On the Verification of Timed Ad Hoc Networks

Parosh Aziz Abdulla[1], Giorgio Delzanno[2], Othmane Rezine[1], Arnaud Sangnier[3], and Riccardo Traverso[2]

[1] Uppsala University [2] University of Genova [3] Liafa-University Paris 7

Abstract. We study decidability and undecidability results for parameterized verification of a formal model of timed Ad Hoc network protocols. The communication topology is represented by a graph and the behavior of each node is represented by a timed automaton communicating with its neighbors via broadcast messages. We consider verification problems formulated in terms of reachability, starting from initial configurations of arbitrary size, of a configuration that contain at least one occurrence of a node in a certain 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. [8,12,11,5,6]. Ad hoc networks 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 [5] 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 parameterize verification problems in the size and shape of the initial configuration as in the untimed case in [5]. We observe however that protocols for ad hoc networks are often based on time-sensitive conditions like time-outs or time-stamps (added to flooded data). Parameterized verification of timed automata has been studied only for fully connected networks and rendez-vous communication (Timed Networks) in [4,2,3].

A natural combination of the AHN model in [5] and of Timed Network is obtained by adding a connectivity graph to a network of timed automata communicating via broadcast messages, we call the resulting model Timed Ad hoc Networks (TAHNs). For a fixed initial configuration, TAHNs can be specified in model checkers like Uppaal [13] by using a shared global matrix to specify the communication topology. This idea has been used to verify safety properties of the LMAC protocol for initial configurations with a small number of nodes [8].

Following [5,6], in this paper we study decidability and undecidability properties of the local state reachability problem parametric on the initial configuration of a TAHN, i.e., the problem of 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.

Compared to the positive results obtained for Timed Networks where the control state reachability problem is decidable for processes with a single clock, for the same type of processes the problem becomes undecidable in a very simple class of topologies in which nodes are connected so as to form stars with diameter five. The undecidability result can be ported to the more general class of graphs with bounded path (for some bound $N \geq 5$ on the length – number of nodes – of paths). In the untimed case local state reachability is decidable for bounded path topologies [5]. Furthermore, 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 timed automaton in each node has a single clock (as in Timed Networks).

For discrete 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 holds for cliques of arbitrary order as in the case of dense time.

Related Work Decidability issues for untimed models of ad hoc Networks have been considered in [5,6]. Model checking for timed automata has been applied to verify protocols for ad hoc networks with a fixed number of nodes in [8]. Models with a discrete global clock and a lazy exploration of configurations of fixed size has been considered in [12]. Formal specification languages for timed models of ad hoc networks have been proposed, e.g., in [10]. In contrast to these works, we consider computatibility 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 transfers. In the untimed case the combination of rendez-vous and transfers is considered in Datanets, a model in which processes have data taken from an ordered domain [9].

Outline In the next section we give some preliminaries. The models of Timed Ad Hoc Networks and Time Networks are introduced in Sections 3–4. In Section 5, we introduce the undecidability results for three different topologies. We give decidability for different topologies under the dense time semantic in Section 6 and under the discrete time semantics in Section 7. Finally, we give some conclusions in Section 8. Due to lack of space, some proofs are moved to the Appendix.

2 Preliminaries

We use $\mathbb{N}$ and $\mathbb{R}^{\geq 0}$ for the set of natural numbers and set of non-negative real numbers respectively. 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 use $f\,[a \hookleftarrow b]$ to

denote the function f' defined as follows: $f'(a) = b$ and $f'(a') = f(a')$ for all $a' \neq a$. We use $[A \mapsto B]$ to denote the set of all total functions from A to B.

3 Timed Ad Hoc Networks

An ad hoc network consists of a graph where the nodes represent processes that run a common predefined protocol. This protocol is defined by a communicating timed automaton. The values of the clocks of the automata inside the processes are incremented continuously all at the same rate. In addition, a process may perform a discrete transition. The latter is either a local transition or the result of a communication event. In a *local* transition, the process changes its local state without interacting with the other processes. Communication is performed through *selective broadcast*, where a process sends a broadcast message to the network. The effect of a broadcast is local to the vicinity of the sender, i.e., only the neighbors of the sending process are able to receive the message. The connectivity of the nodes is reflected by the edges of the graph. Furthermore, transitions are conditioned by values of the clocks of the process, and may reset the values of some clocks.

