Noname manuscript No.
(will be inserted by the editor)

Resource-aware networked control systems under temporal logic specifications

Kazumune Hashimoto · Dimos V. Dimarogonas

Received: date / Accepted: date

Abstract Temporal logics for control of dynamical systems have the potential to automatically synthesize controllers under complex goals expressed by temporal logic formulas. In this paper, we are interested in the situation, where a controller system that implements high and low level controllers is connected to a plant over a *communication network*. In such control architecture, it is known that the limited nature of computation and communication resources should be explicitly taken into account. In view of this, we jointly provide control and communication strategies, such that the resulting state trajectories satisfy the desired temporal logic formula, while at the same time the average communication rate is below a certain threshold. The proposed strategies are illustrated through numerical simulation examples.

Keywords Event and Self-Triggered Control · Temporal Logic Control

1 Introduction

Temporal logic motion/task planning and control of dynamical systems have been receiving an increased attention in recent years (Kress-Gazit, Lahijanian, and Raman 2018; Belta et al. 2007). In contrast to the standard control tasks

This work is supported by Knut och Alice Wallenberg foundation (KAW), and the H2020 ERC Starting Grant BUCOPHSYS.

K. Hashimoto
Graduate School of Engineering Science, Osaka University, Osaka, Japan
Tel.: +819080658331
E-mail: kazumune.hashimoto@z5.keio.jp

D. V. Dimarogonas
School of Electrical Engineering, KTH Royal Institute of Technology, 10044 Stockholm, Sweden
E-mail: dimos@ee.kth.se

that are formulated by well-known stability concepts or point-to-point navigations, temporal logics allow us to automatically synthesize controllers for more complicated specifications involving *temporal* constraints, such as "Survey the region A, B, C, D *infinitely often*, while making sure that the region E is *always* avoided *until* C is visited". The availability of treating such temporal constraints has led to a wide variety of applications, including cooperative task planning of multi-robot systems (Guo, Johansson, and Dimarogonas 2013; Guo and Dimarogonas 2015; Tumova and Dimarogonas 2016; Karimadini and Lin 2011), manufacturing systems (Heddy et al. 2015; Antoniotti, Jafari, and Mishra 1995; He et al. 2015; Morel, Petin, and Lamboley 2001), robot manipulation (Chinchali et al. 2012; Verginis and Dimarogonas 2017), motion planning of dynamic robots (Filippidis et al. 2016; Wongpiromsarn, Topcu, and Murray 2012; Kloetzer and Belta 2008; Livingston and Murray 2013; Fainekos et al. 2009; Kress-Gazit, Fainekos, and Pappas 2009; Gol, Lazar, and Belta 2015; Kress-Gazit, Fainekos, and Pappas 2007), to name a few. In the temporal logic control framework, such temporal tasks are generally expressed by Linear Temporal Logic (LTL) or Computational Tree Logic (CTL) formulas. The basic control synthesis algorithm is given in a hierarchical manner, as briefly described below. First, we obtain a finite transition system that consists of symbolic states and corresponding transitions, which represents an abstracted behavior of the control system. The transition system may be obtained by decomposing the state-space into a finite number of polytopes, and the reachability for each pair of polytopes among them is analysed (Kloetzer and Belta 2008; Fainekos et al. 2009; Coogan et al. 2016). Once the transition system is obtained, a *high level controller* finds an accepting run to satisfy the desired specifications through an implementation of, e.g., automata-based model checking algorithms (Baier and Katoen 2008). If such accepting run is found, a *low level controller* implements a feedback control strategy, such that the generated state trajectory satisfies the desired specification.

In this paper, we are interested in the situation, where the plant aims at achieving desired goals expressed by temporal logic formulas, but the high and low level controllers need to be implemented over a *communication network*. In general, control systems whose plants and controllers are interacted over the communication network are referred to as *Networked Control Systems* (NCSs) (Hespanha, Naghshtabrizi, and Xu 2007). Introducing the NCSs in temporal logics are beneficial, especially when the plant has a limited capacity of memory and computational power, so that it needs to rely on the network to implement both high and low level controllers. Moreover, such situation can be met in emerging system architectures that are of great importance in current and future control technologies. Consider, for example, *cloud robotics* (Kehoe et al. 2015), where autonomous robots such as manufacturing robots or UAVs interact with the clouds for supporting various tasks involving their decision making and learning. In temporal logics, the temporal specification may be obtained by interacting with collaborators such as humans, robots, etc., as well as the remote operators that administrate the control system. The cloud computing can be responsible for implementing the high level controller to obtain robot motion planning, and the low level controller to compute and transmit control actions to operate the robot.

In NCSs, the increased popularity of integrating the communication network has brought new control design and implementation challenges. For example, network delays and packet losses are typically present while transmitting control signals or sensor data over a communication channel. In view of this, various results have been proposed to analyze the relation among network uncertainties, control performance, and stability (see e.g., Zhang, Branicky, and Phillips 2001; Zhang et al. 2005). Moreover, some approaches to obtain symbolic models for NCSs under shared communication resources, communication delays or packet dropouts have been provided in recent years (see, e.g., Borri, Pola, and Benedetto 2018; Pola, Pepe, and Benedetto 2018; Mazo, Davitian, and Tabuada 2010).

Another main challenge lies in the fact that NCSs are subject to a limited range of *computation and communication resources*, which will be the main focus of this paper. For example, in sensor networks, sensor and relay nodes are typically battery driven and are equipped with a frugal battery capacity. Therefore, designing appropriate controllers to save the energy consumption is a crucial problem that needs to be solved. Moreover, NCSs are typically dealing with resource-limited embedded micro processors, which means that the number of tasks that can be executed in real-time is limited. Hence, reducing the amount of communication tasks allows to assign other network tasks that are necessary to be executed in real time. Based on the above motivations, two resource-aware control schemes have been proposed in recent years, namely, *event-triggered control* and *self-triggered control* (Heemels, Johansson, and Tabuada 2012). In both strategies, the objective is to reduce the communication frequency between the plant and the controller. Specifically, sensor data and control inputs are exchanged over a communication network *only when* they are needed based on the prescribed control performances, such as $\mathcal{L}_2/\mathcal{L}_\infty$-gain stability (Wang and Lemmon 2009), so that communication is given aperiodically. Such aperiodic scheme can potentially lead to energy savings of battery powered devices, since the communication over the network is known to be one of the main energy consumers.

The main contribution of this paper is to provide novel control and communication strategies under temporal logic specifications, which is in particular inspired by the resource-aware control paradigm as described above. Specifically, given that the plant is described by a linear discrete-time system with additive disturbances, we jointly design control and communication strategies, such that: (i) the resulting state trajectory satisfies the LTL specification, and (ii) the *average communication rate*, which represents how often control packets are transmitted over the communication network, is below a given threshold. The latter requirement is useful in practical implementation, since it allows us to examine how much energy consumption is necessary for satisfying the LTL specifications. Moreover, it allows to examine the possibility to assign additional network tasks if necessary to be executed in real time. To achieve the goal of this paper, our control and communication synthesis framework has in particular the following technical contributions. First, we provide a reachability analysis based on Rapidly-Exploring Random Trees (RRT), which is adapted such that all state trajectories satisfy the requirements to achieve reachability (in the sense of satisfaction relation of the "trace" definition of the LTL formula), as well as that the corresponding communication strategy can be designed. To design the communication strategy, we revise the original RRT as follows: first, for each iterative step we draw a random sample in the free state-space and pick the closest node to the sample in the tree. Then, we pick the optimal pair of the control input and the inter-communication time steps, such that the corresponding state becomes the closest to the sample. As we will see later, such optimal node will be found by considering local reachability from the dynamics of the plant with different selections of inter-communication time steps. To deal with uncertainties due to initial states and disturbances, we also expand a tree of uncertainties together with states, which will be incorporated to check feasibility (collision avoidance to the obstacles). The approach proceeds by constructing a transition system based on the result of reachability analysis, and implementing both high and low level controllers. In particular, in the high-level controller part we generate a plan such that the LTL formula can be achieved and the average communication rate is below a given threshold. As will be seen later, this is achieved by assigning suitable (communication) costs for each edge of the transition system, and finding an accepting run such that certain conditions on the assigned weights are satisfied.

(Related works) : So far, a wide variety of controller synthesis algorithms under temporal logic specifications have been proposed; in particular, our approach is related to Livingston, Wolff, and Murray 2015; Bhatia, Kavraki, and Vardi 2010; Karaman and Frazzoli 2011a; Karaman and Frazzoli 2012, since the reachabiliy analysis is based on sampling-based techniques. For example, Livingston, Wolff, and Murray 2015 proposed probabilistic sampling schemes to generate optimal state trajectories subject to LTL specifications. Moreover, Karaman and Frazzoli 2012 proposed sampling-based algorithms under deterministic μ-calculus specifications for nonlinear control systems with the integration of RRT* algorithms. When limiting our scope to the reachability analysis, various sampling-based algorithms have been proposed; in particular, the approach is related to B. Luders and How 2010; Pepy, Kieffer, and Walter 2009; Tedrake et al. 2010; Majumdar and Tedrake 2017; Agha-mohammadi, Chakravorty, and Amato 2014, in the sense that they handle uncertainties of the plant dynamics for the reachability analysis. For example, Tedrake et al. 2010 proposed a sampling-based motion planning scheme by designing both a local LQR feedback control law and level sets that guarantee the reachability towards the goal region. The main novelty of our approach with respect to the above previous results is that, we design not only a control strategy but also a *communication* strategy during the reachability analysis. As previously mentioned, this is achieved by extending the RRT algorithm, in which a tree of states is expanded with different selections of inter-communication time steps.

Moreover, the approach is also novel in the sense that we provide a framework to design a communication strategy such that the average communication rate is guaranteed to be below a threshold.

The approach presented in this paper builds upon our previous work in Hashimoto, Adachi, and Dimarogonas 2018. The approach presented in this paper is advantageous over this previous result in the following sense. In the previous approach, the communication strategy was designed in the low-level controller, which dynamically assigns the communication time steps during the online implementation. However, such approach makes it difficult to analyze the communication load; since the number of communication time steps is unknown apriori, it is also unknown how often control packets should be transmitted to satisfy the LTL specifications. On the other hand, the proposed approach in this paper preliminarily assigns the communication time steps during the reachability analysis (before the implementation). Thus, we can evaluate how much communication frequency is needed to satisfy the LTL specifications. As will be seen later, this allows us to design a communication strategy such that the average communication rate is below a threshold during high level controller implementation. The second drawback of the previous result is that the low-level controller needs to solve a finite horizon optimization (feasibility) problem and transmit a (potentially large) sequence of control inputs per each communication time. This formulation may not be suitable, since network delays as well as network congestions may arise due to the load of computing and transmitting control inputs to operate the plant. On the other hand, the proposed approach in this paper considers a simple stabilizing control law that does not rely on any optimization problem, and only one control sample is necessary to be transmitted for each communication time step.

The rest of the paper is organized as follows. We describe some preliminaries and the problem formulation in Section 2 and 3, respectively. In Section 4, reachability analysis and an algorithm to obtain a finite transition system are given. In Section 5, we propose the implementation algorithm involving both high and low level strategies. In Section 6, a simulation example is given to validate the effectiveness of the proposed approach. We finally conclude in Section 7.

Notations. Let $\mathbb{R}$, $\mathbb{R}_+$, $\mathbb{N}$, $\mathbb{N}_+$, $\mathbb{N}_{a:b}$ be the non-negative real, positive real, non-negative integers, positive integers, and the set of integers in the interval $[a, b]$, respectively. For given two sets $\mathcal{X} \subset \mathbb{R}^n$, $\mathcal{Y} \subset \mathbb{R}^n$, denote by $\mathcal{X} \oplus \mathcal{Y}$ the Minkowski sum: $\mathcal{X} \oplus \mathcal{Y} = \{z \in \mathbb{R}^n \mid \exists x \in \mathcal{X}, y \in \mathcal{Y} : z = x + y\}$. Given $A \in \mathbb{R}^{n\times n}$ and $\mathcal{X} \subset \mathbb{R}^n$, let $A\mathcal{X} = \{Ax \in \mathbb{R}^n \mid x \in \mathcal{X}\}$. Given $x \in \mathbb{R}^n$, we denote by $\|x\|$ and $\|x\|_\infty$ the Euclidean norm and the infinity norm of x, respectively. For simplicity, we denote the collection of N sets $\mathcal{X}_1, \ldots, \mathcal{X}_N \subset \mathbb{R}^n$ as $\mathcal{X}_{1:N} = \{\mathcal{X}_1, \ldots, \mathcal{X}_N\}$.

2 Preliminaries

In this section, we review several useful notions and established results for transition systems, Linear Temporal Logic (LTL) formula, and Büchi Automaton.

(*Transition System*): A transition system is a tuple $\mathcal{T} = (S, s_{\rm init}, \delta, \Pi, g, \mathcal{W})$, where S is a set of states, $s_{\rm init} \in S$ is an initial state, $\delta \subseteq S \times S$ is a transition relation, Π is a set of atomic propositions, $g : S \to \Pi$ is a labeling function, and $\mathcal{W} : \delta \to \mathbb{R}$ is a weight function. A *run* of $\mathcal{T}$ is defined as an infinite sequence of states $s_{\rm seq} = s_0 s_1 \cdots$ such that $s_0 = s_{\rm init}$, $(s_i, s_{i+1}) \in \delta$, $\forall i \in \mathbb{N}$. We denote by $\mathrm{Run}(\mathcal{T})$ the set of all runs of $\mathcal{T}$: $\mathrm{Run}(\mathcal{T}) = \{s_{\rm seq} \mid s_{\rm seq} \text{ is a run of } \mathcal{T}\}$. A trace of a run $s_{\rm seq} = s_0 s_1 s_2 \cdots$ is given by $\mathrm{trace}(s_{\rm seq}) = g(s_0)g(s_1)g(s_2)\cdots$. $\mathrm{Trace}(\mathcal{T})$ is defined as a set of all traces generated by the runs of $\mathcal{T}$; $\mathrm{Trace}(\mathcal{T}) = \{\mathrm{trace}(s_{\rm seq}) \mid s_{\rm seq} \in \mathrm{Run}(\mathcal{T})\}$.

