Theoretical Computer Science 612 (2016) 1–22

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

Parameterized verification of time-sensitive models of ad hoc network protocols ☆

Parosh Aziz Abdulla [a], Giorgio Delzanno [b,*], Othmane Rezine [a], Arnaud Sangnier [c], Riccardo Traverso [d]

[a] Uppsala University, Sweden
[b] University of Genova, Italy
[c] LIAFA, Univ. Paris Diderot, CNRS, France
[d] FBK, Trento, Italy

ARTICLE INFO

Article history:
Received 1 February 2014
Received in revised form 8 July 2015
Accepted 25 July 2015
Available online 7 August 2015
Communicated by J.-F. Raskin

Keywords:
Parameterized verification
Timed automata
Ad hoc networks
Graphs
Decidability
Well structured transition systems

ABSTRACT

We study decidability and undecidability results for parameterized verification of a formal model of timed Ad Hoc network protocols. The communication topology is defined by an undirected graph and the behaviour of each node is defined by a timed automaton communicating with its neighbours via broadcast messages. We consider parameterized verification problems formulated in terms of reachability. In particular we are interested in searching for an initial configuration from which an individual node can reach an error state. We study the problem for dense and discrete time and compare the results with those obtained for (fully connected) networks of timed automata.

1. Introduction

In recent years there has been an increasing interest in automated verification methods for ad hoc networks, see e.g. [18,24,23,11,12]. Ad Hoc Networks (AHN) consist of wireless hosts that, in absence of a fixed infrastructure, communicate sending broadcast messages. In this context, protocols are supposed to work independently from a specific configuration of the network. Indeed, discovery protocols are often applied in order to identify the vicinity of a given node. In the AHN model proposed in [11] undirected graphs are used to represent a network in which each node executes an instance of a fixed (untimed) interaction protocol based on broadcast communication. Since individual nodes are not aware of the network topology, in the ad hoc setting it is natural to consider verification problems that are parametric in the size and shape of the initial configuration as in [11].

In this paper we introduce a new model of distributed systems obtained by enriching the AHN model of [11] with time-sensitive specification of individual nodes. In the resulting model, called Timed Ad Hoc Networks (TAHN), the connection

☆ This work is partially supported by the ANR national research program ANR-14-CE28-0002 PACS and by the MIUR PRIN Italian research program CINA.
* Corresponding author.
E-mail addresses: parosh@it.uu.se (P.A. Abdulla), giorgio.delzanno@unige.it (G. Delzanno), sangnier@liafa.univ-paris-diderot.fr (A. Sangnier), rtraverso@fbk.eu (R. Traverso).

http://dx.doi.org/10.1016/j.tcs.2015.07.048
0304-3975/

topology is still modelled as a graph in which nodes communicate via broadcast messages but the behaviour of a node is now defined as a timed automaton. More in detail, each node has a finite set of clocks which all advance at the same rate and transitions describing the behaviour of the nodes are guarded by conditions on clocks and have also the ability to reset clocks.

Following [11,12], we study the decidability status of the parameterized reachability problem taking as parameters the initial configuration of a TAHN, i.e., we aim at checking the existence of an initial configuration that can evolve using continuous and discrete steps into a configuration exposing a given local state (usually representing an error). Our model presents similarities with Timed Networks introduced in [2]. A major difference between TAHN and Timed Networks lies in the fact that in the latter model the connection topology is always a fully-connected graph, i.e., broadcast communication is not selective since a message sent by a node always reaches all other nodes. For Timed Networks, it is known that reachability of a configuration containing a given control location is undecidable in the case of two clocks per node, and decidable in the case of one clock per node.

When constraining communication via a complex connection graph, the decidability frontier becomes much more complex. More specifically, our technical results are as follows:

- For nodes equipped with a single clock, parameterized reachability becomes undecidable in a very simple class of graphs in which nodes are connected so as to form stars with diameter five.
- The undecidability result still holds in the more general class of bounded path graphs, i.e., graphs in which the length of maximal simple paths is bounded by a constant. In our proof we consider a bound $N \geq 5$ on the length of simple paths. Since nodes have no information about the shape of the network topology, the undecidability proof is not a direct consequence of the result for stars. Indeed the undecidability construction requires a preliminary step aimed at discovering a two-star topology in a graph of arbitrary shape but simple paths of at most five nodes.
- The problem turns out to be undecidable in the class of cliques of arbitrary order (that contains graphs with arbitrarily long paths) in which each timed automaton has at least two clocks.
- Decidability holds for special topologies like stars with diameter three and cliques of arbitrary order assuming that the process running in each node is equipped with a single clock (as in Timed Networks).
- Finally when considering discrete time, e.g. to model time-stamps, instead of continuous time, we show that the local state reachability problem becomes decidable for processes with any number of clocks in the class of graphs with bounded path. The same result holds for cliques of arbitrary order.

2. Preliminaries

Let $\mathbb{N}$ be the set of natural numbers and $\mathbb{R}^{\geq 0}$ the set of non-negative real numbers. For sets A and B, we use $f : A \mapsto B$ to denote that f is a total function that maps A to B. For $a \in A$ and $b \in B$, we write $f[a \hookleftarrow b]$ to denote the function f' defined as follows: $f'(a) = b$ and $f'(a') = f(a')$ for all $a' \neq a$. We denote by $[A \mapsto B]$ the set of all total functions from A to B.

We now recall the notion of well-quasi-ordering (which we abbreviate as wqo). A quasi-order $(A, \preceq)$ is a wqo if for every infinite sequence of elements $a_1, a_2, \ldots$ in A, there exist two indices $i < j$ such that $a_i \preceq a_j$. Given a set A with an ordering $\preceq$ and a subset $B \subseteq A$, the set B is said to be *upward closed* in A if $a_1 \in B$, $a_2 \in A$ and $a_1 \preceq a_2$ implies $a_2 \in B$. Given a set $B \subseteq A$, we define the upward closure $\uparrow B$ to be the set $\{a \in A \mid \exists a' \in B \text{ such that } a' \preceq a\}$. For a quasi-order $(A, \preceq)$, an element a is minimal for $B \subseteq A$ if for all $b \in B$, $b \preceq a$ implies $a \preceq b$. If $(A, \preceq)$ is a wqo and if B is upward closed in A, then the set of minimal elements of B is finite. If $\{b_1, \ldots, b_k\}$ is the set of minimal elements of B, then $\uparrow\{b_1, \ldots, b_k\} = B$; hence B can be represented finitely.

3. Timed ad hoc networks

3.1. Syntax

A Timed Ad Hoc Network (TAHN) consists of a graph where the nodes represent processes that run a common predefined protocol defined by a communicating timed automaton. The values of the clocks manipulated by the automaton inside each process are incremented all at the same rate. In addition, processes may perform discrete transitions which are either local transitions or communication events. When firing a *local* transition, a single process changes its local state without interacting with the other processes. For what concerns communication, it is performed by means of *selective broadcast*, a process sends a broadcast message which can be received only by its neighbours in the network. We choose to represent the communication relation as a graph. Finally, transitions are guarded by conditions on values of clocks and may also reset clocks.

We now provide the formal definition of the model. We assume that each process operates on a set of clocks X. A *guard* is a boolean combination of predicates of the form $k \lhd x$ for $k \in \mathbb{N}$, $\lhd \in \{=, <, \leq, >, \geq\}$, and $x \in X$. We denote by $\mathcal{G}(X)$ the set of guards over X. A reset R is a subset of X. The guards will be used to impose conditions on the clocks of processes that participate in transitions and the resets to identify the clocks that will be reset during the transition. A *clock valuation* is a mapping $F : X \mapsto \mathbb{R}^{\geq 0}$. For a guard g and a clock valuation F, we write $F \models g$ to indicate that the formula obtained by replacing in the guard g each clock x by $F(x)$ is valid. For a clock valuation F and a subset of clocks $Y \subseteq X$, we denote by $F[Y]$ the clock valuation such that $F[Y](x) = 0$ for all $x \in Y$ and $F[Y](x) = F(x)$ for all $x \in X \setminus Y$.

For a finite alphabet Σ of messages, we define the set of events associated to this alphabet as follows: $\mathcal{M}(\Sigma) = \{\tau\} \cup \{!!a, ??a \mid a \in \Sigma\}$. These events correspond to the following ideas:

(i) τ is used for a local move;
(ii) $!!a$ represents the broadcast of the message a;
(iii) $??a$ denotes the reception of the message a (that has been broadcasted by another process).

We now give the definition of a protocol which will be executed by the nodes in the network.

Definition 1. A *protocol* P is a tuple $\left(Q, X, \Sigma, \mathcal{R}, q^{init}\right)$ such that Q is a finite set of *states*, X is a finite set of *clocks*, Σ is a finite message alphabet, $\mathcal{R} \subseteq Q \times \mathcal{G}(X) \times \mathcal{M}(\Sigma) \times 2^X \times Q$ is a finite set of *rules* labelled with a guard, a message and a reset, and $q^{init} \in Q$ is an initial state.

Intuitively P defines the protocol that is run by each of the nodes (or entities) present in the network, where Q is the set of *local states* of each node, while $\mathcal{R}$ is a set of *rules* describing the behaviour of each node. We will use the notation $\left(q, g \xrightarrow{e} R, q'\right)$ to represent the rule (q, g, e, R, q'). For a protocol $P = \left(Q, X, \Sigma, \mathcal{R}, q^{init}\right)$, we denote by $nbclocks(P)$ the size of X, i.e., the number of clocks it uses.

A TAHN $\mathcal{T}$ is then simply a pair (G, P) where:

- $G = (V, E)$ is a *connectivity graph* composed of a finite set of nodes V and a set of undirected edges without self-loops, i.e., $E \subseteq V \times V \setminus \{(v, v) \mid v \in V\}$ s.t. E is symmetric;
- P is the protocol which will be executed by the node present in the nodes of the graph.

Intuitively, the graph G characterizes potential process interactions in the network $\mathcal{T}$; the set V represents the nodes and E defines the connectivity relation between the nodes of the network. The nodes belonging to an edge are called the *endpoints* of the edge. For an edge $(u, v) \in E$, we often use the notation $u \sim v$ and say that the vertices u and v are adjacent to each other.

3.2. Operational semantics

We now define the operational semantics of TAHN by means of a timed transition system. Let $\mathcal{T} = (G, P)$ be a TAHN with $G = (V, E)$ and $P = \left(Q, X, \Sigma, \mathcal{R}, q^{init}\right)$. A configuration γ of $\mathcal{T}$ is a pair $(\mathcal{Q}, \mathcal{X})$ where:

- $\mathcal{Q} : V \mapsto Q$ is a function that maps each node of the graph with a state of the protocol;
- $\mathcal{X} : V \mapsto [X \mapsto \mathbb{R}^{\geq 0}]$ is a function that assigns to each node a clock valuation.

An important point is that, in a configuration, each node of the graph has its own set of clocks. We denote by $\mathcal{C}_{\mathcal{T}}$ the set of configurations. The *initial configuration* of $\mathcal{T}$ is the configuration $\left(\mathcal{Q}^{init}, \mathcal{X}^{init}\right)$ with $\mathcal{Q}^{init}(v) = q^{init}$ and $\mathcal{X}^{init}(v)(x) = 0$ for all $v \in V$ and $x \in X$. In other words, in an initial configuration all the nodes are in the initial local state and all their associated clocks have value 0.

We now introduce a notation to characterize the nodes in a configuration that are able to receive a message a. Given a configuration $\gamma = (\mathcal{Q}, \mathcal{X})$ of the TAHN $\mathcal{T} = (G, P)$ (with $G = (V, E)$) and given a message $a \in \Sigma$, let $En_{\mathcal{T}}(\gamma, a)$ be the following set of nodes able to receive a in γ from the connectivity graph G:

$$En_{\mathcal{T}}(\gamma, a) = \{v \in V \mid \exists \left(q, g \xrightarrow{??a} R, q'\right) \in \mathcal{R} \text{ s.t. } \mathcal{Q}(v) = q \text{ and } \mathcal{X}(v) \models g\}$$

In the rest of the paper we will use $En(\gamma, a)$ when $\mathcal{T}$ is clear from the context.

The semantics associated to a TAHN $\mathcal{T}$ is then defined by the timed transition system $(\mathcal{C}_{\mathcal{T}}, \Longrightarrow_{\mathcal{T}})$, where the transition relation $\Longrightarrow_{\mathcal{T}} \subseteq \mathcal{C}_{\mathcal{T}} \times \mathcal{C}_{\mathcal{T}}$ corresponds to the union of a *discrete* transition relation $\Longrightarrow_{\mathcal{T},d}$, representing transitions induced by the rules of $\mathcal{T}$ and a *timed* transition relation $\Longrightarrow_{\mathcal{T},t}$ which characterizes the elapse of time.

The discrete transition relation $\Longrightarrow_{\mathcal{T},d} \subseteq \mathcal{C}_{\mathcal{T}} \times \mathcal{C}_{\mathcal{T}}$ is such that given two configurations $\gamma = (\mathcal{Q}, \mathcal{X})$ and $\gamma' = \left(\mathcal{Q}', \mathcal{X}'\right)$, we have $\gamma \Longrightarrow_{\mathcal{T},d} \gamma'$ if and only if one of the following conditions is satisfied:

Local: There exists a rule $\left(q, g \xrightarrow{\tau} R, q'\right)$ and a vertex $v \in V$ such that $\mathcal{Q}(v) = q$, $\mathcal{X}(v) \models g$, $\mathcal{Q}' = \mathcal{Q}\left[v \hookleftarrow q'\right]$, and $\mathcal{X}' = \mathcal{X}\left[v \hookleftarrow \mathcal{X}(v)[R]\right]$, and, for each $w \in V \setminus \{v\}$, we have $\mathcal{Q}'(w) = \mathcal{Q}(w)$, $\mathcal{X}'(w) = \mathcal{X}(w)$.

Broadcast: There exists a rule $\left(q, g \xrightarrow{!!a} R, q'\right)$ and a vertex $v \in V$ such that $\mathcal{Q}(v) = q$, $\mathcal{X}(v) \models g$, $\mathcal{Q}'(v) = q'$ and $\mathcal{X}'(v) = \mathcal{X}(v)[R]$, and, for each $w \in V \setminus \{v\}$, we have:

- either $w \sim v$ and $w \in En(\gamma, a)$ and there exists a rule of the form $\left(q_1, g_1 \xrightarrow{??a} R_1, q_1'\right)$ such that $\mathcal{Q}(w) = q_1$, $\mathcal{X}(w) \models g_1$, $\mathcal{Q}'(w) = q_1'$, and $\mathcal{X}'(w) = \mathcal{X}(w)[R_1]$.
- or $(w \nsim v$ or $w \notin En(\gamma, a))$, $\mathcal{Q}'(w) = \mathcal{Q}(w)$, and $\mathcal{X}'(w) = \mathcal{X}(w)$.

