleq \text{True}$ and $\bot \triangleq \text{False}$; $\neg$ is the “negation” operator; $\lor$ is the “or” operator; $\Uparrow$ is the “until” operator. Those basic operators can be used to derive other commonly used operators, such as $\wedge$ (and), $\Box$ (eventually), $\square$ (always). Notice that contrary to the standard definition [18], the “next” operator is not allowed here.

The semantics of LTL is defined over discrete-time Boolean signals over the same $AP$, which is also called a word over $AP$. Particularly, given a discrete-time Boolean signal $w: \mathbb{Z} \rightarrow 2^{AP}$, $w(k) \subseteq AP$ is the set of propositions that signal $w$ satisfies at discrete time $k$, $\forall k \in \mathbb{Z}$. Then, consider an LTL formula $\varphi$ and a discrete-time Boolean signal $w$ over $AP$; the satisfiability relation $(w, k) \models \varphi$, i.e., whether $w$ satisfies $\varphi$ at discrete time $k \in \mathbb{Z}$, is determined according to the following recursive definition: $(w, k) \models a \iff a \in w(k)$; $(w, k) \models \neg \varphi \iff (w, k) \not\models \varphi$; $(w, k) \models \varphi_1 \lor \varphi_2 \iff (w, k) \models \varphi_1$ or $(w, k) \models \varphi_2$; $(w, k) \models \varphi_1 \Uparrow \varphi_2 \iff \exists k_1 \geq k, (w, k_1) \models \varphi_2$ and $\forall k_2 \in [k, k_1), (w, k_2) \models \varphi_1$. Intuitively, $\varphi_1 \Uparrow \varphi_2$ requires that $\varphi_1$ must be true over $w$ until $\varphi_2$ becomes true. We say that a formula $\varphi$ is satisfied by $w$ if $(w, 0) \models \varphi$ or $w \models \varphi$ for brevity.

Given an LTL formula $\varphi$ over AP, the set of words that satisfies $\varphi$ can alternatively be captured through a B"{u}chi automaton $A_\varphi = (Q, 2^{AP}, \delta, Q_0, F)$, where $Q$ is a finite set of states; $2^{AP}$ is the allowed input alphabet; $\delta: Q \times 2^{AP} \rightarrow 2^Q$ is a transition function; $Q_0 \subseteq Q$ is a set of initial states; and $F \subseteq Q$ is a set of accepting states. An infinite run over an input word $w = w(0)w(1)\ldots$ is an infinite sequence of states $r = q_0q_1\ldots$ such that $q_0 \in Q_0$ and $q_{k+1} \in \delta(q_k, w(k))$, $\forall k \geq 1$. A run is accepting if it intersects with the accepting set $F$ infinitely many times, and an input word $w$ is accepted if there exists an accepting run over it. The B"{u}chi automaton $A_\varphi$ can be obtained by the translation algorithm [20], such that $A_\varphi$ accepts exactly the words satisfying $\varphi$.

Moreover, one particular class of LTL is the syntactically co-safe LTL (sc-LTL) [21], which contains only the $U$ and $\Diamond$ temporal operators and is written in positive normal form [23]. In contrast to general LTL, the satisfaction of an sc-LTL formula can be achieved in a finite time, i.e., each word satisfying an sc-LTL formula $\varphi$ consists of a satisfying prefix that can be followed by an arbitrary suffix.

B. Real-time Temporal Logic

To analyze properties of real-time signals, we also consider a real-time extension of LTL, i.e., the real-time temporal logic (RTL), originally introduced in [22]. Its syntax is similar to LTL as introduced in Section II-A, but its semantics is defined over continuous-time Boolean signals. Consider such a signal $x: \mathbb{R}_{\geq 0} \rightarrow 2^{AP}$ over the atomic propositions $AP$, where $x(t) \subseteq AP$ is the set of propositions that $x$ satisfies at time $t \geq 0$. Given an RTL formula $\varphi$ over $AP$, the satisfiability relation $(x, t) \models \varphi$, i.e., whether signal $x$ satisfies $\varphi$ at time $t \geq 0$, is determined via the following recursive definition: $(x, t) \models a \iff a \in x(t)$; $(x, t) \models \neg \varphi \iff (x, t) \not\models \varphi$; $(x, t) \models \varphi_1 \lor \varphi_2 \iff (x, t) \models \varphi_1$ or $(x, t) \models \varphi_2$; $(x, t) \models \varphi_1 \Uparrow \varphi_2 \iff \exists t' \geq t, (x, t') \models \varphi_2$ and $\forall t'' \in (t, t'), (x, t'') \models \varphi_1$. We say that an RTL formula $\varphi$ is satisfied by $x$ if $(x, 0) \models \varphi$ or simply $x \models \varphi$. Moreover, similarly to sc-LTL, there exists a particular class of RTL, called sc-RTL that can be satisfied by a real-time Boolean signal in finite time. It only contains the $U$ and $\Diamond$ temporal operators and is written in positive normal form [23]. Note that the RLT semantics does not include the “next” operator as it is not defined in continuous time [23]. Finally, since we consider RTL and LTL with the same syntax but interpreted with different semantics, we define the following correspondence between an RTL formula and an LTL formula:

**Definition 1.** Given an RTL formula $\varphi$, the associated LTL formula is denoted by $[\varphi]$, which has the same expression as $\varphi$. 

---

2325-5870 (c) 2015 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.
but is evaluated under the LTL semantics.

For example, consider the RTL formula \( \varphi_2 = \Box (R_2 \land \Box R_3) \land \Box R_4 \) that will be demonstrated in the simulation section. Its associated LTL formula \( [\varphi_2]_1 \) has the same expression but is evaluated under different semantics, i.e., \( \varphi_2 \) is specified over continuous-time Boolean signals, while \([\varphi_2]_1\) over discrete-time Boolean signals.

### III. Problem Formulation

**A. System Description**

We consider a team of \( N \) autonomous agents obeying the following single-integrator dynamics in a 2D workspace:

\[
\dot{p}_i(t) = u_i(t),
\]

where \( p_i(t), u_i(t) \in \mathbb{R}^2 \) denote the \( i \)-th agent’s position and control input at time \( t \geq 0, \forall i \in \mathcal{N} = \{1, 2, \ldots, N\}. \)

The agents are modeled as point masses without volume, hence inter-agent collisions are not considered. Moreover, all agents have a common sensing radius \( r > 0 \), i.e., agent \( i \) can only exchange information with another agent \( j \) if their relative distance satisfies \( \|p_i(t) - p_j(t)\| \leq r \). We also denote by \( p_i(t) \) the trajectory of agent \( i \) during the time interval \([0, t]\). Similarly, \( p_i([t_1, t_2]) \) stands for the trajectory segment during time \([t_1, t_2]\). Finally, each agent \( i \) has a predefined nonempty neighboring set \( \mathcal{N}_i \subseteq \mathcal{N} \), \( i \in \mathcal{N} \) and \( j \in \mathcal{N}_i \) implies \( i \neq j \) and \( j \notin \mathcal{N}_i \). Nevertheless, agent \( i \) can communicate with agent \( j \in \mathcal{N}_i \) only if \( \|p_i(t) - p_j(t)\| \leq r \), i.e., their relative distance is less than the communication radius.

Additionally, there exists a set \( \Pi = \{\pi_1, \pi_2, \ldots, \pi_M\} \) of \( M \geq 1 \) regions of interest within the 2D workspace. These regions may be in various shapes, such as points [24], triangles [19], polygons [25]. We consider here the circular area around the points of interest:

\[
\pi_\ell = B(c_\ell, r_\ell) = \{ p \in \mathbb{R}^2 | \|p - c_\ell\| \leq r_\ell \},
\]

for \( \ell = 1, 2, \ldots, M \), where \( c_\ell \in \mathbb{R}^2 \) is the center and \( r_\ell \geq r_{\text{min}} \) is the radius, with \( r_{\text{min}} \) denoting the minimal radius. We assume that \( \Pi \) is included within a large circular region denoted by \( \pi_B = B(c_B, r_B) \), which stands for the allowed workspace. However, there is a set of local atomic propositions \( R = \{R_\ell, \ell = 1, 2, \ldots, M\} \) representing the property fulfilled at each region. Hence, the agent state \( p_i(t) \) at time \( t \geq 0 \), it holds that \( R_\ell(t) = 1 \) if \( p_i(t) \in \pi_\ell \) and \( R_\ell(t) = 0 \) otherwise, for \( \ell = 1, 2, \ldots, M \). Finally, it should be noted that the workspace, i.e., the location and property of the regions above, is fully-known and static to all agents.

### B. Contingent Request

As mentioned earlier, the agents may exchange information directly with their neighbors when their relative distance is less than \( r \). Particularly, we allow each agent to send to its neighbors two types of contingent requests:

1. **Service**: agent \( i \) requests its neighbor \( j \in \mathcal{N}_i \) at time \( t = t_0 \) to accomplish its own short-term service described by an sc-RTL formula \( \varphi_{ij,t} \) over \( R \), which is assumed to be feasible for agent \( j \). Notice that \( \varphi_{ij,t} \) is predefined for each neighbor \( j \) and may be different at various request times. This request is communicated by sending the formula \( \varphi_{ij,t} \) directly to agent \( j \).