(*Linear Temporal Logic Formula*): Throughout the paper, we consider that the specification ϕ is described by an LTL formula (see e.g., Chapter 5 in Baier and Katoen 2008) over a set of atomic propositions Π. LTL formulas are constructed according to the following grammars:

$$\phi ::= \mathrm{true} \mid \pi \mid \phi_1 \wedge \phi_2 \mid \neg\phi \mid \phi_1 \mathsf{U} \phi_2, \tag{1}$$

where $\pi \in \Pi$ is the atomic proposition, $\wedge$ (*conjunction*), $\neg$ (*negation*) are Boolean connectives, and U (*until*) is the temporal operator. Other useful temporal operators such as $\square$ (*always*), $\lozenge$ (*eventually*), R (*release*) can be expressed by combining the operators in (1) and how they are derived is omitted for brevity. Note that in contrast to standard LTL formulas, the operator $\bigcirc$ (*next*) is not defined in (1) and will not be used to express the specification ϕ in this paper. Such formulas are often called *LTL*$_X$ *formulas* (see, e.g., Kloetzer and Belta 2008). The semantics of LTL formula is inductively defined over an infinite sequence of sets of atomic propositions $\pi_{\rm seq} = \pi_0\pi_1 \cdots \in (2^{\Pi})^{\omega}$. Roughly speaking, an atomic proposition $\pi \in \Pi$ is satisfied if π holds true at π_0. The formula $\phi_1 \wedge \phi_2$ is satisfied if both ϕ_1 and ϕ_2 hold true. The formula $\phi_1 \mathsf{U} \phi_2$ is satisfied if ϕ_1 is satisfied until ϕ_2 is satisfied. For a given $\pi_{\rm seq} = \pi_0\pi_1 \cdots \in (2^{\Pi})^{\omega}$ and an LTL formula ϕ, we denote by $\pi_{\rm seq} \models \phi$ if $\pi_{\rm seq}$ satisfies the formula ϕ. We further denote by $\mathrm{Words}(\phi)$ the set of all words that satisfy the formula ϕ. That is,

$$\mathrm{Words}(\phi) = \{\pi_{\rm seq} \in (2^{\Pi})^{\omega} \mid \pi_{\rm seq} \models \phi\}. \tag{2}$$

(*Büchi Automaton*): *A Büchi Automaton* is a tuple $\mathcal{B} = (Q, Q_{\rm init}, \Sigma, \delta_B, F)$, where Q is a set of states; $Q_{\rm init} \subseteq Q$ is a set of initial states; Σ is an input alphabet; $\delta_B \subseteq Q \times \Sigma \times Q$ is a non-deterministic transition relation; F is an acceptance set. A *word* is defined as $\sigma_{\rm seq} = \sigma_0\sigma_1 \cdots$, with $\sigma_i \in \Sigma$, $\forall i \in \mathbb{N}$. A *run* of $\mathcal{B}$ over a word $\sigma_{\rm seq}$ is defined as an infinite sequence of states $q_{\rm seq} = q_0 q_1 \cdots$ generated such that $q_0 \in Q_{\rm init}$ and $(q_i, \sigma_i, q_{i+1}) \in \delta_B$, $\forall i \in \mathbb{N}$. A run $q_{\rm seq}$ is called *accepting* if there exists a word $\sigma_{\rm seq}$ such that F is intersected

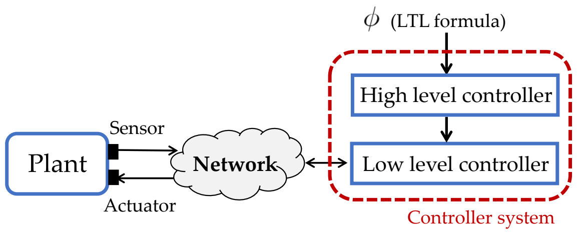


Fig. 1 Networked Control System.

infinitely often. A word $\sigma_{\rm seq}$ is *accepted* by $\mathcal{B}$ if there exists an accepting run for $\sigma_{\rm seq}$. A *language* $\text{Lang}(\mathcal{B})$ is defined as a set of all words accepted by $\mathcal{B}$. It is known that any LTL formula ϕ can be translated into the corresponding Büchi Automaton $\mathcal{B}$ with $\Sigma = 2^{\Pi}$, such that $\text{Words}(\phi) = \text{Lang}(\mathcal{B})$. Many off-the-shelf tools for this translation algorithm have been proposed and can be found online, e.g., LTL2BA (Oddoux and Gastin 2001).

3 Problem formulation

3.1 Plant dynamics

We consider a Networked Control System illustrated in Fig. 1, where the plant and the controller are connected over a communication network. The controller system is responsible for both implementing a *high level controller* that generates symbolic and communication plans for a given LTL specification ϕ, and a *low level controller* that generates (low level) control inputs to operate the plant. How these controllers are designed will be formally given later in this paper. Throughout the paper, we assume that the communication network is ideal; it induces neither packet dropouts nor any network delays. The dynamics of the plant is given by the following Linear-Time-Invariant (LTI) systems:

$$x_{k+1} = Ax_k + Bu_k + w_k, \tag{3}$$

for $k \in \mathbb{N}$, where $x_k \in \mathbb{R}^n$ is the state, $u_k \in \mathbb{R}^m$ is the control input, and $w_k \in \mathbb{R}^n$ is the additive disturbance. We assume that the pair (A, B) is controllable, and that the disturbance is constrained as $w_k \in \mathcal{W}$, $\forall k \in \mathbb{N}$, where $\mathcal{W}$ is a given polytopic set. Regarding the state, we assume $x_k \in \mathcal{X}$, $\forall k \in \mathbb{N}$, where $\mathcal{X}$ is a polygonal set that can be either a convex or non-convex region. The set $\mathcal{X}$ represents the *free space*, in which the state is allowed to move. Inside $\mathcal{X}$, we assume that there exist in total N_I number of polytopic regions $\mathcal{R}_1, \mathcal{R}_2, \ldots, \mathcal{R}_{N_I} \subset \mathcal{X}$, which represent the *regions of interest* in $\mathcal{X}$. These regions are assumed to be disjoint, i.e., $\mathcal{R}_i \cap \mathcal{R}_j = \emptyset$, $\forall i, j \in \mathbb{N}_{1:N_I}$ with $i \neq j$. Moreover, let $x_{{\rm cent},i} \in \mathcal{R}_i$, $i \in \mathbb{N}_{1:N_I}$ denote the Chebyshev center (Borrelli, Bemporad, and Morari 2017) of the polytope $\mathcal{R}_i$. The Chebyshev center is the center of the maximum ball that is included in $\mathcal{R}_i$ and is obtained by solving a linear program (for details, see Section 5.4.5 in Borrelli, Bemporad, and Morari 2017). In addition, the initial state is assumed to be given and is inside one of the regions of interest, i.e., $x_0 \in \mathcal{R}_{\rm init}$ where $\mathcal{R}_{\rm init} \in \mathcal{R}_{1:N_I}$.

Let π_i, $i \in \mathbb{N}_{1:N_I}$ be the atomic proposition assigned to the region $\mathcal{R}_i$. Namely, π_i holds true if and only if $x \in \mathcal{R}_i$. Also, denote by π_0 an atomic proposition associated to the regions of non-interest, i.e., π_0 holds true if and only if $x \in \mathcal{X} \backslash (\cup_{i=1}^{N_I} \mathcal{R}_i)$. The atomic proposition π_0 represents a dummy symbol, which will not be used to describe the specification. Let $\mathit{\Pi} = \{\pi_1, \pi_2, \ldots, \pi_{N_I}\}$ and $h_X : \mathbb{R}^n \to \mathit{\Pi}$ be the mapping from the state to the corresponding atomic proposition, i.e.,

$$h_X(x) = \begin{cases} \pi_i, & \text{if} \quad x \in \mathcal{R}_i, \quad i \in \mathbb{N}_{1:N_I}, & (4) \\ \pi_0, & \text{if} \quad x \in \mathcal{X}_{\backslash \mathcal{R}}, & (5) \end{cases}$$

where $\mathcal{X}_{\backslash \mathcal{R}} = \mathcal{X} \backslash (\cup_{i=1}^{N_I} \mathcal{R}_i)$.

3.2 Satisfaction relation over the state trajectory

Denote by $\mathbf{x} = x_0 x_1 x_2 \cdots$ a state trajectory in accordance to (3), with $x_k \in \mathcal{X}$, $u_k \in \mathbb{R}^m$, $w_k \in \mathcal{W}$, $\forall k \in \mathbb{N}$. We next define the satisfaction relation of the formula ϕ by the state trajectory $\mathbf{x}$. Let us first define the trajectory of interest as follows:

Definition 1 Given a state trajectory $\mathbf{x} = x_0 x_1 x_2 \cdots$, the *trajectory of interest* $\mathbf{x}_I$ corresponding to $\mathbf{x}$ is defined as $\mathbf{x}_I = x_{\ell_0} x_{\ell_1} x_{\ell_2} \cdots$ with $\ell_0 = 0$, $\ell_j < \ell_{j+1}$, $\forall j \in \mathbb{N}$, such that: $h_X(x_{\ell_j}) \in \mathit{\Pi}$, $\forall j \in \mathbb{N}$, and $h_X(x_k) = \pi_0$, $\forall k \in (\ell_j, \ell_{j+1})$, $\forall j \in \mathbb{N}$. □

Stated in words, the trajectory of interest is given by eliminating all states that belong to the regions of non-interest. The trace of the state trajectory is generated based on the trajectory of interest as defined next:

Definition 2 Given a state trajectory $\mathbf{x} = x_0 x_1 \cdots$, the *trace* of $\mathbf{x}$ is given by $\text{trace}(\mathbf{x}) = \rho_0 \rho_1 \cdots$, which is generated over the corresponding trajectory of interest $\mathbf{x}_I = x_{\ell_0} x_{\ell_1} x_{\ell_2} \cdots$, satisfying the following rules for all $L \in \mathbb{N}$, $i \in \mathbb{N}$:

(i) $\rho_0 = h_X(x_{\ell_0})$;
(ii) If $\rho_L = h_X(x_{\ell_i})$ and there exists $j > i$ such that $h_X(x_{\ell_i}) = h_X(x_{\ell_{i+1}}) = \cdots = h_X(x_{\ell_j})$, $h_X(x_{\ell_j}) \neq h_X(x_{\ell_{j+1}})$, then $\rho_{L+1} = h_X(x_{\ell_{j+1}})$;
(iii) If $\rho_L = h_X(x_{\ell_i})$, and $h_X(x_{\ell_j}) = h_X(x_{\ell_i})$, $\forall j \geq i$, then $\rho_m = \rho_L$, $\forall m \geq L$. □

For example, assume that $x_k \in \mathcal{R}_1$ for $k = 0, 1, 2$, $x_k \in \mathcal{X}_{\backslash \mathcal{R}}$ for $k = 4, 5$ and $x_k \in \mathcal{R}_2$ for $k = 6, 7, 8 \cdots$. This means that the state initially starts from $\mathcal{R}_1$, leave $\mathcal{R}_1$ for entering $\mathcal{R}_2$, and remains there for all the time afterwards. The trajectory of interest is given by $x_0 x_1 x_2 x_6 x_7 x_8 \cdots$. The trace of the trajectory according to Definition 2 is $\rho = \rho_0 \rho_1 \rho_2 \cdots$ with $\rho_0 = h_X(x_0) = \pi_1$ and $\rho_L = \pi_2$, $\forall L \geq 1$.

Definition 3 *Given a state trajectory $\mathbf{x} = x_0x_1x_2\cdots$, we say that $\mathbf{x}$ satisfies the formula ϕ if and only if the corresponding trace according to Definition 2 satisfies ϕ, i.e., $\pi_{\rm seq} = {\rm trace}(\mathbf{x}) \models \phi$.*

3.3 Communication strategy

To satisfy the formula ϕ, the plant interacts with the low level controller over the communication network for obtaining control inputs in real time. To indicate the communication times, let $k_m, m \in \mathbb{N}$ with $k_{m+1} > k_m, \forall m \in \mathbb{N}$ be the communication time steps between the plant and the controller. That is, for each k_m, $m \in \mathbb{N}$ the plant transmits the current state information x_{k_m} to the controller, and the controller computes a suitable control input to be applied and transmit it back to the plant. We assume that the control input is given in a *sample-and-hold implementation*, i.e., $u_{k_m+\ell} = u_{k_m}$, $\forall \ell \in \mathbb{N}_{1:k_{m+1}-k_m-1}$, $\forall m \in \mathbb{N}$. In what follows, we introduce the notion of *average communication rate*:

Definition 4 Given k_m, $m \in \mathbb{N}$, the *average communication rate* $\rho_{\rm ave} \in [0, 1]$ is given by

$$\rho_{\rm ave} = \lim_{m\to\infty} \frac{m}{k_m} \tag{6}$$

□

Intuitively, the average communication rate is an asymptotic ratio between the number of communication time steps and the time steps. As we will see below, the communication time steps $k_m, m \in \mathbb{N}$ are designed, such that the state trajectory satisfies the LTL formula, as well as that a certain communication constraint on $\rho_{\rm ave}$ is satisfied.

Remark 1 Note that the *actual* number of communication time steps until k_m is $m + 1$, instead of m. However, since $k_m \to \infty$ as $m \to \infty$, we have $\lim_{m\to\infty}(m + 1)/k_m = \lim_{m\to\infty} m/k_m$. Hence, the lack of "plus one" term does not affect our result provided in this paper. □

3.4 Problem formulation

We now describe the main problem to be solved in this paper:

Problem 1 Given $x_0 \in \mathcal{R}_{\rm init} \in \mathcal{R}_{1:N_I}$, ϕ and $\bar{\rho} \in (0, 1]$, design both control and communication strategies in a sample-and-hold implementation (see Section 3.3), such that:

(A.1) the resulting state trajectory satisfies ϕ;
(A.2) the average communication rate is below $\bar{\rho}$, i.e., $\rho_{\rm ave} < \bar{\rho}$. □

That is, the goal of this paper is to design control and communication strategies, such that the resulting state trajectory satisfies the desired LTL formula ϕ, and the corresponding average communication rate is below a given threshold $\bar{\rho}$. In practice, $\bar{\rho}$ can be provided in terms of the limited communication resources that are present in NCSs, such as the lifetime of the battery powered devices, the number of network tasks that can be executed in real time, and so on. For example, if the relationship between the frequency of communication and the lifetime of the battery powered devices is known, we may select $\bar{\rho}$ in order to ensure that the battery powered devices can stay alive longer than the desired period. As another example, suppose that the embedded micro-processor is able to execute only one task for each k (i.e., there exists only one time slot that the micro-processor can assign the task for each k), and that in addition to the communication task, another network task should be executed within 60% of the total number of executions. In this case, we may select $\bar{\rho}$ as $\bar{\rho} = 1 - 0.6 = 0.4$, so that the rate of executing the communication task is below 40%.