The timed transition relation $\Longrightarrow_{\mathcal{T},t} \subseteq \mathcal{C}_{\mathcal{T}} \times \mathcal{C}_{\mathcal{T}}$ is such that given two configurations $\gamma = (\mathcal{Q}, \mathcal{X})$ and $\gamma' = \left(\mathcal{Q}', \mathcal{X}'\right)$, we have $\gamma \Longrightarrow_{\mathcal{T},t} \gamma'$ if and only if:

Time: There is a $\delta \in \mathbb{R}^{\geq 0}$ such that for all $v \in V$ and $x \in X$, $\mathcal{Q}'(v) = \mathcal{Q}(v)$ and $\mathcal{X}'(v)(x) = \mathcal{X}(v)(x) + \delta$.

As said before, $\Longrightarrow_{\mathcal{T}}$ is then equal to $\Longrightarrow_{\mathcal{T},d} \cup \Longrightarrow_{\mathcal{T},t}$.

3.3. Topologies

As we will see, we will often restrict the connectivity graph of TAHN to belong to a family of graphs. In this paper, we consider different families of graphs that we call *topologies*. A *topology Top* is hence a class of graphs that we use to impose structural restrictions on the communication graph of a set of configurations. In the sequel we write $G \in Top$ to indicate that the graph G belongs to a given *Top*. We now list the topologies we will take into account in this work.

- GRAPH is the topology consisting of all finite graphs.
- For $\ell \geq 0$, STAR(ℓ) is the star topology of depth ℓ. It characterizes graphs $G = (V, E)$ for which there is a partition of V of the form $\{v_0\} \cup V_1 \cup \cdots \cup V_\ell$ such that:
  (i) $v_0 \sim v_1$ for all $v_1 \in V_1$;
  (ii) for each $1 \leq i < \ell$ and $v_i \in V_i$ there is one and only $v_{i+1} \in V_{i+1}$ with $v_i \sim v_{i+1}$ and one and only one $v_{i-1} \in V_{i-1}$ with $v_i \sim v_{i-1}$;
  (iii) no other nodes are adjacent to each other.
  In other words, in a star graph of dimension ℓ, there is a central node v_0 and an arbitrary number of *rays*. A ray consists of a path $v_1, v_2, \ldots, v_\ell$ of ℓ nodes, starting from v_1 adjacent to v_0. We call v_0 the *root*, $v_1, v_2, \ldots, v_{\ell-1}$ *internal nodes*, and v_ℓ a *leaf* of G.
- For $\ell \geq 0$, BOUNDED(ℓ) is the bounded path topology of bound ℓ. It characterizes graphs for which the length of the maximal simple path is bounded by ℓ. Formally, if $G \in$ BOUNDED(ℓ) with $G = (V, E)$ then there does not exist a finite sequence of nodes $(v_i)_{1 \leq i \leq m}$ such that $m > \ell$, and, $v_i \neq v_j$ for all i, j in $\{1, \ldots, m\}$ with $i \neq j$, and, $v_i \sim v_{i+1}$ for all $i \in \{1, \ldots, m-1\}$.
- CLIQUE is the set of cliques which characterizes graphs $G = (V, E)$ such that $v \sim w$ for all $v, w \in V$ with $v \neq w$.

3.4. State reachability problem

We now present the verification problem we study in this work. It consists in determining for a given protocol whether there exists a connectivity graph belonging to a certain topology such that in the obtained TAHN it is possible to reach, from the initial configuration, a configuration exhibiting a specific state (for instance an error state). We insist on the fact that we do not restrict the number of nodes appearing in the considered connectivity graphs. Notice that all the classes of graphs (called topologies) introduced previously have an infinite cardinality hence an algorithm enumerating all the graphs belonging to a given topology cannot be applied to solve our reachability problem. In fact, as we shall see, the main difficulty in this problem is that the set of connectivity graphs to consider is infinite.

Let $\mathcal{T} = (G, P)$ be a TAHN with a protocol $P = \left(Q, X, \Sigma, \mathcal{R}, q^{init}\right)$ and a connectivity graph $G = (V, E)$. We say that a configuration γ_n is *reachable* in $\mathcal{T}$ if there exists a finite path, starting at the initial configuration γ_0, of the form $\gamma_0 \Longrightarrow_{\mathcal{T}} \gamma_1 \Longrightarrow_{\mathcal{T}} \cdots \Longrightarrow_{\mathcal{T}} \gamma_n$ in the associated transition system. Given a state $q \in Q$, we say that q is reachable in the TAHN $\mathcal{T}$ if there exists a reachable configuration $\gamma = (\mathcal{Q}, \mathcal{X})$ and a vertex $v \in V$ such that $\mathcal{Q}(v) = q$.

We now define the state reachability problem TAHN–Reach(Top, K) parameterized by a topology *Top* and a number of clocks K as follows:

Input: A protocol P such that $nbclocks(P) \leq K$ and a control state q;
Output: Is there a TAHN $\mathcal{T} = (G, P)$ with $G \in Top$ such that q is reachable in $\mathcal{T}$?

In [11,12], a model of Ad Hoc Networks without time has been studied; it is the same as the one we have introduced considering protocols without clocks. The authors have shown that when the connectivity graphs are unrestricted, then the state reachability problem is undecidable. However, one can regain the decidability by restricting the graphs to have bounded path (i.e., graphs in which the length of the maximal simple path is bounded). Note also that when the reachability problem is restricted to cliques, then TAHN without clocks are equivalent to Broadcast Protocols (with no rendez-vous communication) which were introduced in [17] and for which the reachability problem is proved to be decidable. A proof, in terms of Ad Hoc Networks, of this latter result can also be found in [12]. The following theorem rephrases these results in our context.

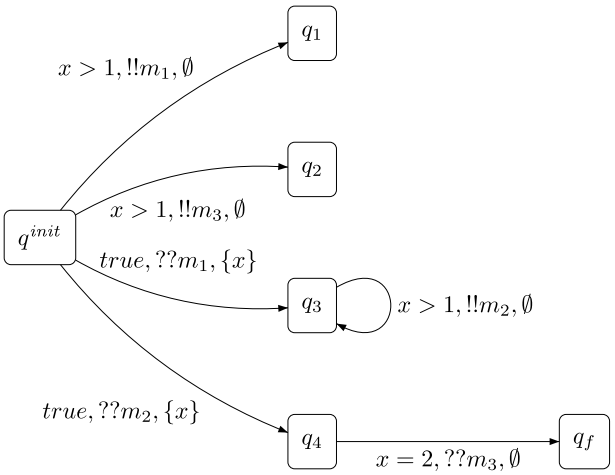

Fig. 1. A protocol P.

(a) A clique graph G_1 (b) A star graph G_2 of depth 2

Fig. 2. Example of two connectivity graphs.

Theorem 1. *(See [11,17,12].)*

1. $\texttt{TAHN-Reach}(\texttt{GRAPH}, 0)$ *is undecidable.*
2. *For all* $N \geq 1$, $\texttt{TAHN-Reach}(\texttt{BOUNDED}(N), 0)$ *is decidable.*
3. $\texttt{TAHN-Reach}(\texttt{CLIQUE}, 0)$ *is decidable*

Remark 1. We point out the fact that for a number of clocks K and given two topologies *Top* and *Top*′, if $Top \subset Top'$, we cannot infer directly any relation between the decidability status of $\texttt{TAHN-Reach}(Top, K)$ and $\texttt{TAHN-Reach}(Top, K')$. For instance if $\texttt{TAHN-Reach}(Top, K)$ is undecidable, then it does not imply necessarily that $\texttt{TAHN-Reach}(Top, K')$ is undecidable, it could be in fact the case that dealing with a larger class of graphs renders the problem solvable. Similarly if $\texttt{TAHN-Reach}(Top, K')$ is undecidable, we know that $\texttt{TAHN-Reach}(Top, K)$ could be decidable (see the above theorem where we have $\texttt{CLIQUE} \subset \texttt{GRAPH}$).

3.5. Example

Consider the protocol P described at Fig. 1 which uses a single clock per process. In this protocol, after more than one time unit, processes can broadcast m_1 or m_3. A process in initial state can then receive a message m_1, and after reception of such a message, it can broadcast a message m_2 if the delay between the reception of m_1 and the broadcast of m_2 is strictly more than one time unit. Finally, a process can reach the state q_f if it receives a message m_2 and exactly 2 times unit after, it receives a message m_3.

Fig. 2 gives two examples of connectivity graphs; the first one G_1 belongs to the topology $\texttt{CLIQUE}$ and the second one G_2 belongs to $\texttt{STAR}(2)$.

We are interested in knowing whether q_f is reachable in (G_1, P) and (G_2, P). We first consider the TAHN (G_1, P). In this model as soon as a process broadcasts a message m_1, then all the processes in initial state have to receive it with the rule $\left(q^{init}, true \xrightarrow{??m_1} \{x\}, q_3\right)$; because of the clique graph, each broadcast message is received by all the processes in the TAHN. Consequently, there does not remain any node in q^{init} ready to receive the message m_2 that is needed to go to q_f. Indeed there does not exist any connectivity graph $G \in \texttt{CLIQUE}$, such that q_f is reachable in the TAHN (G, P).

On the other hand, if we consider the TAHN (G_2, P), then q_f can be reached. We describe a possible scenario. After 2 times units one of the leaf broadcasts m_1, which is received by the adjacent internal node. After 2 times units this latter node broadcasts m_2 (note that this broadcast happens at global time 4). The message m_2 is received by the root node, which resets its clock, and exactly 2 times unit after (the global time is now 6), one of the two internal nodes, which remained in state q^{init}, broadcasts a message m_3, allowing thus the root node to reach q_f (it receives m_3 exactly two times units after the reception of m_2).

4. Undecidability with dense time

In this section, we show undecidability of the reachability problem in TAHN for three different topologies, namely:

- STAR(2): *star* connectivity graphs of depth 2 (one root and several rays with two nodes); the undecidability holds even if each process uses a single clock;
- CLIQUE: *clique* topologies; for this case, we need at least two clocks per process to get the undecidability;
- BOUNDED(5): *bounded path* topologies with maximal simple path of length at most 5; the undecidability holds even if each process uses a single clock.

In the first two cases, the undecidability results are obtained thanks to a reduction into the reachability problem for timed networks where processes are equipped with two clocks. We will hence first recall the definition of this latter model, which was originally presented in [2]. Afterwards, we will provide the reduction allowing to lift the undecidability result for Timed Network to the case of TAHN.

4.1. Timed networks

In [2], the authors introduce a model called Timed Network (TN) which can be used to describe a system consisting of an arbitrary number of processes, each of which being a finite-state system operating on real-valued clocks. The differences between the TN model and TAHN can be summarized as follows:

1. A TN contains a distinguished *controller* that is a finite-state automaton without any clocks [2] (note that adding clocks to the controller does not affect our results).
2. Each process in a TN may communicate with all the other processes and hence it is not meaningful to describe connectivity graphs in the case of TN.
3. Communication takes place through rendez-vous between fixed sets of processes rather than broadcast messages.

Following [2], we provide the syntax and the semantics of Timed Networks.

4.1.1. Syntax

Definition 2. A Timed Network (TN) $\mathcal{N}$ is a tuple $(Q^{ctrl}, Q^{proc}, X, \mathcal{R}, q^{init}_{ctrl}, q^{init}_{proc})$ where Q^{ctrl} is a finite set of *controller states*, Q^{proc} is a finite set of *process states*, X is finite set of clocks, $q^{init}_{ctrl} \in Q^{ctrl}$ is an initial controller stater, $q^{init}_{proc} \in Q^{proc}$ is an initial process state and $\mathcal{R}$ is a finite set of *rules*. A rule is of the form:

$$\begin{bmatrix} q_0 \\ \rightarrow \\ q'_0 \end{bmatrix} \begin{bmatrix} q_1 \\ g_1 \rightarrow R_1 \\ q'_1 \end{bmatrix} \cdots \begin{bmatrix} q_n \\ g_n \rightarrow R_n \\ q'_n \end{bmatrix}$$

such that $q_0, q'_0 \in Q^{ctrl}$, and, $q_i, q'_i \in Q^{proc}$, $g_i \in \mathcal{G}(X)$ and $R_i \in 2^X$ for all $i \in \{1, \ldots, n\}$.

4.1.2. Operational semantics

As for TAHN, we give the semantics associated to a TN in term of a timed transition system. We consider a TN $\mathcal{N} = \left(Q^{ctrl}, Q^{proc}, X, \mathcal{R}, q^{init}_{ctrl}, q^{init}_{proc}\right)$. A configuration γ is a tuple of the form $(I, q, \mathcal{Q}, \mathcal{X})$ with:

- I is a finite set of indices;
- $q \in Q^{ctrl}$;
- $\mathcal{Q} : I \mapsto Q^{proc}$;
- $\mathcal{X} : I \mapsto [X \rightarrow \mathbb{R}^{\geq 0}]$.

Intuitively, the configuration refers to the controller whose state is q, and to a finite set of processes, each one associated to an element of I. The mapping $\mathcal{Q}$ describes the states of the processes and the mapping $\mathcal{X}$ their associated clock values. More precisely, for $i \in I$ and $x \in X$, $\mathcal{X}(i)(x)$ gives the value of clock x in the process of index i. We use $|\gamma|$ to denote the number of processes in γ, i.e., $|\gamma| = |I|$. Let $\mathcal{C}_{\mathcal{N}}$ be the set of configurations of $\mathcal{N}$. A configuration $\gamma^{init} = (I, q, \mathcal{Q}, \mathcal{X})$ is said to be *initial* if $q = q^{init}_{ctrl}$, $\mathcal{Q}(i) = q^{init}_{proc}$, and $\mathcal{X}(i)(x) = 0$ for each $i \in I$ and $x \in X$. This means that an execution of a timed

network starts from a configuration where the controller and all the processes are in their initial states, and the clock values are all equal to 0. Note that there is an infinite number of initial configurations, namely one for each finite index set I.

