Planning Based Semantics for Distributed Real-Time Systems

Mahieddine Dellabani, Jacques Combaz, Saddek Bensalem, Marius Bozga

Verimag Research Report n^o TR-2017-1

April 25, 2017

Reports are downloadable at the following address
http://www-verimag.imag.fr

Unité Mixte de Recherche 5104 CNRS - Grenoble INP - UGA
Bâtiment IMAG
Université Grenoble Alpes
700, avenue centrale
38401 Saint Martin d'Hères
France
tel : +33 4 57 42 22 42
fax : +33 4 57 42 22 22
http://www-verimag.imag.fr/

Planning Based Semantics for Distributed Real-Time Systems

Mahieddine Dellabani, Jacques Combaz, Saddek Bensalem, Marius Bozga

University Grenoble Alpes, VERIMAG, F-38000 Grenoble, France
CNRS, VERIMAG, F-38000 Grenoble, Frnace
{mahieddine.dellabani,saddek.bensalem,jacques.combaz,marius.bozga}@imag.fr
http://www.verimag.fr/rsd

April 25, 2017

Abstract

Design, implementation and verification of distributed real-time systems is acknowledged to be a very hard task. Such systems are prone to different kind of delays, such as execution times of actions or communication delays implied by the distributed platform. The latter, increases considerably the complexity of coordinating the parallel activities of running components. Scheduling such systems must cope with those delays by proposing execution strategies ensuring global consistency and satisfaction of timing constraints. In this paper, we investigate a formal model for such systems as compositions of timed automata subject to multi-party interactions, and propose a method aiming to overcome the communication delays problem through planning ahead interactions. To be effective in a distributed context, planning an interaction should rely on (as much as possible) local information only, namely the state of the participating components. However, as shown in the paper local information is not always sufficient for correctly scheduling interactions as it may introduce deadlocks. Moreover, delays may also affect the statisfaction of timing constraints, which also corresponds to deadlocks in the formal model. In this paper we investigate methods for analyzing such deadlock situations and for computing deadlock-free scheduling strategies when possible.

Keywords: Distributed Real-Time Systems·Timed Automata·Planning·
Compositional Verification

Reviewers:

How to cite this report:

@techreport {TR-2017-1,
title = {Planning Based Semantics for Distributed Real-Time Systems},
author = {Mahieddine Dellabani, Jacques Combaz, Saddek Bensalem, Marius Bozga},
institution = {{Verimag} Research Report},
number = {TR-2017-1},
year = {}
}

1 Introduction

Nowadays, real-time systems are ubiquitous in several application domains, and such an emergence led to an increasing need of performance: resources, availability, concurrency...etc. This expansion initiates a shift from the use of single processor based hardware platforms, to large sets of interconnected and distributed computing nodes. Moreover, it prompts the birth of a new family of systems known as *Networked Embedded Systems*, and that are intrinsically distributed. Such evolution stems from an increase in complexity of real-time software embedded on such platforms (e.g. electronic control in avionics and automotive domains [7]), and the need to integrate formerly isolated systems [11] so that they can cooperate as well as share resources, improving functionality and reducing costs.

To deal with such complexity, the community of safety critical systems often restricts its scope to predictable systems, which are represented with domain specific models (e.g. periodic tasks, synchronous systems, time-deterministic systems) for which the range of possible executions is small enough to be easily analyzed, allowing the pre-computation of optimal control strategies. *Networked Embedded Systems* usually describes a set of real-time systems, distributed across platform(s) and interacting through a network. Because of their adaptive behavior, the standard practice when implementing such systems is not to rely on models for pre-computation execution strategies but rather to design systems dynamically adapting at runtime to the actual context of execution. Such approaches do not offer any formal guarantee of timeliness. The lack of a priori knowledge on system behavior leaves also little room for static optimization.

Model-based development is one promising approach in building distributed real-time system. First, an application model is designed, expressing abstraction of the timed system behavior. This abstraction is platform independent, meaning that it does not consider platform introduced delays or CPU speed, which allows to: *(i)* model the system at early stages without any knowledge of the target platform, and *(ii)* verify the obtained model against some safety properties (functional requirements). Thereafter, the application source code, which represents the actual implementation of the system on a given platform, is automatically generated from the model. The big challenge becomes then how to verify the timing behavior of the implementation, since a lot of assumption drops such as atomic execution of action or timeless communication delays. In this paper, we propose a model-based approach aiming to mitigate the communication delays of distributed platforms. In this approach, systems consist of components represented as timed automata that may synchronize on particular actions to coordinate their activities. We contribute to this research field by proposing a different semantics than the usual semantics of timed automata. This semantics aims to distinguish between the decision date of executing interactions and their actual execution dates. It relies on *planning* interactions with bounded horizons in order to take into account the communication delays of the execution platforms and to reduce their impact on systems execution.

This work is an extension of our work presented in [8]. We extend our previous work by *(1)* defining a more mature and realistic semantics for planning interactions. Particularly, we introduce a lower bound planning horizon representing communication delays. In other words, immediate (timeless) planning is no longer allowed. We also give *(2)* a formal characterization of deadlocks and *(3)* provide a simple execution strategy aiming to avoid deadlocks exhibited in system running under this semantics.

The rest of the paper is organized as follows. Section 2 gives preliminary definitions of timed automata with respect to multiparty interactions as well as predicates definitions needed for the rest of the paper. In Section 3, we present our extended planning semantics, discuss its relation with the usual timed automata semantics, and give a formal characterization of its deadlock states. Then, Section 4 presents an execution strategy aiming to enforce the correctness of the planning semantics and provide a compositional verification approach for checking deadlock-freedom property. Additionally, Section 5 explains how the planning approach can be formalized as a real-time controller synthesis problem and highlights the key issues met during our reflection and the obtained results. Finally, Section 6 describes our implementation of the proposed approach and the obtained results.

2 Timed Systems and Properties

In the framework of the present paper, components are timed automata and systems are compositions of timed automata with respect to multiparty interactions. The timed automata we use are essentially the ones

from [1], however, slightly adapted to embrace a uniform notation throughout the paper.

Definition 1 (Component)**.** *A* component *is a tuple* $B = (\mathcal{L}, \ell_0, \mathcal{A}, \mathcal{T}, \mathcal{X}, tpc)$ *where* $\mathcal{L}$ *is a finite set of* locations*,* $\ell_0 \in \mathcal{L}$ *is an* initial *location,* $\mathcal{A}$ *a finite set of* actions*,* $\mathcal{X}$ *is a finite set of* clocks*,* $\mathcal{T} \subseteq \mathcal{L} \times (\mathcal{A} \times \mathcal{C} \times 2^{\mathcal{X}}) \times \mathcal{L}$ *is a set of* transitions *labeled with an action, a guard, and a set of clocks to be reset, and* $tpc : \mathcal{L} \to \mathcal{C}$ *assigns a* time progress condition*,* tpc_ℓ *to each location* $\ell \in \mathcal{L}$*, where* $\mathcal{C}$ *is the set of* clock constraints *defined by the following grammar:*

$$\mathcal{C} := \mathit{true} \mid x \sim ct \mid x - y \sim ct \mid \mathcal{C} \wedge \mathcal{C} \mid \mathit{false},$$

with $x, y \in \mathcal{X}$*,* $\sim \in \{<, \leq, =, \geq, >\}$ *and* $ct \in \mathbb{Z}$*. Time progress conditions are restricted to conjunctions of constraints of the form* $x \leq ct$*.*

Throughout the paper, we assume components that are deterministic timed automata, that is, at a given location ℓ and for a given action a, there is up to one outgoing transition from ℓ labeled by a. Given a timed automaton $(\mathcal{L}, \ell_0, \mathcal{A}, \mathcal{T}, \mathcal{X}, tpc)$, we write $\ell \xrightarrow{a,g,r} \ell'$ if there exists a transition $\tau = (\ell, (a, g, r), \ell') \in \mathcal{T}$. We also denote by $guard(a, \ell)$ the clock constraints of the transition labeled by a and outgoing from ℓ if it exists, and *false* otherwise and we write:

$$guard(a, \ell) = \begin{cases} g, & \text{if } \exists \tau = (\ell, (a, g, r), \ell') \in \mathcal{T} \\ \mathit{false}, & \text{otherwise} \end{cases}$$

Let $\mathcal{V}$ be the set of all clock valuation functions $v : \mathcal{X} \to \mathbb{R}_{\geq 0}$. For a clock constraint C, $C(v)$ is a boolean value corresponding to the evaluation of C on v. For a valuation $v \in \mathcal{V}$, $v + \delta$ is the valuation satisfying $(v + \delta)(x) = v(x) + \delta$, while for a subset of clocks r, $v[r]$ is the valuation obtained from v by resetting clocks of r, i.e. $v[r](x) = 0$ for $x \in r$, $v[r](x) = v(x)$ otherwise. We also denote by $C + \delta$ the clock constraint C shifted by δ, i.e. such that $(C + \delta)(v)$ iff $C(v + \delta)$. We also consider the classical *backward* and *forward* operators [14] on clock constraints, i.e. $(\swarrow C)(v)$ iff $\exists \delta \geq 0 \,.\, C(v + \delta)$ and $(\nearrow C)(v)$ iff $\exists \delta \geq 0 \,.\, C(v - \delta)$. In what follows, we also use two variants of the backward operator considering lower bounds $u \in \mathbb{Z}_{\geq 0}$ and upper bounds $l \in \mathbb{Z}_{\geq 0} \cup \{+\infty\}$: $(\swarrow_l C)(v)$ iff $\exists \delta \geq l \,.\, C(v + \delta)$ and $(\swarrow_l^u C)(v)$ iff $\exists \delta \,.\, l \leq \delta \leq u \wedge C(v + \delta)$.