4 Constructing transition system

As a first step to solve Problem 1, we construct a finite transition system that represents an *abstracted* model to describe the behavior of the control systems in (3). Specifically, we aim at obtaining $\mathcal{T} = (S, s_{\rm init}, \delta, \Pi, g, \mathcal{W}_1, \mathcal{W}_2)$, where $S = \{s_1, \ldots, s_{N_I}\}$ is a set of symbolic states, $s_{\rm init} \in S$ is an initial state, $\delta \subseteq S \times S$ is a transition relation, $\Pi = \{\pi_1, \ldots, \pi_{N_I}\}$ is a set of atomic propositions, $g : S \to \Pi$ is a labeling function, and $\mathcal{W}_1, \mathcal{W}_2 : \delta \to \mathbb{N}$ are weight functions. Roughly speaking, each symbol $s_i \in S$ indicates the region of interest $\mathcal{R}_i$ (i.e., the region having the same index i). To relate each symbol to the corresponding region of interest, let $\Gamma : S \to \mathcal{R}_{1:N_I}$ be the mapping given by

$$\Gamma(s_i) = \mathcal{R}_i, \quad \forall i \in \mathbb{N}_{1:N_I}. \tag{7}$$

Conversely, let $\Gamma^{-1} : \mathcal{R}_{1:N_I} \to S$ be the mapping from each region of interest to the corresponding symbolic state. The symbol $s_{\rm init} \in S$ represents the symbolic state associated with $\mathcal{R}_{\rm init}$, i.e., $s_{\rm init} = \Gamma^{-1}(\mathcal{R}_{\rm init})$. The labeling function g outputs the atomic proposition assigned to $\mathcal{R}_i$, i.e., $g(s_i) = \pi_i$. The weight functions $\mathcal{W}_1, \mathcal{W}_2$ are defined to output, as we will see later, the number of communication time steps and the total number of time steps for each transition in δ, respectively. The transition relation $(s_i, s_j) \in \delta$ indicates that every $x \in \mathcal{R}_i$ can be steered to $\mathcal{R}_j$ in finite time. A more formal definition of δ is provided in the next subsection.

4.1 Definition of reachability

To characterize δ in the transition system, we next introduce the notion of reachability among the regions of interest. To this end, consider a pair

$(\mathcal{R}_i, \mathcal{R}_j) \in \mathcal{R}_{1:N_I} \times \mathcal{R}_{1:N_I}$ with $i \neq j$. For notational simplicity, let $\mathcal{X}_{ij} \subset \mathcal{X}$ be given by

$$\mathcal{X}_{ij} = \mathcal{X} \backslash \bigcup_{n \in \mathbb{N}_{\backslash ij}} \mathcal{R}_n, \tag{8}$$

where $\mathbb{N}_{\backslash ij} = \{1, \ldots, N_I\} \backslash \{i, j\}$. That is, $\mathcal{X}_{ij}$ represents the free space that we exclude all regions of interest other than $\mathcal{R}_i$ and $\mathcal{R}_j$. Note that $\mathcal{X}_{ij}$ is a polygonal set that can be a non-convex region. Whether the transition is allowed in $\mathcal{T}$, from $s_i = \Gamma^{-1}(\mathcal{R}_i)$ to $s_j = \Gamma^{-1}(\mathcal{R}_j)$ (i.e., $(s_i, s_j) \in \delta$), is determined according to the following notion of *reachability*:

Definition 5 (Reachability) We say that the reachability holds from $\mathcal{R}_i$ to $\mathcal{R}_j$ $(i \neq j)$, which we denote by $(s_i, s_j) \in \delta$, if there exists $k_F \in \mathbb{N}_+$ such that the following holds: for any $x_0 \in \mathcal{R}_i$ and the disturbance sequence $w_0, w_1, \ldots, w_{k_F-1} \in \mathcal{W}$, there exist $u_0, u_1, \ldots, u_{k_F-1} \in \mathbb{R}^m$ such that the resulting state trajectory $x_0, x_1, \ldots, x_{k_F}$ in accordance with (3) satisfies

(C.1) $x_{k_F} \in \mathcal{R}_j$,
(C.2) $x_k \in \mathcal{X}_{ij}$, $\forall k \in \mathbb{N}_{0:k_F}$,
(C.3) If $x_{k'} \in \mathcal{R}_j$ for some $k' \in \mathbb{N}_{1:k_F}$, then $x_k \notin \mathcal{R}_i$, $\forall k \in \mathbb{N}_{k':k_F}$. □

Based on Definition 5, reachability holds from $\mathcal{R}_i$ to $\mathcal{R}_j$ if there exists a controller such that any state in $\mathcal{R}_i$ can be steered to $\mathcal{R}_j$ in finite time. Moreover, we require by (C.2) that the state needs to avoid any other region of interest apart from $\mathcal{R}_i$ and $\mathcal{R}_j$. Also, (C.3) implies that once the state enters $\mathcal{R}_j$ it must not enter $\mathcal{R}_i$ afterwards. The conditions (C.2), (C.3) are essentially required to guarantee that the trace of the state trajectory satisfies the following property:

Proposition 1 *For every $x_0 \in \mathcal{R}_i$, the trace of the state trajectory $x_0, x_1, \ldots,$ x_{k_F} satisfying (C.1)–(C.3) is $\pi_i \pi_j$.*

Proposition 1 implies that the trace of the state trajectory satisfying (C.1)–(C.3), which is generated according to the rules in Definition 2, is consistent with the trace for the transition from s_i to s_j (i.e., $g(s_i)g(s_j)$). As will be seen later, this leads to that the trace of $\mathcal{T}$ that satisfies ϕ implies that the corresponding trace of the actual state trajectory also satisfies ϕ.

4.2 Reachability analysis from $\mathcal{R}_i$ to $\mathcal{R}_j$

This section provides a way to analyze reachability from $\mathcal{R}_i$ to $\mathcal{R}_j$. The reachability analysis presented in this paper is based on the RRT algorithm (LaValle 2006), which is adapted such that the state trajectories satisfy all requirements to achieve reachability (i.e., the conditions (C.1)–(C.3)), as well as that the corresponding communication strategy can be designed. The overall algorithm is illustrated in Algorithm 1 and the details are described below. As the inputs to the algorithm, we give $N_{\max} \in \mathbb{N}_+$ and $L_{\max} \in \mathbb{N}_+$ as the user-defined parameters, which represent the total number of iterations for building a tree

Algorithm 1: Reachability analysis from $\mathcal{R}_i$ to $\mathcal{R}_j$.

```
   input : $\mathcal{R}_i$, $\mathcal{R}_j$, $L_{\max} \in \mathbb{N}_+$, $N_{\max} \in \mathbb{N}_+$, $K \in \mathbb{R}^{m\times n}$
   output: $k_{0:M}$ (communication time steps); $\hat{x}_{0:k_M}, \hat{u}_{0:k_M-1}, \mathcal{X}_{0:k_M}$ (nominal
           states, inputs, and uncertain sets from $\mathcal{R}_i$ to $\mathcal{R}_j$)
1  $\mathcal{V} \leftarrow \{\}$; $\mathcal{E} \leftarrow \{\}$;
2  $\mathcal{I}(x, x') \leftarrow \{\}$, $\forall (x, x') \in \mathcal{X} \times \mathcal{X}$;
3  $\mathcal{V} \leftarrow \mathcal{V} \cup \{(x_{\mathrm{cent},i}, \mathcal{R}_i)\}$;
4  for $N = 1 : N_{\max}$ do
5      $x_{\mathrm{samp}} \leftarrow \mathsf{Sample}(\mathcal{X}_{ij})$;
6      $(x_{\mathrm{nearest}}, \mathcal{X}_{\mathrm{nearest}}) \leftarrow \mathsf{FindNearest}(\mathcal{V}, x_{\mathrm{samp}})$;
7      $\hat{x}_{\mathrm{opt}} \leftarrow \{\}$; $L_{\mathrm{opt}} \leftarrow \{\}$; $\hat{u}_{\mathrm{opt}} \leftarrow \{\}$;
8      for $L = 1 : L_{\max}$ do
9          $\{\hat{x}_L, \hat{u}\} \leftarrow \mathsf{Steer}(x_{\mathrm{nearest}}, x_{\mathrm{samp}}, L)$;
10         if $\hat{x}_{\mathrm{opt}}$ is empty or $\mathsf{IfClose}(\hat{x}_L, \hat{x}_{\mathrm{opt}}, x_{\mathrm{samp}})$ then
11             $\hat{x}_{\mathrm{opt}} \leftarrow \hat{x}_L$; $\hat{u}_{\mathrm{opt}} \leftarrow \hat{u}$;
12             $L_{\mathrm{opt}} \leftarrow L$;
13         end
14     end
15     $\hat{x}_{0:L_{\mathrm{opt}}} \leftarrow \mathsf{GenTraj}(x_{\mathrm{nearest}}, \hat{u}_{\mathrm{opt}}, L_{\mathrm{opt}})$;
16     $\{\mathcal{X}_0, \dots, \mathcal{X}_{L_{\mathrm{opt}}}\} \leftarrow \mathsf{UncertainSets}(\hat{x}_{0:L_{\mathrm{opt}}}, \hat{u}_{\mathrm{opt}})$;
17     if $\mathsf{IfFeasible}(\mathcal{X}_{0:L_{\mathrm{opt}}})$ then
18         $\mathcal{V} \leftarrow \mathcal{V} \cup \{(\hat{x}_{L_{\mathrm{opt}}}, \mathcal{X}_{L_{\mathrm{opt}}})\}$;
19         $\mathcal{E} \leftarrow \mathcal{E} \cup \{(\hat{x}_0, \mathcal{X}_0), \hat{u}_{\mathrm{opt}}, (\hat{x}_{L_{\mathrm{opt}}}, \mathcal{X}_{L_{\mathrm{opt}}})\}$;
20         $\mathcal{I}(\hat{x}_0, \hat{x}_{L_{\mathrm{opt}}}) \leftarrow L_{\mathrm{opt}}$;
21     end
22 end
23 $\{\tilde{x}_{0:M}, \tilde{u}_{0:M-1}, \widetilde{\mathcal{X}}_{0:M}\} \leftarrow \mathsf{FindTraj}(\mathcal{V}, \mathcal{E})$;
24 $\{k_{0:M}\} \leftarrow \mathsf{UpdateTimes}(\tilde{x}_{0:M}, \mathcal{I})$;
25 $\{\hat{x}_{0:k_M}, \hat{u}_{0:k_M-1}, \mathcal{X}_{0:k_M}\} \leftarrow \mathsf{GenAllTraj}(\tilde{x}_{0:M}, \tilde{u}_{0:M-1}, \widetilde{\mathcal{X}}_{0:M}, k_{0:M})$;
26 if $\mathsf{IfReachable}(\mathcal{X}_{0:k_M})$ then
27     return $k_{0:M}, \hat{x}_{0:k_M}, \hat{u}_{0:k_M-1}, \mathcal{X}_{0:k_M}$;
28 end
```

and the maximum inter-communication time steps, respectively. Moreover, we give a matrix $K \in \mathbb{R}^{m\times n}$, where $A + BK$ is stable, which will be used to obtain (state feedback) control inputs and uncertainty propagations around the nominal state trajectory. The algorithm starts by initializing the set of nodes $\mathcal{V}$ and edges $\mathcal{E}$ as empty sets. In addition, we initialize the mapping $\mathcal{I} : \mathcal{X} \times \mathcal{X} \to \mathbb{N}_+$, which will be used to stack the inter-communication time steps required to steer the states in $\mathcal{X}$. As shown in the algorithm (Line 3), we stack the pair $(x_{\mathrm{cent},i}, \mathcal{R}_i)$ in $\mathcal{V}$ as the initial node. Intuitively, the set $\mathcal{R}_i$ represents the *uncertainty* of the (actual) initial state, which is also stacked together with $x_{\mathrm{cent},i}$ in $\mathcal{V}$.