2. **Formation**: agent \( i \) requests its neighbor \( j \in \mathcal{N}_i \) at time \( t = t_0 \) to converge to a desired relative formation described by \( \varphi_{ij} \in \mathbb{R}^2 \) with a predefined transient response imposed by the corresponding performance function \( \rho_{ij}(t) : \mathbb{R}^+ \rightarrow \mathbb{R}^+ \). This formation has to be kept until agent \( i \) accomplishes a short-term formation task described by an sc-RTL formula \( \varphi_{ij,t} \) over \( R \) and a release message is sent to agent \( j \) afterwards. The formation task \( \varphi_{ij,t} \) is also predefined for agent \( i \) and may be different at various request times. More specifically, the relative formation error is given by:

\[
e_{ij}(t) = p_i(t) - p_j(t) - c_{ij}.
\]

Let us also define \( \rho_{ij} \) as a scalar measure of the formation error and

\[
\tilde{\mu}_{ij}(t) = \frac{\mu_{ij}(t)}{\rho_{ij}(t)}
\]

as the normalized error over the associated performance function \( \rho_{ij}(t) \), which is a smooth, bounded and strictly decreasing function of time. Here we adopt exponential functions

\[
\rho_{ij}(t) = \left(\rho_{ij}, t_0 - \rho_{ij,\infty}\right) e^{-\left(\rho_{ij} - t_0\right)} + \rho_{ij,\infty},
\]

where \( \rho_{ij} > 0 \) specifies the decreasing rate of \( \rho_{ij}(t) \), \( \rho_{ij,t} > 0 \) is the initial value of \( \rho_{ij}(t) \) at time \( t_0 \), chosen such that \( \rho_{ij,t} > \rho_{ij,\infty} > 0 \) reflects the maximum allowed steady state error. This formation request is communicated by sending \( e_{ij} \) and the parameters of \( \rho_{ij} \) to agent \( j \).

To monitor the formation performance, we also define a set of controllable atomic propositions \( H_j = \{h_{ij}, j \in N_j\} \):

\[
h_{ij}(t) = \begin{cases} \top & \text{if } \tilde{\mu}_{ij}(t) \in D_{ij} \equiv [0, 1), \\ \bot & \text{otherwise}. \end{cases}
\]

Notice that agent \( j \) needs to satisfy the prescribed formation performance, i.e., \( h_{ij}(t) = \top \) until it receives a release message from agent \( i \).

Upon receiving either a service or formation request from agent \( i \), we assume that agent \( j \) replies immediately to either confirm or deny it. Thus, we introduce a set of observational atomic propositions for each agent \( i \): \( O_i = \{o_{ij}, o_{ij}^\top, j \in N_i\} \), where \( o_{ij}(t) = \top \) if agent \( i \) confirms agent \( j \)’s service request at time \( t \); and \( o_{ij}^\top(t) = \bot \) otherwise (\( o_{ij}^\top(t) \) is defined similarly for formation requests). Additionally, we introduce a set of releasing atomic propositions: \( Z_i = \{z_{ij}, j \in N_i\} \), where \( z_{ij}(t) = \top \) if agent \( i \) sends a release message to its neighbor \( j \in N_i \) at time \( t \); and \( z_{ij}(t) = \bot \) otherwise.

### C. Local Task Specification

Let us denote by \( AP_i = R \cup O_i \cup H_i \cup Z_i \) the complete set of atomic propositions of each agent \( i \in N \). Each agent \( i \) is assigned a local task defined as an RTL formula \( \varphi_i \) over \( AP_i \), which has the following structure:

\[
\varphi_i = \varphi_i^a \land \varphi_i^f \land \varphi_i^s.
\]

Notice that \( \varphi_i^a \) describes a static task specification as a general RTL formula over \( R \) as the local task of agent \( i \). Similarly, \( \varphi_i^f \)
is a contingent task specification regarding the service requests that agent \(i\) may receive from agent \(j\) with \(i \in \mathcal{N}_j\):
\[
\varphi^s_{ij,t} \triangleq \bigwedge_{j \in \mathcal{N}_i} \Box (o^s_{ji,t} \rightarrow \varphi^s_{ji,t}).
\]
\(8\)

Hence, whenever agent \(i\) confirms a service request \(\varphi^s_{ji,t}\) from agent \(j\), then it should fulfill this service task afterwards. Finally \(\varphi^s_{ij,t}\) is the contingent task regarding the formation requests that agent \(i\) may send to its neighbor \(j \in \mathcal{N}_i\):
\[
\varphi^f_{ij,t} \triangleq \bigwedge_{j \in \mathcal{N}_i} \Box (o^f_{ji,t} \rightarrow (\varphi^f_{ji,t} \land (h_{ij} \cup z_{ij}))).
\]
\(9\)

Thus, whenever a formation request from agent \(i\) is confirmed by agent \(j\), the corresponding short-term formation task \(\varphi^f_{ij,t}\) should be fulfilled right afterwards and the formation controllable proposition \(h_{ij}\) should be always kept true, until a release message is sent to agent \(j\). In other words, once agent \(j\) confirms a formation request from agent \(i\), it should achieve the desired relative position with prescribed performance as imposed by the corresponding performance function \(\rho_{ij}(t)\), until agent \(i\) has accomplished its formation task \(\varphi^f_{ij,t}\).

**Remark 1.** While the local task \(\varphi^s_{ij}\) has been commonly-seen [25], [26] as a static task specification, the main novelty of this work lies in the contingent service task \(\varphi^s_{ij}\) by (8) and the formation task \(\varphi^f_{ij}\) introduced by (9) that have not been tackled for multi-agent systems in this context. Moreover, notice that all service and formation requests are exchanged online and hence cannot be known before the system starts. The validity of the observational and controllable propositions can only be determined in real-time. Thus, only the local task \(\varphi^f_{ij}\) and not the complete specification \(\varphi_i\) is determined and assigned to agent \(i\) before the system initializes.

In the sequel we formulate the problem confronted here:

**Problem 1.** Consider a team of agents obeying the dynamics (1), with each member assigned a set of local task specifications defined in (7)-(9). The objective is to synthesize a distributed hybrid control protocol such that all RTL formulas \(\varphi_i\) are satisfied, \(\forall i \in \mathcal{N}\).

**IV. MAIN RESULTS**

The proposed hybrid control strategy consists of four major components as shown in Figure 1 (i.e., the communication, the event monitoring, the discrete planning and the hybrid control modules). In this section, we first present two continuous motion control schemes regarding the navigation and the formation tasks respectively. Subsequently we describe the real-time event monitoring module that handles critical events during the system operation as well as the established communication protocol among the agents regarding the contingent requests. Afterwards, we present the discrete plan synthesis and adaptation algorithms that handle the contingent service and formation requests. Finally, the overall hybrid control strategy is synthesized based on the aforementioned components and its correctness is formally proven.

**A. Continuous Controller Design**

In this subsection, we describe the continuous control design for two different control objectives, i.e., to navigate an agent from any initial position to its goal region, without crossing any undesirable regions; and to establish a desired formation with prescribed transient response.

1) **Navigation Control:** Recall that the set of circular regions of interest within the workspace boundary \(\pi_B\) is given by \(\Pi = \{\pi_1, \pi_2, \ldots, \pi_M\}\). Since there are no explicit representations of “obstacles” and no fixed initial or goal regions, we denote by \(\pi_g = B(c_g, r_g) \in \Pi\) the goal region and \(\pi_s = B(c_s, r_s) \in \Pi\) the initial region. In this work, we rely on the navigation function approach proposed by Rimon and Koditschek in [24] to navigate agent \(i\) from \(\pi_s\) to \(\pi_g\), without crossing other undesirable regions \(\pi_j \in \Pi\) with \(j \neq \{s, g\}\).

In particular, the navigation function \(\Phi_i\) is constructed as:
\[
\Phi_i(p_i) = \frac{\gamma_g}{(r_g + \beta_{gs})^2},
\]
\(10\)

where \(k > 1\) is a design parameter, \(p_i \in \mathbb{R}^2\), \(\Phi_i \in [0, 1]\), \(\gamma_g = \|p_i - c_g\|^2\) represents the attractive potential field to the desired position \(c_g \in \mathbb{R}^2\) and \(\beta_{gs} = \beta_{ij} \bigcup \{1, \ldots, j \neq g\}\) is the repulsive potential field by the workspace boundary and the undesirable regions that should be avoided, with \(\beta_{ij} = r_B - \|p_i - c_B\|^2\) and \(\beta_{ij} = \|p_i - c_j\|^2 - r_j^2\). For brevity, we denote by \(\Pi_\text{avoid} \triangleq \{\pi_j = B(c_j, r_j) \in \Pi\}\) the set of circular regions that should be avoided and by \(\pi_B \setminus \Pi_\text{avoid}\) the free space. It is assumed that \(\pi_B\) and the sphere regions in \(\Pi\) satisfy the condition of a valid workspace in [24], i.e., \(\pi_m \subseteq \pi_B\) and \(\pi_m \cap \pi_n = 0\), \(\forall m, n \in \{1, 2, \ldots, M\}\) with \(m \neq n\).