We assume that each process operates on a set X of clocks. A *guard* is a Boolean combination of predicates of the form $k \lhd x$ for $k \in \mathbb{N}$, $\lhd \in \{=, <, \leq, >, \geq\}$, and $x \in X$. A *reset* R is a subset of X. We will use guards to impose conditions on the clocks of processes that participate in transitions, and use resets to identify the clocks that will be reset during the transition. A *clock valuation* is a mapping $\mathtt{X} : X \mapsto \mathbb{R}^{\geq 0}$. For a guard g and a clock valuation $\mathtt{X}$, we write $\mathtt{X} \models g$ to indicate the validity of the formula we get by replacing each clock x in g by its value $\mathtt{X}(x)$. Also, we will assume a finite alphabet Σ. The alphabet induces a set of events, where an event is of one of three forms: (i) *empty event* τ that represents a local move; (ii) *broadcast event* $!!a$, with $a \in \Sigma$, that represents broadcasting the message a; or (ii) *receive event* $??a$ with $a \in \Sigma$, that represents receiving the message a (that has been broadcast by another process).

Formally, a *Timed Ad Hoc Network* (*TAHN* for short) is defined by a pair $\mathcal{T} = (G, P)$. The first component $G = (V, E)$ is graph where V is a finite set of vertices and $E \subseteq V \times V$ is a set of edges. The second component P, called the *protocol*, is a pair $(Q, \mathcal{R})$ where Q is a finite set of *states*, and $\mathcal{R}$ is a finite set of *rules*. Intuitively, the graph G defines the topology of $\mathcal{T}$ where the set V represents the nodes, and E defines the connectivity of the nodes. The vertices 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. Furthermore, P defines the protocol that runs inside the nodes, where Q is the set of *local states* of each node, while $\mathcal{R}$ is a set of *rules* describing the behavior of each node. A rule $\rho \in \mathcal{R}$ is of the form $\left(q, g \xrightarrow{e} R, q'\right)$ where $q, q' \in Q$, g is a guard, e is an event, and R is a reset. We use $\#P$ to denote that number of clocks inside each process.

Configurations A configuration γ is a pair $(\mathcal{Q}, \mathcal{X})$, where $\mathcal{Q} : V \mapsto Q$ and $\mathcal{X} : V \mapsto [X \to \mathbb{R}^{\geq 0}]$, i.e., the configuration assigns to each node a local state and assigns to each clock in the node a value (in $\mathbb{R}^{\geq 0}$).

Transition Relation For configurations $\gamma = (\mathcal{Q}, \mathcal{X})$ and $\gamma' = (\mathcal{Q}', \mathcal{X}')$, we write $\gamma \Longrightarrow_{\mathcal{T}} \gamma'$ to denote that one of the following conditions is satisfied:

- *Local:* There exists a rule $\rho = \left(q, g \xrightarrow{\tau} R, q'\right)$ and a vertex $v \in V$ such that $\mathcal{Q}(v) = q$, $\mathcal{X}(v) \models g$, $\mathcal{Q}' = \mathcal{Q}[v \hookleftarrow q']$, and $\mathcal{X}' = \mathcal{X}[v \hookleftarrow \mathtt{X}]$ where $\mathtt{X}(x) = 0$ if $x \in R$ and $\mathtt{X}(x) = \mathcal{X}(v)(x)$ otherwise.
- *Broadcast:* There exists a rule $\left(q, g \xrightarrow{!!a} R, q'\right)$ and a vertex $v \in V$ such that $\mathcal{Q}(v) = q$, $\mathcal{X}(v) \models g$, $\mathcal{Q}'(v) = q'$, $\mathcal{X}'(v)(x) = 0$ if $x \in R$, and $\mathcal{X}'(v)(x) = \mathcal{X}(v)(x)$ otherwise. Furthermore, for each $v_1 \in V - \{v\}$, one of the following conditions is satisfied:
  - $v_1 \sim v$ and there is a rule of the form $\left(q_1, g_1 \xrightarrow{??a} R_1, q_1'\right)$ such that $\mathcal{Q}(v_1) = q_1$, $\mathcal{X}(v_1) \models g_1$, $\mathcal{Q}'(v_1) = q_1'$, $\mathcal{X}'(v_1)(x) = 0$ if $x \in R_1$, and $\mathcal{X}'(v_1)(x) = \mathcal{X}(v_1)(x)$ otherwise.
  - $\mathcal{Q}'(v_1) = \mathcal{Q}(v_1)$, $\mathcal{X}'(v_1) = \mathcal{X}(v_1)$, and either $v_1 \not\sim v$ or there is no rule of the form $\left(q_1, g_1 \xrightarrow{??a} R_1, q_1'\right)$ for any g_1, R_1, q_1' with $\mathcal{Q}(v_1) = q_1$ and $\mathcal{X}(v_1) \models g_1$.
- *Time:* There is a $\delta \in \mathbb{R}^{\geq 0}$ such that $\mathcal{Q}(v') = \mathcal{Q}(v)$ and $\mathcal{X}'(v)(x) = \mathcal{X}(v)(x) + \delta$ for all $v \in V$ and $x \in X$.

Topology A *topology Top* restricts the shape of the underlying graph $G = (V, E)$ in a TAHN $\mathcal{T} = (P, G)$. We write $G \in \mathit{Top}$ to indicate that G satisfies *Top*. Below, we give examples of some topologies.

- We denote by $\mathtt{GRAPH}$ the topology consisting of all finite graphs.
- The star topology of depth ℓ (with $\ell \geq 0$), denoted $\mathtt{STAR}(\ell)$, characterizes graphs G for which there is a partitioning $\{v_0\} \cup V_1 \cup \cdots \cup V_\ell$ of V 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 exactly one $v_{i+1} \in V_{i+1}$ with $v_i \sim v_{i+1}$, (iii) for each $1 < i \leq \ell$ and $v_i \in V_i$ there is exactly one $v_{i-1} \in V_{i-1}$ with $v_i \sim v_{i-1}$, and (iv) no other nodes are adjacent to each other. In other words, in a star topology of dimension ℓ, there is a central node v_0 and an arbitrary number of *rays*. A ray consists of a sequence of ℓ nodes, starting from v_0 followed by some $v_1 \in V_1$, and then by some $v_2 \in V_2$, etc. We call v_0 the *root*, call the nodes in $V_1, \ldots, V_{\ell-1}$ the *internal nodes*, and call the nodes in V_ℓ the *leaves* of G.
- The bounded path topology of bound ℓ (with $\ell \geq 0$), denoted $\mathtt{BOUNDED}(\ell)$, characterizes graphs G for which the length of the maximal simple path in G is bounded by ℓ. This means that there does not exist a finite sequence of vertices $(v_i)_{1 \leq i \leq m}$ satisfying the following conditions (1) $m > \ell$, (2) $v_i \neq v_j$ for all i, j in $\{1, \ldots, m\}$ such that $i \neq j$ and (3) $v_i \sim v_{i+1}$ for all i such that $1 \leq i < m - 1$.
- The set of cliques, denoted $\mathtt{CLIQUE}$ characterizes graphs G where $v_1 \sim v_2$ for all $v_1, v_2 \in V$ with $v_1 \neq v_2$.