Definition 2 (Semantics)**.** *A component* $B = (\mathcal{L}, \ell_0, \mathcal{A}, \mathcal{T}, \mathcal{X}, tpc)$ *defines the labeled transition system (LTS)* $(\mathrm{Q}, \mathcal{A} \cup \mathbb{R}_{>0}, \to)$ *where* $\mathrm{Q} \subseteq \mathcal{L} \times \mathcal{V}(\mathcal{X})$ *denotes the* states *of* B*, and* $\to \subseteq \mathrm{Q} \times (A \cup \mathbb{R}_{>0}) \times Q$ *denotes the set of transitions between states according to the rules:*

- $(\ell, v) \xrightarrow{a} (\ell', v[r])$ *if* $\ell \xrightarrow{a,g,r} \ell'$*, and* $g(v)$ *is true (action step).*
- $(\ell, v) \xrightarrow{\delta} (\ell, v + \delta)$ *if* $tpc_\ell(v + \delta)$ *for* $\delta \in \mathbb{R}_{>0}$ *(time step).*

An *execution sequence* of B from a state (ℓ, v) is a path in the LTS starting at (ℓ, v) and that alternates action steps and time steps, that is:

$$(\ell, v) \xrightarrow{\sigma_1} \ldots \xrightarrow{\sigma_i} (\ell_n, v_n), n \in \mathbb{Z}_{>0}, \sigma \in \mathcal{A} \cup \mathbb{R}_{>0}.$$

We say that a state (ℓ, v) is *reachable* if there is an execution sequence from the initial configuration (ℓ_0, v_0) leading to (ℓ, v), where v_0 assigns 0 to all clocks. In this paper, we always assume components with *well formed guards*, that is, transitions $\ell \xrightarrow{a,g,r} \ell'$ satisfy $g(v) \Rightarrow tpc_\ell(v) \wedge tpc_{\ell'}(v[r])$ for any $v \in \mathcal{V}$. This ensures that the reachable states always satisfy the time progress conditions, i.e. if (ℓ, v) is reachable then we have $tpc_\ell(v)$. Notice that the set of reachable states is in general infinite, but it can be partitioned into a finite number of symbolic states [14, 4, 10]. A symbolic state is defined by a pair (ℓ, ζ) where, ℓ is a location of B, and ζ is a zone, i.e. a set of clock valuations defined by a clock constraint (as defined in Definition 1). Efficient algorithms for computing symbolic states and operations on zones are fully described in [4]. Given symbolic states $\{(\ell_j, \zeta_j)\}_{j \in J}$ of B, the predicate $Reach(B)$ characterizing the reachable states can be formulated as:

$$Reach(B) = \bigvee_{j \in J} \mathsf{at}(\ell_j) \wedge \zeta_j,$$

where $\mathsf{at}(\ell_j)$ is true on states whose location is ℓ_j, and clock constraint ζ_j is straightforwardly applied to clock valuation functions of states.

We define the predicate $Enabled(a)$ characterizing states (ℓ, v) at which an action a is enabled, i.e. such that $(\ell, v) \xrightarrow{a} (\ell', v')$. It can be written:

$$Enabled(a) = \bigvee_{\ell \in \mathcal{L}} \mathsf{at}(\ell) \wedge guard(a, \ell).$$

A state (ℓ, v) is said *urgent* if time cannot progress from (ℓ, v), that is, there is no $\delta \in \mathbb{R}_{>0}$ such that $(\ell, v) \xrightarrow{\delta} (\ell, v')$. Urgent states are characterized by the predicate:

$$\bigvee_{\ell \in \mathcal{L}} \mathsf{at}(\ell) \wedge urg(\ell) \tag{1}$$

where $urg(\ell)$ is a clock constraint characterizing the valuations from which time cannot progress with respect to the time progress condition of ℓ, that is, it is defined by $urg(\ell) = \bigvee_{i=1}^{m}(x_i \geq ct_i)$ if $tpc_\ell = \bigwedge_{i=1}^{m} x_i \leq ct_i$. Notice that due to well-formed guards an urgent reachable state satisfies also (1) if inequalities $x_i \geq ct_i$ on clocks are replaced by equalities $x_i = ct_i$ in the expression of $urg(\ell)$.

Definition 3 (Deadlock and action-time-lock)**.** *We say that a state (ℓ, v) of a component $B = (\mathcal{L}, \ell_0, \mathcal{A}, \mathcal{T}, \mathcal{X}, tpc)$ is* deadlock *if, with respect to its semantics, no action can be executed from (ℓ, v) and from its successors, that is:*

$$\nexists a \in \mathcal{A} \,.\, (\ell, v) \xrightarrow{a} (\ell', v') \;\vee\; (\ell, v) \xrightarrow{\delta} (\ell, v + \delta) \xrightarrow{a} (\ell', v').$$

A deadlock (ℓ, v) is called an action-time-lock *when no interaction can execute nor time can progress from (ℓ, v), that is:*

$$\nexists a \in \mathcal{A} \,.\, (\ell, v) \xrightarrow{a} (\ell', v') \wedge \;\; \nexists \delta > 0 \,.\, (\ell, v) \xrightarrow{\delta} (\ell, v + \delta).$$

Deadlocks are situations from which a component is stuck a given location without beging able to progress by executing an action, which must be avoided in reactive systems. Action-time-locks are modeling errors and consists in deadlocks from which time cannot progress.

In our framework, components communicate by means of *multiparty interactions*. A multiparty interaction is a rendez-vous synchronization between actions of a fixed subset of components. It takes place only if all the participants agree to execute the corresponding actions. Given n components B_i, $i = 1, \ldots, n$, with disjoint sets of actions $\mathcal{A}_i$, an interaction is a subset of actions $\alpha \subseteq \cup_{1 \leq i \leq n} \mathcal{A}_i$ containing at most one action per component, i.e. $\alpha \cap \mathcal{A}_i$ is either empty or a singleton $\{a_i\}$. That is, an interaction α can be put in the form $\{a_i\}_{i \in I}$ with $I \subseteq \{1, \ldots, n\}$ and $a_i \in \mathcal{A}_i$ for all $i \in I$. We denote by $\mathrm{part}(\alpha)$, the set of components *participating* in α, that is, $\mathrm{part}(\alpha) = \{B_i\}_{i \in I}$.

In practice we do not explicitly build compositions of components. We rather interpret their semantics at runtime by evaluating enabled interactions based on current states of components. In a composition of n components $B_{i \in \{1, \cdots, n\}}$, denoted by $\gamma(B_1, \ldots, B_n)$, an action a_i can execute only as part of an interaction α such that $a_i \in \alpha$, that is, along with the execution of all other actions $a_j \in \alpha$, which corresponds to the usual notion of multiparty interaction.

Definition 4 (Standard Semantics of a Composition)**.** *Given a set of components $\{B_1, \cdots, B_n\}$ and an interaction set γ. The (standard) semantics of the composition $S = \gamma(B_1, \cdots, B_n)$ w.r.t the set of interactions γ, is the LTS $LTS_g = (\mathrm{Q}_g, \gamma \cup \mathbb{R}_{>0}, \rightarrow_\gamma)$ where:*

- *$\mathrm{Q}_g = \mathcal{L} \times \mathcal{V}(\mathcal{X})$ is the set of global states, where $\mathcal{L} = \mathcal{L}_1 \times \cdots \times \mathcal{L}_n$ and $\mathcal{X} = \bigcup_{i=1}^{n} \mathcal{X}_i$. We write a state $q = (\ell, v)$ where $\ell = (\ell_1, \cdots, \ell_n) \in \mathcal{L}$ is a global location and $v \in \mathcal{V}(\mathcal{X})$ is global clocks valuations.*
- *$\rightarrow_\gamma$ is the set of labeled transitions defined by the rules:*
  - *$(\ell, v) \xrightarrow{\alpha}_\gamma (\ell', v')$ for $\ell = (\ell_1, \cdots, \ell_n)$ and $\alpha = \{a_i\}_{i \in I} \in \gamma$, if $\forall i \in I \; (\ell_i, v_i) \xrightarrow{a_i} (\ell'_i, v'_i)$ and $\forall i \notin I \; (\ell_i, v_i) = (\ell'_i, v'_i)$.*

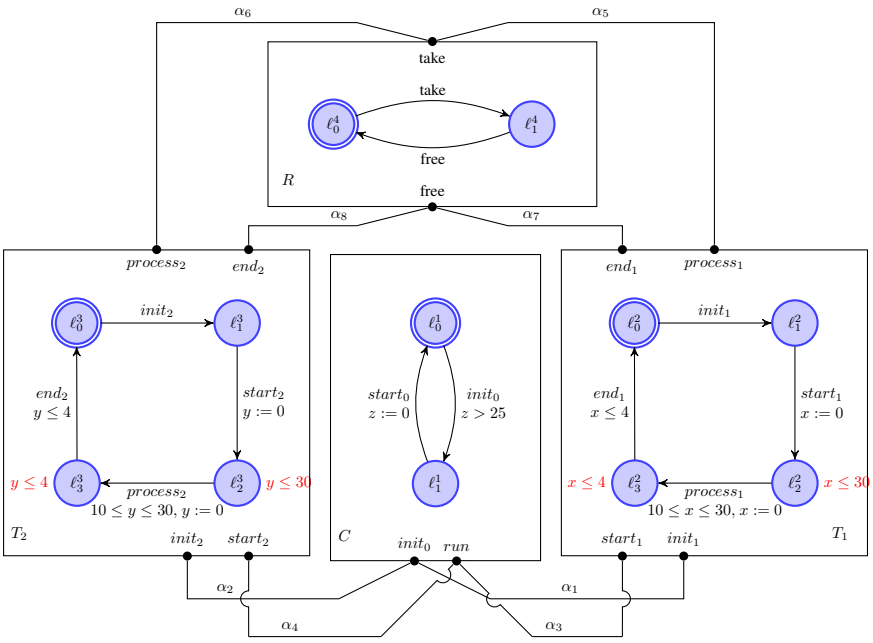

Figure 1: Task Manager

– $(\ell, v) \xrightarrow{\delta}_\gamma (\ell, v+\delta)$ *for* $\delta \in \mathbb{R}_{>0}$ *if* $\forall i \in \{1, \cdots, n\}$ $tpc_{\ell_i}(v_i + \delta)$ *where* $\ell = (\ell_1, \ldots, \ell_n)$ *and* v_i *denotes the restriction of* v *to clocks* $\mathcal{X}_i$ *of* B_i.