The algorithm proceeds by expanding a tree of states (Line 4–Line 22). First, $\mathsf{Sample}(\mathcal{X}_{ij})$ draws a state randomly chosen from $\mathcal{X}_{ij}$. For given $\mathcal{V}$ and $x_{\mathrm{samp}} \in \mathcal{X}_{ij}$, $\mathsf{FindNearest}(\mathcal{V}, x_{\mathrm{samp}})$ finds the closest node in $\mathcal{V}$ to x_{samp}, i.e.,

$$\mathsf{FindNearest}(\mathcal{V}, x_{\mathrm{samp}}) = \arg\min_{(x,\mathcal{X})\in\mathcal{V}} \|x - x_{\mathrm{samp}}\|. \tag{9}$$

The function $\mathsf{Steer}(x_{\rm nearest}, x_{\rm samp}, L)$ finds a control input and the corresponding state by solving the following problem:

$$\min_{u \in \mathbb{R}^m} \left\| A^L x_{\rm nearest} + \sum_{\ell=1}^{L} A^{\ell-1} B u - x_{\rm samp} \right\|. \tag{10}$$

In (10), the term $A^L x_{\rm nearest} + \sum_{\ell=1}^{L} A^{\ell-1} B u$ represents the nominal state from $x_{\rm nearest}$ by applying u *constantly* for L time steps. Namely, the function finds an *L-step constant* control input such that the corresponding state is close to $x_{\rm samp}$ as much as possible from $x_{\rm nearest}$. Since the control input is applied constantly for L time steps, we will utilize L as the inter-communication time steps to steer the state from $x_{\rm nearest}$. Let $\hat{u}$ be an optimal control input by solving (10) and let $\hat{x}_L = A^L x_{\rm nearest} + \sum_{\ell=1}^{L} A^{\ell-1} B \hat{u}$. Then, Steer returns $\hat{x}_L$ and $\hat{u}$, i.e.,

$$\mathsf{Steer}(x_{\rm nearest}, x_{\rm samp}, L) = \{\hat{x}_L, \hat{u}\}. \tag{11}$$

The function $\mathsf{IfClose}(\hat{x}_L, \hat{x}_{\rm opt}, x_{\rm samp})$ is defined as follows:

$$\mathsf{IfClose}(\hat{x}_L, \hat{x}_{\rm opt}, x_{\rm samp}) = \begin{cases} \mathsf{True}, & \text{if } \|\hat{x}_L - x_{\rm samp}\| < \|\hat{x}_{\rm opt} - x_{\rm samp}\| \\ & \text{or } \|\hat{x}_L - x_{\rm samp}\| < \varepsilon, \\ \mathsf{False}, & \text{otherwise}, \end{cases} \tag{12}$$

where $\varepsilon > 0$ is a given threshold. That is, it examines if $x_{\rm samp}$ is closer to $\hat{x}_L$ than to $\hat{x}_{\rm opt}$ or *it is close enough to* $x_{\rm samp}$. If $\mathsf{IfClose}(\hat{x}_L, \hat{x}_{\rm opt}, x_{\rm samp}) = \mathsf{True}$, then $\hat{x}_{\rm opt}$ is replaced by $\hat{x}_L$ (Line 11). Note that if $\|\hat{x}_L - x_{\rm samp}\| < \varepsilon$ is satisfied for several values of L, then $\hat{x}_L$ with the *largest* L is chosen (since the algorithm computes $\mathsf{IfClose}$ starting with $L = 1$). The function $\mathsf{GenTraj}(x_{\rm nearest}, \hat{u}_{\rm opt}, L_{\rm opt})$ is executed to obtain the nominal state trajectory by applying the optimal control input: $\mathsf{GenTraj}(x_{\rm nearest}, \hat{u}_{\rm opt}, L_{\rm opt}) = \hat{x}_{0:L_{\rm opt}}$, where $\hat{x}_0 = x_{\rm nearest}$ and $\hat{x}_{\ell+1} = A\hat{x}_\ell + B\hat{u}_{\rm opt}$, $\forall \ell \in \mathbb{N}_{0:L_{\rm opt}-1}$. Then, the function $\mathsf{UncertainSets}(\hat{x}_{0:L_{\rm opt}}, \hat{u}_{\rm opt})$ (Line 16) yields a sequence of polytopic sets as follows:

$$\mathsf{UncertainSets}(\hat{x}_{0:L_{\rm opt}}, \hat{u}_{\rm opt}) = \{\mathcal{X}_0, \ldots, \mathcal{X}_{L_{\rm opt}}\}, \tag{13}$$

where $(\hat{x}_0, \mathcal{X}_0) \in \mathcal{V}$ and $\mathcal{X}_1, \ldots, \mathcal{X}_{L_{\rm opt}}$ are given by

$$\mathcal{X}_\ell = \hat{x}_\ell \oplus \left(A^\ell + \sum_{\ell'=1}^{\ell} A^{\ell'-1} BK \right) (-\hat{x}_0 \oplus \mathcal{X}_0) \oplus \sum_{\ell'=0}^{\ell-1} A^{\ell'} \mathcal{W}, \tag{14}$$

for all $\ell \in \mathbb{N}_{1:L_{\rm opt}}$. Intuitively, and as will be clearer later in this section (see in particular Lemma 1), the sets $\mathcal{X}_0, \ldots, \mathcal{X}_{L_{\rm opt}}$ represent *uncertainty propagations* of the actual state trajectory around the nominal one $\hat{x}_0, \ldots, \hat{x}_{L_{\rm opt}}$, starting from any initial state from $\mathcal{X}_0$ by applying a suitable control strategy. The function $\mathsf{IfFeasible}(\mathcal{X}_{0:L_{\rm opt}})$ (Line 17) examines if all $\mathcal{X}_0, \ldots, \mathcal{X}_{L_{\rm opt}}$ are inside $\mathcal{X}_{ij}$, i.e.,

$$\mathsf{IfFeasible}(\mathcal{X}_{0:L_{\rm opt}}) = \begin{cases} \mathsf{True}, & \text{if } \mathcal{X}_\ell \subseteq \mathcal{X}_{ij}, \forall \ell \in \mathbb{N}_{0:L_{\rm opt}}, & (15) \\ \mathsf{False}, & \text{otherwise}. & (16) \end{cases}$$

If the feasibility holds, the pair $(\hat{x}_{L_{\rm opt}}, \mathcal{X}_{L_{\rm opt}})$, and the triple $((\hat{x}_0,\ \mathcal{X}_0), \hat{u}_{\rm opt}, (\hat{x}_{L_{\rm opt}}, \mathcal{X}_{L_{\rm opt}}))$ are stored in $\mathcal{V}$ and $\mathcal{E}$, respectively. Moreover, we assign the corresponding inter-communication time steps to the mapping $\mathcal{I}$ (Line 18–Line 22).

Once a set of nodes and edges $(\mathcal{V}, \mathcal{E})$ are obtained, the function $\mathsf{FindTraj}$ is executed to find nominal states, inputs and the corresponding uncertain sets from $\mathcal{R}_i$ to $\mathcal{R}_j$ in the following way: if there exist $M \in \mathbb{N}_+$ and $\tilde{x}_{0:M}, \tilde{u}_{0:M-1}, \widetilde{\mathcal{X}}_{0:M}$, where $(\tilde{x}_0, \widetilde{\mathcal{X}}_0) = (x_{{\rm cent},i}, \mathcal{R}_i)$, and

$$((\tilde{x}_m, \widetilde{\mathcal{X}}_m), \tilde{u}_m, (\tilde{x}_{m+1}, \widetilde{\mathcal{X}}_{m+1})) \in \mathcal{E},\ \forall m \in \mathbb{N}_{0:M-1}, \tag{17}$$

$$\widetilde{\mathcal{X}}_M \subseteq \mathcal{R}_j, \tag{18}$$

then we set $\mathsf{FindTraj}(\mathcal{V}, \mathcal{E}) = \{\tilde{x}_{0:M}, \tilde{u}_{0:M-1}, \widetilde{\mathcal{X}}_{0:M}\}$. The illustration of the trajectory $\tilde{x}_{0:M}$ is shown in Fig. 2(a). Roughly speaking, the condition in (18) indicates that the uncertainty around the terminal state (i.e., $\widetilde{\mathcal{X}}_M$) is small enough such that the actual state trajectory is guaranteed to enter $\mathcal{R}_j$. The above sequences can be found by implementing a graph search from the initial node $(x_{{\rm cent},i}, \mathcal{R}_i)$ to some node in $\mathcal{V}$, whose state and the set are both inside $\mathcal{R}_j$. If multiple feasible sequences satisfying (17) and (18) are found, we select the one with the minimum value of the communication rate M/k_M. Note that to steer the state from $\tilde{x}_m$ to $\tilde{x}_{m+1}$ ($m \in \mathbb{N}_{0:M-1}$), $\tilde{u}_m$ needs to be applied constantly for $\mathcal{I}(\tilde{x}_m, \tilde{x}_{m+1})$ time steps, i.e., $\tilde{x}_{m+1} = A^{L_m}\tilde{x}_m + \sum_{\ell=1}^{L_m} A^{\ell-1}B\tilde{u}_m$, where $L_m = \mathcal{I}(\tilde{x}_m, \tilde{x}_{m+1})$.

The function $\mathsf{UpdateTimes}$ (Line 24) is defined as follows: $\mathsf{UpdateTimes}(\tilde{x}_{0:M}, \mathcal{I}) = \{k_0, k_1, \ldots, k_M\}$, where $k_0 = 0$ and

$$k_m = k_{m-1} + \mathcal{I}(\tilde{x}_{m-1}, \tilde{x}_m),\ \forall m \in \mathbb{N}_{1:M}. \tag{19}$$

Intuitively, $k_0, k_1, \ldots, k_{M-1}$ indicate the time steps when control inputs are updated, which, as will be seen below, represent the *communication time steps* between the plant and the controller. In addition, k_M indicates the terminal time step when the state trajectory enters $\mathcal{X}_j$. The function $\mathsf{GenAllTraj}$ is defined to generate all nominal states, inputs and the corresponding uncertain sets including the ones during the inter-communication time steps: $\mathsf{GenAllTraj}(\tilde{x}_{0:M}, \tilde{u}_{0:M-1}, \widetilde{\mathcal{X}}_{0:M}, k_{0:M}) = \{\hat{u}_{0:k_M-1}, \hat{x}_{0:k_M}, \mathcal{X}_{0:k_M}\}$, where

$$\hat{u}_{k_m} = \tilde{u}_m,\ \ \forall m \in \mathbb{N}_{0:M-1}, \tag{20}$$

$$\hat{u}_{k_m+\ell} = \hat{u}_{k_m},\ \forall \ell \in \mathbb{N}_{1:k_{m+1}-k_m-1}, \forall m \in \mathbb{N}_{0:M-1}, \tag{21}$$

and

$$\hat{x}_{k+1} = A\hat{x}_k + B\hat{u}_k,\ \forall k \in \mathbb{N}_{0:k_M-1}. \tag{22}$$

An example of the trajectory $\hat{x}_0, \ldots, \hat{x}_{k_M}$ is illustrated in Fig. 2(b). Note that from (20), (21) and (22), it follows that $\hat{x}_{k_m} = \tilde{x}_m$, $\forall m \in \mathbb{N}_{0:M}$. The sequence

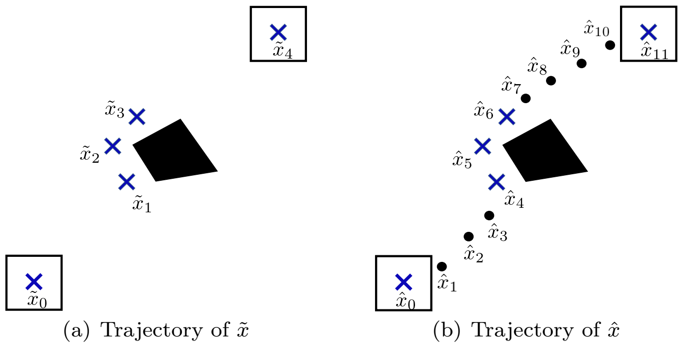


Fig. 2 Illustration of $\tilde{x}_0, \dots, \tilde{x}_M$ from (17) and (18) and $\hat{x}_0, \dots, \hat{x}_{k_M}$ from (22).

$\mathcal{X}_{0:k_M}$ is given as follows: $\mathcal{X}_0 = \widetilde{\mathcal{X}}_0 (= \mathcal{R}_i)$ and

$$\mathcal{X}_{k_m+\ell} = \hat{x}_{k_m+\ell} \oplus \left(A^\ell + \sum_{\ell'=1}^{\ell} A^{\ell'-1} BK \right) (-\hat{x}_{k_m} \oplus \mathcal{X}_{k_m}) \oplus \sum_{\ell'=0}^{\ell-1} A^{\ell'} \mathcal{W}, \quad (23)$$

for all $\ell \in \mathbb{N}_{1:k_{m+1}-k_m}, m \in \mathbb{N}_{0:M-1}$. Note that from (23) and (14), it follows that $\mathcal{X}_{k_m} = \widetilde{\mathcal{X}}_m$, $\forall m \in \mathbb{N}_{0:M}$.

Finally, the function $\mathsf{IfReachable}$ examines if the reachability holds from $\mathcal{R}_i$ to $\mathcal{R}_j$ in the following way: $\mathsf{IfReachable}(\mathcal{X}_{0:k_M}) = \mathsf{True}$ if the following holds:

$$\mathcal{X}_{k'} \cap \mathcal{R}_j \neq \emptyset, \ k' \in \mathbb{N}_{0:k_M} \implies \mathcal{X}_k \cap \mathcal{R}_i = \emptyset, \ \forall k \in \mathbb{N}_{k':k_M}, \quad (24)$$

and $\mathsf{IfReachable}(\mathcal{X}_{0:k_M}) = \mathsf{False}$ otherwise. The condition in (24) indicates that once some uncertain set intersects $\mathcal{R}_j$, it does not intersect $\mathcal{R}_i$ afterwards, which aims at fulfilling the condition (C.3) in Definition 5. Finally, if $\mathsf{IfReachable}$ returns True, the algorithm returns the sequences $k_{0:M}$, $\hat{x}_{0:k_M}$, $\hat{u}_{0:k_M-1}$, $\mathcal{X}_{0:k_M}$.