It has been proved in [24] that \(\Phi_i(p_i)\) has only one global minimum at \(c_g\) and \(M - 2\) saddle points within the allowed free space with zero set-measure for a sufficiently large constant \(k\). Hence, a feasible path within the free workspace that leads an agent from its initial position in \(\pi_s\) to its goal region \(\pi_g\) can be generated by following the negated gradient of \(\Phi_i(p_i)\) or equivalently by adopting the subsequent control law:
\[
u_i = -\nabla p_i \Phi_i(p_i).
\]
\(11\)

Based on [27], it is proven that \(\gamma_g \rightarrow 0\) as \(t \rightarrow \infty\) and \(\beta_{gs} > 0\) holds, \(\forall t \geq 0\). Moreover, a collision free path is
ensured for almost any initial position in the free space (except a set of measure zero) to any goal region in the free space. Furthermore, the asymptotic stability of the aforementioned controller guarantees the convergence to the neighborhood of $p_g$ (i.e., the goal region $\pi_g$) in finite time [27].

**Lemma 1.** The motion control law (11) remains always bounded, i.e., $\|u_i\| < u_{\text{max}}$, where $u_{\text{max}} > 0$ is finite.

**Proof.** By computing the gradient of $\Phi_i(p_i)$ directly and owing to the fact that $p_i$ remains within the allowed free space without crossing $\Pi_\text{avoid}$, all terms of $\nabla_p \Phi_i$ remain bounded and hence $\|u_i\| = \|\nabla_p \Phi_i\| \leq u_{\text{max}}$ with an upper bound $u_{\text{max}}$ depending on the configuration of $\Pi_i$. 

2) Formation Control with Prescribed Performance: As described in Section III-B, a contingent request may also involve a formation task that should be executed with prescribed performance (i.e., agent $i$ requests agent $j \in N_i$ to converge to a desired formation described by the relative position $c_{ij}$ and the predefined transient response by the associated performance function $\rho_{ij}(t)$ defined in (5)). Moreover, this formation should be maintained until the associated short-term formation task $\varphi_{ij,t}$ is satisfied by agent $i$. Since agent $i$ may send a formation request to more than one neighbors, we denote the set of neighbors that have confirmed its formation request by $N^f_i \subseteq N_i$. In the sequel, we propose the motion control scheme for both agent $i$ and all agents $j \in N^f_i$. Notice that at the request time $t = t^0_i$ in case agent $i$ lies inside a region $\pi_i \in \Pi_i$ then the associated navigation function can be constructed similarly to (10), i.e., $\Phi_i^4(p_i) \triangleq (\pi^4_i + \beta_g) \pi_i$; otherwise, the repulsive potential $\beta_g$ should be slightly modified since all regions in $\Pi_i$ except for $\pi_g$ should be treated as undesirable regions to be avoided, i.e., $\Phi_i^4(p_i) \triangleq (\pi^4_i + \beta_g)$, where $\beta_g \triangleq \beta_B \prod_{j=1,j \neq i}^M (\pi^4_j + \beta_g)$.

The controller of agent $i$ is:

$$u_i = -\nabla_p \Phi_i(p_i). \quad (12)$$

On the other hand, for each agent $j \in N_i^f$ that has confirmed the formation request of agent $i$, we design a motion control law that ensures the convergence to the desired relative formation performance as follows:

$$u_j = K_{ij} \frac{\varepsilon_{ij}}{\rho_{ij}(t)} \hat{\mu}_{ij}, \quad (13)$$

where $K_{ij} > 0$ is a control gain; $\varepsilon_{ij}$ is the relative formation error from (3); $\rho_{ij}(t)$ is the performance function; and

$$\varepsilon_{ij} \triangleq \ln \left( \frac{1}{1 - \hat{\mu}_{ij}} \right), \quad (14)$$

with $\hat{\mu}_{ij}$ denoting the normalized formation error from (4).

**Lemma 2.** Under the motion control laws (12) and (13), initialized at time $t = t^0_i$, agent $i$ will arrive to its goal region $\pi_g$ within finite time $T_{g_i} > t^0_i$ while avoiding the undesirable regions included in $\Pi_{\text{avoid}}$. Regarding the agents $j \in N_i^f$, the dynamics of the normalized formation error is calculated as follows:

$$\dot{\hat{\mu}}_{ij} = \frac{2 \varepsilon_{ij}^T u_j \rho_{ij}(t) - \varepsilon_{ij}^T \varepsilon_{ij} \hat{\mu}_{ij}(t)}{\rho_{ij}(t)}$$

where $u_j(t) = u_i(t) - u_j(t)$. Substituting the control input (13) and $\varepsilon_{ij}^T u_j(t) \equiv \rho_{ij}(t) \hat{\mu}_{ij}(t)$, we obtain:

$$\dot{\hat{\mu}}_{ij} = -\frac{\hat{\mu}_{ij}}{\rho_{ij}(t)} \left[ K_{ij} \varepsilon_{ij} + \hat{\mu}_{ij} \right] + \frac{2 \varepsilon_{ij}^T u_i(t)}{\rho_{ij}(t)}$$

$$= -\frac{\hat{\mu}_{ij}}{\rho_{ij}(t)} \left[ K_{ij} \ln \left( \frac{1}{1 - \hat{\mu}_{ij}} \right) + \hat{\mu}_{ij} \right] + \frac{2 \varepsilon_{ij}^T u_i(t)}{\rho_{ij}(t)} \quad (15)$$

Since the right-hand side of (15) is continuous on $t$ and locally Lipschitz on $\hat{\mu}_{ij}$ over $D_{ij}$, as defined in (6), we may apply Theorem 54 of [17] (page 476) to derive a maximal solution of (15) on a time interval $[t^0_i, \tau_{\text{max}}]$ such that $\hat{\mu}_{ij}(t) \in D_{ij}, \forall t \in [t^0_i, \tau_{\text{max}}]$. Then consider the Lyapunov-like function:

$$V_{ij}(t) = \frac{1}{2} \varepsilon_{ij} \hat{\mu}_{ij} = \frac{\varepsilon_{ij}}{1 - \hat{\mu}_{ij}} \hat{\mu}_{ij}. \quad (16)$$

which owing to (14) is well-defined for all $t \in [t^0_i, \tau_{\text{max}}]$. Differentiating $V_{ij}(t)$ with respect to time, we get:

$$\dot{V}_{ij} = \varepsilon_{ij} \hat{\mu}_{ij} - \frac{\varepsilon_{ij}}{1 - \hat{\mu}_{ij}} \hat{\mu}_{ij}. \quad (17)$$

Moreover, invoking (15), we arrive at:

$$\dot{V}_{ij} = -m_{ij} \left[ \hat{\mu}_{ij} (K_{ij} \varepsilon_{ij} + \hat{\mu}_{ij} \varepsilon_{ij} - 2 \varepsilon_{ij}^T u_i(t)) \right], \quad (18)$$

where $m_{ij}(t) = \hat{\mu}_{ij}(1 - \hat{\mu}_{ij}) > 0$. Hence, since $\varepsilon_{ij} > 0$, $\rho_{ij}(t) > 0$ and $\varepsilon_{ij}^T u_i(t) \leq \varepsilon_{ij}^T \|u_i(t)\|$, we conclude that:

$$\dot{V}_{ij} \leq -m_{ij} \left[ K_{ij} \hat{\mu}_{ij} \varepsilon_{ij} + \hat{\mu}_{ij} \varepsilon_{ij} - 2 \|\varepsilon_{ij}\| \|u_i(t)\| \right]$$

$$\leq -m_{ij} \left[ K_{ij} \hat{\mu}_{ij} \varepsilon_{ij} - 2 \sqrt{\rho_{ij}(t)} \|u_i(t)\| \right]$$

where we employed the fact that $0 \leq \hat{\mu}_{ij} < 1$ and $\|\varepsilon_{ij}\| = \sqrt{\rho_{ij}(t)} \rho_{ij}(t) \leq \sqrt{\rho_{ij}(t)} \|u_i(t)\|$. Thus, invoking the inverse of (14) (i.e., $\hat{\mu}_{ij} = 1 - e^{-\varepsilon_{ij}}$), we conclude that $V_{ij} < 0$ when $\varepsilon_{ij}(1 - e^{-\varepsilon_{ij}})$

$$> \sup_{j \in [t^0_i, \tau_{\text{max}}]} \|\varepsilon_{ij}(t)\| + 2 \sqrt{\rho_{ij}(t)} \|u_i(t)\| \right] \cong b_{ij}. \quad (19)$$

Notice that $b_{ij}$ is finite as $K_{ij} > 0$, $\rho_{ij}(t)$, $\rho_{ij}(t)$ are bounded by construction and $\|u_i(t)\| \leq u_{\text{max}}$ as proven in Lemma 1.