To simplify notations, predicates defined on individual components B_i are straightforwardly interpreted on states (ℓ, v) of a composition $S = \gamma(B_1, \cdots, B_n)$ by considering the projection (ℓ_i, v_i) of (ℓ, v) on B_i, which is such that $\ell = (\ell_1, \ldots, \ell_n)$ and v_i is restriction of v to clocks $\mathcal{X}_i$ of B_i. For instance, $\mathsf{at}(\ell_i)$ evaluates to true on (ℓ, v) iff $\ell \in \mathcal{L}_1 \times \ldots \times \mathcal{L}_{i-1} \times \{\ell_i\} \times \mathcal{L}_{i+1} \times \ldots \times \mathcal{L}_n$. Similarly, clock constraints of components B_i are applied to clock valuation functions v of the composition by restricting v to clocks $\mathcal{X}_i$ of B_i. This allows to write the predicate $Enabled(\alpha)$ characterizing states (ℓ, v) from which an interaction $\alpha = \{a_i\}_{i \in I} \in \gamma$ can be executed, i.e. such that $(\ell, v) \xrightarrow{\alpha}_\gamma (\ell', v')$, as:

$$Enabled(\alpha) = \bigwedge_{i \in I} Enabled(a_i) = \bigvee_{\{\ell_i \in \mathcal{L}_i\}_{i \in I}} \bigwedge_{i \in I} \mathsf{at}(\ell_i) \wedge guard(a_i, \ell_i).$$

Notice that the above formulation of $Enabled(\alpha)$ corresponds to locations enumeration of all components participating in interaction α. In practice, we rather consider only a subset of locations $\mathcal{L}_{a_i} \subseteq \mathcal{L}_i$, from which there exists a transition labeled by action $a_i \in \alpha$.

The definitions of execution sequences, reachable states, deadlocks and action-time-locks of components are also trivially extended to composition of components. Deadlocks of a composition $S = \gamma(B_1, \cdots, B_n)$ can be characterized as follows:

$$\bigvee_{\ell = (\ell_1, \ldots \ell_n) \in \mathcal{L}} \mathsf{at}(\ell) \wedge \Big[\bigwedge_{\alpha \in \gamma} \neg \swarrow \Big(Enabled(\alpha) \wedge \bigwedge_{1 \leq i \leq n} tpc_{\ell_i} \Big) \Big], \tag{2}$$

and action-time-locks by:

$$\Big(\bigwedge_{\alpha \in \gamma} \neg Enabled(\alpha) \Big) \wedge \Big(\bigvee_{1 \leq i \leq n} \bigvee_{\ell_i \in \mathcal{L}_i} \mathsf{at}(\ell_i) \wedge urg(\ell_i) \Big] \Big).$$

Example 1 (Running Example)**.** *Let us consider as a running example the composition of four components* C, T_1, T_2, *and* R *of Figure 1. Component* C *represents a controller that initializes then releases tasks*

T_1 and T_2. Tasks use the shared resource R during their execution. To implement such behavior, we consider the following interactions between C, R, and T_1: $\alpha_1 = \{init_0, init_1\}$, $\alpha_3 = \{start_0, start_1\}$, $\alpha_5 = \{take, process_1\}$, $\alpha_7 = \{free, end_1\}$, and similar interactions α_2, α_4, α_6, α_8 for task T_2, as shown by connections on Figure 1. The controller is responsible for firing the execution of each task. First, it non-deterministically initializes one of the two tasks, i.e. executes α_1 or α_2, and then releases it through interaction α_3 or α_4. Tasks perform their processing independently of the controller, after being granted an access to the shared resource (α_5 or α_6). When finished, a task releases the resource (interactions α_7 or α_8) and go back to its initial location. An example of execution sequence of the system of Figure 1 is given below, in which valuations v of clocks x, y, and z are represented as a tuples $(v(x), v(y), v(z))$:

$$((\ell_0^1, \ell_0^2, \ell_0^3, \ell_0^4), (0,0,0)) \xrightarrow{26}_\gamma ((\ell_0^1, \ell_0^2, \ell_0^3, \ell_0^4), (26,26,26)) \xrightarrow{\alpha_1}_\gamma ((\ell_1^1, \ell_1^2, \ell_0^3, \ell_0^4), (26,26,26))$$
$$\xrightarrow{\alpha_3}_\gamma ((\ell_0^1, \ell_2^2, \ell_0^3, \ell_0^4), (0,26,0)) \xrightarrow{10}_\gamma ((\ell_0^1, \ell_2^2, \ell_0^3, \ell_0^4), (10,36,10)) \xrightarrow{\alpha_5}_\gamma$$
$$((\ell_0^1, \ell_3^2, \ell_0^3, \ell_1^4), (0,36,10)) \xrightarrow{2}_\gamma ((\ell_0^1, \ell_3^2, \ell_0^3, \ell_1^4), (2,38,12)) \xrightarrow{\alpha_2}_\gamma ((\ell_1^1, \ell_3^2, \ell_1^3, \ell_1^4), (2,38,12))$$

3 Local Planning of Interactions

High-level coordination primitives, such as multi-party synchronizations (interactions) defined in the previous section, are rarely built-in primitives of distributed platforms. Hence, their implementation on a distributed platforms requires synchronization protocols responsible for realizing global synchronizations using simpler primitives such as point-to-point messages passing as explained by [13]. This is classically implemented using one ore more additional coordination components observing the system state and deciding on interaction execution. However, due to communication delays, to meet timing constraints of components, scheduling decisions must be taken before actual executions.

This motivates the introduction of the *planning semantics*, which distinguishes between the decision of the execution of an interaction (its *planning*), and the execution itself. The delay between the planning of an interaction and its execution is constrained by the (maximal) communication latency induced by the execution platform, which is a parameter of the semantics. In the proposed semantics, interaction planning is based only on the states of its participating components, which allows to decide locally without monitoring the entire system. It is correct in the sense that it refines (it is included in) the semantics of Section 2. However, being based on local states, planning decisions are too permissive and may introduce deadlocks when they are not compatible with the global state of the system.

3.1 Definition of the Planning Semantics

Let $S = \gamma(B_1, \cdots, B_n)$ be a composition of components $B_1, \ldots, B_n$ as defined in Section 2. We define the predicate $Plannable(\alpha, \delta)$ characterizing states (ℓ, v) from which an interaction $\alpha = \{a_i\}_{i \in I} \in \gamma$ is enabled in $\delta \in \mathbb{R}_{\geq 0}$ units of time (if time progress by δ units of time), that is, such that $Enabled(\alpha)$ evaluates to true on state $(\ell, v + \delta)$. It is characterized by:

$$Plannable(\alpha, \delta) \Leftrightarrow \bigvee_{\{\ell_i \in \mathcal{L}_i\}_{i \in I}} \bigwedge_{i \in I} \mathsf{at}(\ell_i) \wedge \big(guard(a_i, \ell_i) + \delta\big) \tag{3}$$

Notice that for an interaction α the predicate $Plannable(\alpha, \delta)$ depends only on states of components of $\mathrm{part}(\alpha)$, which motivates the following property.

Property 1. *Let (ℓ, v) be a state of the composition S. For any interactions $\alpha, \beta \in \gamma$ such that, $(\ell, v) \xrightarrow{\beta}_\gamma (\ell', v')$ and $\mathrm{part}(\alpha) \cap \mathrm{part}(\beta) = \emptyset$, if $Plannable(\alpha, \delta)$ holds at state (ℓ, v) then it still holds at state (ℓ', v').*

This property derives directly from fact that executing an interaction β does not change the states of components participating in an interaction α, provided that α and β have disjoint sets of participating components, and thus $Plannable(\alpha, \delta)$ is not affected by the execution of β in this case. In the following, we say that two interactions α and β *conflicts* when they share common participating components, that is, when $\mathrm{part}(\alpha) \cap \mathrm{part}(\beta) \neq \emptyset$, and we write $\alpha \# \beta$. We denote by $conf(\alpha)$ the set of interactions conflicting with α, that is, $conf(\alpha) = \{\beta \in \gamma \mid \alpha \# \beta\}$.

Property 2. *Let (ℓ, v) and $(\ell, v+\delta')$, with $\delta' \in \mathbb{R}_{>0}$ be two states of the composition S. For an interaction $\alpha \in \gamma$, if $Plannable(\alpha, \delta)$ holds at state (ℓ, v) then $Plannable(\alpha, \delta - \delta')$ also holds at any state $(\ell, v+\delta')$ such that $\delta' \leq \delta$.*

This property can be found directly by writing Equation 3 on state $(\ell, v + \delta')$.