The fact that the reachability holds based on the above is validated by the following result:

Lemma 1 Suppose that $\mathsf{IfReachable}(\mathcal{X}_{0:k_M}) = \mathsf{True}$ and Algorithm 1 returns $k_{0:M}, \hat{x}_{0:k_M}, \hat{u}_{0:k_M-1}, \mathcal{X}_{0:k_M}$. Then, it follows that the reachability holds from $\mathcal{R}_i$ to $\mathcal{R}_j$. □

Proof : Suppose that $\mathsf{IfReachable}(\mathcal{X}_{0:k_M}) = \mathsf{True}$ and Algorithm 1 returns $k_{0:M}, \hat{x}_{0:k_M}, \hat{u}_{0:k_M-1}, \mathcal{X}_{0:k_M}$. For any $x_0 \in \mathcal{R}_i$, let x_k, $k \in \mathbb{N}_{0:k_M}$ be the actual state trajectory by applying u_k, $k \in \mathbb{N}_{0:k_M-1}$, i.e., $x_{k+1} = Ax_k + Bu_k + w_k$, $\forall k \in \mathbb{N}_{0:k_M-1}$, where

$$u_k = K(x_k - \hat{x}_k) + \hat{u}_k, \quad \text{if } k = k_m, m \in \mathbb{N}_{0:M-1} \quad (25)$$

$$u_k = u_{k-1}, \quad \text{otherwise.} \quad (26)$$

That is, the state feedback controller is utilized for the update times $k_m, m \in \mathbb{N}_{0:M-1}$, and the constant controller is utilized for all the other time steps. In the following, we first show by induction that $x_k \in \mathcal{X}_k$, $\forall k \in \mathbb{N}_{0:k_M}$. For the initial time step $k = k_0 = 0$, we have $x_0 \in \mathcal{R}_i = \mathcal{X}_0$. Assume that $x_{k_m} \in \mathcal{X}_{k_m}$ for some $m \in \mathbb{N}_{0:M-1}$. Then the difference between the actual state and the nominal one for $k_m + \ell$, $\ell \in \mathbb{N}_{1:k_{m+1}-k_m}$ is given by

$$\begin{aligned} x_{k_m+\ell} - \hat{x}_{k_m+\ell} &= A^{\ell} x_{k_m} + \sum_{\ell'=1}^{\ell} A^{\ell'-1} B \left(K(x_{k_m} - \hat{x}_{k_m}) + \hat{u}_{k_m} \right) \\ &\qquad + \sum_{\ell'=1}^{\ell} A^{\ell'-1} w_{k_m+\ell'} - \left(A^{\ell} \hat{x}_{k_m} + \sum_{\ell'=1}^{\ell} A^{\ell'-1} B \hat{u}_{k_m} \right) \\ &= \left(A^{\ell} + \sum_{\ell'=1}^{\ell} A^{\ell'-1} BK \right) (x_{k_m} - \hat{x}_{k_m}) + \sum_{\ell'=0}^{\ell-1} A^{\ell'} w_{k_m+\ell'}, \end{aligned}$$

for all $\ell \in \mathbb{N}_{1:k_{m+1}-k_m}$. Thus, from (23), it follows that $x_{k_m+\ell} \in \mathcal{X}_{k_m+\ell}$, $\forall \ell \in \mathbb{N}_{1:k_{m+1}-k_m}$. Therefore, we have $x_{k_m} \in \mathcal{X}_{k_m} \implies x_{k_m+\ell} \in \mathcal{X}_{k_m+\ell}, \forall \ell \in \mathbb{N}_{1:k_{m+1}-k_m}$, and it is inductively shown that $x_k \in \mathcal{X}_k$, $\forall k \in \mathbb{N}_{0:k_M}$. Moreover, from the feasibility condition in (15), it follows that $\mathcal{X}_k \subseteq \mathcal{X}_{ij}$, $\forall k \in \mathbb{N}_{0:k_M}$. Thus, we obtain $x_k \in \mathcal{X}_{ij}$, $\forall k \in \mathbb{N}_{0:k_M}$ and the condition (C.2) in Definition 5 holds. Moreover, from (18), it follows that $x_{k_M} \in \mathcal{X}_{k_M} = \widetilde{\mathcal{X}}_M \subseteq \mathcal{X}_j$. Hence, the condition (C.1) in Definition 5 holds. Finally, from the definition of IfReachable, it follows that

$$x_{k'} \in \mathcal{R}_j,\ k' \in \mathbb{N}_{1:k_M} \implies x_k \notin \mathcal{R}_i,\ \forall k \in \mathbb{N}_{k':k_M}, \tag{27}$$

which directly shows that the condition (C.3) in Definition 5 holds. Therefore, it is shown that the reachability holds from $\mathcal{R}_i$ to $\mathcal{R}_j$. □

From Lemma 1, if IfReachable returns True there exists a control strategy, as shown in (25) and (26), such that the reachability holds from $\mathcal{R}_i$ to $\mathcal{R}_j$. Moreover, from (25) and (26), each u_{k_m}, $m \in \mathbb{N}_{0:M-1}$ is applied constantly for all $[k_m, k_{m+1})$, which means that the control inputs are updated at $k_0, k_1, \ldots, k_{M-1}$. In other words, the communication time steps when the plant needs to transmit the current state information to the controller are given by $k_0, k_1, \ldots, k_{M-1}$. For the notational use in the next section, let $\mathcal{L}_{x0}, \mathcal{L}_x$, $\mathcal{L}_u$, $\mathcal{L}_I$ be the mappings given by

$$\mathcal{L}_{x0}(\mathcal{R}_i, \mathcal{R}_j) = \hat{x}_{0:k_M},\quad \mathcal{L}_x(\mathcal{R}_i, \mathcal{R}_j) = \hat{x}_{1:k_M}, \tag{28}$$

$$\mathcal{L}_u(\mathcal{R}_i, \mathcal{R}_j) = \hat{u}_{0:k_M-1},\quad \mathcal{L}_I(\mathcal{R}_i, \mathcal{R}_j) = L_{0:M-1}, \tag{29}$$

where $L_m = k_{m+1} - k_m$, $\forall m \in \mathbb{N}_{0:M-1}$. That is, the above mappings yield the nominal state and control trajectories, and the inter-communication time steps to achieve the reachability from $\mathcal{R}_i$ to $\mathcal{R}_j$.

Some remarks on Algorithm 1 are in order as follows:

Remark 2 *(On achieving the minimum communication frequency)*: In this paper, we employ an extended version of the RRT algorithm, so that the communication strategy can be designed while ensuring reachability. Note that the algorithm is not guaranteed to provide the *optimal* communication strategy, which means the resulting communication scheduling may not provide the minimum communication frequency. The minimum number of communication times may be achieved by solving an appropriate optimal control problem for the steering function and by employing the RRT* algorithm (Karaman and Frazzoli 2011b) that integrates the rewiring procedure. Since we aim at reducing the number of control updates, one might attempt to solve the following optimization problem for the steering function:

$$\begin{aligned}
&\underset{\hat{x}_{0:N}, u_{0:N-1}, N}{\arg\min} \ \sum_{\ell=0}^{N-1} \|u_{\ell+1} - u_\ell\|_0, \\
&\text{s.t. } \hat{x}_{\ell+1} = A\hat{x}_\ell + Bu_\ell, \ \forall \ell \in \mathbb{N}_{0:N-1} \\
&\qquad x_0 = x_{\text{nearest}} \\
&\qquad x_N = x_{\text{sample}},
\end{aligned}$$

where $\|\cdot\|_0$ denotes the ℓ_0-norm that represents the number of non-zero components. In other words, we aim to find control inputs and the time step such that the number of control updates is minimized. Using the above problem for the steering function and by the rewiring procedure as in Karaman and Frazzoli 2011b, one may obtain the reachable trajectory that minimizes the number of control updates (i.e., the communication frequency). However, the above optimization problem is a computationally expensive problem, as it involves the ℓ_0-norm cost function. Since the steering function would be utilized for both generating a new sample and the rewiring procedure, using the above problem is clearly unrealistic. Although one may relax the problem by using the ℓ_1-norm cost function, there is no guarantee how such relaxation results in reducing the number of control updates. Therefore, while Algorithm 1 may not minimize the communication frequency, it is more suitable for practical implementations than the above approach in terms of the computation load. $\square$

Remark 3 *(On the computational complexity of Algorithm 1)* : The computational complexity of Algorithm 1 can be analyzed by looking at some primitive procedures in Algorithm 1. The complexity of drawing a sample (Sample) is constant. The complexity of finding the nearest neighbor FindNearest is $O(N)$ (N denotes the iteration number as in Algorithm 1) by using a naive brute-force search, while other efficient but approximate solutions exist (see, e.g., Karaman and Frazzoli 2011b). The steering procedure (Steering) requires to solve a quadratic programming, in which the computational complexity is polynomial with the size of the variables (see, e.g., Monteiro and Adler 1989), which is m in this case, i.e., the dimension of the control input. To analyze the complexity of IfFeasible($\mathcal{X}_{0:L}$), suppose that $\mathcal{X} = \overline{\mathcal{X}} \backslash \mathcal{O}_1 \cup \cdots \cup \mathcal{O}_{N_o}$, where $\overline{\mathcal{X}} \subset \mathbb{R}^n$ denotes a polytopic set, $\mathcal{O}_1, \ldots \mathcal{O}_{N_o}$ are the polytopic obstacles to be avoided in $\widetilde{\mathcal{X}}$, and N_o is the number of the obstacles. For given inter-communitation time steps $L \in \mathbb{N}_{1:L_{\max}}$, IfFeasible examines if L polytopic sets are all inside $\mathcal{X}_{ij}$, which can be done as follows: (i) compute the intersections between each

$\mathcal{X}_\ell$, $\ell \in \mathbb{N}_{1:L}$ and each obstacle and check if these are all empty; (ii) compute the intersections between each $\mathcal{X}_\ell$, $\ell \in \mathbb{N}_{1:L}$ and each region of interest except $\mathcal{R}_i$, $\mathcal{R}_j$, and check if these are all empty. The complexity of computing the intersection between each pair $(\mathcal{X}_\ell, \mathcal{O}_n)$, $(\ell, n) \in \mathbb{N}_{1:L} \times \mathbb{N}_{1:N_o}$ (or each pair $(\mathcal{X}_\ell, \mathcal{R}_n)$, $(\ell, n) \in \mathbb{N}_{1:L} \times \mathbb{N}_{\setminus ij}$) is $O(N_{\rm ver} \log N_{\rm ver})$, where $N_{\rm ver}$ denotes the sum of the number of verticies of $\mathcal{X}_\ell$ and $\mathcal{O}_n$ (or $\mathcal{X}_\ell$ and $\mathcal{R}_n$), see, e.g., Monteiro and Adler 1989. Hence, the computational complexity of IfFeasible depends on the workspace environment, such as the number of obstacles in the state-space. □

4.3 Construction of $\mathcal{T}$

Let us now go back to the beginning of Section 4 and consider constructing the transition system $\mathcal{T}$. Suppose that reachability holds from $\mathcal{R}_i$ to $\mathcal{R}_j$ (i.e., $\mathsf{IfReachable}(\mathcal{X}_{0:k_M}) = \mathsf{True}$) and Algorithm 1 returns $k_{0:M}, \hat{x}_{0:k_M}, \hat{u}_{0:k_M-1}$, $\mathcal{X}_{0:k_M}$. Then, to construct the transition system $\mathcal{T}$, we set $(s_i, s_j) \in \delta$ and assign the values for the weight functions $\mathcal{W}_1, \mathcal{W}_2$ as follows:

$$\mathcal{W}_1(s_i, s_j) = M, \quad \mathcal{W}_2(s_i, s_j) = k_M. \tag{30}$$

That is, we assign for $\mathcal{W}_1(s_i, s_j)$ and $\mathcal{W}_2(s_i, s_j)$ the number of communication time steps and the total number of time steps to achieve reachability from $\mathcal{R}_i$ to $\mathcal{R}_j$, respectively. The weight functions $\mathcal{W}_1$ and $\mathcal{W}_2$ will be utilized to obtain the accepting run of $\mathcal{T}$, such that the corresponding average communication rate is below the threshold $\bar{\rho}$. Based on the above, by applying Algorithm 1 for every pair of the regions of interest, we can characterize both the transition relation δ and the functions $\mathcal{W}_1$, $\mathcal{W}_2$. As a consequence, the transition system $\mathcal{T}$ can be constructed as an abstraction of the control system (3).

5 Implementation

Based on the transition system $\mathcal{T}$ given in the previous section, we now present our control and communication strategies as a solution to Problem 1. Following the hierarchical-based approach, the proposed algorithm consists of *high* and *low level controllers*. The details of each implementation is described below.

5.1 High level controller

In the high level controller part, the controller produces an infinite sequence of the regions of interest that the state should follow to satisfy the formula ϕ, as well as that the average communication rate is below $\bar{\rho}$. Since the reachability among the regions of interest can be captured by the transition system $\mathcal{T}$, this can be done by finding a run $s_{\rm seq} = s_0 s_1 \cdots$ of $\mathcal{T}$, such that $\mathrm{trace}(s_{\rm seq}) \models \phi$ holds. Although there exist several methodologies to achieve this, this paper adopts an automata-based model checking algorithm; we only describe the overview of the approach here and refer the reader to Chapter 5 in Baier and Katoen 2008 for a more detailed explanation. The approach relies on the idea that checking the existence of a run to satisfy ϕ is equivalent to checking the non-emptiness of Trace($\mathcal{T}$) $\cap$ Words(ϕ). Since Words(ϕ) = Lang($\mathcal{B}_\phi$), where $\mathcal{B}_\phi = (Q, Q_0, 2^{\Pi}, \delta_B, F)$ denotes the Büchi Automaton corresponding to the LTL formula ϕ, the above problem is also equivalent to checking the non-emptiness of Trace($\mathcal{T}$) $\cap$ Lang($\mathcal{B}_\phi$), i.e., a language of the product automaton between $\mathcal{T}$ and $\mathcal{B}_\phi$, which is defined below:

Definition 6 (Product Automaton) A product automaton between $\mathcal{T} = (S, s_{\text{init}}, \delta, \Pi, g, \mathcal{W}_1, \mathcal{W}_2)$, and $\mathcal{B}_\phi = (Q, Q_{\text{init}}, \delta_B, 2^{\Pi}, F)$ is defined as a tuple $\mathcal{B}_p = \mathcal{T} \otimes \mathcal{B}_\phi = (Q_p, Q_{\text{init},p}, \delta_p, \Sigma_p, F_p)$, where

– $Q_p = Q \times S$ is a set of states;
– $Q_{\text{init},p} = Q_{\text{init}} \times s_{\text{init}} \subseteq Q_p$ is a set of initial states;
– $\delta_p \subseteq Q_p \times Q_p$ is a transition relation, where $((q,s),(q',s')) \in \delta_p$ iff $(s,s') \in \delta$ and $(q, g(s'), q') \in \delta_B$;
– $\Sigma_p = 2^{\Pi}$ is an input alphabet;
– $F_p = F \times S$ is a set of accepting states.