Consider now the smooth function $g(x) = x(1 - e^{-x})$, which is monotonically increasing for $x > 0$ with $g(0) = 0$. Let $\varepsilon^*_i = 1$ be a constant that satisfies $\varepsilon^*_i (1 - e^{-\varepsilon^*_i}) = b_{ij}$, which exists and is unique owing to the monotonicity of $g(x)$. Hence, $V_{ij} < 0$ when $\varepsilon_{ij} > \varepsilon^*_i$, from which we conclude that $\varepsilon_{ij}(t) < \max \{ \varepsilon^*_i, \varepsilon_{ij}(t^0) \} \cong 0^{+}$, $\forall t \in [t^0_i, \tau_{\text{max}}]$ and

$$\hat{\mu}_{ij}(t) < 1 - e^{-\varepsilon_{ij}} \cong 0^{+}, \quad \forall t \in [t^0_i, \tau_{\text{max}}].$$

Thus, we deduce that $\hat{\mu}_{ij}(t) \in [0, \tau_{\text{max}}] \cong D_{ij}, \forall t \in [t^0_i, \tau_{\text{max}}]$, where $D_{ij}$ is a nonempty and compact subset of $D_{ij}$.

Finally, what remains to be shown is that $\tau_{\text{max}}$ can be any time greater than $t^0_i$. Therefore, assume that $\tau_{\text{max}} = \infty$. Since
\(D_{ij}^{"} \subset D_{ij}\) and \(\hat{\mu}_{ij}(t)\) is a maximal solution of (15) over \([t_0, \tau_{max})\). Proposition C3.6 in [17] (page 481) dictates the existence of a time instant \(t' \in [t_0, \tau_{max})\) such that \(\hat{\mu}_{ij}(t) \notin D_{ij}^{"}\), which contradicts with the fact that \(\hat{\mu}_{ij}(t) \in D_{ij}^{"}\), \(\forall t \in [t_0, \tau_{max})\). Thus \(\tau_{max}\) can be extended to any time greater than \(t_0\) (i.e., \(\tau_{max}\) can be selected to be equal to \(T_{p}\)) and \(\hat{\mu}_{ij}(t) \in [0, \hat{\mu}_{ij}] \subset D_{ij}\), which ensures that \(c_{ij}^{T}(t)c_{ij}(t) < \rho_{ij}\) and consequently \(h_{ij}(t) = \tau\), \(\forall t \in [T_0, T_0]\).

### B. Real-time Event Monitoring Scheme

It is apparent that monitoring several real-time events plays a crucial role in the satisfaction of all tasks. For each agent \(i \in \mathcal{N}\), four types of events need to be monitored online:

(a) **Region cross event.** It occurs when an agent enters or leaves a region in \(\mathcal{N}\). More precisely, agent \(i\) enters the region \(\pi_e \in \mathcal{N}\) at time \(t_0 > 0\) if \(p_{ij}(t_0 - \delta_t) \notin \pi_e\) and \(p_{ij}(t_0) \in \pi_e\), where \(\delta_t > 0\) is a design parameter as the dwell time [28]. Agent \(i\) leaves the region \(\pi_e \in \mathcal{N}\) at time \(t_0 > 0\) if \(p_{ij}(t_0 - \delta_t) \notin \pi_e\) and \(p_{ij}(t_0) \notin \pi_e\). This event is directly related to the validity of proposition \(R_t \in \mathcal{R}\). \(\delta_t\) can be enforced by setting the crossing event true only if agent \(i\) has been inside region \(\pi_t\) for at least time period of length \(\delta_t\).

(b) **Request and reply event.** It occurs when agent \(i\) sends a service or formation request to its neighbor \(j \in \mathcal{N}_i\) at time \(t_0\) and agent \(j\) replies this request at the same time. Depending on the replies, we consider two cases: if agent \(j\) confirms this service request, the observation proposition \(o_{ij}^s(t) \in \mathcal{O}_i\) satisfies \(o_{ij}(t) = \top\), \(\forall t \in [t_0, t_1 + \delta]\), and \(o_{ij}(t) = \bot\), \(\forall t \in [t_0 + \delta, t_1]\), where \(t_1 > t_0 + \delta\) is the next time instant when agent \(j\) receives a request from agent \(i\); otherwise, if agent \(j\) denies this request, \(o_{ij}(t) = \bot\), \(\forall t \in [t_0, t_1]\). The same rules apply to propositions related to formation requests \(\{o_{ij}, \forall j \in \mathcal{N}_i\}\). In addition, if agent \(i\) sends a release message to its neighbor \(j \in \mathcal{N}_i\) at \(t_0\), then \(z_{ij}(T) = \top\), \(\forall t \in [t_0, t_1 + \delta]\) and \(z_{ij}(t) = \bot\), \(\forall t \in [t_1 + \delta, t_1]\), where \(t_1\) is the next time instant when agent \(i\) sends a release message to agent \(j\).

(c) **Service finish event.** It occurs when agent \(i\) has finished a service request that it has confirmed. For instance, suppose that \(o_{ij}^s(t_0) = \top\) and agent \(i\) confirms the service request \(\varphi_{ij,t_0}\) by agent \(j\) at time \(t_0\). Then \(\varphi_{ij,t_0}\) is satisfied by agent \(i\) at time \(t_f \geq t_0\), if its trajectory \(p_{ij}(t)\) satisfies \(\varphi_{ij,t_0}\) on the basis of the semantics presented in Section II-B. Since all service tasks are feasible and can be finished in finite time, the finishing time \(t_f > t_0\) is finite for each service request.

(d) **Formation finish event.** It occurs when agent \(i\) has finished a formation task that has been confirmed. For instance, suppose that \(o_{ij}^f(t_0) = \top\) and agent \(j\) confirms the formation request by agent \(i\) at time \(t_0 > 0\). The corresponding formation task for agent \(i\) is given by \(\varphi_{ij,t_0}\). Then \(\varphi_{ij,t_0}\) is satisfied at time \(t_f \geq t_0\) if its trajectory \(p_{ij}(t)\) satisfies \(\varphi_{ij,t_0}\) on the basis of the semantics presented in Section II-B. Since all formation tasks are also assumed to be feasible in finite time, the time \(t_f\) is also finite for each formation request.

### D. Discrete Plan Synthesis and Adaptation

In this subsection, we show how to synthesize the initial discrete plan for each agent, based on its own local task, and how to adapt it online upon receiving contingently service and formation requests from neighboring agents.

#### 1) Initial Discrete Plan Synthesis: At time \(t = 0\), we assume that no requests have been sent yet, i.e., \(o_{ij}(0) = \varphi_{ij}(0) = \bot, \forall j \in \mathcal{N}_i\) and \(\forall i \in \mathcal{N}\). As a result, only the static task \(\varphi_i\) of the RTL formula \(\varphi_i\) defined in (7) is initially pursued. Hence, the initial plan synthesis aims at finding a discrete plan that satisfies \(\varphi_i\). Our approach relies on the automaton-based model-checking algorithm [18]. Thus, we
first define a Finite Transition System (FTS) as an abstraction of the agent’s motion among the regions of interest.

**Definition 2.** The motion of agent $i$ is abstracted by an FTS:  
$$T_i \triangleq (\Pi_i, \rightarrow, R, L, \Pi_i, 0, W).$$  
where $\Pi_i \triangleq \{\pi_0\} \cup \Pi$ is the set of states with $\pi_0 \triangleq B(p_i(0), 0)$; and $\rightarrow \triangleq (\Pi \times \Pi) \cup \{\tau_0\} \times \Pi$ denotes the transition relation; $L : \Pi \rightarrow 2^R$ is the labeling function; $\Pi_i, 0 \triangleq \{\pi_0\}$ is the initial state; and $W : \Pi \rightarrow \mathbb{R}^+$ computes the cost of each transition with $W(\pi_m, \pi_n) \triangleq |c_m - c_n|, \forall (\pi_m, \pi_n) \in \rightarrow.$

Notice that only the local propositions of $R$ are allowed in $T_i$ and that the initial state $\pi_0$ is solely determined by the initial position of agent $i$. A path of $T_i$ is given by $\tau = \pi_0^{i_0} \pi_1^{i_1} \cdots$, where $(\pi_0^{i_0}, \pi_1^{i_1}) \in \Pi, k \geq 0$ and it’s associated trace, which corresponds to a discrete-time Boolean signal over $R$, is defined as the sequence of propositions that are true along the path, i.e., $\text{trace}(\tau) = L(\pi_0) L(\pi_1) \cdots$.