Reachability We assume a distinguished local state $q^{init} \in Q$. A configuration γ^{init} is said to be *initial* if it is of the form $\left(\mathcal{Q}_Q^{init}, \mathcal{X}^{init}\right)$ where $\mathcal{Q}_Q^{init}(v) = q^{init}$, and $\mathcal{Q}_Q^{init}(v)(x) = 0$ for all $v \in V$ and $x \in X$. In other words, all the processes are in their initial local states and all the clocks have value 0. A *computation* π of $\mathcal{T}$ is a sequence $\gamma_0 \Longrightarrow_{\mathcal{T}} \gamma_1 \Longrightarrow_{\mathcal{T}} \cdots \Longrightarrow_{\mathcal{T}} \gamma_n$ where γ_0 is an initial configuration. In such a case, we say that γ_n is *reachable* in $\mathcal{T}$. For a local state $q \in Q$, we say that q is *reachable* in $\mathcal{T}$ if there is a configuration $\gamma = (\mathcal{Q}, \mathcal{X})$ such that γ is reachable in $\mathcal{T}$ and $\mathcal{Q}(v) = q$ for some $v \in V$.

The *(local state) reachability problem* for a topology *Top* and a number K, denoted $\texttt{TAHN}-\texttt{Reach}\,(\mathit{Top}, K)$, is defined as follows:

> Given a protocol P with $\#P = K$ and a local state $q \in Q$, is there a TAHN $\mathcal{T} = (P, G)$ such that $G \in \mathit{Top}$ and q is reachable in $\mathcal{T}$.

The following results concerning untimed Ad Hoc Networks have been proved.

Theorem 1 ([5]). $\texttt{TAHN}-\texttt{Reach}\,(\texttt{GRAPH}, 0)$ *is undecidable. For each* $K \geq 0$*, the problem* $\texttt{TAHN}-\texttt{Reach}\,(\texttt{BOUNDED}(K), 0)$ *is decidable.*

4 Timed Networks

In this section, we recall the model of *Timed Networks* (*TN* for short) [4]. In a similar manner to TAHNs, a TN contains an arbitrary number of identical *timed processes* that operate on a finite number of local real-valued clocks. However, there are three main differences between TNs and TAHNs. First, a TN contains a distinguished *controller* that is a finite-state automaton without any clocks[1]. Second, each process in a TN may communicate with all other processes and hence it is not meaningful to describe topologies in the case of TNs. Finally, communication takes place through rendez-vous between fixed sets of processes rather than broadcast messages. In a similar manner to TAHNs, the values of all clocks in a TN are incremented continuously at the same rate. In addition, the TN can change its configuration according to a finite number of *rules*. Each rule describes a set of transitions in which the controller and a fixed number of processes synchronize and simultaneously change their states. A rule may be conditioned on the local state of the controller, together with the local states and clock values of the processes. If the conditions for a rule are satisfied, then a transition may be performed where the controller and each participating process changes its state. During a transition, a process may reset some of its clocks to 0.

We assume a finite set X of clocks and define guards and resets in a similar manner to TAHNs. A *family of timed networks* (*timed network* for short) $\mathcal{N}$ is a pair $(Q, \mathcal{R})$, where Q is a finite set of *states*, partitioned into a set Q^{ctrl} of

[1] This is the model defined in [4]. Adding clocks to the controller does not affect the decidability results.

controller states, and a set Q^{proc} of *process states*; and $\mathcal{R}$ is a finite set of *rules* where each rule is of the form

$$[q_0 \to q_0'] \; [q_1; g_1 \to R_1; q_1'] \cdots \; [q_n; g_n \to R_n; q_n']$$

such that $q_0, q_0' \in Q^{ctrl}$, for all $i : 1 \leq i \leq n$ we have: $q_i, q_i' \in Q^{proc}$, g_i is a guard, and R_i is a reset. Intuitively, the set Q^{ctrl} represents the states of the controller and the set Q^{proc} represents the states of the processes. A rule of the above form describes a set of transitions of the network. The rule is enabled 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 executed by simultaneously changing the state of the controller to q_0' and the states of the n processes to $q_1', \cdots, q_n'$, and resetting the clocks belonging to the sets $R_1, \ldots, R_n$.