For a given word $\sigma_{\text{seq}} = \sigma_0\sigma_1\cdots$ with $\sigma_i \in \Sigma_p, \forall i \in \mathbb{N}$, a run of $\mathcal{B}_p$ over σ_{seq} is defined as an infinite sequence of states: $(q_{\text{seq}}, s_{\text{seq}}) = (q_0, s_0)\ (q_1, s_1)\cdots$, such that $(q_0, s_0) \in Q_{\text{init},p}$ (i.e., $s_0 = s_{\text{init}}, q_0 \in Q_{\text{init}}$) and $((q_i, s_i), (q_{i+1}, s_{i+1})) \in \delta_B, \forall i \in \mathbb{N}$. A run of $\mathcal{B}_p$ is called accepting if there exists a word $\sigma_{\text{seq}} = \sigma_0\sigma_1\cdots$ such that F_p is intersected infinitely often. Let

$$(q_{\text{seq}}, s_{\text{seq}}) = (q_0, s_0)\ (q_1, s_1)\ (q_2, s_2)\cdots \tag{31}$$

be one of the accepting runs of $\mathcal{B}_p$. It is known that any accepting run can be represented by a *prefix-suffix* structure, i.e., there exist $n_1, n_2 \in \mathbb{N}$, such that

$$\begin{aligned}(q_{seq}, &s_{\text{seq}}) \\ &= \underbrace{(q_0, s_0)\cdots(q_{n_1-1}, s_{n_1-1})}_{\text{prefix part}}[\underbrace{(q_{n_1}, s_{n_1})\cdots(q_{n_1+n_2-1}, s_{n_1+n_2-1})}_{\text{suffix part}}]^{\omega}.\end{aligned} \tag{32}$$

As shown in (32), n_1, n_2 represent the length of the prefix and the suffix part, respectively. Based on (32) and the weight functions defined in (30), denote by $\mathcal{A}(s_{seq})$ the ratio between the total number of time steps and the communication time steps required for the suffix part of s_{seq}, i.e.,

$$\mathcal{A}(s_{\text{seq}}) = \frac{\displaystyle\sum_{n=n_1+1}^{n_1+n_2-1} \mathcal{W}_1(s_{n-1}, s_n)}{\displaystyle\sum_{n=n_1+1}^{n_1+n_2-1} \mathcal{W}_2(s_{n-1}, s_n)}. \tag{33}$$

Based on the above definitions, in this paper the high level controller finds an appropriate run of $\mathcal{B}_p$ in the following way. First, it finds a set of accepting runs:

$$Q_{p,seq} = \{(q_{\text{seq}}, s_{\text{seq}}) \mid (q_{\text{seq}}, s_{\text{seq}}) \text{ is an accepting run of } \mathcal{B}_p\}, \tag{34}$$

which can be obtained by finding strongly connected components through depth-search methods (see, e.g., Baier and Katoen 2008). Then, we select the accepting run such that the average communication time step for the suffix part is minimized, i.e.,

$$(q^*_{\rm seq}, s^*_{\rm seq}) = \underset{(q_{\rm seq}, s_{\rm seq}) \in Q_{p,seq}}{\arg\min} \ \mathcal{A}(s_{\rm seq}). \tag{35}$$

Since $q^*_{p,\rm seq} = (q^*_{\rm seq}, s^*_{\rm seq})$ is an accepting run of $\mathcal{B}_p$, it is shown that ${\rm trace}(s^*_{\rm seq}) \models \phi$ (see, e.g., Baier and Katoen 2008) and we can obtain the corresponding sequence of regions of interest that is projected from $s^*_{\rm seq}$, i.e.,

$$\mathcal{R}^*_{\rm seq} = R^*_0 R^*_1 R^*_2 \cdots \tag{36}$$

where we denote $s^*_{\rm seq} = s^*_0 s^*_1 \cdots$ and $\mathcal{R}^*_i = \Gamma(s^*_i)$, $\forall i \in \mathbb{N}$. Note that since $s^*_0 = s_{\rm init}$, we have $\mathcal{R}^*_0 = \mathcal{R}_{\rm init}$. Namely, the sequence $\mathcal{R}^*_{\rm seq}$ represents the infinite sequence of the regions of interest that the state trajectory should traverse to satisfy ϕ.

Remark 4 (On selecting the accepting run) As shown in (35), in this paper we select the accepting run such that the average communication rate is minimized. However, we could consider several other criteria in order to select the accepting run. For example, one could select the accepting run according to the following:

$$\begin{aligned}(q^*_{\rm seq}, s^*_{\rm seq}) = \underset{(q_{\rm seq}, s_{\rm seq}) \in Q_{p,seq}}{\arg\min} & \sum_{n=n_1+1}^{n_1+n_2-1} \mathcal{W}_2(s_{n-1}, s_n) \\ \text{s.t. } & \mathcal{A}(s_{\rm seq}) < \bar{\rho}. \end{aligned} \tag{37}$$

That is, among all accepting runs such that the average communication rate for the suffix is below $\bar{\rho}$, we select the one that minimizes the total time steps (in order to satisfy the formula ϕ "as soon as possible"). Here, the constraint $\mathcal{A}(s_{\rm seq}) < \bar{\rho}$ is enforced in order to deduce that the average communication rate is indeed below the threshold $\bar{\rho}$; for details, see Theorem 1.

5.2 Low level controller

Based on the obtained sequence as in (36), the low-level controller iteratively implements the control and communication strategies, such that the resulting state trajectory satisfies ϕ as well as that the average communication rate is below $\bar{\rho}$. To this end, let $\hat{x}^*_0 \hat{x}^*_1 \hat{x}^*_2, \cdots$, $\hat{u}^*_0 \hat{u}^*_1 \hat{u}^*_2 \cdots$, $L^*_0 L^*_1 L^*_2 \cdots$ be an infinite sequence of nominal states, inputs, and inter-communication time steps that are generated based on $\mathcal{R}^*_{seq}$:

$$\underbrace{\mathcal{L}_{x0}(\mathcal{R}^*_0,\mathcal{R}^*_1)}_{\hat{x}^*_0\ \cdots\ \hat{x}^*_{k_{M_1}}}\ \underbrace{\mathcal{L}_{x}(\mathcal{R}^*_1,\mathcal{R}^*_2)}_{\hat{x}^*_{k_{M_1}+1}\ \cdots\ \hat{x}^*_{k_{M_2}}}\ \underbrace{\mathcal{L}_{x}(\mathcal{R}^*_2,\mathcal{R}^*_3)}_{\hat{x}^*_{k_{M_2}+1}\ \cdots\ \hat{x}^*_{k_{M_3}}}\ \cdots, \tag{38}$$

$$\underbrace{\mathcal{L}_{u}(\mathcal{R}^*_0,\mathcal{R}^*_1)}_{\hat{u}^*_0\ \cdots\ \hat{u}^*_{k_{M_1}-1}}\ \underbrace{\mathcal{L}_{u}(\mathcal{R}^*_1,\mathcal{R}^*_2)}_{\hat{u}^*_{k_{M_1}}\ \cdots\ \hat{u}^*_{k_{M_2}-1}}\ \underbrace{\mathcal{L}_{u}(\mathcal{R}^*_2,\mathcal{R}^*_3)}_{\hat{u}^*_{k_{M_2}}\ \cdots\ \hat{u}^*_{k_{M_3}-1}}\ \cdots, \tag{39}$$

$$\underbrace{\mathcal{L}_{I}(\mathcal{R}^*_0,\mathcal{R}^*_1)}_{L^*_0\ \cdots\ L^*_{M_1-1}}\ \underbrace{\mathcal{L}_{I}(\mathcal{R}^*_1,\mathcal{R}^*_2)}_{L^*_{M_1}\ \cdots\ L^*_{M_2-1}}\ \underbrace{\mathcal{L}_{I}(\mathcal{R}^*_2,\mathcal{R}^*_3)}_{L^*_{M_2}\ \cdots\ L^*_{M_3-1}}\ \cdots, \tag{40}$$

where for the notational simplicity we let $\mathcal{L}_{x0}(\mathcal{R}^*_0,\mathcal{R}^*_1)=\hat{x}^*_0\ \cdots\ \hat{x}^*_{k_{M_1}}$ and

$$\mathcal{L}_x(\mathcal{R}^*_i,\mathcal{R}^*_{i+1})=\hat{x}^*_{k_{M_i}+1}\ \cdots\ \hat{x}^*_{k_{M_{i+1}}}, \tag{41}$$

$$\mathcal{L}_u(\mathcal{R}^*_i,\mathcal{R}^*_{i+1})=\hat{u}^*_{k_{M_i}}\ \cdots\ \hat{u}^*_{k_{M_{i+1}}-1}, \tag{42}$$

$$\mathcal{L}_I(\mathcal{R}^*_i,\mathcal{R}^*_{i+1})=L^*_{M_i}\ \cdots\ L^*_{M_{i+1}-1}, \tag{43}$$

with $M_i \in \mathbb{N}$, $i \in \mathbb{N}_+$ appropriately chosen to line up the sequences:

$$\hat{x}^*_0\hat{x}^*_1\hat{x}^*_2\hat{x}^*_3\cdots,\ \hat{u}^*_0\hat{u}^*_1\hat{u}^*_2\hat{u}^*_3\cdots,\ L^*_0L^*_1L^*_2L^*_3\cdots. \tag{44}$$

Moreover, let k^*_m, $m \in \mathbb{N}$ be given by

$$k^*_0=0,\ k^*_{m+1}=k^*_m+L^*_m,\ \forall m\in\mathbb{N}, \tag{45}$$

i.e., k^*_m, $m \in \mathbb{N}$ are the communication time steps between the plant and the controller. Based on the above, a complete algorithm of the low-level implementation is summarized in Algorithm 2. As shown in the algorithm, for each communication time step k^*_m the plant transmits the current state information to the controller, based on which the controller updates the control input according to (25) and (26), and transmits it back to the plant. The main result of this paper is now given as a solution to Problem 1.

Theorem 1 Suppose that for given $x_0 \in \mathcal{R}_{\rm init}$, ϕ and $\bar{\rho} \in (0,1]$, the high level controller finds an accepting run $q^*_{p,{\rm seq}} = (q^*_{\rm seq}, s^*_{\rm seq})$ according to (35) and that we have $\mathcal{A}(s^*_{\rm seq}) < \bar{\rho}$. Moreover, suppose that the low level controller (Algorithm 2) is implemented. Then, the resulting state trajectory satisfies ϕ for any $w_k \in \mathcal{W}$, $\forall k \in \mathbb{N}$. Moreover, the average communication rate is below $\bar{\rho}$, i.e., $\rho_{\rm ave} < \bar{\rho}$. □

Proof : For a given $i \in \mathbb{N}$, suppose that $x_{k_{M_i}} \in \mathcal{R}^*_i$, where k_{M_i} is defined in (38). Since the reachability holds from $\mathcal{R}^*_i$ to $\mathcal{R}^*_{i+1}$ and from the proof of Lemma 1, it follows that the state trajectory enters $\mathcal{R}^*_{i+1}$ (i.e., $x_{k_{M_{i+1}}} \in \mathcal{R}^*_{i+1}$) by applying a control strategy according to (25) and (26), and this holds for any disturbance sequence $w_k \in \mathcal{W}$, $k \in \mathbb{N}_{k_{M_i}:k_{M_{i+1}}-1}$. Hence, starting from $x_0 \in \mathcal{R}^*_0 = \mathcal{R}_{\rm init}$, it is inductively shown that the state trajectory traverses all regions of interest $\mathcal{R}^*_{\rm seq} = \mathcal{R}^*_0\mathcal{R}^*_1\cdots$ by applying Algorithm 2. Moreover, from

Algorithm 2: Low level controller

```
input : $x_0$ (initial state);
        $\hat{x}_0^*\hat{x}_1^*\hat{x}_2^*\cdots$, $\hat{u}_0^*\hat{u}_1^*\hat{u}_2^*\cdots$ (nominal states and inputs);
        $L_0^*L_1^*L_2^*\cdots$ (inter-communication time steps)
        $k_0^*k_1^*k_2^*\cdots$ (communication time steps);
output: State trajectory that solves Problem 1;
1  $k \leftarrow 0$ (initialization);
2  while do
3      if $k = k_m^*$ $(m \in \mathbb{N})$ then
4          the plant transmits the current state $x_k$ to the controller;
5          the controller computes $u_k$ as follows:
               $u_k = K(x_k - \hat{x}_k^*) + \hat{u}_k;$    (46)
           the controller transmits $u_k$ to the plant, and the plant applies $u_k$;
6      else
7          set $u_k = u_{k-1}$ and the plant applies $u_k$ (no communication is given);
8      end
9      $k \leftarrow k + 1$;
10 end
```

the proof of Lemma 1 the state trajectory from $\mathcal{R}_i^*$ to $\mathcal{R}_{i+1}^*$ for each $i \in \mathbb{N}$ satisfies (C.1)–(C.3) in Definition 5. That is, the trace of the state trajectory while moving from $\mathcal{R}_i^*$ to $\mathcal{R}_{i+1}^*$ is $g(s_i^*)g(s_{i+1}^*)$ for all $i \in \mathbb{N}$ (see Proposition 1), which leads to the fact that the trace of the overall state trajectory is given by $\mathrm{trace}(\mathbf{x}) = g(s_0^*)g(s_1^*)g(s_2^*)\cdots$. Thus, we obtain $\mathrm{trace}(\mathbf{x}) = \mathrm{trace}(s_{\mathrm{seq}}^*) \models \phi$ and so the satisfaction of ϕ is achieved.