As previously explained, due to communication latencies induced by the platform we assume that interactions cannot be planned in δ units of time if $\delta < h_{\min}$, where $h_{\min} \in \mathbb{Z}_{\geq 0}$ is a parameter representing the minimal *planning horizon*, which should represent the upper bound communication latencies. Notice that for the sake of simplicity, we consider a global parameter $h_{\min}$ but we could also assume different parameters for each interaction. For an interaction α we define the predicate $Plannable(\alpha)$ characterizing states from which α can be planned in a delay respecting the planning horizon $h_{\min}$, that is:

$$Plannable(\alpha) \Leftrightarrow \exists \delta \geq h_{\min} \,.\, Plannable(\alpha, \delta),$$

It can be written formally as follows:

$$\bigvee_{\{\ell_i \in \mathcal{L}_i\}_{i \in I}} \Big(\bigwedge_{i \in I} \mathsf{at}(\ell_i) \Big) \wedge \Big(\not\swarrow_{h_{\min}} \big[\bigwedge_{i \in I} guard(a_i, \ell_i) \big] \Big). \tag{4}$$

Definition 5 (Plan)**.** *A plan π is a function $\pi : \gamma \to \mathbb{R}_{\geq 0} \cup \{+\infty\}$ defining relative times for executing interactions, with the convention that an interaction α if planned to execute in $\pi(\alpha)$ time units only if $\pi(\alpha) < +\infty$. Plans satisfy that for any two interactions $\alpha \neq \beta$ such that $\pi(\alpha) < +\infty$ and $\pi(\beta) < +\infty$, then interactions α and β are not conflicting (i.e. $\neg(\alpha \# \beta)$).*

We denote by $conf(\pi)$ the set of interactions conflicting with the plan π, i.e. $conf(\pi) = \{\alpha \mid \exists \beta \# \alpha \,.\, \pi(\beta) < +\infty\}$, and $\mathrm{part}(\pi)$ the set of components participating in interactions planned by π, i.e. $\mathrm{part}(\pi) = \{B_i \mid \exists \alpha \,.\, \pi(\alpha) < +\infty \wedge B_i \in \mathrm{part}(\alpha)\}$. Notice that since π stores relative times, whenever time progresses by δ, the value $\pi(\alpha)$ assigned by π to an interaction α should be decreased by δ until it reaches 0, meaning that α have to execute. We write $\pi - \delta$ to describe the progress of time over the plan, that is, $(\pi - \delta)(\alpha) = \pi(\alpha) - \delta$ for interactions α such that $\pi(\alpha) < +\infty$. Similarly, $\pi[\alpha \mapsto \delta]$ assigns relative time δ to α, $\alpha \notin conf(\pi)$, into existing plan π, i.e. $(\pi[\{\alpha \mapsto \delta])(\beta) = \delta$ for $\beta = \alpha$, $(\pi[\alpha \mapsto \delta])(\beta) = \pi(\beta)$ otherwise.

Definition 6 (Planning Semantics)**.** *Given a set of components $\{B_1, \cdots, B_n\}$ and an interaction set γ, we define the planning semantics of the composition $\gamma(B_1, \cdots, B_n)$, as the LTS $LTS_p = (\mathrm{Q}_p, \gamma \cup \mathbb{R}_{>0} \cup \{\mathbf{plan}\}, \rightsquigarrow_{>\gamma})$ where:*

- $\mathrm{Q}_p = \mathcal{L} \times \mathcal{V}(\mathcal{X}) \times \Pi$*, where $\mathcal{L}$ is the set of global location, $\mathcal{V}(\mathcal{X})$ is the set of global clocks valuations, and Π is the set of plans. Again, in the following to simplify notations predicates defined on states $(\ell, v) \in \mathrm{Q}_g = \mathcal{L} \times \mathcal{V}(\mathcal{X})$ of the standard semantics are straightforwardly interpreted on states $(\ell, v, \pi) \in \mathrm{Q}_p$ considering the projection (ℓ, v) of (ℓ, v, π) on Q_g.*
- **plan** *defines the action of planning interactions*
- $\rightsquigarrow_{>\gamma}$ *is the set of labeled transitions defined by the rules:*

– $(\ell, v, \pi) \overset{\mathbf{plan}(\alpha,\delta)}{\rightsquigarrow}_{>\gamma} (\ell, v, \pi[\alpha \mapsto \delta])$ *for $\alpha \in \gamma$ and $\delta \geq h_{\min}$ if $\alpha \notin conf(\pi)$ and $Plannable(\alpha, \delta)$ holds on (ℓ, v, π).*

– $(\ell, v, \pi) \overset{\alpha}{\rightsquigarrow}_{>\gamma} (\ell', v', \pi[\alpha \mapsto +\infty])$ *for $\alpha \in \gamma$ if $\pi(\alpha) = 0$.*

– $(\ell, v, \pi) \overset{\delta}{\rightsquigarrow}_{>\gamma} (\ell, v + \delta, \pi - \delta)$ *for $\delta \leq \min\ \pi$, $\ell = (\ell_1, \ldots, \ell_n)$, if $tpc_{\ell_i}(v + \delta)$ for components $B_i \in \mathrm{part}(\pi)$ and $tpc_{\ell_i}(v + \delta + h_{\min})$ for components $B_i \notin \mathrm{part}(\pi)$.*

States of the planning semantics does not include only locations and clocks valuations, but also the relative execution times of the planned interactions stored by π. Initially, no interaction is planned, that is, initial states (ℓ_0, v_0, π_0) satisfy $\pi_0 = +\infty$. Planning an interaction α to be executed at a relative time $\delta \geq h_{\min}$ corresponds to the operation $\pi[\alpha \mapsto \delta]$ on the plan, which can only be done if α is not conflicting

with the plan, and if it becomes enabled if time progress by δ (i.e. if $Plannable(\alpha, \delta)$). On the other hand, time progress not only updates clocks value but also the plan by decreasing the relative execution times of the planned interactions. To force the execution of planned interactions when their relative execution times reach 0, time cannot progress more than the relative execution times of the interactions (more than $\delta \leq \min\ \pi$). As for the standard semantics, time progress is limited by the time progress conditions of the components, but with the following significant difference. Components $B_i \in \mathrm{part}(\pi)$ participating in planned interactions behaves as in the standard semantics, that is, time can progress until their time progress conditions expire. For components $B_i \notin \mathrm{part}(\pi)$ that are not participating in planned interactions, we take into account the minimal delay $h_{\min}$ needed for planning and then executing an interaction: in components $B_i \notin \mathrm{part}(\pi)$ time can progress only up to $h_{\min}$ time units before their time progress conditions expire. By doing so we are sure that there always remains enough time to plan interactions involving $B_i \notin \mathrm{part}(\pi)$, if they exist, and execute them before their time progress condition expire.

Example 2. *Let us consider the following execution sequence for example of Figure 1 under the planning semantics with* $h_{\min} = 2$.

$$((\ell_0^1,\ell_0^2,\ell_0^3,\ell_0^4),(0,0,0),+\infty) \overset{\mathbf{plan}(\alpha_1,26)}{\rightsquigarrow}_\gamma ((\ell_0^1,\ell_0^2,\ell_0^3,\ell_0^4),(0,0,0),\{\alpha_1 \mapsto 26\}) \overset{26}{\rightsquigarrow}_\gamma$$
$$((\ell_0^1,\ell_0^2,\ell_0^3,\ell_0^4),(26,26,26),\{\alpha_1 \mapsto 0\}) \overset{\alpha_1}{\rightsquigarrow}_\gamma ((\ell_1^1,\ell_1^2,\ell_0^3,\ell_0^4),(26,26,26),+\infty)$$
$$\overset{\mathbf{plan}(\alpha_3,2)}{\rightsquigarrow}_\gamma ((\ell_1^1,\ell_1^2,\ell_0^3,\ell_0^4),(26,26,26),\{\alpha_3 \mapsto 2\}) \overset{2}{\rightsquigarrow}_\gamma ((\ell_1^1,\ell_1^2,\ell_0^3,\ell_0^4),(28,28,28),$$
$$\{\alpha_3 \mapsto 0\}) \overset{\alpha_3}{\rightsquigarrow}_\gamma ((\ell_0^1,\ell_2^2,\ell_0^3,\ell_0^4),(0,28,0),+\infty) \overset{\mathbf{plan}(\alpha_2,26)}{\rightsquigarrow}_\gamma ((\ell_0^1,\ell_2^2,\ell_0^3,\ell_0^4),$$
$$(0,28,0),\{\alpha_2 \mapsto 26\}) \overset{26}{\rightsquigarrow}_\gamma ((\ell_0^1,\ell_2^2,\ell_0^3,\ell_0^4),(26,54,26),\{\alpha_2 \mapsto 0\}) \overset{\alpha_2}{\rightsquigarrow}_\gamma$$
$$((\ell_1^1,\ell_2^2,\ell_1^3,\ell_0^4),(26,54,26),+\infty) \overset{\mathbf{plan}(\alpha_4,2)}{\rightsquigarrow}_\gamma ((\ell_1^1,\ell_2^2,\ell_1^3,\ell_0^4),(26,54,26),\{\alpha_4 \mapsto 2\})$$
$$\overset{2}{\rightsquigarrow}_\gamma ((\ell_1^1,\ell_2^2,\ell_1^3,\ell_0^4),(28,56,28),\{\alpha_4 \mapsto 0\}) \overset{\alpha_4}{\rightsquigarrow}_\gamma ((\ell_0^1,\ell_2^2,\ell_2^3,\ell_0^4),(28,0,0),+\infty)$$
$$\overset{\mathbf{plan}(\alpha_6,30)}{\rightsquigarrow}_\gamma ((\ell_0^1,\ell_2^2,\ell_2^3,\ell_0^4),(28,0,0),\{\alpha_6 \mapsto 30\})$$

This execution sequence represents a path that alternates plan actions, time progress and execution of some interactions, and leads to the action-time-lock state $((\ell_0^1,\ell_2^2,\ell_2^3,\ell_0^4),(0,0,28),\{\alpha_6 \mapsto 30\})$. *In fact, the time progress condition* $x \leq 30$ *in component* T_1*, imposes the planning of interaction* α_7 *at the latest* h_{min} *units of time before it becomes urgent. However, since interaction* α_6 *was planned in 28 units of time,* α_7 *cannot be planned since it is conflicting with* α_6*. This execution sequence shows that a given system can action-time-locks under the planning semantics, even if it is deadlock-free in the standard semantics.*

3.2 Relation between Standard Semantics and Planning Semantics

We use weak simulation to compare the model under the standard semantics and the planning semantics by considering **plan**-transitions unobservable. As shown in Example 2, the planning semantics does not preserve the deadlock freedom property of our system. Nevertheless, the following proves weak simulation relations between the two semantics.

Lemma 1. *Given a reachable state* (ℓ, v, π) *of the planning semantics. If for* $\alpha \in \gamma$, $\pi(\alpha) < +\infty \Rightarrow Plannable(\alpha, \pi(\alpha))$.