Before we give the formal definition of the transition relation associated to $\mathcal{N}$, let us explain intuitively the behaviour of a rule of the form:

$$\begin{bmatrix} q_0 \\ \rightarrow \\ q_0' \end{bmatrix} \begin{bmatrix} q_1 \\ g_1 \rightarrow R_1 \\ q_1' \end{bmatrix} \cdots \begin{bmatrix} q_n \\ g_n \rightarrow R_n \\ q_n' \end{bmatrix}$$

Such a rule is enabled from a given configuration, if the state of the controller is q_0 and if there are n processes with states $q_1, \cdots, q_n$ whose clock values satisfy the corresponding guards. The rule is then executed by simultaneously changing the state of the controller to q_0', the states of the n processes to $q_1', \cdots, q_n'$ and by resetting the clocks belonging to the sets $R_1, \ldots, R_n$.

The semantics associated to the TN $\mathcal{N}$ is given by the timed transition system $(\mathcal{C}_\mathcal{N}, \longrightarrow_\mathcal{N})$ where the transition relation $\longrightarrow_\mathcal{N} \subseteq \mathcal{C}_\mathcal{N} \times \mathcal{C}_\mathcal{N}$ is defined as the union of a *discrete* transition relation $\longrightarrow_{\mathcal{N},d}$, representing transitions induced by the rules of $\mathcal{N}$ and a *timed* transition relation $\longrightarrow_{\mathcal{N},t}$ which characterizes the elapse of time.

We begin by describing the transition relation $\longrightarrow_{\mathcal{N},d} \subseteq \mathcal{C}_\mathcal{N} \times \mathcal{C}_\mathcal{N}$. For this matter, we define a transition relation $\longrightarrow_{\mathcal{N},r}$ for each rule $r \in \mathcal{R}$ of the TN $\mathcal{N}$. We consider a rule r described as above. Let $\gamma = (I, q, \mathcal{Q}, \mathcal{X})$ and $\gamma' = (I', q', \mathcal{Q}', \mathcal{X}')$ be two configurations in $\mathcal{C}_\mathcal{N}$. We have $\gamma \longrightarrow_{\mathcal{N},r} \gamma'$ if and only if $I = I'$ and there exists an injection $h : \{1, \ldots, n\} \rightarrow I$ such that, for all $i \in \{1, \ldots, n\}$:

1. $q = q_0$, $\mathcal{Q}(h(i)) = q_i$ and $\mathcal{X}(h(i)) \models g_i$ (i.e., the rule r is enabled);
2. $q' = q_0'$ and $\mathcal{Q}'(h(i)) = q_i'$ (i.e. the states of the processes are changed according to r);
3. $\mathcal{X}'(h(i)) = \mathcal{X}(h(i))[R_i]$ (i.e. a clock is reset to 0 if it occurs in the set R_i, otherwise its value remains unchanged).
4. $\mathcal{Q}'(j) = \mathcal{Q}(j)$ and $\mathcal{X}'(j) = \mathcal{X}(j)$ for all $j \in I \setminus \{h(i) \mid i \in \{1, \ldots, n\}\}$.

The discrete transition relation $\longrightarrow_{\mathcal{N},d}$ is then equal to $\bigcup_{r \in \mathcal{R}} \longrightarrow_{\mathcal{N},r}$.

We now provide the definition of the timed transition relation $\longrightarrow_{\mathcal{N},t} \subseteq \mathcal{C}_\mathcal{N} \times \mathcal{C}_\mathcal{N}$. Given two configurations $\gamma = (I, q, \mathcal{Q}, \mathcal{X})$ and $\gamma' = (I', q', \mathcal{Q}', \mathcal{X}')$ in $\mathcal{C}_\mathcal{N}$, we have $\gamma \longrightarrow_{\mathcal{N},t} \gamma'$ if and only if $I' = I$, $q' = q$, $\mathcal{Q}' = \mathcal{Q}$ and there exists $\delta \in \mathbb{R}^{\geq 0}$ such that $\mathcal{X}'(i)(x) = \mathcal{X}(i)(x) + \delta$ for all $i \in I$ and all $x \in X$. Hence, as in TAHN, a timed transition has no effect on the states of the different processes but its effect changes the value of the different clocks making them evolve at the same rate.

Finally we define $\longrightarrow_\mathcal{N}$ to be $\longrightarrow_{\mathcal{N},d} \cup \longrightarrow_{\mathcal{N},t}$. Note that if $\gamma \longrightarrow_\mathcal{N} \gamma'$ then the index sets of γ and γ' are identical (by definition of the transition relation) and therefore $|\gamma| = |\gamma'|$. This reflects the fact that in the considered networks, the number of processes is indeed parametric but once fixed it does not change during an execution, in other words there is no dynamic creation or deletion of processes.

4.1.3. State reachability problem

Similarly to the case of TAHN, we will present here the state reachability problem for TN. Here also this problem is parameterized by the number of processes involved in the execution, that is why we do not impose any bounds on the size of the initial configurations and we investigate whether there exists an initial configuration from which the system can reach another configuration in which the controller is in a given control state (for instance an error state).

Let $\mathcal{N} = (Q^{ctrl}, Q^{proc}, X, \mathcal{R}, q_{ctrl}^{init}, q_{proc}^{init})$ be a TN. We say that a configuration γ_n in $\mathcal{C}_\mathcal{N}$ is reachable in $\mathcal{N}$ if there exists a finite path $\gamma_0 \longrightarrow_\mathcal{N} \gamma_1 \longrightarrow_\mathcal{N} \cdots \longrightarrow_\mathcal{N} \gamma_n$ in the transition system associated to $\mathcal{N}$. As for TAHN, a controller state q is said to be reachable in $\mathcal{N}$ if there is a reachable configuration of the form $(I, q, \mathcal{Q}, \mathcal{X})$. The TN state reachability problem `TN-Reach(K)`, parametric in the a number of clocks K, is defined as follows:

Input: A TN $\mathcal{N} = (Q^{ctrl}, Q^{proc}, X, \mathcal{R}, q_{ctrl}^{init}, q_{proc}^{init})$ with $|X| \leq K$ and a controller state $q \in Q^{ctrl}$;
Output: Is q reachable in $\mathcal{N}$?

As said earlier, Timed Networks have already been introduced in [2] where results for the state reachability problems are also presented. These latter result can be expressed as follows:

Theorem 2. *(See [4].)*

1. `TN-Reach(2)` *is undecidable.*
2. `TN-Reach(1)` *is decidable.*

4.2. Two-star topologies

In this section we prove that the reachability problem for the star topology is undecidable even when the rays are restricted to have length 2 and the nodes are restricted to have a single clock. The proof is based on the encoding of a

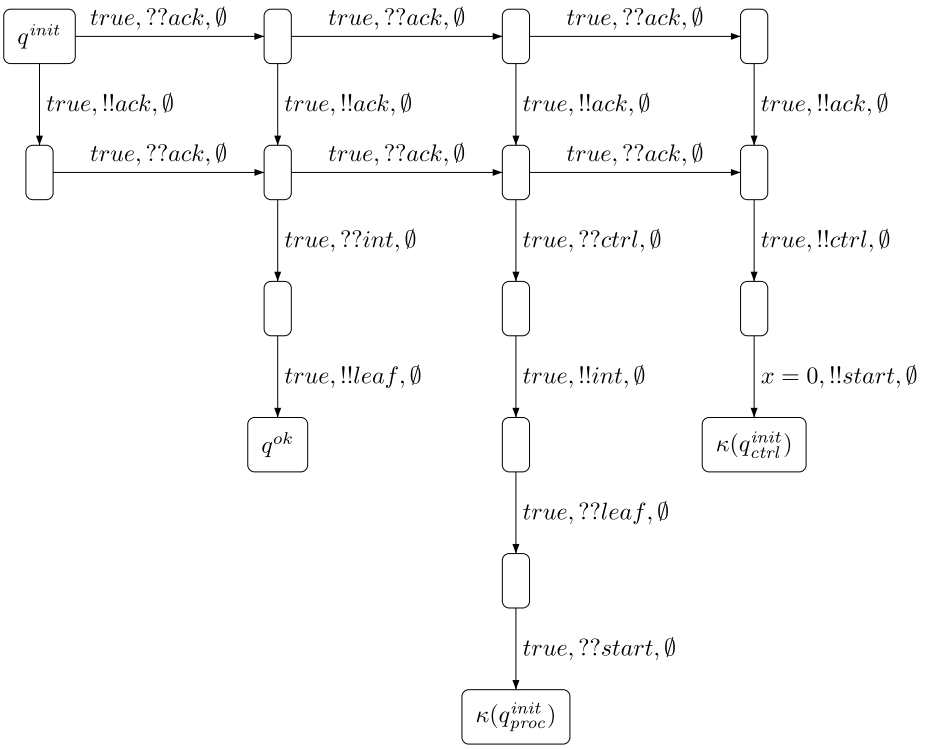

Fig. 3. Initializing the simulation.

generic TN $\mathcal{N}$ with two clocks per process into a protocol P of TAHNs. We will refer to the clocks inside a process of $\mathcal{N}$ as x_1 and x_2 respectively. For each state q in $\mathcal{N}$, we will have a corresponding state $\kappa(q)$ in the protocol P. Furthermore, we will have a number of auxiliary states in P that we need to perform the simulation. We omit state labels in the automata representation when their names are not relevant.

Given a TN $\mathcal{N} = \left(Q^{ctrl}, Q^{proc}, X, \mathcal{R}, q^{init}_{ctrl}, q^{init}_{proc}\right)$ and a controller state q in $\mathcal{N}$, we define a protocol P with $nbclocks(P) = 1$ together with a local state $\kappa(q)$ satisfying the following property: there exists a $\mathcal{T} = (G, P)$ with $G \in \mathtt{STAR}(2)$ such that such that $\kappa(q)$ is reachable in $\mathcal{T}$ iff q is reachable in $\mathcal{N}$. The root of G plays the role of the controller in $\mathcal{N}$, while each ray in G plays the role of one process in $\mathcal{N}$. The local state of a process in $\mathcal{N}$ is stored in the internal node of the corresponding ray. Furthermore, the two clocks x_1 and x_2 of a process are represented respectively by the clock of the internal node and by the clock of the leaf of the ray. For technical reasons, we require that the connectivity G of the considered TAHNs has at least three rays needed in the initialization phase. In case $\mathcal{N}$ has fewer than three processes, the additional rays will not simulate any processes, and remain passive (except during the initialization phase; see below).

Notation We will assume without loss of generality that the guards present in the TN $\mathcal{N}$ are conjunctions of predicates of the form $k \lhd x$ for $k \in \mathbb{N}$, $\lhd \in \{=, <, \leq, >, \geq\}$, and $x \in X$. In the sequel, (e.g. Figs. 4, 5, and 6), we will write $g(x_j \leftarrow x)$ to denote the guard obtained by first projecting g on the constraints involving only the variable x_j (this is done by deleting the predicates on the other variables), and then by replacing x_j (a clock of $\mathcal{N}$) with x (the clock of P) in the resulting formula. For instance, if g is $x_1 \geq 2 \wedge x_2 = 4$, then $g(x_1 \leftarrow x)$ is equal to $x \geq 2$ and $g(x_2 \leftarrow x)$ equals $x = 4$. For a reset R, we will write $R(x_j \leftarrow x)$ for the reset $\{x\}$ if $x_j \in R$, or for $\emptyset$ otherwise, i.e., we map a reset on x_j to a reset on the clock variable x of P. For instance, if $R = \{x_1\}$, then $R(x_1 \leftarrow x) = \{x\}$ and $R(x_2 \leftarrow x) = \emptyset$. We are now ready to describe the simulation protocol. It consists of two phases.

Initialization Recall that the nodes of a TAHN are identical in the sense that they execute the same (predefined) protocol. This means that the nodes are not *a priori* aware of their positions inside the network. The purpose of the initialization phase (Fig. 3) is to identify the nodes that play the roles of the controller and those that play the roles of the different processes.

As shown in Fig. 3, a node starts by broadcasting/receiving an *ack* message to/from his neighbours. The messages of type *ack* are used for the election phase. The elected node becomes the controller of the TN $\mathcal{N}$. To be elected, a node has to receive acknowledgements (messages *ack*) from at least three other processes. This implies that only the root of our star configuration can be elected. Indeed, it is the only node that is connected to more than two other nodes (the internal nodes are connected to two other nodes while the leafs are connected to only one other node). Notice that a node can become a

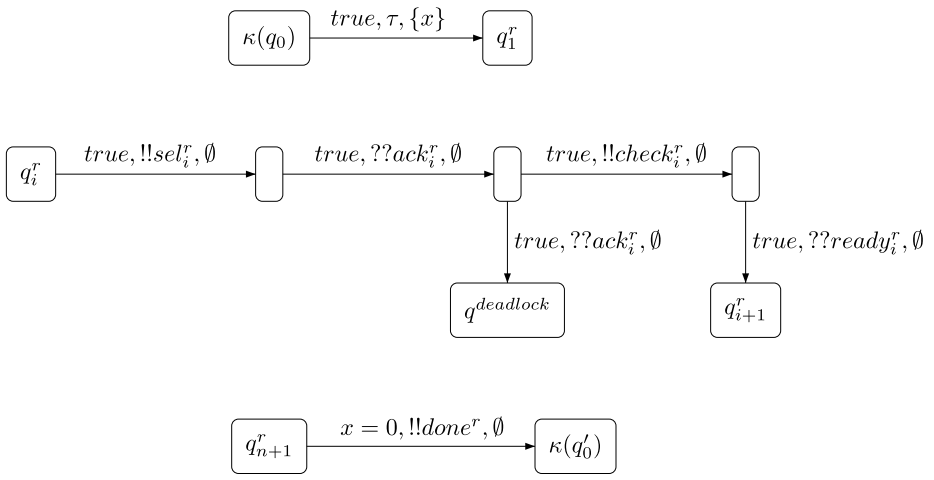

Fig. 4. Ray selection: root node.