On the other hand, $\pi_1^i$ is a general RTL formula over $R$ with a corresponding LTL formula denoted by $[\pi_1^i]$ based on Definition 1. Thus, following the notations defined in Section II-A, we may construct the NBA associated with $\pi_1^i$ and denote it by $B_{s,i} = (Q_{s,i}, 2^R, \delta_{s,i}, Q_{s,0}, F_{s,i})$. Subsequently, we may construct the weighted product automaton as follows:

**Definition 3.** The weighted product Büchi automaton $P_i = T_i \times B_{s,i}$ is defined by $Q'_{s,i} = \Pi_i \times Q_{s,i}$ with $\delta'_{s,i} : Q'_{s,i} \rightarrow 2^{Q'_{s,0}} \times \Pi_i$ and $\forall q \in Q_{s,i}$; $\delta'_{s,i} : Q'_{s,i} \rightarrow 2^{Q'_{s,0}} \times \Pi_i$ and $\forall q \in Q_{s,i}$, $\forall (\delta_{s,i}(q, q_{\pi_0}), \delta_{s,i}(q, q_{\pi_1})) \in \delta'_{s,i}$, where if only if $(\delta_{s,i}(q, q_{\pi_0}), \delta_{s,i}(q, q_{\pi_1})) \in \delta'_{s,i}$. If $\delta_{s,i}(q, q_{\pi_0}) \in \Pi_i$, $\delta_{s,i}(q, q_{\pi_1}) \in \Pi_i$. The set of initial states is $F'_{s,i} = \Pi_i \times F_{s,i}$. The set of accepting states is $W_{s,i} : \delta'_{s,i} \rightarrow \mathbb{R}^+$ with $W_{s,i}(\delta_{s,i}(q, q_{\pi_0}), \delta_{s,i}(q, q_{\pi_1})) = W(\delta_{s,i}(q, q_{\pi_0}), \delta_{s,i}(q, q_{\pi_1})) \in \delta'_{s,i}$.

After $P_i$ is constructed, we search for one of its accepting runs denoted by $R_i$ that has a prefix-suffix structure (i.e., $R_i = R_{i,0} \cdot R_{i,suf}$) and minimizes the total cost defined by $\text{cost}(R_{i,0}) + \text{cost}(R_{i,suf})$, where $\text{cost}(R_{i,0})$ and $\text{cost}(R_{i,suf})$ are simply the accumulated weight of the transitions along the finite sequence of product states in $R_{i,0}$ and $R_{i,suf}$. In this respect, $P_i$ may be viewed as a directed graph with initial and accepting states, a variation of the Dijkstra’s shortest path algorithm can be used to find such an optimal accepting run. The worst-case computational complexity is $|Q'_{s,i}| \cdot \log(|\delta'_{s,i}|)$, where $|Q'_{s,i}|$, $|\delta'_{s,i}|$ are the number of states and transitions. For algorithmic details, we refer the readers to Algorithms 1 and 2 in [26]. Hence denoting by $R_{i,0}$ this optimal accepting run and projecting it back onto $\Pi$, yields the initial plan of agent $i$ as $\tau_{i,0} = R_{i,0}[i]$. Notice that $\tau_{i,0}$ has the prefix-suffix structure and that the trace of $\tau_{i,0}$ satisfies $[\pi_1^i]$ automatically [18]. In this way, the plan $\tau_{i,0}$ obtains the following sequence:  
$$\tau_{i,0}^\omega = \pi_0 \pi_1 \cdots \pi_{i,k} \pi_{i,(k+1)} \cdots \pi_{i,K_i}^\omega,$$  
and its trace given by $\text{trace}(\tau_{i,0}) = L(\pi_0) L(\pi_1) \cdots$, where $\pi_0 \in \Pi_i, \forall i \in 1, 2, \cdots, K_i, \pi_0 \pi_1 \cdots \pi_{i,k}$ is the plan prefix defined as a finite sequence of goal regions to reach; and $\pi_{i,k}, \pi_{i,(k+1)} \cdots \pi_{i,K_i}$ is the plan suffix, which is also finite but should be repeated infinitely often in order to satisfy $[\pi_1^i]$.

If no contingent requests are sent or confirmed by agent $i$ for all $t > 0$, its initial discrete plan $\tau_{i,0}$ should remain unchanged and be executed as follows. Initializing at $p_i(0)$, agent $i$ moves to and enters region $\pi_1$ employing the motion controller described in Section IV-A1. Then, an event that agent $i$ entered region $\pi_{i,1}$ should be detected. Afterwards, it leaves region $\pi_{i,1}$ and moves towards region $\pi_{i,2}$ under the same control scheme. Such procedure is repeated until it reaches $\pi_{i,K_i}$, after which the next goal region is set to $\pi_{i,k}$, and continues to $\pi_{i,k}$, and back to $\pi_{i,k}$ again. In this way, the plan suffix is repeated infinitely often as $t \to \infty$. Thus, there exists an infinite sequence of time instants $0 \leq t^i_1 \leq t^i_2 \leq t^i_3 \cdots$ and $t^i_{k+1} > t^i_k > t^i_{k+1} > t^i_k > 0$, $\forall k \geq 1$, such that $p_i(t) \in [\pi_{i,k}, \forall k \geq 1$, (22)  
 where $\pi_{i,k} \triangleq \pi_{i,k}^\ell$ with $k' \triangleq \text{mod}(k - k_i, K_i - k_i) + k_i, k_k > K_i$ (mod is the modulo operation). Thus the trajectory $p_i(t)$ intersects with the sequence of goal regions by $\tau_{i,0}$.

**Theorem 1.** If no contingent requests are sent or confirmed by agent $i$ such that, $\forall j \in N_i$ and $\forall t \geq 0$, then the resulting trajectory $p_i(t)$ of $\tau_{i,0}$ satisfies $[\pi_1^i]$.  
**Proof.** If $\pi^i_j(t) = \pi^i_j(t) = \perp$, $\forall j \in N_i$ and $\forall t \geq 0$, the service task $\pi^f_1^i$ and formation task $\pi^f_2^i$ defined in (8) and (9) be may be ignored owing to the semantics of the implication operator. Thus, $[\pi^f_1^i]$ is equivalent to $[\pi_1^i]$ and $[\pi^f_2^i]$ is equivalent to $[\pi_1^i]$. Notice also that $[\pi^f_1^i]$ is specified over $R$ and therefore depends solely on the agent’s trajectory. Moreover, we have shown that the trace of $\pi_{i,0}$ satisfies the LTL formula $[\pi^f_1^i]$ as well as that the trajectory $p_i(t)$ satisfies (22). Hence, what we need further to show is that $p_i(t)$ satisfies the RTL formula $[\pi_1^i]$. Let us define $\pi = \text{trace}(\pi_{i,0}),$ which is a discrete-time Boolean signal over $R$ that $(\pi_0, 0) \Rightarrow [\pi_1^i]$.  

- If $(\pi_0, 0) \Rightarrow [\pi_1^i]$ for $R_{i,0}$, i.e., agent $i$ starts from region $\pi_{i,0}$, then (22) guarantees that there exists a time interval $[0, t_1)$ with $t_1 > 0$ such that $p_i(t) \in \pi_{i,t_1} \forall t \in [0, t_1)$. Thus, invoking the RTL semantics we conclude that $p_i(t) \Rightarrow [\pi_1^i]$. Similar arguments may also apply for $[\pi_{i,0}]$ with $R_{i,0}$.  

- If $(\pi_0, 0) \Rightarrow [\pi_{i,0}]$ for $R_{i,0}$, $R_{i,0} \Rightarrow [\pi_{i,0}]$, i.e., agent $i$ stays at region $\pi_{i,0}$, before it moves to region $\pi_{i,t_2}$, then (22) guarantees that there exists a time interval $[0, t_1]$ with $t_1 > 0$ such that $p_i(t) \in \pi_{i,t_1} \forall t \in [0, t_1)$ and a subsequent one $[t_1, t_2)$ with $t_2 > t_1$ such that $p_i(t) \in \pi_{i,t_2} \forall t \in [t_1, t_2)$. Thus, invoking the RTL semantics we also conclude that $p_i(t) \Rightarrow [\pi_{i,0}]$. Similar arguments also apply inductively to other operators like $\perp$, $\Rightarrow$ and $\Rightarrow$. Thus, $(\pi_{i,0})$, we conclude that $p_i(t) \Rightarrow [\pi_1^i]$. The above results are valid only if no contingent requests are exchanged, which however is not the case here. Thus, in

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCNS.2016.2555581, IEEE Transactions on Control of Network Systems

2325-5870 (c) 2015 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.
the sequel we study how service or formation requests should be handled by adapting the local discrete plans.