Configurations A configuration γ of a timed network $(Q, \mathcal{R})$ is a tuple of the form $(I, q, \mathcal{Q}, \mathcal{X})$, where I is a finite *index set*, $q \in Q^{ctrl}$, $\mathcal{Q} : I \to Q^{proc}$, and $\mathcal{X} : I \to X \to \mathbb{R}^{\geq 0}$. Intuitively, the configuration γ refers to the controller whose state is q, and to $|I|$ processes, whose states are defined by $\mathcal{Q}$. The clock values of the processes are defined by $\mathcal{X}$. More precisely, for $i \in I$ and $x \in X$, $\mathcal{X}(x)(i)$ gives the value of clock x in the process with index i. We use $|\gamma|$ to denote the number of processes in γ, i.e., $|\gamma| = |I|$.

Transition Relation The timed network $\mathcal{N}$ above induces a transition relation $\longrightarrow_{\mathcal{N}}$ on the set of configurations. The relation $\longrightarrow_{\mathcal{N}}$ is the union of a *discrete* transition relation $\longrightarrow_D$, representing transitions induced by the rules, and a *timed* transition relation $\longrightarrow_T$ which represents passage of time.

The discrete relation $\longrightarrow_D$ is the union $\bigcup_{r \in \mathcal{R}} \longrightarrow_r$, where $\longrightarrow_r$ represents a transition performed according to rule r. Let r be a rule of the form described in the above definition of timed networks. Consider two configurations $\gamma = (I, q, \mathcal{Q}, \mathcal{X})$ and $\gamma' = (I, q', \mathcal{Q}', \mathcal{X}')$. We use $\gamma \longrightarrow_r \gamma'$ to denote that there is an injection $h : \{1, \ldots, n\} \to I$ such that for each $i : 1 \leq i \leq n$ we have:

1. $q = q_0$, $\mathcal{Q}(h(i)) = q_i$, and $\mathcal{X}(h(i)) \models g_i$. That is, the rule r is enabled.
2. $q' = q_0'$, and $\mathcal{Q}'(h(i)) = q_i'$. The states are changed according to r.
3. If $x \in R_i$ then $\mathcal{X}'(h(i))(x) = 0$, while if $x \notin R_i$ then $\mathcal{X}'(h(i))(x) = \mathcal{X}(h(i))(x)$. In other words, a clock is reset to 0 if it occurs in the corresponding set R_i. Otherwise its value remains unchanged.
4. $\mathcal{Q}'(j) = \mathcal{Q}(j)$ and $\mathcal{X}_k'(j) = \mathcal{X}_k(j)$, for $j \in I \setminus \text{range}(h)$, i.e., the process states and the clock values of the non-participating processes remain unchanged.

A *timed transition* is of the form $\gamma \longrightarrow_{T=\delta} \gamma'$ where $\gamma = (I, q, \mathcal{Q}, \mathcal{X})$, $\delta \in \mathbb{R}^{\geq 0}$, $\gamma' = (I, q, \mathcal{Q}, \mathcal{X}')$, $\mathcal{X}'(j)(x) = \mathcal{X}(j)(x) + \delta$ for all $j \in I$ and $x \in X$. We use $\gamma \longrightarrow_T \gamma'$ to denote that $\gamma \longrightarrow_{T=\delta} \gamma'$ for some $\delta \in \mathbb{R}^{\geq 0}$.

We define $\longrightarrow_{\mathcal{N}}$ to be $\longrightarrow_D \cup \longrightarrow_T$ and use $\overset{*}{\longrightarrow}_{\mathcal{N}}$ to denote the reflexive transitive closure of $\longrightarrow_{\mathcal{N}}$. Notice that if $\gamma \longrightarrow_{\mathcal{N}} \gamma'$ then the index sets of γ and γ' are identical and therefore $|\gamma| = |\gamma'|$. For a configuration γ and a controller

state q, we use $\gamma \xrightarrow{*}_{\mathcal{N}} q$ to denote that there is a configuration γ' of the form $(I', q, \mathcal{Q}', \mathcal{X}')$ such that $\gamma \xrightarrow{*}_{\mathcal{N}} \gamma'$.