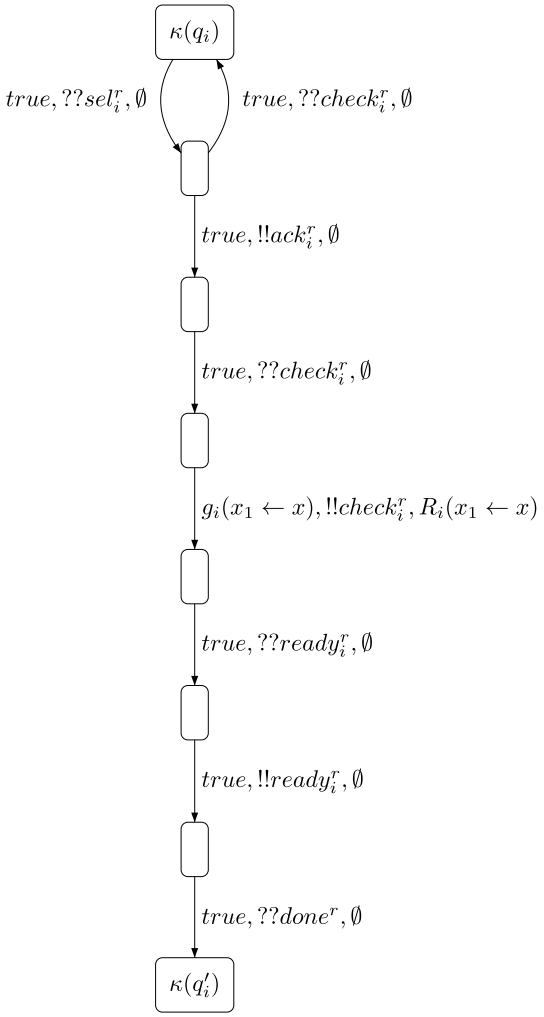

Fig. 5. Ray selection: internal node.

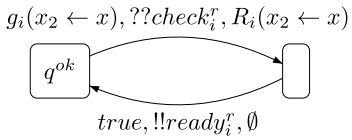

Fig. 6. Ray selection: leaf node.

controller via several different sequences of receive and send actions, the important points is that they contain three ??*ack* actions and one !!*ack* actions in any possible order. Sending !!*ack* after ??*ack*-actions is necessary to synchronize with the other nodes.

Once the root has become the controller, it will make the internal nodes aware of their positions by sending the broadcast message *ctrl*. Due to the star topology, this message is received only by the internal nodes. A node receiving this broadcast message will initiate a subprotocol defined as follows.

(i) It changes local state to accept the role of internal node.
(ii) It makes the leaf of the ray aware of its position by broadcasting a message *int*. Such a message is received only by the leaf of the ray and by the root (controller). The root ignores the message.
(iii) The leaf broadcasts the acknowledgement message *leaf* that can only be received by the internal node of the ray and goes to state q^{ok}.
(iv) The internal node changes state when it receives the acknowledgement and declares itself ready for the next step.

Remark that the internal node and the leaf may choose to ignore performing steps (ii) or (iv). In such a case we say that the protocol fails for the considered ray, otherwise we declare the ray to be successful.

In the last step of the initialization, the root will send one more broadcast where the following step take place:

(i) It changes local state to $\kappa(q_{ctrl}^{init})$ which means that it is now simulating the initial controller state.
(ii) It checks that its clock is equal to 0 which means that the initialization phase has been performed instantaneously.
(iii) The internal nodes of the successful rays will change state to $\kappa(q_{proc}^{init})$. The rest of the nodes will remain passive throughout the rest of the simulation.

Now all the nodes are ready: the root of G in $\mathcal{T}$ is in the initial state of the controller of $\mathcal{N}$; the internal nodes of the successful rays are in the initial states of the processes of $\mathcal{N}$; the leafs are in state q^{ok} and all clocks have values equal to 0.

Simulating discrete transitions Below, we show how $\mathcal{T}$ simulates a rule r of the form

$$\begin{bmatrix} q_0 \\ \rightarrow \\ q_0' \end{bmatrix} \quad \begin{bmatrix} q_1 \\ g_1 \rightarrow R_1 \\ q_1' \end{bmatrix} \quad \cdots \quad \begin{bmatrix} q_n \\ g_n \rightarrow R_n \\ q_n' \end{bmatrix}$$

The behaviour of the root, internal, and leaf nodes is detailed respectively in Fig. 4, Fig. 5, and Fig. 6. At first, the root of G in $\mathcal{T}$ is in the state $\kappa(q_0)$ and executes a transition to reset its clock to 0. The reset is used later to ensure that a simulation step has taken no time. The simulation consists of different phases, where in each phase the root tries to identify a ray that can play the role of process k for $1 \leq k \leq n$. To find the first ray, it sends a broadcast message $!!sel_1^r$. An (internal) node that receives the message and whose local state is q_1 may either decide to ignore the message or to try to become the node that simulates the first process in the rule. In the latter case it will enter a temporary state from which it initiates a sub-protocol whose goal is to confirm its status as the simulator of the first process. In doing so, the node guesses that its clocks satisfy the values specified by the guard. If the guess is not correct it will eventually be excluded from the rest of the simulation (will remain passive in the rest of the simulation). At the end of this phase, exactly one node will be chosen among the ones that have correctly guessed that their clocks satisfy g_1. The successful node will be the one that plays the role of the first process. The sub-protocol proceeds as follows:

(i) The internal node checks whether the value of its clock satisfies the guard g_1. Recall that each node contains one clock. Since the guard g_1 only compares the clocks x_1, x_2 with constants, the conditions of g_1 can be tested on separate nodes. Namely a node v can deal with the sub-guard involving x_1 and another node w can deal with the sub-guard involving clock x_2. The condition is then satisfied if both v and w acknowledge a certain request. If the clock of the node does not satisfy g_1 (which means that x_1 does not satisfy g_1), the node will remain passive from now on (it has made the wrong guess). Otherwise, the node resets its clock if R_1 contains x_1, and then broadcasts a message (such a message is received by the leaf of the ray).
(ii) The leaf checks whether the value of its clock satisfies the guard g_1 (i.e., if x_2 satisfies g_1); if *yes* it resets its clock if x_2 is included in R_1, and then broadcasts an acknowledgement.

(iii) Upon receiving the above acknowledgement, the internal node declares itself ready for the next step by broadcasting an acknowledgement. At the same time, it moves to new local state and waits for a last acknowledgement from the root (described below) after which it will move to local state $\kappa(q'_1)$.
(iv) When the root receives the acknowledgement it sends a broadcast declaring that it has successfully found a ray to simulate the first process. All the nodes in temporary states will now enter local states from which they remain passive. To prevent multiple nodes from playing the role of the first process, the root enters an error state on reception of an acknowledgement from more than one internal node.

The root now proceeds to identify the ray to simulate the second process. This continues until all n processes have been identified. Then the root makes one final move where the following events take place: (i) It moves its local state to $\kappa(q'_0)$. (ii) It sends a final broadcast where the node ready for simulating the ith process will now move to $\kappa(q'_i)$ for all $i : 1 \leq i \leq n$ (notice that there is at most one such node for each i). (iii) It checks that its clock is equal to 0 (the simulation of the rule has not taken any time).

Simulating timed transitions This is done in a straightforward manner by letting time pass in $\mathcal{T}$ by the same amount as in $\mathcal{N}$.

Putting together the different phases, we obtain a complete simulation of a TN with two clocks per node. Since reachability of a given control location (from an arbitrary initial configuration) is undecidable for TN, we deduce the following negative result.

Theorem 3. `TAHN-Reach(STAR(2), 1)` *is undecidable.*

4.3. Cliques and nodes with two clocks

We now show that the reachability problem for the clique topology is undecidable if each node manipulates two clocks. For this purpose, we build a protocol P with $nbclocks(P) = 2$ which will simulate $\mathcal{N}$ on connectivity graphs belonging to the clique topology. In a similar manner to the case of star topologies, the simulation consists of two phases.

Initialization phase The purpose of the initialization phase it to choose a node that will simulate the controller. This choice is done non-deterministically through a protocol that is initialized by a broadcast message. Notice that this protocol exists in all the nodes since they run the same pre-defined protocol. The first node which will perform the broadcast will become the controller (from now on we refer to this node as the *controller node*). When the controller node performs the above broadcast it moves to the state $\kappa(q^{init}_{ctrl})$, while all the other nodes will move to $\kappa(q^{init}_{proc})$.

Simulating discrete transitions Below, we show how a rule of the form of the previous sub-section is simulated. In a similar manner to the case of stars, the controller node first resets its clock to 0. The simulation again consists of different phases, where in each phase the controller node tries to identify a node that can play the role of process i for $1 \leq i \leq n$. To find the first process it sends a broadcast. A node that receives the broadcast, whose local state is q_1, and whose clocks (x_1 and x_2) satisfy the guard g_1, may decide to ignore the message or try to become the node that simulates the first process in the rule. In the latter case, the node declares itself ready for the next step by broadcasting an acknowledgement. At the same time, it moves to new local state and waits for a last acknowledgement from the controller node (described below) after which it will move to local state $\kappa(q'_1)$. To prevent multiple nodes from playing the role of the first process, the controller node enters an error state on reception of an acknowledgement from more than one node. The controller node now proceeds to identify the node to simulate the second process. This continues until all n processes have been identified. Then the controller node performs the same three steps as the ones in the final phase of the simulation described above for stars.

By exploiting undecidability of control state reachability for Timed Networks, we obtain the following theorem.

Theorem 4. `TAHN-Reach(CLIQUE, 2)` *is undecidable.*

4.4. Bounded path topologies

Using the result of Theorem 3 we now show that the undecidability proof for the reachability problem can be extended to bounded path topologies. The result uses a reduction to the two-star case, thus we need to consider topologies in which the simple paths can have 5 nodes in order to be able to rebuild stars with rays of depth 2.

For such a reduction, we need a preliminary protocol that discovers a two-star topology in a graph of arbitrary shape but simple paths of (at most) five nodes. The discovery protocol first selects root, internal and leaf candidates and then verifies that they are connected in the desired way by sending all other nodes in their vicinities to a special null state.

The discovery protocol is defined as P with $nbclocks(P) = 1$ with $q^{init} \in Q$ as initial state. We denote by x the clock used by P. We first define three transitions labelled with empty event that non-deterministically select the role of each node: the root (control state q_0), an internal node (control state r_0) or a leaf (control state s_0) of the star topology. These three rules have then following form:

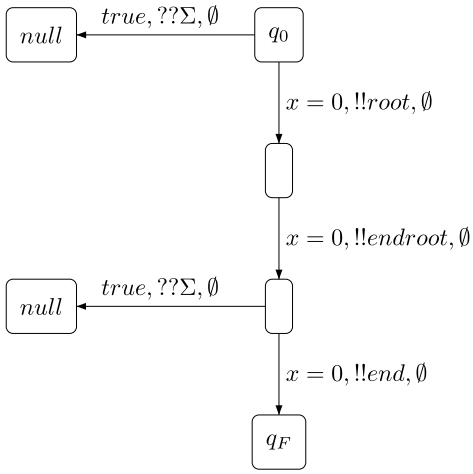

Fig. 7. Discovery protocol: node chosen as the root.

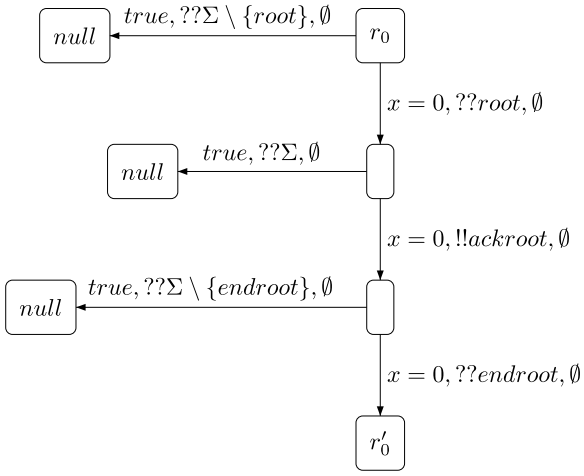

Fig. 8. Discovery protocol: node chosen as an internal node (communication with the root).

$$\left(q^{init}, x = 0 \xrightarrow{\tau} \emptyset, q_0\right)$$
$$\left(q^{init}, x = 0 \xrightarrow{\tau} \emptyset, r_0\right)$$
$$\left(q^{init}, x = 0 \xrightarrow{\tau} \emptyset, s_0\right)$$

The behaviour of the root node (state q_0) is given in Fig. 7. It broadcasts message *root* to notify its neighbours that it is the root. A node in state q_0 moves to an error state if it receives notifications/requests from other nodes. This protocol ensures that all the nodes in state q_0 connected to the root move to an error state (remember that communication is synchronous). On reception of a message of type *root*, a node in state r_0 runs the protocol in Fig. 8. Specifically, it first reacts by sending *ackroot*. This message is needed to send all of its neighbours in states derived from r_0 in the *null* error state. In fact if two adjacent nodes in state r_0 receive a message *root*, the first one sending *ackroot* will send the other one in the state *null*. The *ackroot* message is also needed to ensure that an internal node is never connected to two different root nodes. On reception of a message of type *endroot* from the root, the considered node moves to state r'_0. By the above described properties, when a node reaches state r'_0, then it is connected to at most one root node, and it is not connected to any node which was previously in state r_0 and which is not in state *null*. At this point several *leaf* nodes can still be connected to nodes in state r'_0. The last part of the protocol deals with this case.

Figs. 9 and 10 show the handshaking protocol between leaf and internal nodes. A node in state s_0 sends a message *leaf* to its adjacent nodes. An internal node can react to the message only in state r'_0, otherwise it goes to an error state. Furthermore, a leaf node that receives the *leaf* notification moves to an error state. By construction, the following properties then hold.

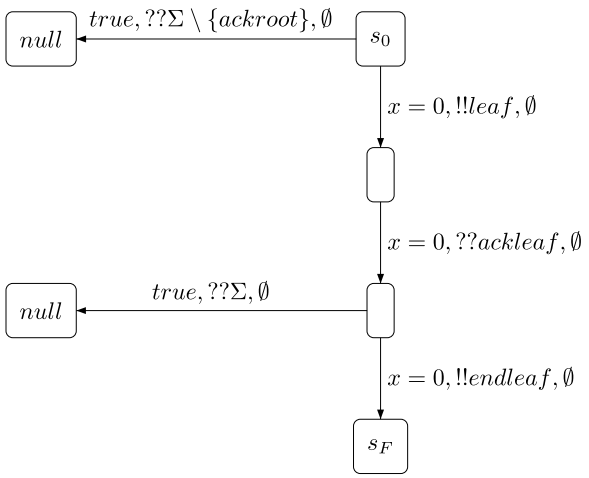

Fig. 9. Discovery protocol: node chosen as a leaf.