Proposition 1. *An interaction can execute from a state* (ℓ, v, π) *in the planning semantics only if it can execute from* (ℓ, v) *in the standard semantics, that is:*

$$\forall \alpha \in \gamma.(\ell, v, \pi) \overset{\alpha}{\rightsquigarrow}_\gamma (\ell', v', \pi') \Rightarrow (\ell, v) \xrightarrow{\alpha}_\gamma (\ell', v').$$

Proposition 1 is a consequence of Lemma 1: an interaction α can execute in the planning semantics only if $\pi(\alpha) = 0$ (see Definition 5). That is, a state (ℓ, v, π) of the planning semantics from which α can execute satisfy $Plannable(\alpha, 0)$ or equivalently $Enabled(\alpha)$, which demonstrates that α can execute from (ℓ, v) in the standard semantics.

Proposition 2. *Time can progress by δ at a state (ℓ, v, π) in the planning semantics only if time can progress by δ at (ℓ, v) in the standard semantics, that is:*

$$\forall \delta \in \mathbb{R}_{>0}.(\ell, v, \pi) \overset{\delta}{\rightsquigarrow}_{>\gamma} (\ell', v', \pi') \Rightarrow (\ell, v) \overset{\delta}{\rightarrow}_{\gamma} (\ell', v').$$

Proposition 2 is a direct consequence of the definition of time progress in the planning semantics which is a restriction of the one of the standard semantics.

Corollary 1. *If a state (ℓ, v, π) is reachable in the planning semantics the state (ℓ, v) is reachable in the standard semantics.*

Corollary 1 is obtained from Propositions 1 and 2 and the fact that **plan**$-$
transitions (labeled by $\mathbf{plan}(\alpha, \delta)$) affect only the plan π in states (ℓ, v, π) of the planning semantics.

Definition 7 (Weak Simulation)**.** *A weak simulation over $A = (\mathrm{Q}_A, \sum \cup \{\beta\}, \rightarrow_A)$ and $B = (\mathrm{Q}_B, \sum \cup \{\beta\}, \rightarrow_B)$ is a relation $R \subseteq \mathrm{Q}_A \times \mathrm{Q}_B$ such that we have: $\forall (q, r) \in R, a \in \sum . q \overset{a}{\rightarrow}_A q' \implies \exists r' : (q', r') \in R \wedge r \overset{\beta^* a \beta^*}{\longrightarrow}_B r'$ and $\forall (q, r) \in R : q \overset{\beta}{\rightarrow}_A q' \implies \exists r' : (q', r') \in R \wedge r \overset{\beta^*}{\longrightarrow} r'$. B simulates A, denoted by $A \sqsubseteq_R B$, means that B can do everything A does.*

The definition of weak simulation is based on the unobservability of $\beta-$transitions. In our case, $\beta-$transitions corresponds to **plan**$-$transitions.

Corollary 2. *$LTS_p \sqsubseteq_R LTS_g$ with $R = \{(q, \pi); q) \in \mathrm{Q}_p \times \mathrm{Q}_g\}$.*

Corollary 2 corresponds to a notion of correctness of the planning semantics: any execution in the planning semantics corresponds to an execution in the standard semantics. In addition, if interactions are allowed to be planned with relative execution times of 0 (i.e. $h_{\min} = 0$) then the planning semantics simulates the standards semantics [8]. However, this is no longer true in general if $h_{\min} > 0$ which means that not all execution sequences of the standard semantics are preserved by the planning semantics. Notice that the planning semantics also preserves non zenoness of the standard semantics (by Corollary 2 and the fact that it is not possible to have infinite sequences of **plan**-transitions without interaction execution).

Proposition 3. *For a given composition $\gamma(B_1, \cdots, B_n)$, if its standard semantics is deadlock-free then its planning semantics is deadlock-free if and only if it is action-time-lock free.*

Proposition 3. We use proof by contradiction to proof Proposition 3. Let us assume that the system under the standard (resp. planning) semantics is deadlock free (resp. action-time-lock free). Let (ℓ, v, π) be a reachable deadlock state of the planning semantics. We have:

$$\nexists \sigma \in \gamma \cup \{\mathbf{plan}\}.(\ell, v, \pi) \overset{\sigma}{\rightsquigarrow}_{>\gamma} (\ell', v', \pi') \vee (\ell, v, \pi) \overset{\delta}{\rightsquigarrow}_{>\gamma} (\ell, v + \delta, \pi - \delta) \overset{\sigma}{\rightsquigarrow}_{>\gamma} (\ell', v', \pi')$$

We denote by $wait(\ell, v, \pi)$ the set of allowed waiting times at state (ℓ, v, π), that is:

$$wait(\ell, v, \pi) = \{0\} \cup \{\delta \in \mathbb{R}_{>0} | (\ell, v, \pi) \overset{\delta}{\rightsquigarrow}_{>\gamma} (\ell, v + \delta, \pi - \delta)\}$$

We also put $\max(wait(\ell, v, \pi))$ to denote the maximal waiting time at state (ℓ, v, π).

Lemma 2. *Let (ℓ, v, π) be a reachable state of the planning semantics. For $k \in \mathbb{R}_{\geq 0}$, such that $k = \max(wait(\ell, v, \pi))$, we have the following properties:*

P1 *If $k < +\infty$ then $(\ell, v, \pi) \overset{k}{\rightsquigarrow}_{>\gamma} (\ell, v + k, \pi - k) \wedge wait(\ell, v + k, \pi - k) = \{0\}$*

P2 *If $\pi \neq +\infty$ then $k < +\infty \wedge k \leq \min \pi$*

We distinguish 2 cases:

Case 1: no interaction was planned (i.e. $\pi = +\infty$)
By definition of the planning semantics, it is clear that for $\pi = +\infty$, there is no interaction to execute from (ℓ, v, π) or any of its successor $(\ell, v + \delta, \pi - \delta)$.

1. $wait(\ell, v, \pi) = \{0\}$:
   This means that time progress is not allowed at state (ℓ, v, π). We also have $\nexists \sigma \in \{\textbf{plan}\}.(\ell, v, \pi) \overset{\sigma}{\rightsquigarrow}_\gamma (\ell', v', \pi')$ (deadlock assumption). We can conclude that (ℓ, v, π) is a reachable action-time-lock state, which contradicts the assumption that the system under the planning semantics is action-time-lock free.

2. $wait(\ell, v, \pi) \neq \{0\}$:

   (a) $\max(wait(\ell, v, \pi)) = +\infty$:

   Lemma 3. *Let (ℓ, v, π) be a reachable state of the planning semantics. If $\forall \delta \in \mathbb{R}_{>0}.(\ell, v, \pi) \overset{\delta}{\rightsquigarrow}_\gamma (\ell, v + \delta, \pi - \delta) \wedge \neg Plannable(\alpha)$ at (ℓ, v, π), then we have $\neg Enabled(\alpha)$ at $(\ell, v + \delta, \pi - \delta)$ with $\delta \geq h_{\min}$.*

   Using the deadlock assumption and P1 of Lemma 3 we can deduce that $\exists \delta \geq h_{\min}.(\ell, v, \pi) \overset{\delta}{\rightsquigarrow}_\gamma (\ell, v + \delta, \pi - \delta)$ such that we have: $\bigvee_{\alpha \in \gamma} \neg \not\swarrow (Enabled(\alpha))$. Finally, since the state $(\ell, v + \delta, \pi - \delta)$ is reachable in the standard semantics, and by evaluating the deadlock characterization 2 on state $(\ell, v + \delta, \pi - \delta)$, we can conclude that the system under the standard semantics deadlocks, which contradicts the assumption of deadlock freedom of the system under the standard semantics.

   (b) $\max(wait(\ell, v, \pi)) < +\infty$:
   Considering that $k = \max(wait(\ell, v, \pi))$, then we have by P1 of Lemma 2: $(\ell, v, \pi) \overset{k}{\rightsquigarrow}_\gamma (\ell, v + k, \pi - k) \wedge wait(\ell, v + k, \pi - k) = 0$. Using the deadlock assumption we have: $\bigvee_{\alpha \in \gamma} \neg Plannable(\alpha)$ at state $(\ell, v + k, \pi - k)$. Since the system cannot progress beyond this state ($wait(\ell, v + k, \pi - k) = 0$), we can conclude that $(\ell, v + k, \pi - k)$ is a reachable action-time-lock state, which contradicts the assumption that the system under the planning semantics is action-time-lock free.

Case 2: at least an interaction was planned (i.e. $\pi \neq +\infty$)
Considering that $k = \max(wait(\ell, v, \pi))$, since $\pi \neq +\infty$, we have by P2 of Lemma 2: $k < +\infty \wedge k \leq \min \pi$. Using the deadlock assumption we can infer that $k < \min \pi$, since no executions is possible from (ℓ, v, π) or any of its successors. This means that $(\ell, v + k, \pi - k)$ is a reachable action-time-lock state, which contradicts the assumption that the system under the planning semantics is action-time-lock free. □ □

Proposition 4. *A state (ℓ, v, π) of the planning semantics is an action-time-lock if and only if $\pi > 0$ and:*

$$\bigwedge_{\alpha \notin conf(\pi)} \neg Plannable(\alpha) \ \wedge \bigvee_{\substack{\ell_i \in \mathcal{L}_i \\ B_i \notin \text{part}(\pi)}} \mathsf{at}(\ell_i) \ \wedge (urg(\ell_i) + h_{\min}).$$

As shown in Example 2, the planning semantics may also introduce deadlocks. The source of deadlocks is twofold: *(i)* due to communication delays consecutive execution in a component are separated by at least $h_{\min}$ units of time which may be incompatible with its timings constraints, and *(ii)* conditions for planning interactions are too permissive as they only take into account timing constraints of participating components whereas they may block additional components, namely the ones participating in conflicting interactions. In the rest of the paper we study how to generate planning strategies for preserving deadlock-freedom. They restrict the **plan**-transitions of the planning semantics so that deadlock states become unreachable. Such a strategy may not exist when timing constraints cannot accommodate with the communication delays $h_{\min}$. Notice that by Proposition 3 it is sufficient to focus on the action-time-locks of the planning semantics for systems that are initially deadlock-free.