2) Event-based Discrete Plan Adaptation: Initially, we show how to handle contingent service requests. Based on the previous analysis, an agent may confirm and serve multiple service requests simultaneously. In such case, agent \(i\) needs to satisfy three types of tasks: (i) the static local task; (ii) all the service requests received so far that have not been satisfied; and (iii) the newly-received service request. These tasks need to be treated differently since the first type should be satisfied by the whole trajectory from \(t = 0\); the second type by the trajectory starting from the time the service requests are confirmed; the third type by the future trajectory. Thus, it is important to keep track of how much the local task as well as the past and current service requests have been satisfied. In that respect, we consider three different cases:

**Case I:** Suppose the first service request \(\varphi_{ji,t}^\prime\) received by agent \(i\) at time \(t_f > 0\) was sent by agent \(j\). Agent \(i\) needs to incorporate \(\varphi_{ji,t}^\prime\) into its static task specification \(\varphi_i^s\) and update its current plan \(\pi_{i,0}\) to satisfy this request. As mentioned earlier, the service task specification defined in (7) requires \(\varphi_{ji,t}^\prime\) to be satisfied within a finite time. The associated LTL formula is denoted by \([\varphi_{ji,t}^\prime]\) and \(B[\varphi_{ji,t}^\prime] = (Q, 2^R, \delta, Q_{i,0}, F_i)\) is the corresponding NBA, with similar notation to Section II-A.

Recall that \(B[\varphi_i] = (Q, 2^R, \delta, Q_{i,0}, F_i)\) is the NBA associated with \([\varphi_i]\) and assume that at time \(t_j\) the product state of agent \(i\) in \(P_i^s\) is \(q_i,t_j \in Q_{i,0}\) and the associated Büchi state in \(B[\varphi_i]\) is \(q_{s,t_j} = q_{s,t_j}Q_i.\) Thus, we define the request-prioritized and layered intersection of \([\varphi_{ji,t}^\prime]\) and \([\varphi_i]\) below:

**Definition 4.** The intersection of \(B[\varphi_{ji,t}^\prime]\) and \(B[\varphi_i]\) is an NBA:

\[
A[\varphi_i] = (Q, 2^R, \delta, Q, F_i),
\]

where \(Q = Q_j \times Q_s \times \{1, 2\}; Q_0 = Q_{i,0} \times \{q_{s,t_j}\} \times \{1\}; F = F_j \times F_s \times \{2\}; \delta : Q \times 2^R \rightarrow 2^Q,\) with \((q_j, q_s, c) \in \delta((q_j, q_s, c), l)\) when the following conditions hold:
\(\)ii) \((q_j, q_s, c) \in \delta((q_j, q_s, c), 1)c = 1; or \(q_j \in F_j, c = 1\) and \(c = 2; or \(c = c = 2.\)

The aforementioned definition is different from the conventional way of computing intersections of Büchi automata [18] owing to the fact that \([\varphi_i]\) is a general LTL formula that should be satisfied at \(t = 0\) and \([\varphi_{ji,t}^\prime]\) is an sc-LTL formula that should be satisfied after it is received. In particular, \(A[\varphi_i]\) has two layers and it transits from the first layer to the second only if it reaches \(F_j\) (i.e., \([\varphi_{ji,t}^\prime]\) is satisfied); afterwards, it stays at the second layer in order to satisfy \([\varphi_i]\). In the following lemma, we prove the correctness of Definition 4 by showing that \(A[\varphi_i]\) accepts all words that satisfy both \([\varphi_i]\) and \([\varphi_{ji,t}^\prime]\).

**Lemma 3.** If there exists an \(\omega\)-word \(w\) in \(R^\omega\) such that \(w \models [\varphi_{ji,t}^\prime]\) and \(w \models [\varphi_i]\), then \(\varphi_{ji,t}^\prime\) and \(\varphi_i^s\) are mutually feasible, and \(A[\varphi_i]\) has at least one accepting run.

**Proof.** Owing to the fact that \(w \models [\varphi_{ji,t}^\prime]\), at least one of the resulting runs of \(w\) in \(B[\varphi_{ji,t}^\prime]\) is an accepting run, denoted by \(r_{j} = q_{i,0}q_{j,1} \cdots q_{j,L_k}(q_j,K)\), for which \(q_{i,0}q_{j,1} \cdots q_{j,K}\) is a finite sequence from an initial state \(q_{i,0} \in Q_{i,0}\) to an accepting state \(q_{j,K} \in F_j\) and \((q_j,K)\) is a repetitive suffix over \(q_j,K\).

Such argument holds true since all accepting states of \(B[\varphi_{ji,t}^\prime]\) have a self-cycle that accepts any input alphabet (see Remark 4.31 of [18]). On the other hand, given that \(w \models [\varphi_i]\), then \(w\) results in at least one accepting run of \(B[\varphi_i]\), denoted by \(r_s = q_{s,0}q_{s,1} \cdots (q_{s,L_s}q_{s,K_s+1}q_{s,K_s+2} \cdots q_{s,K_s+K_s})\), which starts from an initial state \(q_{s,0} \in Q_{s,0}\) to an accepting state \(q_{s,K_s+K_s} \in F_s\). We assume \(K_s > K_j\). If not so, the suffix of \(r_s\) can be extended by repeating itself until \(K_f > K_j\). In this sense, we can easily verify that a resulting run of \(w\) in \(A[\varphi_i]\) can be constructed as \(r = (q_j,0,q_s,0,1 \cdots (q_j,K_s),(q_j,K_s+1),(q_j,K_s+2),(q_j,K_s+3,2 \cdots ((q_j,K_s), q_j,K_s+K_s+1,2 \cdots ((q_j,K_s +K_s+1), K_j)^\omega).\)

Since \((q_j,0,q_s,0,1) \in Q_0\) and the state within the repetitive suffix \((q_j,K_s),(q_j,K_s+2), F_s\), then by definition \(r\) is an accepting run of \(A[\varphi_i]\) and hence \(A[\varphi_i]\) accepts \(w\).

From the proof aforementioned, it can be seen that \(r\) reaches the accepting states in \(F_j\) that satisfy \([\varphi_{ji,t}^\prime]\) before it repeats the suffix that satisfies \([\varphi_i]\). Thus, the service request has a higher priority than the static specification and will be satisfied earlier. Moreover, the layered structure of \(A[\varphi_i]\) allows us to track the satisfaction of \([\varphi_i]\) and \([\varphi_{ji,t}^\prime]\) separately.

**Case II:** Suppose that agent \(i\) has confirmed \(m\) service requests \([\varphi_{gi,t_m}^s]\) at time \(t_j\) from agent \(g\), \(\forall g \in N_{i}^s \subseteq N^s_{i}\). Without loss of generality, we can re-order the neighbors in \(N_{i}^s\) by \(\{1, 2, \ldots, m\} \triangleq N_{i}^s\), according to the time their service request was received, e.g., \([\varphi_{gi,t_m}^s]\) denotes the earliest and \([\varphi_{gi,t_m}^s]\) denotes the latest received request. Furthermore, let \(B[\varphi_{gi,t_m}^s] = (Q_g, 2^R, \delta, Q_{g,0}, F_g)\) be the NBA associated with \([\varphi_{gi,t_m}^s]\), \(\forall g \in N_{i}^s\), with similar notations to Section II-A. Additionally, assume that at time \(t_m\) the corresponding product state of agent \(i\) is \(q_{i,t_m} \in Q_{i,0}\). In that respect, its associated Büchi state in \(B[\varphi_{gi,t_m}^s]\) is \(q_{s,t_m} = q_{s,t_m}Q_i\), and the associated Büchi state in each \(B[\varphi_{gi,t_m}^s]\) is \(q_{s,t_m} = q_{s,t_m}Q_i\), \(\forall g \in N_{i}^s\). Thus, we may define the request-prioritized intersection of \([\varphi_{gi,t_m}^s]\), \(\forall g \in N_{i}^s\) and \([\varphi_i]\) as follows:

**Definition 5.** The intersection of \(B[\varphi_{gi,t_m}^s], \forall g \in N_{i}^s\) and \(B[\varphi_i]\) is an NBA defined by:

\[
A[\varphi_i] = (Q, 2^R, \delta, Q_0, F),
\]

where \(Q = Q_1 \times Q_2 \cdots \times Q_m \times Q_s \times \{1, 2, \ldots, m+1\}; Q_0 = \{q_{i,0}\} \times \{q_{g,0}\} \times \cdots \times \{q_{g,m-1}\} \times Q_{s,0} \times \{q_{s,t_m}\} \times \{1\}; F = F_1 \times F_2 \cdots \times F_m \times F_s \times \{m+1\}; \delta : Q \times 2^R \rightarrow 2^Q,\) with \((q_g, q_s, c) \in \delta((q_g, q_s, c), l)\) when the following conditions hold:
\(\)i) \((q_g, q_s, c) \in \delta((q_g, q_s, c), \) \(c = 1; or \(q_g \in F_g, c = 1\) and \(c = 2; or \(c = c = 2.\)

Notice that \(A[\varphi_i]\) has \(m+1\) layers and transits to the \((m+1)\)-th layer only if the set of accepting states \(F_g\) is reached for \(c = 1, 2, \ldots, m\) (i.e., \([\varphi_{gi,t_m}^s]\) is satisfied), and then stays at the \((m+1)\)-th layer in order to satisfy \(B[\varphi_i]\). The definition of \(Q_0\) ensures that the past progress of serving \([\varphi_i]\) and \([\varphi_{gi,t_m}^s]\)

\(g \in N_{i}^s\) is preserved, while the definition of \(\delta\) allows
us to keep track of such progress separately. Clearly, \( A[\varphi_i] \) has \(|Q_1|\cdot|Q_2|\cdots|Q_m|\cdot|Q_s|\cdot(m+1)\) states and maximally \(|\delta_1|\cdot|\delta_2|\cdots|\delta_m|\cdot|\delta_s|\cdot(m+1)\) transitions. Finally, the correctness of the aforementioned definition relies on the following lemma.