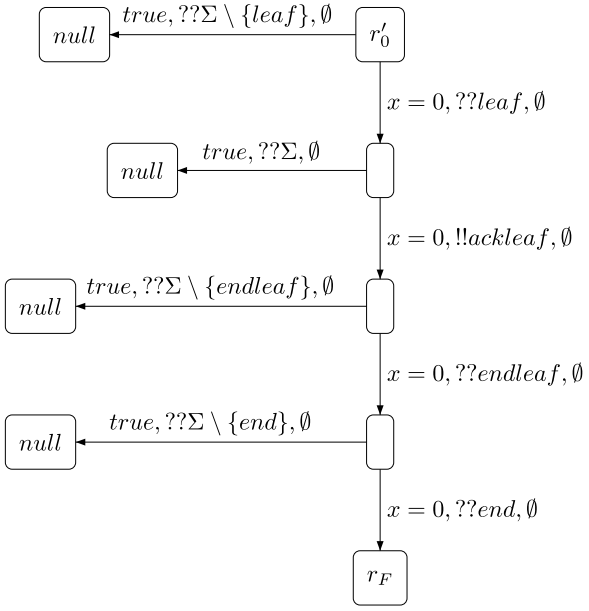

Fig. 10. Discovery protocol: node chosen as an internal one (communication with the leaf).

Proposition 1. *In a TAHN $\mathcal{T} = (G, P)$ with $G \in \mathtt{BOUNDED}(5)$, all configurations $\gamma = (G, \mathcal{Q}, \mathcal{X})$ reachable in $\mathcal{T}$ satisfy the following properties:*

- *for all nodes $v \in V$ such that $\mathcal{Q}(v) = q_F$, for all $v' \in V$ such that $v \sim v'$, we have $\mathcal{Q}(v') = r_F$ or $\mathcal{Q}(v') = null$,*
- *for all $v \in V$ such that $\mathcal{Q}(v) = r_F$, there exists two nodes $v_1, v_2 \in V$ such that $v \sim v_1$, $v \sim v_2$ and $\mathcal{Q}(v_1) = q_F$ and $\mathcal{Q}(v_2) = s_F$ and for all nodes $v' \in V \setminus \{v_1, v_2\}$, $v' \sim v$ implies $\mathcal{Q}(v') = null$,*
- *for all $v \in V$ such that $\mathcal{Q}(v) = s_F$, there exists at most one vertex $v' \in V$ such that $v \sim v'$ and $\mathcal{Q}(v') \neq null$ and furthermore it is such that $\mathcal{Q}(v') = r_F$.*

In other words if γ is a configuration reachable in a TAHN $\mathcal{T} = (G, P)$ with $G \in \mathtt{BOUNDED}(5)$ then all the nodes in the state q_F can be seen as the root node of a star of depth 2 where the internal nodes are in state r_F, the leaves are in state s_F, and all the other nodes connected to these nodes are in state *null* and will not take part to the further communications. Hence from q_F using the protocol proposed in the proof of Theorem 3, we can now simulate the behaviour of a TN as if we were in a star of depth 2. Indeed, combining the above described discovery protocol and the undecidability results for two-star topology, we obtain the following theorem.

Theorem 5. TAHN–Reach(BOUNDED(5), 1) *is undecidable.*

Proof. We reduce reachability of a TN $\mathcal{N}$ with two clocks. Specifically, we define a new protocol P' with a single clock that combines the discovery protocol and the simulation protocol described in the proof of Theorem 3. The final (non-null) states of the discovery protocol (namely q_F, r_F, and s_F) become the initial states of the simulation protocol. The following properties then hold:

- From Theorem 3, we know that there exists a protocol P and a TAHN $\mathcal{T} = (G, P)$ with $G \in \mathtt{STAR}(2)$ where it is possible to correctly simulate a TN $\mathcal{N}$.
- From Proposition 1, we know that any star of depth 2 can be obtained with our pre-protocol. Furthermore, the protocol which uses a single clock guarantees the existence of an initial configuration from which we can mark (using states q_F, r_F, and s_F) a subgraph in STAR(2) when the graphs of the initial configurations belongs to BOUNDED(5).

Combining the two properties, we deduce that the there exists a protocol P' using a single clock and a TAHN $\mathcal{T}' = (G', P')$ with $G' \in \mathtt{BOUNDED}(5)$ which can correctly simulate an $\mathcal{N}$ with two clocks per node. □

5. Decidability with dense time

In the previous sections, we have shown that TAHN–Reach(STAR(2), 1) is undecidable. We consider here two other topologies for which reachability becomes decidable when nodes have a single clock, namely the topologies STAR(1) and CLIQUE. For this we mix the technique, proposed in [2], to prove that the reachability problem is decidable in Timed Networks where each process is equipped with a single clock (see Theorem 2) and the one, used in [12], to show that, for TAHN with no clock and restricted to cliques, the reachability problem is decidable (see Theorem 1).

Our proof will be based on the following steps:

1. Define a symbolic way to represent graphs and their associated configurations.
2. Exhibit a well-quasi-ordering over the symbolic configurations which corresponds to the inverse of set inclusion on the associated concrete configurations.
3. Show that it is possible to compute symbolically the predecessors of a symbolic configuration.
4. Give an iterative method to compute all the elements from which a given symbolic configuration can be reached. Termination is ensured by the well-quasi-ordering of symbolic configurations.

5.1. Decidability of TAHN–Reach(CLIQUE, 1)

In the sequel, we fix a protocol $P = \left(Q, X, \Sigma, \mathcal{R}, q^{init}\right)$.

5.1.1. Symbolic representation of configurations

We recall that a TAHN is composed both by a connectivity graph $G = (V, E)$ and the protocol P and that the configurations of such a TAHN are of the form $(\mathcal{Q}, \mathcal{X})$ where $\mathcal{Q} : V \mapsto Q$ and $\mathcal{X} : V \mapsto [X \mapsto \mathbb{R}^{\geq 0}]$ is a function that assigns to each node a clock valuation. We will now introduce a way to represent symbolically connectivity graphs and associated configurations. Note that in this part, we focus on TAHN whose connectivity graphs are cliques, hence we only need to take into account the number of nodes in the graphs (the edges can indeed be deduced from this information). The symbolic representation we propose is very similar to the one from [2] used for the analysis of Timed Networks, the main differences being that in our model we do not have a special process playing the role of controller, and the discrete symbolic predecessor relation is different, since we do not deal with rendez-vous communication but with broadcast.

In what follows, we denote by *max* the maximal constant occurring in the guards of P. Furthermore for a quasi-order $\sqsubseteq$, we use the notation $a \equiv b$ whenever $a \sqsubseteq b$ and $b \sqsubseteq a$ (resp. $\equiv'$ for a quasi-order $\sqsubseteq'$, and, $\equiv_i$ for $\sqsubseteq_i$). A symbolic configuration φ for the protocol P is a tuple $\left(m, \mathcal{Q}^{symb}, \mathcal{X}^{symb}, \sqsubseteq\right)$ where:

- m is a natural number so that $\{1, \ldots, m\}$ is a set of indices for the processes present in the network;
- $\mathcal{Q}^{symb} : \{1, \ldots, m\} \mapsto Q$ maps indices to protocol states;
- $\mathcal{X}^{symb} : \{1, \ldots, m\} \mapsto \{0, \ldots, max\}$ maps process indices to a natural number less or equal than the constant max;
- $\sqsubseteq$ is a total preorder on the set $\{1, \ldots, m\} \cup \{\bot, \top\}$ such that:
  - $\bot$ and $\top$ are respectively the minimal and maximal elements of $\sqsubseteq$ with $\bot \neq \top$;
  - for $j \in \{1, \ldots, m\}$, if $\mathcal{X}^{symb}(j) = max$ then $j \equiv \bot$ or $j \equiv \top$;
  - for $j \in \{1, \ldots, m\}$, if $j \equiv \top$ then $\mathcal{X}^{symb}(j) = max$.

We denote by $\mathcal{S}_P$ the set of symbolic configurations for P. The intuition behind a symbolic configuration $\varphi = (m, \mathcal{Q}^{symb}, \mathcal{X}^{symb}, \sqsubseteq)$ is the following: it corresponds to a set of clique graphs and associated configurations where at least m processes are involved, and each of this m process is given an index $j \in \{1, \ldots, m\}$ such that $\mathcal{Q}^{symb}(j)$ is the state of the process, $\mathcal{X}^{symb}(j)$ is either the integral part of the clock value or *max*, and, the relation $\sqsubseteq$ provides an ordering for the processes

corresponding to the ordering of the fractional part of their respective clock values. Finally, if $j \equiv \bot$, this means that the clock value of process j is at most max and its fractional part is equal to 0, and, if $j \equiv \top$, then the clock value of process j is strictly greater than max.

Note that the couple $(\mathcal{X}^{symb}, \sqsubseteq)$ corresponds exactly to the clock regions for the m clocks represented in the abstract configuration φ. This region construction was originally introduced in [5] for the analysis of timed automaton since it allows to get rid of the precise value of the clocks by keeping an abstraction over the possible different values. It was then reused in [2] in the context of timed networks equipped with a single clock. In this latter work, the authors show how to adapt a quasi-order over such abstract configurations, since we need the same tool, we adopt in this work the same presentation for abstract configurations.

As done in [2] for the case of Timed Networks, we formalize the previous intuition by providing a formal definition of the set $[\![\varphi]\!]$ of graphs and concrete configurations represented by the symbolic configuration φ. We consider a graph $G = (V, E)$ in CLIQUE and a configuration $\gamma = (\mathcal{Q}, \mathcal{X})$ of the TAHN (G, P) and a symbolic configuration $\varphi = \left(m, \mathcal{Q}^{symb}, \mathcal{X}^{symb}, \sqsubseteq\right)$ of P. We have $(G, \gamma) \in [\![\varphi]\!]$ if and only if there exists an injective function $h : \{1, \ldots, m\} \mapsto V$ such that for all $j, j' \in \{1, \ldots, m\}$:

- $\mathcal{Q}(h(j)) = \mathcal{Q}^{symb}(j)$;
- $\min(max, \lfloor \mathcal{X}(h(j)) \rfloor) = \mathcal{X}^{symb}(j)$ (where $\lfloor \mathcal{X}(h(j)) \rfloor$ denotes the integral part of $\mathcal{X}(h(j))$);
- $j \equiv \bot$ if and only if $\mathcal{X}(h(j)) \leq max$ and $frac(\mathcal{X}(h(j))) = 0$ (where $frac(\mathcal{X}(h(j)))$ denotes the fractional part of $\mathcal{X}(h(j))$);
- $j \equiv \top$ if and only if $\mathcal{X}(h(j)) > max$;
- if $j \not\equiv \top$ and $j' \not\equiv \top$ then $frac(\mathcal{X}(h(j))) \leq frac(\mathcal{X}(h(j')))$ if and only if $j \sqsubseteq j'$.

When it exists, such an injective function h will be called a *mapping associated to* $((G, \gamma), [\![\varphi]\!])$. Note that in the above definition, we do not require the number of nodes in G and the number of processes in φ to be the same, but only that each process of φ can be matched with a process of the TAHN (G, P) in the configuration γ. For a set of symbolic configurations $\Phi \subseteq \mathcal{S}_P$, we denote by $[\![\Phi]\!]$ the set $\bigcup_{\varphi \in \Phi} [\![\varphi]\!]$.

We will now equip the set of symbolic configurations $\mathcal{S}_P$ with a quasi-order $\preceq$. Given two symbolic configurations $\varphi_1 = \left(m_1, \mathcal{Q}_1^{symb}, \mathcal{X}_1^{symb}, \sqsubseteq_1\right)$ and $\varphi_2 = \left(m_2, \mathcal{Q}_2^{symb}, \mathcal{X}_2^{symb}, \sqsubseteq_2\right)$, we have $\varphi_1 \preceq \varphi_2$ if and only if there exists an injective mapping $g : \{1, \ldots, m_1\} \mapsto \{1, \ldots, m_2\}$ such that for all $j, j' \in \{1, \ldots, m_1\}$:

- $\mathcal{Q}_2^{symb}(g(j)) = \mathcal{Q}_1^{symb}(j)$;
- $\mathcal{X}_2^{symb}(g(j)) = \mathcal{X}_1^{symb}(j)$;
- $g(j) \equiv_2 \bot$ if and only if $j \equiv_1 \bot$;
- $g(j) \equiv_2 \top$ if and only if $j \equiv_1 \top$;
- $g(j) \sqsubseteq_2 g(j')$ if and only if $j \sqsubseteq_1 j'$.

We have then the following proposition concerning this order.

Proposition 2.

1. *Given $\varphi_1, \varphi_2 \in \mathcal{S}_P$, we have $\varphi_1 \preceq \varphi_2$ if and only if $[\![\varphi_2]\!] \subseteq [\![\varphi_1]\!]$.*
2. *$(\mathcal{S}_P, \preceq)$ is a well-quasi-order.*

Proof. We will show the first point. Let $\varphi_1 = \left(m_1, \mathcal{Q}_1^{symb}, \mathcal{X}_1^{symb}, \sqsubseteq_1\right)$ and $\varphi_2 = \left(m_2, \mathcal{Q}_2^{symb}, \mathcal{X}_2^{symb}, \sqsubseteq_2\right)$ be two symbolic configurations in $\mathcal{S}_P$.

Suppose that $\varphi_1 \preceq \varphi_2$ and let $g : \{1, \ldots, m_1\} \mapsto \{1, \ldots, m_2\}$ be the corresponding mapping associated to the definition of the quasi-order $\preceq$. We then take $(G, \gamma) \in [\![\varphi_2]\!]$ with $G = (V, E)$ in CLIQUE and $\gamma = (\mathcal{Q}, \mathcal{X})$ and let $h : \{1, \ldots, m_2\} \mapsto V$ be the injective function associated to $((G, \gamma), [\![\varphi_2]\!])$. It is then clear that the composed function $h \circ g : \{1, \ldots, m_1\} \mapsto V$ is an injective function matching the condition for $(G, \gamma) \in [\![\varphi_1]\!]$. From this we deduce that $[\![\varphi_2]\!] \subseteq [\![\varphi_1]\!]$.