4 Enforcing Deadlock-Free Planning

As explained above, the planning semantics is based on local conditions for planning interactions and may exhibit deadlocks even when the system is deadlock-free with the standard semantics. Such deadlocks are partly due to the fact that planning an interaction may block, in addition to the participating components, extra components whose timing constraints are not considered by these local conditions. In this section, we investigate simple execution strategies that only restrict the horizon used for planning interactions with upper bounds. By reducing the period of time during which components are blocked, they tend to remove deadlocks from the reachable states. In addition we provide sufficient conditions for checking deadlock-freedom of the planning semantics subject to upper bounded horizons. In what follows, we consider a composition of components $S = \gamma(B_1, \cdots, B_n)$ such that it is deadlock-free in the standard semantics.

4.1 Planning with Upper Bounded Horizon

We slightly modify the planning semantics of Definition 6 of Section 3.1 by considering upper bounds $h_{\max} : \gamma \to \mathbb{Z}_{\geq 0} \cup \{+\infty\}$ for interactions such that for any interaction α we have $h_{\max}(\alpha) \geq h_{\min}$. In the modified semantics interactions α can be planned only using planning horizon δ satisfying $h_{\min} \leq \delta \leq h_{\max}(\alpha)$. Notice that planning semantics as presented in Section 2 corresponds to the choice $h_{\max}(\alpha) = +\infty$ for all interactions $\alpha \in \gamma$. The predicate $\mathit{Plannable}(\alpha)$ introduced in Section 3.1 and characterizing states from which α can be planned in a delay respecting the constraints on the planning horizon becomes:

$$\mathit{Plannable}(\alpha) \Leftrightarrow \exists \delta \in \mathbb{R}_{\geq 0} \,.\, h_{\min} \leq \delta \leq h_{\max}(\alpha) \ \wedge \ \mathit{Plannable}(\alpha, \delta),$$

more formally:

$$\bigvee_{\{\ell_i \in \mathcal{L}_i\}_{i \in I}} \Big(\bigwedge_{i \in I} \mathsf{at}(\ell_i) \Big) \wedge \Big(\swarrow_{h_{\min}}^{h_{\max}(\alpha)} \big[\bigwedge_{i \in I} \mathit{guard}(a_i, \ell_i) \big] \Big).$$

It can easily be shown that all results of Section 3.2 also apply to this variant of the planning semantics. The characterization of action-time-locks of Proposition 4 is still valid provided $\mathit{Plannable}(\alpha)$ is computed using its refined version presented above. In the rest of the section we always consider this modified version of the planning semantics.

4.2 Checking Deadlock-Freedom

As explained in Section 3.2, action-time-lock-freedom is a sufficient condition for deadlock-freedom of the planning semantics. By Proposition 4 a state (ℓ, v, π) is an action-time-lock in the planning semantics iff $\pi > 0$, that is, no interaction can be executed from (ℓ, v, π), and:

$$\bigwedge_{\alpha \in \gamma \setminus conf(\pi)} \neg \mathit{Plannable}(\alpha) \ \wedge \bigvee_{\substack{\ell_i \in \mathcal{L}_i \\ B_i \notin \mathrm{part}(\pi)}} \mathsf{at}(\ell_i) \ \wedge (\mathit{urg}(\ell_i) + h_{\min}).$$

The above predicate characterizes the fact that no interaction can be planned, nor time can progress in component $B_i \notin \mathrm{part}(\pi)$. Consequently, we deduce that a necessary condition of action-time-lock is, besides $\pi > 0$, the existence of a component $B_i \notin \mathrm{part}(\pi)$ such that time cannot progress in B_i and B_i cannot be planned in an interaction, that is:

$$\bigwedge_{\alpha \in \gamma(B_i) \setminus conf(\pi)} \neg \mathit{Plannable}(\alpha) \ \wedge \bigvee_{\ell_i \in \mathcal{L}_i} \mathsf{at}(\ell_i) \ \wedge (\mathit{urg}(\ell_i) + h_{\min}).$$

where $\gamma(B_i)$ denotes the subset of interactions in which B_i participate, that is, $\gamma(B_i) = \{\beta \in \gamma \mid B_i \in \mathrm{part}(\beta)\}$. Notice that the above equation strongly depends on the plan π, which is diffcult to characterize in practice. The following theorem proposes sufficient plan-independent conditions characterizing action-time-lock states of the planning semantics.

Theorem 1. *Let ϕ be the following predicate:*

$$\bigvee_{1\leq i\leq n}\Big[\bigvee_{\ell_i\in\mathcal{L}_i}\mathsf{at}(\ell_i)\ \wedge(\mathit{urg}(\ell_i)+h_{\min})\ \wedge\bigwedge_{\alpha\in\gamma(B_i)}\Big(\neg\mathit{Plannable}(\alpha)\ \vee\bigvee_{\substack{\beta\in\mathit{conf}(\alpha)\\ B_i\notin\mathrm{part}(\beta)}}\overline{\mathit{Plannable}(\beta)}\Big)\Big].$$

We prove that a reachable action-time-lock state (ℓ,v,π), satisfies ϕ.

Theorem 1. A reachable action-time-lock state of the planning semantics satisfies:

$$(\pi>0)\ \wedge\bigwedge_{\alpha\in\gamma(B_i)\setminus\mathit{conf}(\pi)}\neg\mathit{Plannable}(\alpha)\ \wedge\bigvee_{\substack{\ell_i\in\mathcal{L}_i\\ B_i\notin\mathrm{part}(\pi)}}\mathsf{at}(\ell_i)\ \wedge(\mathit{urg}(\ell_i)+h_{\min}).$$

In order to approximate the above equation, we distinguish two cases:

Case 1: no interaction was planned (i.e. $\pi=+\infty$)
From $\pi=+\infty$ we deduce directly there exists an urgent component B_i such that no interaction α involving B_i can be planned, that is:

$$\bigvee_{1\leq i\leq n}\Big(\bigvee_{\ell_i\in\mathcal{L}_i}\mathsf{at}(\ell_i)\ \wedge(\mathit{urg}(\ell_i)+h_{\min})\ \wedge\bigwedge_{\alpha\in\gamma(B_i)}\neg\mathit{Plannable}(\alpha)\Big).\tag{1}$$

Case 2: at least an interaction was planned (i.e. $\pi\neq+\infty$)
In this case there exists an urgent component $B_i\notin\mathrm{part}(\beta)$ such that no interaction α involving B_i can be planned either because it conflicts with a planned interaction β or because $\mathit{Plannable}(\alpha)$ is not satisfied, that is $\exists\beta\in\pi,\exists B_i\notin\mathrm{part}(\beta)$ satisfying:

$$(0<\pi(\beta)<+\infty)\ \wedge\bigwedge_{\alpha\in\gamma(B_i)\setminus\mathit{conf}(\beta)}\neg\mathit{Plannable}(\alpha)\ \wedge\bigvee_{\substack{\ell_i\in\mathcal{L}_i\\ B_i\notin\mathrm{part}(\beta)}}\mathsf{at}(\ell_i)\ \wedge(\mathit{urg}(\ell_i)+h_{\min}).$$

or equivalently $\exists\beta\in\pi,\exists B_i\notin\mathrm{part}(\beta)$ satisfying:

$$\bigvee_{\substack{\ell_i\in\mathcal{L}_i\\ B_i\notin\mathrm{part}(\beta)}}\mathsf{at}(\ell_i)\ \wedge(\mathit{urg}(\ell_i)+h_{\min})\ \wedge\bigwedge_{\alpha\in\gamma(B_i)}\Big(\neg\mathit{Plannable}(\alpha)\vee\big(\beta\in\mathit{conf}(\alpha)\wedge(0<\pi(\beta)<+\infty)\big)\Big).$$

By Noticing that we have the following implication between quantifiers $\exists y,\forall x.Q(x,y)\implies\forall x,\exists y.Q(x,y)$, we can deduce that the above condition implies:

$$\bigvee_{1\leq i\leq n}\Big[\bigvee_{\ell_i\in\mathcal{L}_i}\mathsf{at}(\ell_i)\ \wedge(\mathit{urg}(\ell_i)+h_{\min})\ \wedge\bigwedge_{\alpha\in\gamma(B_i)}\Big(\neg\mathit{Plannable}(\alpha)\ \vee\bigvee_{\substack{\beta\in\mathit{conf}(\alpha)\\ B_i\notin\mathrm{part}(\beta)}}0<\pi(\beta)<+\infty\Big)\Big].$$

As $\pi>0$, and if we consider only reachable action-time-locks, we have $0<\pi(\beta)\leq h_{\max}(\beta)$, and by Lemma 1 we have $\mathit{Plannable}(\beta,\pi(\beta))$. That is, β satisfies $\mathit{Plannable}(\beta)$ in which the lower bound $h_{\min}$ is replace by the strict lower bound 0, i.e.:

$$\overline{\mathit{Plannable}(\beta)}\Leftrightarrow\exists\delta>0\,.\,\delta\leq h_{\max}(\beta)\ \wedge\ \mathit{Plannable}(\beta,\delta).$$

Then, the above equation becomes:

$$\bigvee_{1\leq i\leq n}\Big[\bigvee_{\ell_i\in\mathcal{L}_i}\mathsf{at}(\ell_i)\ \wedge(\mathit{urg}(\ell_i)+h_{\min})\ \wedge\bigwedge_{\alpha\in\gamma(B_i)}\Big(\neg\mathit{Plannable}(\alpha)\ \vee\bigvee_{\substack{\beta\in\mathit{conf}(\alpha)\\ B_i\notin\mathrm{part}(\beta)}}\overline{\mathit{Plannable}(\beta)}\Big)\Big].\tag{2}$$

By remarking that Equations 1 implies Equation 2, we can conclude that an action-time-lock of the planning semantics satisfies:

$$\bigvee_{1\leq i\leq n}\Big[\bigvee_{\ell_i\in\mathcal{L}_i}\mathsf{at}(\ell_i)\ \wedge(\mathit{urg}(\ell_i)+h_{\min})\ \wedge\bigwedge_{\alpha\in\gamma(B_i)}\Big(\neg \mathit{Plannable}(\alpha)\ \vee\bigvee_{\substack{\beta\in \mathit{conf}(\alpha)\\ B_i\notin \mathrm{part}(\beta)}}\overline{\mathit{Plannable}(\beta)}\Big)\Big].$$

□ □

In order to attest that planning interactions does not introduce deadlocks, we use an SMT solver to check the satisfiability of ϕ. As explained earlier, a given system is deadlock-free under the restricted planning semantics if $Reach(LTS_p)\wedge\phi$ is unsatisfiable. Since $Reach(LTS_p)\subset Reach(LTS_g)$ (Corollary 2), we can verify the above on $Reach(LTS_g)$.