**Lemma 4.** If there exists an \( \omega\)-word \( w \in \mathcal{R}^\omega \) such that \( w \models [\varphi_{gi,t_i}] \), \( \forall g \in N_i^\omega \) and \( w \models [\varphi_i] \), then the received service tasks \( \varphi_{gi,t_i} \), \( \forall g \in N_i^\omega \) and \( \varphi_i \) are mutually feasible, and \( A[\varphi_i] \) has at least one accepting run.

**Proof.** Following the same line of proof with Lemma 3, we may construct an accepting run of \( A[\varphi_i] \) by employing the resulting runs of \( w \) over \([\varphi_i]\) and \([\varphi_{gi,t_i}]\), \( \forall g \in N_i^\omega \). Similarly, to the one constructed in Lemma 3, such accepting run reaches an accepting state of \( B[\varphi_{gi,t_i}] \) first, transits to the second layer, reaches an accepting state of \( B[\varphi_{gi,t_i}] \), transits to the third layer and so on. This process is repeated until it reaches the \((m+1)\)th layer and stays there. Afterwards its suffix is repeated infinitely often to satisfy \([\varphi_i]\). Thus, \( A[\varphi_i] \) accepts the common words of \([\varphi_i]\) and all the service requests \([\varphi_{gi,t_i}]\), \( \forall g \in N_i^\omega \). Finally notice that \( A[\varphi_i] \) is updated recursively by Definition 5, whenever agent \( i \) confirms new service requests.

**Case III:** Assume that agent \( i \) has confirmed a service request \( \varphi_{gi,t_i}^s \) from its neighbor \( l \in N_i \) at time \( t^3_i > 0 \) and accomplished it at \( t^2_i > t^1_i \). Afterwards, at time \( t^1_i > t^2_i \), agent \( i \) confirms a new service request \( \varphi_{li,t_i}^s \) from the same neighbor \( l \). In the sequel, we discuss how we should adjust \( A[\varphi_i] \) in order to remove the old request \( \varphi_{li,t_i}^s \) and incorporate the new request \( \varphi_{gi,t_i}^s \). Let us denote by \( N_i^s \) the set of neighbors, whose service requests agent \( i \) has to satisfy as well as by \( A[\varphi_i] \) the associated intersection NBA. Since the service request from agent \( l \) has changed from \( \varphi_{li,t_i}^s \) to \( \varphi_{gi,t_i}^s \), \( A[\varphi_i] \) needs to be updated as follows. First, the service requests need to be reorganized according to the time they were received. Hence, the service request of agent \( l \) should be assigned the index \( m = |N_i^s| \) as it was the latest received one. Subsequently, given the product state \( q'_{gi,t_i}^s \) of agent \( i \) at time \( t^3_i \), the Büchi state associated with each \( [\varphi_{gi,t_i}^s] \) is derived by \( q'_{gi,t_i}^s|Q_i^s, \forall g \in N_i^\omega \), and \( g \neq l \). Thus, \( A[\varphi_i] \) can be recomputed by Definition 5, with the service request of agent \( l \) being moved to the \( m \)-th layer. The rest of the details are similar to **Case II**.

Given the updated intersection automaton \( A[\varphi_i] \), the corresponding product automaton \( \mathcal{P}_i \) should be updated following Definition 3. Thus, we search for an accepting run of the updated \( \mathcal{P}_i \) that minimizes the cost function mentioned in Section IV-D1, of which the computational complexity is derived analogously with the updated \( \mathcal{P}_i \). This accepting run always exists due to the assumption that the local task and the received service tasks are mutually feasible for each agent \( i \in \mathcal{N} \). Subsequently, this accepting run can be projected onto \( \mathcal{I} \), thus yielding the updated discrete plan of agent \( i \). Finally, notice that the local plan should be adapted whenever \( A[\varphi_i] \) is updated in any of the cases above.

In the sequel, we prove the correctness of the proposed discrete plan adaptation approach. In this respect, assume that agent \( i \) receives a service request \( \varphi_{gi,t_i}^s \) from its neighbor \( g \in N_i^\omega \) at time \( t^3_i > 0 \). After that, at time \( t > t^3_i > 0 \), agent \( i \) has crossed the sequence of goal regions \( \pi_{1,i,1} \pi_{2,i,2} \cdots \pi_{K,i,K} \) during \([t^3_i, t]\), which is uniquely determined by its discrete plan \( \tau_i([t^3_i, t]) \), where \( \pi_{i,k} \in \mathcal{I}, \forall i, 1, 2, \ldots, K \). In this way the corresponding trace is defined by \( \text{trace}_i([t^3_i, t]) = L_1(\pi_{i,0}, L_2(\pi_{i,1}), \ldots, L_m(\pi_{i,K})) \). Furthermore, let \( q'_{i,t} \in Q_i^\omega \) be the corresponding product state in \( \mathcal{P}_i \) at time \( t \).

**Theorem 2.** Given that \( \varphi_{gi,t_i}^s, \forall g \in N_i^\omega \) and \( \varphi_i^s \) are mutually feasible, there exists a finite time \( t > t_i \) such that \( p_i([t^3_i, t]) \) satisfies the RTL formula \( \varphi_{gi,t_i}^s \) for each agent \( g \in N_i^\omega \).

**Proof.** Lemma 4 ensures that the intersection automaton \( A[\varphi_i] \) accepts the satisfying words of \([\varphi_{gi,t_i}^s]\). Irrespective of the layer that \([\varphi_{gi,t_i}^s]\) is in \( A[\varphi_i] \), since all service requests are co-safe, there exists a finite time \( t > t_i \) when \( q'_{i,t} \) reaches the accepting state \( F_g \). Furthermore, from the definition of \( A[\varphi_i] \), we know that if \( q_0 \in F_g \) for any \( g \in N_i^\omega \), then the past trace during \([t^3_i, t]\) (i.e., \( \text{trace}_i([t^3_i, t]) \)) results in an accepting run of \( B[\varphi_{gi,t_i}^s] \) from the initial state to the accepting state \( q_0 \). Thus, \( \text{trace}_i([t^3_i, t]) \) satisfies \([\varphi_{gi,t_i}^s]\) finally. It can be easily deduced, following the proof of Theorem 1, that the resulting trajectory \( p_i([t^3_i, t]) \) satisfies the RTL formula \( \varphi_{gi,t_i}^s \).
Figure 2. The hybrid controller module. The arrows indicate the discrete transitions labeled by the associated guards for agent $i \in \mathcal{N}$.

of $B_{ij,t}$ and thus $\text{trace}_i([t_j, t^f])$ satisfies $[\varphi^\pi_{ij,t}]$. Subsequently, it can be shown as in Theorem 1 that the resulting trajectory $p_i([t_g, t^f])$ satisfies the RTL formula $\varphi^\pi_{ij,t}$. At the same time, a confirmation message is sent from agent $i$ to $j$. Secondly, Lemma 2 indicates that the formation control law (13) guarantees that agent $j$ will converge to the desired relative formation with prescribed performance, i.e., the controllable proposition $h_i(t) = \top, \forall t \in [t_j, t^f]$ until a release signal is received at time $t$ and $z_{ij}(t) = \top$.

E. Overall Structure

Now we formalize the overall hybrid control architecture, based on the aforementioned analysis. As depicted in Figure 1, the architecture is organized in four interconnected modules that run concurrently in real time. It includes the communication module from Section IV-C, the discrete planner module from Section IV-D1, the event monitoring module from Section IV-B and the hybrid control module.

Hybrid control module. Agent $i$ is in charge of switching between the navigation and formation control modes. As shown in Figure 2, the hybrid control automaton is defined as a tuple $H_\text{cont} \triangleq (Q, P, \text{Init}, f, D, O, E, G)$, where $Q = \{q_{\text{nav}}, q_{\text{form}}\}$ is the set of discrete states; $P \subseteq \pi_{i0}$ is the continuous state; $\text{Init} = q_{\text{nav}} \times p_i(0)$ is the initial state; $f(q_{\text{nav}}, p_i) = -\nabla q_i \Phi_i(p_i, \pi_{i0})$ is the continuous dynamics (10) within the discrete state $q_{\text{nav}}$, where $\pi_{i0}$ is the goal region imposed by the discrete state $q_{\text{nav}}$; $f(q_{\text{form}}, p_i) = K_{ij} z_{ij} \epsilon_{ij}/\rho_{ij}(t)$ is the continuous dynamics (13) within the discrete state $q_{\text{form}}$, where $z_{ij}, \rho_{ij}(t)$ are the formation request conveyed via the communication module; $D(q_{\text{nav}}) = D(q_{\text{form}}) \subseteq \pi_{i0}$ are the domains of the continuous state; $O = \{o_1, o_2, o_3\}$ is the set of external events, where $o_1$ is “$R_g = \top$”, $o_2$ is “$\alpha_{ij} = \top$”, and $o_3$ is “$\beta_{ij} = \top$”; $E = \{(q_{\text{nav}}, q_{\text{nav}}), (q_{\text{nav}}, q_{\text{form}}), (q_{\text{form}}, q_{\text{nav}})\}$ involves the allowed discrete transition edges: $G(q_{\text{nav}}, q_{\text{nav}}) = o_1$, $G(q_{\text{nav}}, q_{\text{form}}) = o_2$, $G(q_{\text{form}}, q_{\text{nav}}) = o_3$ indicate the guards that should hold over each discrete transition.