We assume that $[\![\varphi_2]\!] \subseteq [\![\varphi_1]\!]$. We consider the graph $G = (\{v_1, \ldots, v_{m_2}\}, E)$ in CLIQUE and we build the configuration $\gamma = (\mathcal{Q}, \mathcal{X})$ of the TAHN (G, P) in order that it verifies $\mathcal{Q}(v_i) = \varphi_2(i)$ for all $i \in \{1, \ldots, m_2\}$ and $\mathcal{X}$ verifies for all $i, i' \in \{1, \ldots, m_2\}$ the following points:

- if $i \equiv_2 \top$ then $\mathcal{X}(v_i) = max + 1$;
- if $i \not\equiv_2 \top$ then $\lfloor \mathcal{X}(v_i) \rfloor = \mathcal{X}_2^{symb}(i)$;
- if $i \equiv_2 \bot$ then $frac(\mathcal{X}(v_i)) = 0$;
- if $i \not\equiv_2 \bot$ then $frac(\mathcal{X}(v_i)) > 0$;
- if $i \not\equiv_2 \top$ and $i' \not\equiv_2 \top$ and $i \sqsubseteq_2 i'$ and $i \equiv_2 i'$ then $frac(\mathcal{X}(v_i)) = frac(\mathcal{X}(v_{i'}))$;
- if $i \not\equiv_2 \top$ and $i' \not\equiv_2 \top$ and $i \sqsubseteq_2 i'$ and $i \not\equiv_2 i'$ then $frac(\mathcal{X}(v_i)) < frac(\mathcal{X}(v_{i'}))$.

We consider then the bijective function $h_2 : \{1, \ldots, m_2\} \mapsto V$ such that $h_2(i) = v_i$ for all $i \in \{1, \ldots, m_2\}$. It is clear that h_2 satisfies the different conditions of a mapping associated to $((G, \gamma), [\![\varphi_2]\!])$. Hence $(G, \gamma) \in [\![\varphi_2]\!]$ and since $[\![\varphi_2]\!] \subseteq [\![\varphi_1]\!]$, we also have $(G, \gamma) \in [\![\varphi_1]\!]$. Let $h_1 : \{1, \ldots, m_1\} \mapsto V$ be the injective mapping associated to $((G, \gamma), [\![\varphi_1]\!])$. We consider now the composed function $h_2^{-1} \circ h_1 : \{1, \ldots, m_1\} \mapsto \{1, \ldots, m_2\}$. On the way we build the pair (G, γ) and the function h_2 and by definition of the mapping associated to $((G, \gamma), [\![\varphi_1]\!])$, one can easily deduce that the mapping $g = h_2^{-1} \circ h_1$ is effectively injective and satisfies the conditions required in the definition of the quasi-order $\preceq$ and hence this reasoning allows to obtain $\varphi_1 \preceq \varphi_2$.

For what concerns the proof that $(\mathcal{S}_P, \preceq)$ is a well-quasi-order, it can be found in [2], where quasi-identical symbolic configurations are used (and the same quasi-order is defined), the only difference being that, in our case, we do not have a controller state in the symbolic configurations. □

5.1.2. Computing the symbolic predecessors

We describe next how to compute symbolically the set of predecessors of the graphs and configurations described by a symbolic configuration. For a symbolic configuration $\varphi \in \mathcal{S}_P$, we will see how to build a finite set of symbolic configurations corresponding to the union of the two following sets:

$$\begin{aligned} pre_d(\varphi) \;=\; & \{(G, \gamma) \mid G \in \texttt{CLIQUE} \text{ and } \gamma \in \mathcal{C}_{(G,P)} \text{ and} \\ & \exists \gamma' \in \mathcal{C}_{(G,P)} \text{ s.t. } (G, \gamma') \in [\![\varphi]\!] \text{ and } \gamma \Longrightarrow_{(G,P),d} \gamma'\} \\ pre_t(\varphi) \;=\; & \{(G, \gamma) \mid G \in \texttt{CLIQUE} \text{ and } \gamma \in \mathcal{C}_{(G,P)} \text{ and} \\ & \exists \gamma' \in \mathcal{C}_{(G,P)} \text{ s.t. } (G, \gamma') \in [\![\varphi]\!] \text{ and } \gamma \Longrightarrow_{(G,P),t} \gamma'\} \end{aligned}$$

Hence $pre_d(\varphi)$ characterizes the symbolic predecessors for the discrete transition relation and $pre_t(\varphi)$ does the same for the timed transition relation. We will in fact show that it is possible to build a finite set of symbolic configurations Φ such that

$$[\![\Phi]\!] = pre_d(\varphi) \cup pre_t(\varphi)$$

First we begin by the predecessors obtained by considering the discrete transition relation. Following the idea used in [2] for Timed Networks, we begin with seeing how to test whether a guard is satisfied by the clock value of a process in the symbolic configuration. For a guard $g \in \mathcal{G}(X)$ (we recall that $|X| = 1$) in which the maximal constant is *max*, a symbolic configuration $\varphi = \left(m, \mathcal{Q}^{symb}, \mathcal{X}^{symb}, \sqsubseteq\right)$ and a natural number $j \in \{1, \ldots, m\}$, we define the relation $(\varphi, j) \models g$ inductively as follows:

- $(\varphi, j) \models k \leq x$ for $k \in \{0, \ldots, max\}$ iff $k \leq \mathcal{X}^{symb}(j)$;
- $(\varphi, j) \models k < x$ for $k \in \{0, \ldots, max\}$ iff either $k < \mathcal{X}^{symb}(j)$ or $(k = \mathcal{X}^{symb}(j)$ and $\bot \sqsubseteq j$ and $j \not\equiv \bot)$;
- $(\varphi, j) \models k \geq x$ for $k \in \{0, \ldots, max\}$ iff either $k > \mathcal{X}^{symb}(j)$ or $(k = \mathcal{X}^{symb}(j)$ and $j \equiv \bot)$;
- $(\varphi, j) \models k > x$ for $k \in \{0, \ldots, max\}$ iff $k > \mathcal{X}^{symb}(j)$;
- $(\varphi, j) \models k = x$ for $k \in \{0, \ldots, max\}$ iff $k = \mathcal{X}^{symb}(j)$ and $j \equiv \bot$;
- $(\varphi, j) \models g_1 \wedge g_2$ iff $(\varphi, j) \models g_1$ and $(\varphi, j) \models g_1$;
- for what concerns the negation, we assume that there are pushed inwards in the standard way before applying the definition.

Adapting the proof proposed in [2] for a similar result, we can deduce the following lemma about the satisfiability relation $\models$ on symbolic configurations.

Lemma 1. *Let $\varphi = \left(m, \mathcal{Q}^{symb}, \mathcal{X}^{symb}, \sqsubseteq\right)$ be a symbolic configuration, $G \in \texttt{CLIQUE}$ be a connectivity clique and $\gamma = (\mathcal{Q}, \mathcal{X})$ be a concrete configuration such that $(G, \gamma) \in [\![\varphi]\!]$ and let g be a guard in $\mathcal{G}(X)$ (for which the maximal appearing constant is max). Then for any mapping h associated to $((G, \gamma), [\![\varphi]\!])$ and $j \in \{1, \ldots, m\}$, we have that $(\varphi, j) \models g$ if and only if $\mathcal{X}(h(j)) \models g$.*

We will now show, for each rule $r \in \mathcal{R}$ of P and each symbolic configuration $\varphi \in \mathcal{S}_P$, how to compute the set $\texttt{Pre}(r, \varphi)$ of symbolic configurations corresponding to the symbolic predecessors of φ with respect to the rule r. Let $r = \left(q, g \xrightarrow{e} R, q'\right)$ be a rule of the protocol P and $\varphi = \left(m, \mathcal{Q}^{symb}, \mathcal{X}^{symb}, \sqsubseteq\right)$ be a symbolic configuration. We now provide the conditions for a symbolic configuration $\varphi_2 = \left(m_2, \mathcal{Q}_2^{symb}, \mathcal{X}_2^{symb}, \sqsubseteq_2\right)$ to belong to the set $\texttt{Pre}(r, \varphi)$. This definition is done by a case analysis on the message labelling the rule r. We have $\varphi_2 \in \texttt{Pre}(r, \varphi)$ iff $m \leq m_2 \leq m + 1$ and one of the following conditions is satisfied:

1. $e = !!a$ and there exists $j \in \{1, \ldots, m_2\}$ such that, if $m_2 = m + 1$, then $j = m + 1$, and such that $\mathcal{Q}_2^{symb}(j) = q$ and $\mathcal{X}_2^{symb}(j) \models g$ and such that the following conditions are satisfied:
   - if $m_2 = m$, then $\mathcal{Q}^{symb}(j) = q'$ and
     - if $R = \emptyset$ then $\mathcal{X}^{symb}(j) = \mathcal{X}_2^{symb}(j)$ and $j \equiv \bot$ iff $j \equiv_2 \bot$, and, $j \equiv \top$ iff $j \equiv_2 \top$;

  - if $R \neq \emptyset$ then $\mathcal{X}^{symb}(j) = 0$ and $j \equiv \bot$;
- for all $i \in \{1, \ldots, m_2\}$ such that $i \neq j$ we have:
  - either, there does not exists in $\mathcal{R}$ a rule $\left(q'', g' \xrightarrow{??a} R', q'''\right)$ such that $\mathcal{Q}_2^{symb}(i) = q''$ and $\mathcal{X}_2^{symb}(i) \models g'$, then we have $\mathcal{Q}_2^{symb}(i) = \mathcal{Q}^{symb}(i)$ and $\mathcal{X}_2^{symb}(i) = \mathcal{X}^{symb}(i)$, and, $i \equiv \bot$ iff $i \equiv_2 \bot$, and, $i \equiv \top$ iff $i \equiv_2 \top$;
  - or, there exists in $\mathcal{R}$ a rule of the form $\left(q'', g' \xrightarrow{??a} R', q'''\right)$ such that $\mathcal{Q}_2^{symb}(i) = q''$ and $\mathcal{X}_2^{symb}(i) \models g'$ and $\mathcal{Q}^{symb}(i) = q'''$ and:
    * either $R' = \emptyset$ and $\mathcal{X}^{symb}(i) = \mathcal{X}_2^{symb}(i)$ and $i \equiv \bot$ iff $i \equiv_2 \bot$, and, $i \equiv \top$ iff $i \equiv_2 \top$;
    * or, $R' \neq \emptyset$ and $\mathcal{X}^{symb}(i) = 0$ and $i \equiv \bot$.
- for all $i, i' \in \{1, \ldots .m\}$, if $i \not\equiv \bot$ and $i' \not\equiv \bot$, then $i \sqsubseteq_2 i'$ iff $i \sqsubseteq i'$.

2. $e = \tau$ and there exists $j \in \{1, \ldots, m_2\}$ such that, if $m_2 = m + 1$, then $j = m + 1$, and such that $\mathcal{Q}_2^{symb}(j) = q$ and $\mathcal{X}_2^{symb}(j) \models g$ and such that the following conditions are satisfied:
- if $m_2 = m$, then $\mathcal{Q}^{symb}(j) = q'$ and
  - if $R = \emptyset$ then $\mathcal{X}^{symb}(j) = \mathcal{X}_2^{symb}(j)$ and $j \equiv \bot$ iff $j \equiv_2 \bot$, and, $j \equiv \top$ iff $j \equiv_2 \top$;
  - if $R \neq \emptyset$ then $\mathcal{X}^{symb}(j) = 0$ and $j \equiv \bot$;
- for all $i \in \{1, \ldots, m_2\}$ such that $i \neq j$, we have $\mathcal{Q}_2^{symb}(i) = \mathcal{Q}^{symb}(i)$ and $\mathcal{X}_2^{symb}(i) = \mathcal{X}^{symb}(i)$, and, $i \equiv \bot$ iff $i \equiv_2 \bot$, and, $i \equiv \top$ iff $i \equiv_2 \top$.
- for all $i, i' \in \{1, \ldots .m\}$, if $i \not\equiv \bot$ and $i' \not\equiv \bot$, then $i \sqsubseteq_2 i'$ iff $i \sqsubseteq i'$.

The intuition behind the definition of the set $\texttt{Pre}(r, \varphi)$ is that first we consider only rules performing a broadcast or a local action, because the rules labelled with receptions are performed together with a broadcast. Then for a rule $\left(q, g \xrightarrow{e} R, q'\right)$ we need to ensure that, in the set of predecessors, the control state q is present, for this reason, either we add a process to the symbolic configuration (when $m_2 = m + 1$) which will perform the broadcast or the internal action, or, we consider that a process present in φ does this action (in that case $m_2 = m$). It is useless to add additional receiver nodes in the symbolic representation of configurations. They will produce redundant configurations.

In the case of a broadcast, we have to ensure that all the processes that could react, have reacted to the broadcast message, whereas in the case of an empty event, we need to ensure that the state of the other processes stays unchanged in the symbolic configuration. Finally, in $\texttt{Pre}(r, \varphi)$, we have to include all the possible symbolic clock mappings which satisfy the conditions of the fired rules, as well as the associated possible reset of the second clocks.

Note that by definition of symbolic configurations, we know that there exists a finite set of symbolic configurations of the form $\left(m, \mathcal{Q}^{symb}, \mathcal{X}^{symb}, \sqsubseteq\right)$ for a fixed m, this allows us to deduce that $\texttt{Pre}(r, \varphi')$ can be computed since it is finite. Furthermore, by definition of $pre(\varphi)$ and by the way we build the set $\texttt{Pre}(r, \varphi)$, we can show that the symbolic configuration $\bigcup_{r \in \mathcal{R}} \texttt{Pre}(r, \varphi)$ represents symbolically all the configuration in $pre_d(\varphi)$. In fact the previous construction covers all the possible cases. This allows us to state the next result.

Lemma 2.

- $\texttt{Pre}(r, \varphi)$ *is computable for all rules* $r \in \mathcal{R}$.
- $pre_d(\varphi) = [\![\bigcup_{r \in \mathcal{R}} \texttt{Pre}(r, \varphi)]\!]$.

Example. We show an example of computation for the set $\texttt{Pre}(r, \varphi)$. For this purpose, we use the protocol given in Fig. 1 of Section 3. We take the symbolic configuration $\varphi = \left(\{1\}, \{1 \mapsto q_f\}, \{1 \mapsto 2\}, \bot \equiv 1 \sqsubseteq \top\right)$ with a single process whose associated state is q_f and its associated symbolic clock value is 2 and we consider the rule $r = \left(q^{init}, (x > 1) \xrightarrow{!!m_3} \emptyset, q_2\right)$. Then in $\texttt{Pre}(r, \varphi)$, we have the symbolic configuration $(\{1, 2\}, \{1 \mapsto q_4, 2 \mapsto q^{init}\}, \{1 \mapsto 2, 2 \mapsto 2\}, \bot \equiv 1 \sqsubseteq 2 \sqsubseteq \top)$. In fact, it is possible to have predecessors from the symbolic configuration φ considering the rule r but, for this, we need to add a process to the symbolic configurations, which will represent the process performing the broadcast of m_3. Note that on the other hand, if we took the following symbolic configuration $\varphi' = (\{1\}, \{1 \mapsto q_4\}, \{1 \mapsto 2\}, \bot \equiv 1 \sqsubseteq \top)$ instead of φ, then the set $\texttt{Pre}(r, \varphi')$ would have been empty. In fact, it is not possible to have configurations with processes in state q_4 and associated clock value equal to 2 after the transition r has been fired, because the broadcast of the message m_3 would have sent all such processes in state q_f (since we are considering clique connectivity graphs exclusively).