Effectively, we do not compute $Reach(LTS_g)$ to avoid the combinatorial explosion problem, inherent to composition of timed automata. In fact, we rather build an over-approximation, $\widetilde{Reach(LTS_g)}$ of the latter, an use it during our verification. The computed over-approximation combines the reachable states of individual components. However, in general not all combinations are reachable since components are not fully independent and may synchronize together through interactions. Moreover, individual invariants alone do not express the fact that time progresses the same way in components. In order to capture these additional information, we use different kind of invariants: *(1)* linear invariants [5] that capture the constraints on location configurations of the components induced by their synchronizations (interactions), and *(2)* interaction inequalities for history clocks [12], allowing to relate the local constraints obtained individually on components, as well as separation constraints for interaction [12] that describes dis-equalities (or separations) constraints between interactions.

5 Planning Semantics as Real-Time Controller Synthesis

In Section 4, we presented a method that provides execution strategies by restricting the upper bounds planning horizon for each interaction. Since the given approach checks a sufficient condition for deadlock-freedom, it may give false-positives results, that is, it will find a state verifying condition ϕ of Theorem 1 but which is not deadlock state. In such cases, an alternative is to tackle the problem as a real-time controller synthesis problem. Expressing the planning problem as a real-time controller synthesis problem is not an easy task. Effectively, in order to encode the planning in timed automata, horizon values must be integer. Moreover, for timing constraints not restricted with upper bounds, we end up by an infinity of plan transitions. Consequently, the first thing to do is to discretize the planning horizons in order to obtain finite values in $\mathbb{Z}_{>0}$. For a matter of simplicity, we restrict the planning horizon values by putting for each interaction $\alpha\in\gamma/h_{\max}(\alpha)=h_{\min}$. Notice that in this case, the method checks if a given system is deadlock-free when being planned with a $h_{\min}$ horizon.

Real-time controller synthesis is a common method used to extract an execution strategy from formal specifications satisfying certain properties. Usually, these properties express the reachability (resp. non-reachability) of a set of wining states (resp. bad states). In case of planning interactions with a $h_{\min}$ horizon, the idea is to restrict the transition relation so that all the remaining behaviors do not lead to states where a component is urgent and no possible execution including this component may occur. This can be formalized as a reachability games for a timed game automaton [6], where the main idea consists in trying to find an execution strategy guaranteeing that a given set of namely *bad states* of the system are never reached. In order to apply this approach, it is required to encode the planning of interactions and their effects on the system, that is, *(i)* encode interactions planning as synchronizations between components, *(ii)* reserve the components of the planned interactions until their chosen execution date, i.e, keep track of the planned interactions and their executions date, and *(iii)* characterize the set of bad states. Thereafter, tools such as UPAALL-Tiga [3] can be used to find an execution strategy of the planning semantics avoiding the set of bad states, that is, deadlock states.

5.1 Planning Zones

From Equation 4, we can see that the clocks values for planning an interaction are calculated at a global level, that is, by applying the $\swarrow_{h_{\min}}^{h_{\max}(\alpha)}$ on the conjunction of its participating actions timing constraints. Notice that for a timing constraint $g = g_1 \wedge g_2$, we have:

$$\swarrow_{h_{\min}}^{h_{\min}} g = \swarrow_{h_{\min}}^{h_{\min}} (g_1 \wedge g_2) \not\equiv \swarrow_{h_{\min}}^{h_{\min}} g_1 \wedge \swarrow_{h_{\min}}^{h_{\min}} g_2$$

The above equation bears out the fact that planning states must be encoded on the composition of the system model and not on individual components. Therefore, in what follows, we consider models with interactions having timing constraints on up to one of its components participating action. In fact, considering interactions with up to one action with timing constraints, will allow to encode the planning on individual components that, additionally to the defined synchronizations (interactions), will also synchronize their planning actions.

The idea is to split each transition of the initial model into two transitions: *(1)* a planning transition, *(2)* followed by an execution transition $h_{\min}$ after the plan transition being performed. This is achieved by equipping each component with an additional clock x_p, that will be used to track the time. Additionally, time progress conditions must also be translated to enforce planning at the latest $h_{\min}$ units of time before their expiry. Figure 2 depicts such transformation:

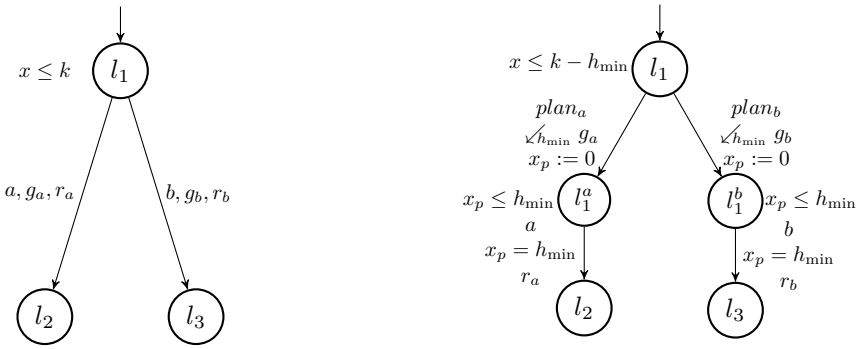

Figure 2: A Timed system

Definition 8 (Planning Timed Automaton)**.** *Given n timed components $\mathcal{B}_i = (\mathcal{L}_i, \ell_0^i, \mathcal{A}_i, \mathcal{T}_i, \mathcal{X}_i, \mathcal{I}_i)$ synchronizing through the interaction set γ such that, for each interaction $\alpha \in \gamma$ the guard of α is equal to the guard of one of its included actions. We define the corresponding planning model as the composition of the n timed automata $\mathcal{B}_i^p = (\mathcal{L}_i^p, \ell_0, \mathcal{A}_i \cup \mathcal{P}_i, \mathcal{T}_i^p, \mathcal{X}_i \cup \{x_i^p\}, \mathcal{I}_i^p)$, w.r.t the interaction set $\gamma \cup \mathcal{P}$, where:*

- $\mathcal{P}_i = \cup_{a \in \mathcal{A}_i} p_a$ *is the set of* Planning Actions
- $\mathcal{P} = \{p_\alpha = \{p_{a_i}\}_{i \in I} | \alpha \in \gamma \wedge \alpha = \{a_i\}_{i \in I}\}$ *is the set of* Planning Interactions
- x_i^p *is a* Tracking Clock *for interactions execution in each component*
- $\mathcal{L}_i^p = (\mathcal{L}_i \cup \mathcal{L}_i^p)$ *is the set of control locations, where $\mathcal{L}_i^p$ is the set of locations following planning actions*
- $\mathcal{T}_i^p$ *is such that for each $(\ell, a, g, r, \ell') \in \mathcal{T}_i$, $a \in \alpha$:*
  - *if $g_\alpha \neq true$ we have:*

$$\textit{Planning transitions: } \begin{cases} \ell \xrightarrow{p_a, true, \emptyset} \ell^a, \textit{ if } g = true \\ \ell \xrightarrow{p_a, \swarrow_{h_{\min}}^{h_{\min}} g, r(x_i^p)} \ell^a, \textit{ otherwise} \end{cases}$$

$$\textit{Execution transitions: } \begin{cases} \ell^a \xrightarrow{a, true, r} \ell', \textit{ if } g = true \\ \ell^a \xrightarrow{a, x_i^p = h_{\min}, r} \ell', \textit{ otherwise} \end{cases}$$

where $\ell^a \in \mathcal{L}_i^p$.

– *if* $g_\alpha = true$, *we choose one action* $b \in \alpha$:

Planning transitions: $\begin{cases} \ell \xrightarrow{p_a, true, \emptyset} \ell^a, \ \textit{if } a \neq b \\ \ell \xrightarrow{p_a, true, r(x_i^p)} \ell^a, \ \textit{otherwise} \end{cases}$

Execution transitions: $\begin{cases} \ell^a \xrightarrow{a, true, r} \ell', \ \textit{if } a \neq b \\ \ell^a \xrightarrow{a, x_i^p = h_{\min}, r} \ell', \ \textit{otherwise} \end{cases}$

- $\mathcal{I}_i^p$ *is the set of* Location Invariants *, such that:* $\mathcal{I}_i^p(\ell) = \begin{cases} \mathcal{I}_i(\ell) - h_{\min}, \ \textit{if } \ell \in \mathcal{L}_i \\ x_a \leq h_{\min}, \ \textit{if } \ell = \ell^a \in \mathcal{L}_i^p, \end{cases}$

For a composition $\gamma(B_1, \cdots, B_n)$, let $LTS_p = (\mathrm{Q}_p, \gamma \cup \mathbb{R}_{>0} \cup \{\textbf{plan}\}, \rightsquigarrow_\gamma)$ its respective labeled transition system under the planning semantics. Let also $LTS_{p'} = (\mathrm{Q}_{p'}, \gamma' \cup \mathbb{R}_{>0}, \longrightarrow_{\gamma'})$, where $\gamma' = \gamma \cup \mathcal{P}$, be the corresponding labeled transition system of its planning model under the standard semantics.