Now, it remains to show that the communication rate is below $\bar{\rho}$. From Section 5.1, the sequences s_{seq}^* and $\mathcal{R}_{\mathrm{seq}}^*$ can be expressed by the following prefix-suffix structure:

$$s_{\mathrm{seq}}^* = s_0^* s_1^* \cdots s_{n_1^*-1}^* \left(s_{n_1^*}^* \cdots s_{n_1^*+n_2^*-1}^*\right)^\omega, \tag{47}$$

$$\mathcal{R}_{\mathrm{seq}}^* = \mathcal{R}_0^* \mathcal{R}_1^* \cdots \mathcal{R}_{n_1^*-1}^* \left(\mathcal{R}_{n_1^*}^* \cdots \mathcal{R}_{n_1^*+n_2^*-1}^*\right)^\omega, \tag{48}$$

where n_1^*, n_2^* denote the length of the prefix and the suffix parts, respectively. Note that we have $\mathcal{A}(s_{\mathrm{seq}}^*) < \bar{\rho}$. Let $M_{\mathrm{pref}}^*, M_{\mathrm{suf}}^*, K_{\mathrm{pref}}^*, K_{\mathrm{suf}}^* \in \mathbb{N}_+$ be given by

$$M_{\mathrm{pref}}^* = \sum_{n=1}^{n_1^*-1} \mathcal{W}_1(s_{n-1}^*, s_n^*), \quad M_{\mathrm{suf}}^* = \sum_{n=n_1^*+1}^{n_1^*+n_2^*-1} \mathcal{W}_1(s_{n-1}^*, s_n^*), \tag{49}$$

$$K_{\mathrm{pref}}^* = \sum_{n=1}^{n_1^*-1} \mathcal{W}_2(s_{n-1}^*, s_n^*), \quad K_{\mathrm{suf}}^* = \sum_{n=n_1^*+1}^{n_1^*+n_2^*-1} \mathcal{W}_2(s_{n-1}^*, s_n^*). \tag{50}$$

That is, M_{pref}^* and M_{suf}^* represent the number of communication time steps for the prefix and the suffix parts, respectively, and K_{pref}^* and K_{suf}^* represent the total number of time steps for the prefix and the suffix parts, respectively.

Hence, the total number of time steps when the state trajectory traverses the prefix part (i.e., $\mathcal{R}^*_0\mathcal{R}^*_1\cdots\mathcal{R}^*_{n^*_1-1}$) and p-cycles of the suffix part (i.e., the p repetitions of $\mathcal{R}^*_{n^*_1}\cdots\mathcal{R}^*_{n^*_1+n^*_2-1}$) is given by $K^*_{\rm pref}+pK^*_{\rm suf}$. Similarly, the total number of communication time steps when the state trajectory traverses the prefix part and p-cycles of the suffix part is given by $M^*_{pref}+pM^*_{suf}$. Hence, the average communication rate is given by

$$\begin{aligned}\rho_{\rm ave} &= \lim_{p\to\infty}\frac{M^*_{\rm pref}+pM^*_{\rm suf}}{K^*_{\rm pref}+pK^*_{\rm suf}}\\ &= \lim_{p\to\infty}\left(\frac{M^*_{\rm pref}}{K^*_{\rm pref}+pK^*_{\rm suf}}+\frac{pM^*_{\rm suf}}{K^*_{\rm pref}+pK^*_{\rm suf}}\right)\\ &= \frac{M^*_{\rm suf}}{K^*_{\rm suf}}. \end{aligned} \tag{51}$$

Thus, it follows that $\rho_{\rm ave}<\bar{\rho}$ since $\mathcal{A}(s^*_{\rm seq})=M^*_{\rm suf}/K^*_{\rm suf}<\bar{\rho}$. Thus, the average communication rate is below $\bar{\rho}$. □

6 Illustrative examples

In this section we illustrate the effectiveness of the proposed approach. As a simulation example, we consider the motion planning problem of a vehicle moving in a given free space. Let $x=[x_{\rm pos};x_{\rm vel}]\in\mathbb{R}^4$ denote the state of the vehicle, where $x_{\rm pos}=[x_{\rm pos1};x_{\rm pos2}]\in\mathbb{R}^2$ and $x_{\rm vel}=[x_{\rm vel1};x_{\rm vel2}]\in\mathbb{R}^2$ are the position and the velocity of the vehicle, respectively. The dynamics of the vehicle is given by the following double integrator:

$$\dot{x}=\begin{bmatrix}0&0&1&0\\0&0&0&1\\0&0&0&0\\0&0&0&0\end{bmatrix}x+\begin{bmatrix}0&0\\0&0\\1&0\\0&1\end{bmatrix}u+w, \tag{52}$$

where $u\in\mathbb{R}^2$ is the control input and $w\in\mathbb{R}^4$ is the disturbance. To obtain the discrete-time model, we discretize the system in (52) under zero-order hold controller with 0.5 sampling time interval. The disturbance set is given by $\mathcal{W}=\{w\in\mathbb{R}^4:\|w\|_\infty\leq 0.1\}$. The position space of the vehicle, denoted as $\mathcal{X}_{\rm pos}\subset\mathbb{R}^2$ is shown in Fig. 3(a). In the figure, the white regions represent $\mathcal{X}_{\rm pos}$ in which the state (vehicle) can move freely, and the black regions represent obstacles to be avoided. As shown in Fig. 3(a) there exist 4 regions of interest $\mathcal{R}_1,\mathcal{R}_2,\mathcal{R}_3,\mathcal{R}_4$, which are all 1×1 squares. The velocity space, denoted as $\mathcal{X}_{\rm vel}\subset\mathbb{R}^2$, is given by $\mathcal{X}_{\rm vel}=\{x_{\rm vel}\in\mathbb{R}^2:\|x_{\rm vel}\|_\infty\leq 2.0\}$. The (free) state space is thus given by $\mathcal{X}=\mathcal{X}_{\rm pos}\times\mathcal{X}_{\rm vel}$. Based on the above setting, we implement Algorithm 1 for each pair of the regions of interest. While implementing Algorithm 1, we assume $L_{\max}=20$, $N_{\max}=500$ and the matrix K is given

Table 1 The average execution time to compute a control input per each communication time step and the total number of time steps to achieve reachability from $\mathcal{R}_1$ to $\mathcal{R}_3$.

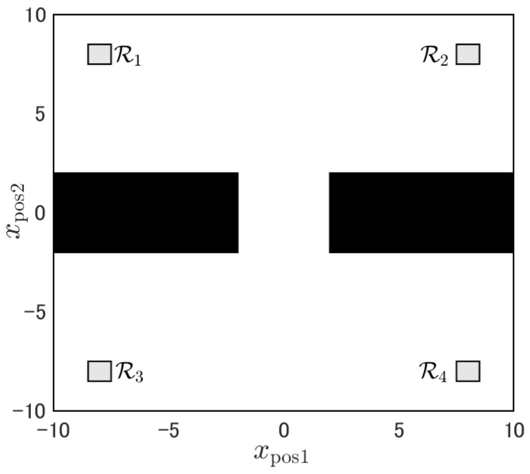
	this paper	Hashimoto et al. 2018
Execution time [s]	5.0×10^{-4}	0.25
The num. time steps	78	64

by

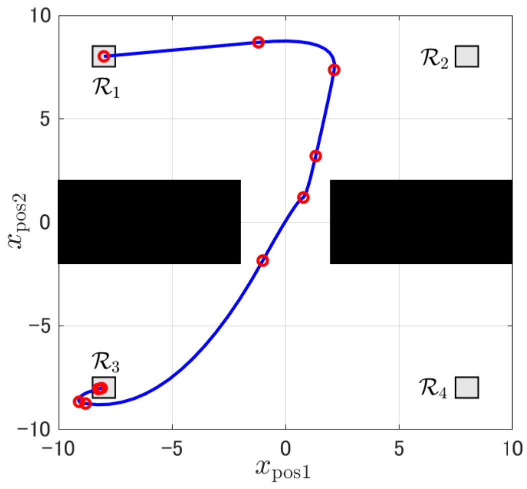
$$K = \begin{bmatrix} 0.004 & 0 & 0.139 & 0 \\ 0 & 0.004 & 0 & 0.139 \end{bmatrix} \tag{53}$$

so that the matrix $A + BK$ is stable. Fig. 3(b) illustrates the nominal state trajectory $\hat{x}_{0:k_M}$ obtained by implementing Algorithm 1 for the pair $(\mathcal{R}_1, \mathcal{R}_3)$. In the figure, the red circles represent the instants when the communication is given (i.e., $\hat{x}_{k_0}, \hat{x}_{k_1}, \ldots \hat{x}_{k_{M-1}}$). The total number of communication time steps and the total number of time steps are given by $M = 10$ and $k_M = 78$, respectively. From the figure, it is shown that the nominal state trajectory reaches $\mathcal{R}_3$, while avoiding all the obstacles. Moreover, Fig. 3(c) illustrates 100 state trajectories by applying the control strategies in (25) and (26), starting from different initial states randomly selected from $\mathcal{R}_1$. It can be shown from the figure that all state trajectories satisfy (C.1)–(C.3) in Definition 5. Similarly, we analyze reachability for all the other pairs of the regions of interest and construct the transition system. The resulting transition system $\mathcal{T}$ contains 4 symbolic states and 16 transitions. The total time to construct $\mathcal{T}$ is 4398s on Windows 10, Intel(R) Core(TM) 2.40GHz, 8GB RAM. To compare with the previous result in Hashimoto, Adachi, and Dimarogonas 2018, we illustrate in Table 1 the average execution time to compute a control input per each communication time step by employing the approach in this paper (i.e., (46)) and the one presented in our previous work (i.e., Eq.(24) and Eq.(25) in Hashimoto, Adachi, and Dimarogonas 2018). The table shows that, the approach presented in this paper results in a significant reduction of the computation load, since it does not need to solve an optimal control problem. On the other hand, the table shows that the approach in this paper requires longer time steps than Hashimoto, Adachi, and Dimarogonas 2018 to achieve the reachability, which may be due to the fact that only one control action can be applied during the inter-communication time steps (while different control actions can be used in the previous approach).

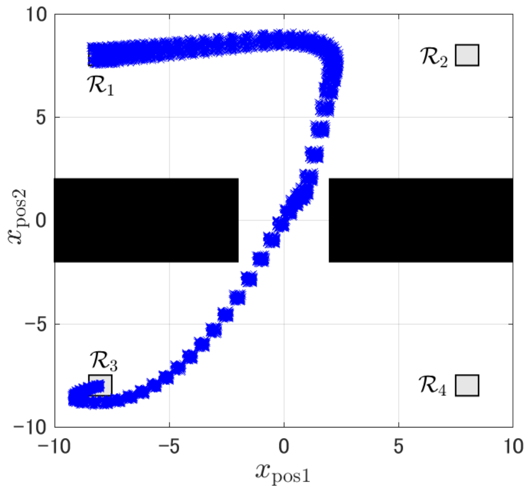
To illustrate the proposal, we consider the following specification: $\phi = (\Diamond\pi_1 \wedge \Diamond\pi_2 \wedge \Diamond\pi_3 \wedge \Diamond\pi_4) \wedge \{(\Box\Diamond\pi_1 \wedge \Box\Diamond\pi_2) \vee (\Box\Diamond\pi_1 \wedge \Box\Diamond\pi_2 \wedge \Box\Diamond\pi_3)\}$. The specification means that, the vehicle should visit all regions of interest at least once, and visit $\mathcal{R}_1, \mathcal{R}_2$ or $\mathcal{R}_1, \mathcal{R}_2, \mathcal{R}_3$ infinitely often. Moreover, we assume that the threshold of the average communication rate is given by $\bar{\rho} = 0.1$. Based on the above problem setup, we implemented both high and low level controllers. Fig. 4(a) illustrates the resulting state trajectory by implementing the low level controller (Algorithm 2) with $x_0 = [8.3;\ -8.4;\ 0;\ 0] \in \mathcal{R}_4$. The figure shows



(a) The position space $\mathcal{X}_{\text{pos}}$

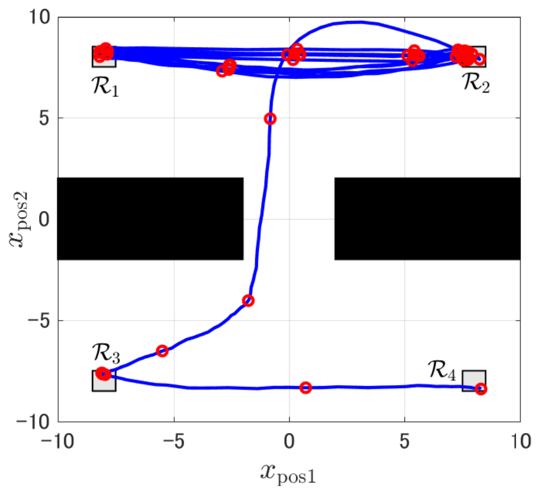


(b) Nominal state trajectory $\hat{x}_{0:k_M}$ from $\mathcal{R}_1$ to $\mathcal{R}_3$ obtained by applying Algorithm 1. The red circles indicate the communication instants (i.e., $\hat{x}_{k_0}, \hat{x}_{k_1}, \ldots, \hat{x}_{k_{M-1}}$)

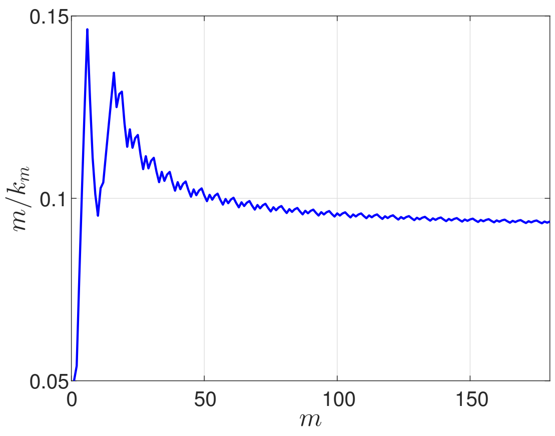


(c) 100 state trajectories by applying (25) and (26) from the initial state randomly selected from $\mathcal{R}_1$.

Fig. 3 Illustration of the position space $\mathcal{X}_{\text{pos}}$ and the results by applying Algorithm 1 for $(\mathcal{R}_1, \mathcal{R}_3)$.



(a) Resulting state trajectory by implementing the low level controller (Algorithm 2).



(b) The sequence of m/k_m, $m = 1, 2, \ldots$.

Fig. 4 Resulting state trajectory by implementing the low level controller (Algorithm 2) and the corresponding sequence of m/k_m, $m = 1, 2, \ldots$.

that the state trajectories visit all the regions of interest, as well as that they visit $\mathcal{R}_1$, $\mathcal{R}_2$ infinitely often. Hence, the state trajectories are shown to satisfy the formula ϕ. In addition, Fig. 4(b) illustrates the corresponding sequence m/k_m, $m = 1, 2, \ldots$. The figure shows that the sequence m/k_m converges below $\bar{\rho} = 0.1$, which shows that that the average communication rate is indeed below $\bar{\rho}$.