For what concerns the set $pre_t(\varphi)$, the rules of the systems are not taken into account and hence in the case of TAHN computing this set is exactly the same as in Timed Networks, we can consequently reuse the result proved in [2].

Lemma 3. *(See [2].) There exists a computable finite set of symbolic configurations* Φ *such that* $[\![\Phi]\!] = pre_t(\varphi)$.

Sketch of proof. We can in fact characterize the symbolic configurations belonging to the set Φ. For a configuration $\varphi = (m, \mathcal{Q}^{symb}, \mathcal{X}^{symb}, \sqsubseteq)$ a configuration $\varphi_2 = \left(m_2, \mathcal{Q}_2^{symb}, \mathcal{X}_2^{symb}, \sqsubseteq_2\right)$ will belong to the set Φ representing $pre_t(\varphi)$ if it has the same number of process (i.e. $m = m_2$), the state mapping is the same (i.e. $\mathcal{Q}_2^{symb} = \mathcal{Q}^{symb}$) and for what concerns $\mathcal{X}_2^{symb}$ and the relation $\sqsubseteq_2$ over fractional part, they can be deduced from $\mathcal{X}^{symb}$ and $\sqsubseteq$ by iteratively making a rotation in the order $\sqsubseteq$ and in the same time by decreasing the integral part of a process whose fractional part is 0 (i.e. the number of this process is equivalent to $\perp$). Since the details of this construction are exactly the same as in Section 6.2 of [2], we do not provide them here. □

According to the two previous lemmas, for a symbolic configuration φ we can compute a finite set of predecessor symbolic configurations of φ. This is summed up by the next lemma.

Lemma 4. *There exists a computable finite set of symbolic configurations* $\texttt{Pre}(\varphi)$ *such that* $[\![\texttt{Pre}(\varphi)]\!] = pre_d(\varphi) \cup pre_t(\varphi)$.

5.1.3. Solving TAHN–Reach(CLIQUE, 1)

We now show how the facts that we have a well-quasi-order on the set of symbolic configurations which is related to the inclusion of sets of configuration (see Proposition 2) and that we can reason symbolically to compute the symbolic predecessors are enough to solve TAHN–Reach(CLIQUE, 1).

For this purpose, we need one more tool to manipulate sets of symbolic configurations. Given two sets of symbolic configurations $\Phi_1, \Phi_2 \subseteq \mathcal{S}_P$, we define the symbolic union of these two sets $\Phi_1 \sqcup \Phi_2$ as follows: $\varphi \in \Phi_1 \sqcup \Phi_2$ iff $(\varphi \in \Phi_1)$ or $(\varphi \in \Phi_2$ and there does not exist $\varphi' \in \Phi_1$ such that $\varphi' \preceq \varphi)$. Note that there are more than one set respecting these conditions, but each time we do the symbolic union we choose non-deterministically one of them.

From Proposition 2, we deduce the following lemma.

Lemma 5. $[\![\Phi_1 \sqcup \Phi_2]\!] = [\![\Phi_1]\!] \cup [\![\Phi_2]\!]$

Proof. Assume $(G, \gamma) \in [\![\Phi_1 \sqcup \Phi_2]\!]$, then there exists $\varphi \in \Phi_1 \sqcup \Phi_2$ such that $(G, \gamma) \in [\![\varphi]\!]$, and since by definition of $\sqcup$ we have $\varphi \in \Phi_1 \cup \Phi_2$, we deduce that $(G, \gamma) \in [\![\Phi_1]\!] \cup [\![\Phi_2]\!]$.

Assume now $(G, \gamma) \in [\![\Phi_1]\!] \cup [\![\Phi_2]\!]$, then there exists $\varphi \in \Phi_1 \cup \Phi_2$ such that $(G, \gamma) \in [\![\varphi]\!]$. We consider then $\varphi' \in \Phi_1 \sqcup \Phi_2$ such that $\varphi' \preceq \varphi$ (by definition of $\sqcup$ such a φ' exists). Then thanks to the first item of Proposition 2, we deduce $(G, \gamma) \in [\![\varphi']\!]$. Consequently $(G, \gamma) \in [\![\Phi_1 \sqcup \Phi_2]\!]$. □

We show that if we compute iteratively the symbolic predecessors of a symbolic configuration, then such a computation will converge after a finite number of iterations. We define, for a symbolic configuration $\varphi \in \mathcal{S}_P$, the following sequence of sets of symbolic configurations $(\mathcal{P}_\varphi^i)_{i \in \mathbb{N}}$:

- $\mathcal{P}_\varphi^0 = \{\varphi\}$;
- $\mathcal{P}_\varphi^{i+1} = \mathcal{P}_\varphi^i \sqcup \bigsqcup_{\varphi' \in \mathcal{P}_\varphi^i} \texttt{Pre}(\varphi')$

The next statement shows that the computation of the $\mathcal{P}_\varphi^i$ converges after a finite number of steps and furthermore that the obtained set characterize all the configurations from which it is possible to reach a configuration in $[\![\varphi]\!]$. The second point is quite obvious and the first point is obtained thanks to the fact that $(\mathcal{S}_P, \preceq)$ is a well-quasi-order as said by Proposition 2.

Lemma 6. *There exists* $N \in \mathbb{N}$ *such that* $\mathcal{P}_\varphi^i = \mathcal{P}_\varphi^N$ *for all* $i \geq N$.

Proof. We reason by contradiction and suppose that for all $i \in \mathbb{N}$, we have $\mathcal{P}_\varphi^{i+1} \neq \mathcal{P}_\varphi^i$. Since $\mathcal{P}_\varphi^{i+1} = \mathcal{P}_\varphi^i \sqcup \bigsqcup_{\varphi' \in \mathcal{P}_\varphi^i} \texttt{Pre}(\varphi')$, by definition of the operator $\sqcup$, this means that for all $i \in \mathbb{N}$, there exists φ_{i+1} such that $\varphi_{i+1} \in \bigsqcup_{\varphi' \in \mathcal{P}_\varphi^i} \texttt{Pre}(\varphi')$ and for which there does not exist $\varphi'' \in \mathcal{P}_\varphi^i$ such that $\varphi'' \preceq \varphi_{i+1}$. We consider then the infinite sequence $(\varphi_i)_{i \in \mathbb{N} \setminus \{0\}}$ of symbolic configurations in $\mathcal{S}_P$. By construction, this infinite sequence is such that for all $j \geq 1$ there does not exist $i \geq 1$ such that $i < j$ and $\varphi_i \preceq \varphi_j$. This is a contradiction with the claim of Proposition 2 which says that $(\mathcal{S}_P, \preceq)$ is a well-quasi-order. □

In the sequel we will denote $\mathcal{P}_\varphi$ the set $\mathcal{P}_\varphi^N$ defined by the previous lemma. Note that a consequence of this lemma is that such a set is finite and from Lemma 4, we know it can be effectively computed. Furthermore, we have also the following result which states the completeness and soundness of the symbolic reasoning.

Lemma 7.

1. *For all* $(G,\gamma) \in [\![\varphi]\!]$, *if* γ *is reachable in* $\mathcal{T} = (G,P)$ *from the initial configuration* γ_0 *then* $(G,\gamma_0) \in [\![\mathcal{P}_\varphi]\!]$.
2. *For all* $(G,\gamma_0) \in [\![\mathcal{P}_\varphi]\!]$, *there exists a reachable configuration* γ *in* $\mathcal{T} = (G,P)$ *such that* $(G,\gamma) \in [\![\varphi]\!]$.

Proof. Let $(G,\gamma) \in [\![\varphi]\!]$. Suppose that γ is reachable in $\mathcal{T} = (G,P)$ from the initial configuration γ_0. This means that there exists from γ_0 a finite path of the form $\gamma_0 \Longrightarrow_{\mathcal{T}} \gamma_1 \Longrightarrow_{\mathcal{T}} \cdots \Longrightarrow_{\mathcal{T}} \gamma_n$ in $\mathcal{T}$ with $\gamma_n = \gamma$. But thanks to Lemma 4, fixing $\varphi_n = \varphi$, we know that for all $i \in \{0,\ldots,n-1\}$, there exists φ_i such that $\varphi_i \in \mathtt{Pre}(\varphi_{i+1})$ and $(G,\gamma_i) \in [\![\varphi_i]\!]$. Then using Lemma 5 and the definition of $\mathcal{P}_\varphi$, we deduce that $(G,\gamma_0) \in [\![\mathcal{P}_\varphi]\!]$.

Similarly, if we suppose $(G,\gamma_0) \in [\![\mathcal{P}_\varphi]\!]$, thanks to Lemmas 4 and 5 and by definition of $\mathcal{P}_\varphi$, we know that there exists a finite path of the form $\gamma_0 \Longrightarrow_{\mathcal{T}} \gamma_1 \Longrightarrow_{\mathcal{T}} \cdots \Longrightarrow_{\mathcal{T}} \gamma_n$ in $\mathcal{T} = (G,P)$ such that $(G,\gamma_n) \in [\![\varphi]\!]$. □

For a control state $q \in Q$, we build the (finite) set of symbolic configurations Φ_q such that a symbolic configuration $\varphi = \left(m, \mathcal{Q}^{symb}, \mathcal{X}^{symb}, \sqsubseteq\right)$ belongs to this set if and only if $m = 1$ and $\mathcal{Q}^{symb}(1) = q$. And we define $\mathcal{P}_{\Phi_q}$ as the set $\bigcup_{\varphi\in\Phi_q} \mathcal{P}_\varphi$. Using the previous lemma, we can deduce that there exists a TAHN $\mathcal{T} = (G,P)$ with $G \in \mathtt{CLIQUE}$ such that q is reachable in $\mathcal{T}$ iff we have a symbolic configuration $\varphi_0 \in \mathcal{P}_{\Phi_q}$ such that $\varphi_0 = \left(m_0, \mathcal{Q}_0^{symb}, \mathcal{X}_0^{symb}, \sqsubseteq_0\right)$ verifies the following points:

- $\mathcal{Q}_0^{symb}(i) = q^{init}$ for all $i \in \{1,\ldots,m_0\}$;
- $\mathcal{X}_0^{symb}(i) = 0$ for all $i \in \{1,\ldots,m_0\}$;
- $i \equiv_0 \perp$ for all $i \in \{1,\ldots,m_0\}$.

Note that this last condition can be effectively tested on the set of symbolic configurations $\mathcal{P}_{\Phi_q}$ which is finite and computable. Hence this allows us to state the main result of this section.

Theorem 6. $\mathtt{TAHN\text{-}Reach}(\mathtt{CLIQUE},1)$ *is decidable.*

5.2. Decidability of $\mathtt{TAHN\text{-}Reach}(\mathtt{STAR}(1),1)$

A similar positive result can be obtained for TAHN with 1 clock restricted to star connectivity graphs of depth 1. The only difference with the previous result is that in a star of depth 1 we have to distinguish the root (the central node) from the leaves. In fact, when the root performs a broadcast, it is transmitted to all the leaf nodes, but when a leaf performs it, only the root can receive it. However the previous proof can be easily adapted to this case. The main trick consists in using symbolic configurations of the form $\left(m, \mathcal{Q}^{symb}, \mathcal{X}^{symb}, \sqsubseteq\right)$, as for the case of cliques, except that this time the index of the processes will go from 0 to m and 0 will be the index of the central node. The rest of the proof is then very similar to the previous construction; the bigger difference being in the computation of symbolic discrete predecessors, where one need to make the difference with a broadcast from the process 0 and one from the other processes. This allows us to state our second decidability result for the reachability problem restricted to protocols equipped with a single clock.

Theorem 7. $\mathtt{TAHN\text{-}Reach}(\mathtt{STAR}(1),1)$ *is decidable.*

We point out the fact, that such a reasoning could not be adapted for star topologies of depth strictly bigger than 1, because in that case we would need to have a relation in the symbolic configurations to know which process is connected to which other processes, and such a relation will break the possibility to have a well-quasi-order on the symbolic configurations (as a matter of fact, we have seen previously that the reachability problem is undecidable when considering protocols with a single clocks and star topologies of depth 2).

6. Decidability with discrete time

In this section we consider the state reachability problem for Discrete Time Ad Hoc Networks (DTAHN). In this model clocks range over the natural numbers instead of the reals. When using discrete time, it is enough to consider time steps that advance the clocks one unit per time. Furthermore, we can restrict the valuation of clocks to the finite range $\Omega = \{0,\ldots,\mu\}$ where $\mu = max + 1$ and max is the maximum constant used in the protocol rules. This follows from the fact that, as soon as the clock associated to variable x reaches a value greater than or equal to μ, guards of the form $x > c$ [resp. $x < c$] remain enabled [resp. disabled] forever. Therefore, beyond μ we need not distinguish between different values for the same clock [4].

Given a protocol P and a topology $G = (V,E)$, a configuration γ of the associated DTAHN $\mathcal{D}$ is a pair $(\mathcal{Q},\mathcal{X})$ defined as for TAHN except that the clock valuation mapping is of the form $\mathcal{X} : V \mapsto [X \mapsto \Omega]$. We denote by $\mathcal{C}_\mathcal{D}$ the set of configuration of $\mathcal{D}$. Initial configurations are defined as for TAHN. In the sequel, to simplify the handling of transition guards, without loss of generality we will assume that guards occurring in rules of a protocol $P = \left(Q, X, \Sigma, \mathcal{R}, q^{init}\right)$ have

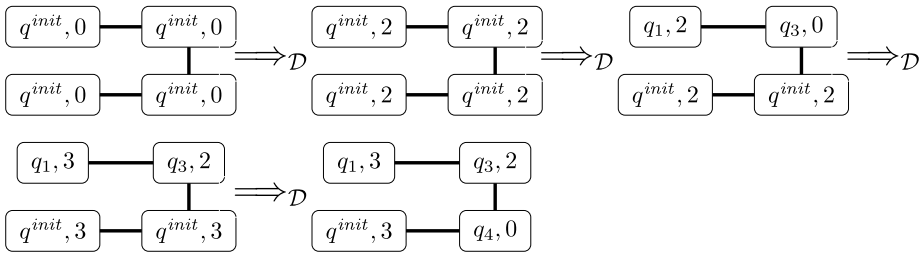

Fig. 11. An example of discrete time execution.

form $\bigwedge_{x\in X} x \geq a_x^g \wedge x \leq b_x^g$ with $a_x, b_x \in \{0, \ldots, max\}$ for all $x \in X$. This normal form is well-defined since clocks have always an explicit lower bound (which can be 0) and in case they do not have an explicit upper bound we set it to the constant μ. Since clock values range in $\{0, \ldots, \mu\}$, the previous restriction on guards does not affect the semantics. Furthermore, it is possible to encode disjunctions and negations by adding multiple rules between the same two states.