Lemma 4. $LTS_{p'} \sqsubseteq LTS_g$.

The above Lemma is obtained by restricting the horizons values in the planning semantics to $h_{\min}$.

Once interactions planning encoded, one last thing to do is to add the set of bad states to each planning automaton (if needed) and find a strategy to avoid those states. Figure 3 depicts the corresponding planning automaton for a task component of the running example w.r.t Definition 8. Here, x_p is the clock used for tracking the execution date of an interaction within the task component. Locations suffixed by *p*, correspond to locations following planning actions, whereas locations ending with *err* defines the bad states, that is, states with urgent time progress condition(s) and no possible execution removing the urgency. In this example, we chose a $h_{\min}$ value of 2.

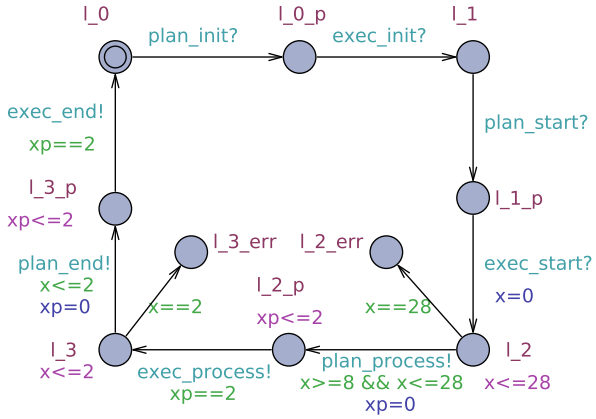

Figure 3: Planning automaton for the task component

5.2 Discussion

In this section, we explained how the problem of planning interactions can be formalized into a real-time controller synthesis approach. However, this approach has some drawbacks. In order to encode planning of interactions in components as timed automata, this approach restricts its scope to discretized horizon values which results in having less control over the planning dates of interactions, and leads in case of a high number of discretized values, to an explosion in the number of planning transitions. Additionally, it considers only a class of systems where interactions have timing constraints on up to one of its participating components action, otherwise, the planning should be encoded on the composition which represents a tedious work.

6 Implementation and Experiments

For our experiments, we used the BIP framework [2] as a modeling language to define systems and their synchronizations. BIP (Behavior, Interaction, Priority), is a component-based framework with a rigorous semantics that allows to model systems as a set of atomics components coordinating their behaviors through multiparty interactions. The proposed method has been implemented as a middleend filter of the real-time BIP compiler, aiming to generate information that could be used by the backend during the code generation process. The presented approach requires a substantial knowledge of the system model, since the satisfiability verification of $\widetilde{Reach(LTS_g)} \wedge \phi$, needs a deep analysis of the system, in order to generate the predicates used in the latter. The implementation takes a BIP model as input. Then, the Frontend of the BIP compiler builds the system metamodel and creates an abstract representation of it. Thereafter, the named middleend planner performs a model analysis in order to construct the predicates needed in ϕ, while keeping interactions upper horizons as free variables. In order to ease the verification process and remove the back and forth process between the predicates generation and their verification, we fully integrated the generation of the invariants over-approximation in our middleend. Finally, for each non *true* component time progress condition a Yices [9] file, including system invariants and the predicates approximating action-time-locks, is generated. The Yices solver then checks the satisfiability of $\widetilde{Reach(LTS_g)} \wedge \phi$. If the result is unsatisfiable, then planning does not miss the deadline expressed by the considered time progress condition. Otherwise, if the result is satisfiable, Yices generates a counter-example. Since for each interaction $h_{\min}$ is a fixed value defined by the worst case estimation of the target platform communication delays, and due to the monotony of ϕ on $h_{\max}$ side, the generated counter-example is used to find the maximal value of $h_{\max}$ satisfying the above condition. Due to the early development stage of the implementation, this part is done manually, but we intend to replace it by a binary search algorithm in the near future. For the experiments, we chose three additional benchmarks: Temperature controller, Controller workers and an adaptation of the Train gate controller. A complete description of the benchmarks is available in [12].

Table 1: Detailed Results of the Task Manager Experiments

$h_{\min}$	$h_{\max}(\alpha_1), h_{\max}(\alpha_2)$	$h_{\max}(\alpha_3), h_{\max}(\alpha_4)$	$h_{\max}(\alpha_5), h_{\max}(\alpha_6)$	$h_{\max}(\alpha_7), h_{\max}(\alpha_8)$
4	UB	UB	29	UB
3	UB	UB	28	UB
2	UB	UB	27	UB
1	UB	UB	26	UB

Table 1 depicts the values $h_{\max}$ for each interaction of the running example, obtained while fixing $h_{\min}$. Notice that the symmetry of the system implies the same $h_{\max}$ for interactions $\alpha_i, \alpha_{i+1}, i \in \{1, 3, 5, 7\}$. By remarking that location ℓ_3^2 (resp. ℓ_3^3) has a time progress condition $x \leq 4$ (resp. $y \leq 4$), and by observing that the clock x is reset on the transition leading to this location, we can conclude that planning the system with $h_{\min} > 4$ will lead to an action-time-locks. Particularly, in Example 2, for $h_{\min} = 2$ interaction α_6 was planned with a horizon $\delta = 30$, and consequently, leads to a action-time-lock state. Our method detects such cases and thus, find that the maximum horizon for this interaction is 27. Likewise, the $h_{\max}$ for interactions α_2, α_4 and α_8 (resp. α_1, α_3 and α_7) is found to be unbounded.

Table 2 summarizes the experiments obtained on the benchmarks stated above. For each model, the column $\max h_{\min}$ presents the maximum value for $h_{\min}$ for which the system is deadlock-free in the planning semantics. On the other hand, column $h_{\max}$ shows if a restriction on the upper horizons is required to avoid deadlocks.

7 Conclusion and Future Work

We presented a local planning semantics for scheduling real-time systems in a distributed context. The proposed approach intends to mitigate the effect of communication delays through planning interaction ahead. Sufficient deadlock-freedom condition has been proved, a compositional verification method for checking action-time-lock-freedom was provided, and a simple execution strategy, based on restricting

Table 2: Results of Experiments

Model	$\max h_{\min}$	$h_{\max}$
Task Manager	4	B
Temperature Controller	450	UB
Train Gate	10	B
Controller Worker	5	UB

upper bounds horizons planning of interactions, has been presented. Additionally, a formalization of the planning problem as a real-time controller synthesis approach has been provided. This work shows how to express the planning semantics as timed game automata and highlights the encountered issues met during the formalization.

This approach opens a number of directions for future work. In case of action-time-locks of the planning semantics, a first idea consistes to study their origins and derive a refinement method for models in order to take into account the communication delays. Another interesting direction is the characterization of the reachable states of the planning semantics. Instead of using an over-approximation of systems reachable states under the standard semantics, a more accurate approach could be to define a method for deriving invariants w.r.t the planning semantics. Finally, an interesting idea is to investigate how scheduler(s) can benefits from the information provided by the presented method in order to optimize their scheduling policy.

References

[1] R. Alur and D. L. Dill. A theory of timed automata. *Theor. Comput. Sci.*, 1994. 2

[2] A. Basu, L. Mounier, M. Poulhies, J. Pulou, and J. Sifakis. Using bip for modeling and verification of networked systems – a case study on tinyos-based networks. In *Sixth IEEE International Symposium on Network Computing and Applications (NCA 2007)*, pages 257–260, July 2007. 6

[3] G. Behrmann, A. Cougnard, A. David, E. Fleury, K. G. Larsen, and D. Lime. Uppaal-tiga: Time for playing games! In *Computer Aided Verification, 19th International Conference, CAV 2007, Berlin, Germany, July 3-7, 2007, Proceedings*, pages 121–125, 2007. 5

[4] J. Bengtsson and W. Yi. On clock difference constraints and termination in reachability analysis of timed automata. In *ICFEM*, 2003. 2

[5] S. Bensalem, M. Bozga, B. Boyer, and A. Legay. Incremental generation of linear invariants for component-based systems. In *2013 13th International Conference on Application of Concurrency to System Design*, pages 80–89, 2013. 4.2

[6] F. Cassez, A. David, E. Fleury, K. G. Larsen, and D. Lime. *Efficient On-the-Fly Algorithms for the Analysis of Timed Games*, pages 66–80. Springer Berlin Heidelberg, Berlin, Heidelberg, 2005. 5

[7] R. N. Charette. This car runs on code. *IEEE Spectrum*, 2009. 1

[8] M. Dellabani, J. Combaz, M. Bozga, and S. Bensalem. Local planning of multiparty interactions with bounded horizons. In *FM 2016: Formal Methods - 21st International Symposium, Limassol, Cyprus, November 9-11, 2016, Proceedings*, pages 199–216, 2016. 1, 3.2

[9] B. Dutertre and L. de Moura. The yices smt solver. Technical report, SRI International, 2006. 6

[10] T. A. Henzinger, X. Nicollin, J. Sifakis, and S. Yovine. Symbolic model checking for real-time systems. *Inf. Comput.*, 1994. 2

[11] H. Kopetz. An integrated architecture for dependable embedded systems. In *Proceedings of the 23rd IEEE International Symposium on Reliable Distributed Systems*, SRDS '04, pages 160–161, 2004. 1

[12] S. B. Rayana, L. Astefanoaei, S. Bensalem, M. Bozga, and J. Combaz. Compositional verification for timed systems based on automatic invariant generation. *Logical Methods in Computer Science*, 11(3), 2015. 4.2, 6

[13] A. Triki, B. Bonakdarpour, J. Combaz, and S. Bensalem. Automated conflict-free concurrent implementation of timed component-based models. In *NASA Formal Methods - 7th International Symposium, NFM 2015, Pasadena, CA, USA, April 27-29, 2015, Proceedings*, pages 359–374, 2015. 3

[14] S. Tripakis. *The analysis of timed systems in practice*. PhD thesis, Joseph Fourier University, 1998. 2, 2