Note that the state trajectory *could* satisfy ϕ if the high level controller would select the run of $\mathcal{T}$ such that the state trajectory traverses $\mathcal{R}_1$, $\mathcal{R}_2$, $\mathcal{R}_3$ (instead of $\mathcal{R}_1$, $\mathcal{R}_2$) infinitely often. Fig. 5 illustrates the sequence m/k_m, $m =$

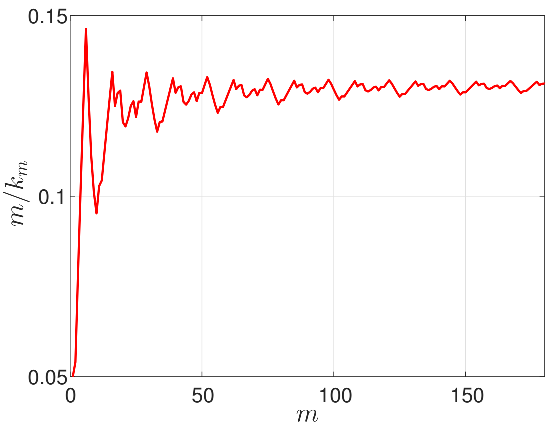


Fig. 5 The sequence of m/k_m, $m = 1, 2, \ldots$ if the state trajectory would traverse $\mathcal{R}_1, \mathcal{R}_2, \mathcal{R}_3$ infinitely often.

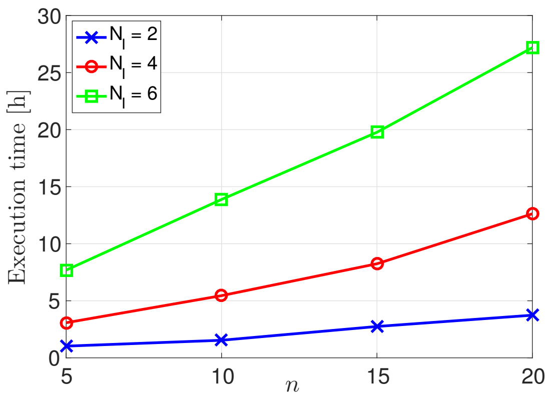


Fig. 6 Execution time against the state dimension n and N_I.

$1, 2, \ldots$ for the case when the state trajectory traverses $\mathcal{R}_1$, $\mathcal{R}_2$, $\mathcal{R}_3$ infinitely often. The figure illustrates that, the corresponding average communication rate cannot be below $\bar{\rho} = 0.1$. This means that, if the high level controller would select a run such that the state trajectory traverses $\mathcal{R}_1$, $\mathcal{R}_2$, $\mathcal{R}_3$, it would then violate the specification of the average communication rate (i.e., $\rho_{\rm ave} > \bar{\rho}$). Hence, it is shown that the high level controller appropriately selected the accepting run, such that the average communication rate is below $\bar{\rho}$.

Finally, it is worth analyzing the execution time with respect to the number of regions of interest N_I as well as the state dimension n. Fig. 6 plots of the execution time for constructing $\mathcal{T}$ (i.e., the total execution time of Algorithm 1 for all pairs of the regions of interest) against the state dimension $n \geq 5$, with different selections of $N_I = 2, 4, 6$. Here, we fix the input dimension as $m = 5$ and the matrices A and B are randomly generated over the reals in the interval $[0, 1]$. The state space is given by $\mathcal{X} = \{x \in \mathbb{R}^n : \|x\|_\infty \leq 15\}$, and we assume that the regions of interest are the full dimensional polytopes with $n + 1$ vertices, which are randomly generated from $\mathcal{X}$. The figure shows that, for fixed N_I, the execution time increases as n increases. This is due to some primitive procedures in Algorithm 1, such as IfFeasible, some vertex operations in (14), etc. In addition, for fixed n the execution time also increases as N_I increases. This is mainly due to the fact that the total number of combinations to select the pair of the regions of interest is $N_I(N_I - 1)/2$, implying that the execution time increases quadratically with N_I. Note that, thanks to the implementation of the sampling-based algorithm, the algorithm is tractable even for high-dimensional systems (e.g., $n = 20$), if the number of the regions of interest is small enough. This point may be an advantage over the previous discretization based approaches (e.g., Wongpiromsarn, Topcu, and Murray 2012; Nilsson et al. 2012), in which the algorithm becomes intractable even with much lower state dimensions.

7 Conclusions and Future work

In this paper, we propose control and communication strategies, such that the state trajectories satisfy the LTL specification and the average communication rate is below a given threshold. We start by providing RRT-based reachability analysis, which is adapted such that the state trajectories satisfy the reachability specifications, as well as that the communication strategy can be designed. Then, we provide a high level controller that aims to find an accepting run of the transition system, and then provide a low level controller that aims to steer the state trajectories satisfying the desired specifications. Finally, we illustrate the benefits of the proposed approach through numerical simulations.

As previously stated in Remark 2, the reachability algorithm is not guaranteed to provide a communication strategy that minimizes the number of communication frequency. Hence, future work involves investigating the reachability algorithm that improves the optimality of the communication strategy. Moreover, the control synthesis that handles delays and packet dropouts explicitly for NCSs will be considered for our future work.

References

Agha-mohammadi, A., S. Chakravorty, and N. M. Amato (2014). “FIRM: Sampling-based feedback motion planning under motion uncertainty and imperfect measurements”. In: *The International Journal of Robotics Research* 33.2, pp. 268–304.

Antoniotti, M., M. Jafari, and B. Mishra (1995). “Applying temporal logic verification and synthesis to manufacturing systems”. In: *IEEE International Conference on Systems, Man and Cybernetics.*

B. Luders, M. Kothari and J. How (2010). “Chance Constrained RRT for Probabilistic Robustness to Environmental Uncertainty”. In: *AIAA guidance, navigation, and control conference.*

Baier, C. and J.-P Katoen (2008). *Principles of model checking.* The MIT Press.

Belta, C. et al. (2007). “Symbolic planning and control of robot motion [Grand Challenges of Robotics]”. In: *IEEE Robotics and Automation Magazine* 14.1, pp. 61–70.

Bhatia, A., L. E. Kavraki, and M. Y. Vardi (2010). “Sampling-based Motion Planning with Temporal Goals”. In: *Proceedings of IEEE International Conference on Robotics and Automation*, pp. 2689–2696.

Borrelli, F., A. Bemporad, and M. Morari (2017). *Predictive Control for Linear and Hybrid Systems.* Cambridge University Press.

Borri, A., G. Pola, and M. D. Di. Benedetto (2018). “Design of Symbolic Controllers for Networked Control Systems”. In: *IEEE Transactions on Automatic Control* 64.3, pp. 1034–1046.

Chinchali, S. et al. (2012). “Towards formal synthesis of reactive controllers for dexterous robotic manipulation”. In: *IEEE International Conference on Robotics and Automation (ICRA).*

Coogan, S. et al. (2016). “Traffic Network Control From Temporal Logic Specifications”. In: *IEEE Transactions on Control of Network Systems* 3.2, pp. 162–172.

Fainekos, G. E. et al. (2009). “Temporal Logic Motion Planning for Dynamic Robots”. In: *Automatica* 45.2, pp. 343–352.

Filippidis, I. et al. (2016). “Control design for hybrid systems with TuLiP: The Temporal Logic Planning toolbox”. In: *2016 IEEE Conference on Control Applications (IEEE CCA).*

Gol, E. A., M. Lazar, and C. Belta (2015). “Temporal logic model predictive control”. In: *Automatica* 56.11, pp. 78–85.

Guo, M. and D. V. Dimarogonas (2015). “Multi-agent plan reconfiguration under local LTL specifications”. In: *The International Journal of Robotics Research* 34.2, pp. 218–235.

Guo, M., K. H. Johansson, and D. V. Dimarogonas (2013). “Revising Motion Planning under Linear Temporal Logic Specifications in Partially Known Workspaces”. In: *IEEE International Conference on Robotics and Automation (ICRA).*

Hashimoto, K., S. Adachi, and D. V. Dimarogonas (2018). “Energy-aware networked control systems under temporal logic specifications”. In: *Proceedings of the 57th IEEE Conference on Decision and Control (IEEE CDC).*

He, K. et al. (2015). “Towards Manipulation Planning with Temporal Logic Specifications”. In: *IEEE International Conference on Robotics and Automation (ICRA).*

Heddy, G. et al. (2015). “Linear Temporal Logic (LTL) Based Monitoring of Smart Manufacturing Systems”. In: *Proceedings of the Annual Conference of the Prognostics and Health Management Society.*

Heemels, W. P. M. H., K. H. Johansson, and P. Tabuada (2012). "An Introduction to Event-triggered and Self-triggered Control". In: *Proceedings of the 51st IEEE Conference on Decision and Control (IEEE CDC)*, pp. 3270–3285.

Hespanha, J. P., P. Naghshtabrizi, and Y. Xu (2007). "A survey of recent results in networked control systems". In: *Proceedings of the IEEE* 95.1, pp. 138–162.

Karaman, S. and E. Frazzoli (2011a). "Linear temporal logic vehicle routing with applications to multiUAV mission planning". In: *International Journal of Robust and Nonlinear Control* 21.12, pp. 1372–1395.

— (2011b). "Sampling-based algorithms for optimal motion planning". In: *The International Journal of Robotics Research* 30.7, pp. 846–894.

— (2012). "Sampling-based Algorithms for Optimal Motion Planning with Deterministic μ-Calsulus Specifications". In: *Proceedings of 2012 American Control Conference*, pp. 735–742.

Karimadini, M. and H. Lin (2011). "Guaranteed global performance through local coordinations". In: *Automatica* 47.5, pp. 890–898.

Kehoe, B. et al. (2015). "A survey of research on cloud robotics and automation". In: *IEEE Transactions on Automation Science and Engineering* 12.2, pp. 398–409.

Kloetzer, M. and C. Belta (2008). "A Fully Automated Framework for Control of Linear Systems from Temporal Logic Specifications". In: *IEEE Transactions on Automatic Control* 53.1, pp. 287–297.

Kress-Gazit, H., G. E. Fainekos, and G. J. Pappas (2007). "Where's Waldo? Sensor-Based Temporal Logic Motion Planning". In: *IEEE International Conference on Robotics and Automation (ICRA).*

— (2009). "Temporal-Logic-Based Reactive Mission and Motion Planning". In: *IEEE Transactions on Robotics* 25.6, pp. 1370–1381.

Kress-Gazit, H., M. Lahijanian, and V. Raman (2018). "Synthesis for Robots: Guarantees and Feedback for Robot Behavior". In: *Annual Review of Control, Robotics, and Autonomous Systems* 1, pp. 211–236.

LaValle, S. M. (2006). *Planning algorithms.* Cambridge, UK: Cambridge University Press.

Livingston, S. C. and R. M. Murray (2013). "Just-in-time synthesis for reactive motion planning with temporal logic". In: *Proceedings of IEEE International Conference on Robotics and Automation (ICRA).*

Livingston, S. C., E. M. Wolff, and R. M. Murray (2015). "Cross-entropy Temporal Logic Motion Planning". In: *Proceedings of the 18th International Conference on Hybrid Systems: Computation and Control (HSCC)*, pp. 269–278.

Majumdar, A. and R. Tedrake (2017). "Funnel libraries for real-time robust feedback motion planning". In: *The International Journal of Robotics Research* 36.8, pp. 947–982.

Mazo, M., A. Davitian, and P. Tabuada (2010). "PESSOA: A Tool for Embedded Controller Synthesis". In: *Proceedings of the 22nd International Conference on Computer Aided Verification*, pp. 566–569.

Monteiro, R. D. C. and I. Adler (1989). “Interior path following primal-dual algorithms. Part II: Convex Quadratic Programming”. In: *Mathematical Programming* 44, pp. 43–66.

Morel, G., J. F. Petin, and P. Lamboley (2001). “Formal Specification for Manufacturing Systems Automation”. In: *Proceedings of the 10th IFAC Symposium on Information Control Problems in Manufacturing.*

Nilsson, P. et al. (2012). “Temporal Logic Control of Switched Affine Systems with an Application in Fuel Balancing”. In: *IEEE Conference on Decision and Control (IEEE CDC).*

Oddoux, D. and P. Gastin (2001). “LTL2BA software: fast translation from LTL formulae to Büchi automaton”. In: URL: `http://www.lsv.ens-cachan.fr/{~}gastin/ltl2ba/`.

Pepy, R., M. Kieffer, and E. Walter (2009). “Reliable Robust Path Planning with application to Mobile Robots”. In: *International Journal of Applied Mathematics and Computer Science* 19.3, pp. 413–424.

Pola, G., P. Pepe, and M. D. Di. Benedetto (2018). “Decentralized Supervisory Control of Networks of Nonlinear Control Systems”. In: *IEEE Transactions on Automatic Control* 63.9, pp. 2803–2817.

Tedrake, R. et al. (2010). “LQR-Trees: Feedback Motion Planning via Sums-of-Squares Verication”. In: *The International Journal of Robotics Research* 29.8, pp. 1038–1052.

Tumova, J. and D. V. Dimarogonas (2016). “Multi-agent planning under local LTL specifications and event-based synchronization”. In: *Automatica* 70, pp. 239–248.

Verginis, C. K. and D. V. Dimarogonas (2017). “Distributed Cooperative Manipulation under Timed Temporal Specifications”. In: *Proceedings of 2017 American Control Conference.*

Wang, X. and M. D. Lemmon (2009). “Self-triggered feedback control systems with finite $\mathcal{L}_2$ gain stability”. In: *IEEE Transactions on Automatic Control* 54.3, pp. 452–467.

Wongpiromsarn, T., U. Topcu, and R. M. Murray (2012). “Receding Horizon Temporal Logic Planning”. In: *IEEE Transactions on Automatic Control* 57.11, pp. 2817–2830.

Zhang, L. et al. (2005). “A New Method for Stabilization of Networked Control Systems with Random delays”. In: *IEEE Transactions on Automatic Control* 50.8, pp. 1177–1181.

Zhang, W., M. S. Branicky, and S. M. Phillips (2001). “Stability of Networked Control Systems”. In: *IEEE Control Systems* 21.1, pp. 84–99.