The semantics of the DTAHN $\mathcal{D}$ built over a protocol P is given by the transition system $(\mathcal{C}_{\mathcal{D}}, \Longrightarrow_{\mathcal{D}})$. The transition relation $\Longrightarrow_{\mathcal{D}} \subseteq \mathcal{C}_{\mathcal{D}} \times \mathcal{C}_{\mathcal{D}}$ is similar to the one of TAHN for the discrete transition and by replacing the time step by a discrete time step. For configurations $\gamma = (\mathcal{Q}, \mathcal{X})$ and $\gamma' = (\mathcal{Q}', \mathcal{X}')$, we write $\gamma \Longrightarrow_{\mathcal{D}} \gamma'$ iff these two configurations are in relation following the local or broadcast rules defined for TAHN, or via a discrete time step defined as follows: For all $v \in V$ and $x \in X$, the following conditions are satisfied: $\mathcal{Q}(\gamma') = \mathcal{Q}(\gamma)$, $\mathcal{X}'(v)(x) = \mathcal{X}(v)(x) + 1$, if $\mathcal{X}(v)(x) < \mu$, $\mathcal{X}'(v)(x) = \mathcal{X}(v)(x) = \mu$, otherwise.

Example. On Fig. 11, we present the example of a discrete time execution for the DTAHN composed of the protocol given in Fig. 1 and of the graph represented in Fig. 11. As we will see later, it is often convenient to represent the graph together with the configuration. Note that we have labelled the node of the graph with the associated control states and clock value (the protocol of Fig. 1 is equipped of a single clock). This run corresponds to the following step: a discrete time step of two units, then a broadcast of message m_1 then a discrete time steps of two units and finally a broadcast of message m_2. Note that we perform the second time step, some clocks get stuck to the maximal value 3 as described by the operational semantics for DTAHN.

For a topology class *Top* and $K \geq 0$, DTAHN–Reach(*Top*, K) denotes the state reachability problem for the new model. We show next that state reachability is decidable when restricting the topology to the class of bounded path graphs BOUNDED(N) for some $N > 1$.

In the sequel we consider a DTAHN $\mathcal{D}$ built over a protocol P. We first introduce an ordering between the configurations with connectivity graph. For this purpose, it is convenient to embed the connectivity graph G in the representation of a configuration. Specifically, we consider extended configurations defined by triples of the form $\gamma = (G, \mathcal{Q}, \mathcal{X})$. Given two (extended) configurations $\gamma = (G, \mathcal{Q}, \mathcal{X})$ with $G = (V, E)$ and $\gamma' = (G', \mathcal{Q}', \mathcal{X}')$ with $G' = (V', E')$ in $\mathcal{C}_{\mathcal{D}}$, we will write $\gamma \preceq \gamma'$ iff there exists an injective function $h : V \mapsto V'$ such that: $\forall u, u' \in V$, $(u, u') \in E$ if and only if $(h(u), h(u')) \in E'$, and $\forall u \in V$, $\mathcal{Q}(u) = \mathcal{Q}'(h(u))$ and $\mathcal{X}(u) = \mathcal{X}'(h(u))$.

In the sequel we will restrict ourselves to configurations whose graphs belong to BOUNDED(N) for some $N > 1$. We define $\mathcal{C}_{\mathcal{D}}^N$ as the set of configurations $\{(G, \mathcal{Q}, \mathcal{X}) \in \mathcal{C}_{\mathcal{D}} \mid G \in \text{BOUNDED}(N)\}$ and $(\mathcal{C}_{\mathcal{D}}, \preceq)$ as the ordering over the configurations of $\mathcal{D}$. For a set of configuration $S \subseteq \mathcal{C}_{\mathcal{D}}$ of the DTAHN $\mathcal{D}$, we denote *Pre*(S) the set $\{\gamma \in \mathcal{C}_{\mathcal{D}} \mid \gamma \Longrightarrow_{\mathcal{D}} \gamma',\ \gamma' \in S\}$. The following properties then holds.

Proposition 3. *The following properties hold:*

(1) $(\mathcal{C}_{\mathcal{D}}^N, \preceq)$ *is a wqo for all* $N > 1$.
(2) *For* γ *in* $\mathcal{C}_{\mathcal{D}}$, *we can algorithmically compute a finite set* B *such that* $\uparrow B = Pre(\uparrow\{\gamma\})$.

Property (1) follows from the observation that $\preceq$ is the induced subgraph relation for graphs with finitely many labels and from the wqo property of this relation proved by Ding in [14]. Properties (2) follows from the results for untimed AHN in [11]. To extend the algorithm for computing a basis for $Pre(\uparrow\gamma')$ described in [11] to discrete time steps we observe that, since the range of clocks is restricted to the interval Ω, we just need to collect all configurations obtained by subtracting in the configuration γ' the same constant value $\delta \geq 0$ s.t. the resulting clock values remain all greater or equal than zero.

Example. Consider a configuration of the protocol of Fig. 1 containing a single node whose associated control state is q_f and with clock value equal to 2. To compute predecessors for this configuration, we assume that we are working over graph in BOUNDED(2). To reach q_f, a process needs to receive a message m_3. Therefor we need to extend the configuration (ensuring we remain in the topology BOUNDED(2)) with an additional node that corresponds to a process from which this message has been broadcasted. The resulting configurations are shown in Fig. 12.

Fig. 12. Example of predecessors.

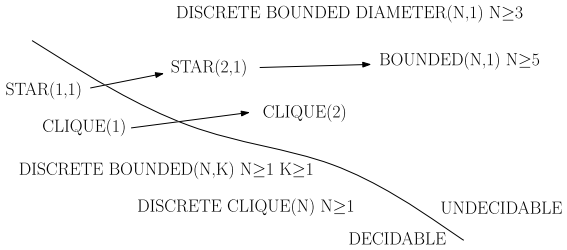

Fig. 13. Decidability and undecidability results for TAHN.

From Proposition 3, we can apply the general results in [3] to decide state reachability via a backward search algorithm working on upward closed sets of extended configurations represented by their finite basis. The following theorem then holds.

Theorem 8. `DTAHN-Reach(BOUNDED(`N`),` K`)` *is decidable for* $N \geq 1$, $K \geq 0$.

7. Related work

In [20] German and Sistla propose a general framework for parameterized verification of concurrent systems based on counting abstractions and reductions to Petri nets-like formalisms. The German–Sistla model is defined for fully connected topologies, individual processes modelled via finite-state automata and communication based on rendez-vous synchronization. Parameterized verification of concurrent systems in which the underlying communication topology is modelled as a special class of graphs, e.g., rings, have been proposed in [15,16,7,6]. In [15] Emerson and Namjoshi provide small model properties (cutoff properties) for a token-passing protocols in unidirectional rings that can be applied to prove fragments of indexed CTL* properties. The results have been extended by Aminof et al. in [6]. Decidability for token passing protocols for arbitrary graphs have been studied in [7,6].

Parameterized verification for broadcast communication has been studied in [15,17]. A forward, possibly non-terminating, reachability algorithm has been proposed in [15]. In [17] Esparza, Finkel and Mayr give a reduction of the problem to coverability in an extension of Petri nets with transfer arcs. Coverability is decidable in this model. The property can be proved by applying the general results in [3,19].

In [11,12] the authors study decidability issues for parameterized verification of a concurrent model with broadcast communication and communication topology restricted by a graph, called AHN. The model is an untimed abstraction that can be applied to specify protocols used for Ad Hoc Networks. Variations of the model with node and link failures, asynchronous communication, and local mailboxes has been studied in [10,13,9]. In [8] Clemente et al. give decidability results for different classes of topologies for systems defined by communicating automata with FIFO and bag channels.

Model checking for timed automata has been applied to verify protocols for ad hoc networks with a fixed number of nodes in [18]. Models with a discrete global clock and lazy exploration of configurations of fixed size has been considered in [24]. Formal specification languages for timed models of ad hoc networks have been proposed, e.g., in [22]. In contrast to these works, we consider here computability issues for verification of timed ad hoc networks with parametric initial configurations.

Decidability of some cases is proved by resorting to an extension of Timed Networks with Transfer. In the untimed case the combination of rendez-vous and transfer is considered in a model called datanets, an untimed extension of Petri nets in which processes have data taken from an ordered domain [21].

This paper extends with detailed proofs the preliminary work presented at FORMATS '11 [1].

8. Conclusions

We have studied local state reachability for Timed Ad Hoc Networks in different classes of topologies and considering the number of clocks of each node as a parameter. Fig. 13 shows a summary of our analysis. We also mention decidability for DTAHN on cliques since, as for bounded paths, it derives from an application of the theory of wsts. Undecidability for DTAHN on graphs with bounded diameter follows instead from the result obtained in the untimed case in [12].

References

[1] P.A. Abdulla, G. Delzanno, O. Rezine, A. Sangnier, Riccardo Traverso, On the verification of timed ad hoc networks, in: FORMATS, 2011, pp. 256–270.

[2] P.A. Abdulla, B. Jonsson, Model checking of systems with many identical timed processes, Theoret. Comput. Sci. 290 (1) (2003) 241–264.
[3] P.A. Abdulla, K. Cerans, B. Jonsson, Y.K. Tsay, General decidability theorems for infinite-state systems, in: LICS'96, IEEE Computer Society, 1996, pp. 313–321.
[4] P.A. Abdulla, J. Deneux, P. Mahata, Multi-clock timed networks, in: LICS'04, IEEE Computer Society, 2004, pp. 345–354.
[5] Rajeev Alur, David L. Dill, A theory of timed automata, Theoret. Comput. Sci. 126 (2) (1994) 183–235.
[6] B. Aminof, S. Jacobs, A. Khalimov, S. Rubin, Parameterized model checking of token-passing systems, in: Verification, Model Checking, and Abstract Interpretation – 15th International Conference, Proceedings, VMCAI 2014, San Diego, CA, USA, January 19–21, 2014, 2014, pp. 262–281.
[7] E.M. Clarke, M. Talupur, T. Touili, H. Veith, Verification by network decomposition, in: CONCUR 2004 – Concurrency Theory, 15th International Conference, Proceedings, London, UK, August 31 – September 3, 2004, 2004, pp. 276–291.
[8] L. Clemente, F. Herbreteau, G. Sutre, Decidable topologies for communicating automata with FIFO and bag channels, in: CONCUR 2014 – Concurrency Theory – 25th International Conference, Proceedings, Rome, Italy, September 2–5, 2014, 2014, pp. 281–296.
[9] G. Delzanno, A. Sangnier, R. Traverso, Parameterized verification of broadcast networks of register automata, in: Reachability Problems – 7th International Workshop, Proceedings, RP 2013, Uppsala, Sweden, September 24–26, 2013, 2013, pp. 109–121.
[10] G. Delzanno, A. Sangnier, R. Traverso, G. Zavattaro, On the complexity of parameterized reachability in reconfigurable broadcast networks, in: IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2012, December 15–17, 2012, Hyderabad, India, 2012, pp. 289–300.
[11] G. Delzanno, A. Sangnier, G. Zavattaro, Parameterized verification of ad hoc networks, in: CONCUR'10, in: LNCS, vol. 6269, Springer, 2010.
[12] G. Delzanno, A. Sangnier, G. Zavattaro, On the power of cliques in the parameterized verification of ad hoc networks, in: FoSSaCS'11, in: LNCS, vol. 6604, Springer, 2011, pp. 441–455.
[13] G. Delzanno, R. Traverso, Decidability and complexity results for verification of asynchronous broadcast networks, in: Language and Automata Theory and Applications – 7th International Conference, Proceedings, LATA 2013, Bilbao, Spain, April 2–5, 2013, 2013, pp. 238–249.
[14] G. Ding, Subgraphs and well quasi ordering, J. Graph Theory 16 (5) (1992) 489–502.
[15] E. Allen Emerson, K.S. Namjoshi, On model checking for non-deterministic infinite-state systems, in: Thirteenth Annual IEEE Symposium on Logic in Computer Science, Indianapolis, Indiana, USA, June 21–24, 1998, 1998, pp. 70–80.
[16] E. Allen Emerson, K.S. Namjoshi, On reasoning about rings, Internat. J. Found. Comput. Sci. 14 (4) (2003) 527–550.
[17] J. Esparza, A. Finkel, R. Mayr, On the verification of broadcast protocols, in: LICS'99, IEEE Computer Society, 1999, pp. 352–359.
[18] A. Fehnker, L. van Hoesel, A. Mader, Modelling and verification of the LMAC protocol for wireless sensor networks, in: IFM'07, in: LNCS, vol. 4591, Springer, 2007, pp. 253–272.
[19] A. Finkel, Ph. Schnoebelen, Well-structured transition systems everywhere!, Theoret. Comput. Sci. 256 (1–2) (2001) 63–92.
[20] S.M. German, A. Prasad Sistla, Reasoning about systems with many processes, J. ACM 39 (3) (1992) 675–735.
[21] R. Lazic, T. Newcomb, J. Ouaknine, A.W. Roscoe, J. Worrell, Nets with tokens which carry data, Fund. Inform. 88 (3) (2008) 251–274.
[22] M. Merro, F.F. Ballardin, E. Sibilio, A timed calculus for wireless systems, in: Proc. of the 3rd Conference on Fundamentals of Software Engineering, FSEN'09, in: LNCS, vol. 5961, Springer, 2010, pp. 228–243.
[23] M. Saksena, O. Wibling, B. Jonsson, Graph grammar modeling and verification of Ad Hoc Routing Protocols, in: TACAS'08, in: LNCS, vol. 4963, Springer, 2008, pp. 18–32.
[24] A. Singh, C.R. Ramakrishnan, S.A. Smolka, Query-based model checking of ad hoc network protocols, in: CONCUR'09, in: LNCS, vol. 5710, Springer, 2009, pp. 603–619.