Theorem 4. Given the hybrid control architecture above and that all static tasks and service/formation requests are mutually feasible, then each local task $\varphi_i$ is satisfied by the trajectory $p_i(t)$ of agent $i$ as $t \to \infty, \forall i \in \mathcal{N}$.

Proof. The local task $\varphi_i$ consists of three parts: $\varphi^\pi_i$, $\varphi^\sigma_i$ and $\varphi^\pi_i$. Firstly, Theorem 1 guarantees that if no service or formation requests are confirmed by agent $i$, then its trajectory will satisfy the local task $\varphi^\pi_i$. Subsequently, since $\varphi^\pi_i$, $\varphi^\sigma_i$ and $\varphi^\pi_i$ are assumed to be mutually feasible, we show that they are all fulfilled when agent $i$ exchanges requests with its neighbors. In case agent $i$ confirms a service request $\varphi^\sigma_{ji,t}$ from agent $g$, then Theorem 2 ensures that there exists a finite time $t^f_i > t_g$ such that the trajectory $p_i([t_g, t^f_i])$ satisfies $\varphi^\sigma_{ji,t}$. Moreover, the same holds for all service requests confirmed by agent $i$, which implies that $\varphi^\sigma_i$ is fulfilled. Alternatively, if a formation request $\varphi^\pi_{ji,t}$ is confirmed, then Theorem 3 ensures that there exists a finite time $t^f_i > t_g$ such that the trajectory $p_i([t_g, t^f_i])$ of agent $g$ fulfills the formation task $\varphi^\pi_{ji,t}$ in finite time $t > t_g$ and meanwhile the formation between agents $i$ and $g$ satisfies the desired performance specifications during $[t_g, t^f_i]$, i.e., the controllable propositions $h_i(t) = \top, \forall t \in [t_j, t^f]$ and $\forall j \in \mathcal{N}_i$. Furthermore, since the same holds for all formation requests confirmed by agent $i$, it implies that $\varphi^\pi_i$ is fulfilled. As a consequence, all three parts of $\varphi_i$ are fulfilled and thus $\varphi_i$ is satisfied for each agent $i \in \mathcal{N}$.

Remark 2. Notice that this hybrid controller will not exhibit Zeno behavior since agent $i$ will stay within the region for at least the dwell time $\delta$ defined in Section IV-B and a formation is requested only after the current formation is finished.

V. CASE STUDY

In this section, we present a simulated paradigm of four autonomous robots with heterogeneous capabilities. The proposed algorithms were implemented in Python 2.7 and all simulations were carried out on a desktop computer (3.06 GHz Duo CPU with 8GB of RAM).

A. Workspace and Task Description

The area of size 4m x 4m is shown in Figure 3, within which there are seven cyclic regions $\pi_0, \pi_1, \cdots, \pi_6$ of interest.
This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCNS.2016.2555581, IEEE Transactions on Control of Network Systems

Agents is given by \( \tau_4 = \pi_6(\pi_4 \pi_3)^s \). It can be easily verified that all satisfy the respective local tasks. After the system initialized, the predefined formation and service tasks were performed by the agents as shown in the left plot of Figure 4. Snapshots of the agents’ trajectories at key time instants are shown in Figure 3. In particular, at \( t = 0.1s \), \( R_3 \) receives \( R_1 \)'s formation request and executes it until \( t = 5.5s \), when \( R_4 \) finishes the formation task \( \Omega R_3 \) by reaching \( \pi_3 \). The evolution of the formation error along with the corresponding performance function is shown in the right plot of Figure 4. At \( 0.6s \), \( R_2 \) confirms the service request \( \Omega R_6 \) from \( R_1 \) and changes its local plan to visit \( \pi_1 \) first, which is done at \( 4.2s \). At the same time, \( R_1 \) confirms service request \( \Omega (R_1 \wedge R_2) \) from \( R_3 \) and changes its local plan to visit \( \pi_1 \) and then \( \pi_2 \), which is done at \( 7.1s \). During \([7.7s, 9.4s]\), \( R_4 \) carries out a relative formation task with \( R_3 \) until \( R_4 \) finishes the formation task \( \Omega R_3 \) by reaching \( \pi_3 \). Then \( R_3 \) confirms the service request \( \Omega (R_2 \wedge R_3) \) from \( R_1 \) at \( 5.5s \). This service is accomplished at \( 14.8s \) by \( R_3 \)'s detour to \( \pi_2 \) first and \( \pi_3 \) afterwards. Finally, \( R_1 \) establishes a relative formation with \( R_3 \) at \( 14.8s \) until agent \( R_3 \) finishes the formation task \( \Omega R_3 \) by reaching \( \pi_3 \) at \( 17.1s \). By then, all predefined contingent tasks were fulfilled and no further contingent service or formation tasks will be exchanged. Subsequently, each agent continues executing its local plan to fulfill its local task. Note that since some agents have an infinite plan, we simulate the system until \( 20s \).

VI. CONCLUSIONS

We presented a hybrid control strategy for multi-agent systems with contingent temporal tasks and prescribed formation constraints that fulfill all local high-level temporal tasks and contingent service/formation requests. Future research efforts will address physically interacting cooperative tasks and more complex agent dynamics with non-zero agent volume.

REFERENCES

Charalampos P. Bechlioulis was born in Arta, Greece, in 1983. He is currently a postdoctoral researcher in the Control Systems Laboratory at the School of Mechanical Engineering of the National Technical University of Athens. He received a diploma in electrical and computer engineering in 2006 (first in his class), a bachelor of science in mathematics in 2011 (second in his class) and a Ph.D. in Electrical Engineering in 2014, all from the Aristotle University of Thessaloniki, Thessaloniki, Greece. His research interests include nonlinear control with prescribed performance, system identification, control of robotic vehicles, multi-agent systems and object grasping. He has authored more than 55 papers in scientific journals, conference proceedings and book chapters.

Kostas J. Kyriakopoulos was born in Athens, Greece (1962). He received the Mechanical Eng. Dipl., with Honours, from the National Technical University of Athens (NTUA), Greece in 1985 and M.S & Ph.D. degrees in Computer & Systems Eng. at Rensselaer Polytechnic Institute (RPI), Troy, NY, in 1987 and 1991, respectively. Between 1988-91 he did research at the NASA Cntr for Intelligent Robotic Systems for Space Exploration while between 1991-93 he was a Research Assistant Professor at the Electrical, Computer and Systems Eng. Dept. of RPI and the New York State Cntr for Advanced Technology in Automation & Robotics. Since December 1994 he has been with Mechanical Eng. Dept at NTUA where he serves as a Professor and Director of the Graduate Program on Automation Systems. His research interests are in the areas of: (i) Nonlinear Control applications to Sensor Based Motion Planning & Control of Robotic multi-agent systems: Manipulators & Vehicles (Mobile, Marine, Aerial) and (ii) Neuro-Robotics.

Dimos V. Dimarogonas was born in Athens, Greece, in 1978. He received the Diploma in Electrical and Computer Engineering in 2001 and the Ph.D. in Mechanical Engineering in 2007, both from the National Technical University of Athens (NTUA), Greece. Between May 2007 and February 2009, he was a Postdoctoral Researcher at the Automatic Control Laboratory, School of Electrical Engineering, ACCESS Linnaeus Center, KTH Royal Institute of Technology, Stockholm, Sweden. Between February 2009 and March 2010, he was a Postdoctoral Associate at the Laboratory for Information and Decision Systems (LIDS) at the Massachusetts Institute of Technology (MIT), Boston, MA, USA. He is currently an Associate Professor at the Automatic Control Laboratory, ACCESS Linnaeus Center, KTH Royal Institute of Technology, Stockholm, Sweden. His current research interests include Multi-Agent Systems, Hybrid Systems and Control, Robot Navigation and Networked Control. He was awarded a Docent in Automatic Control from KTH in 2012. He serves in the Editorial Board of Automatica, the IEEE Transactions on Automation Science and Engineering and the IET Control Theory and Applications and is a member of IEEE and the Technical Chamber of Greece. He received an ERC Starting Grant from the European Commission for the proposal BUCOPHYS in 2014 and was awarded a Wallenberg Academy Fellow grant in 2015.