Given $\gamma_0 \longrightarrow_{\mathcal{N}} \gamma_1 \longrightarrow_{\mathcal{N}} \gamma_2 \ldots \longrightarrow_{\mathcal{N}} \gamma_n$, we say that $\gamma_0, \ldots, \gamma_n$ is a computation of $\mathcal{N}$.

Reachability. We assume a distinguished initial controller state $q^{init}_{ctrl} \in Q^{ctrl}$ and a distinguished initial process state $q^{init}_{proc} \in Q^{proc}$. 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. Notice that there is an infinite number of initial configurations, namely one for each index set I. Concepts such as that of computations and reachability are extended from TAHNs to TNs in the obvious way.

The *(controller state) reachability problem* $\texttt{TN}-\texttt{Reach}\,(K)$ is defined by a timed network $(Q, \mathcal{R})$ with K clocks (in each process), and a controller state q. The task is to check whether q is reachable or not. In [2], the following result is shown.

Theorem 2. $\texttt{TN}-\texttt{Reach}\,(2)$ *is undecidable.*

5 Undecidability with Dense Time

In this section, we show undecidability of the reachability problem for three classes of TAHNs, namely (i) those with *star* topologies of depth 2 (one root and several rays with two nodes) and with a single clock in each node; (ii) those with *clique* topologies provided that each node has two clocks; and (iii) those with *bounded depth* topologies if the depth of the underlying graph is 4 and with a single clock in each node. In the first two cases, the undecidability result is shown through a reduction from $\texttt{TN}-\texttt{Reach}\,(2)$ (that is undecidable by Theorem 2). The main idea of the proofs is to show that we can simulate broadcast (the communication model of TAHNs) by rendez-vous (the communication model of TNs). In each case we will simulate a TN $\mathcal{N}$ with two clocks per process by a TAHN $\mathcal{T}$. 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 $\mathcal{T}(q)$ in $\mathcal{T}$. Furthermore, we will have a number of auxiliary states in $\mathcal{T}$ that we need to perform the simulation. The third undecidability result (for bounded depth topologies) is shown by simulating the star case.

Two-Star Topologies We show that the reachability problem for the star topology is undecidable even when the rays are restricted to have depth 2 and the nodes are restricted to have a single clock.

Theorem 3. $\texttt{TAHN}-\texttt{Reach}\,(\texttt{STAR}(2), 1)$ *is undecidable.*

Given a TN $\mathcal{N} = (Q, \mathcal{R})$ and a controller state q in $\mathcal{N}$, we define a TAHN $\mathcal{T} = (P, G)$ such that $G \in \texttt{STAR}(2)$ and $\#P = 1$, together with a local state

$\mathcal{T}(q)$, such that q is reachable in $\mathcal{T}$ iff q is reachable in $\mathcal{N}$. The root of $\mathcal{T}$ plays the role of the controller in $\mathcal{N}$. Furthermore, each ray in $\mathcal{T}$ plays the role of one process in $\mathcal{N}$. The local state of a process in $\mathcal{T}$ is stored in the internal node of the corresponding ray. Furthermore, the two clocks x_1, x_2 of a process are represented by the clock of the internal node resp. the clock of the leaf of the ray. For technical reasons, we require that $\mathcal{T}$ has at least three rays. 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). The simulation 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 is to identify the nodes that play the roles of the controller and those that play the roles of the different processes. First, a node may broadcast a message where it requests to become the node that simulates the controller in $\mathcal{N}$. In order to succeed, P requires that it should receive acknowledgements from at least three other processes.

Notice that only the root of $\mathcal{T}$ can be successful since 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).

Once the root has become the controller, it will make the internal nodes aware of their positions. It does that by sending a broadcast message. Due to the star topology, this message is received only by the internal nodes. A node receiving this broadcast message will initiate a "local protocol" inside its ray as follows: (i) It changes local state to reflect that it now knows that it is indeed an internal node. (ii) It makes the leaf of the ray aware of its position by broadcasting a message. Such a message is received only by the leaf of the ray and by the root (the latter simply ignores the message). (iii) The leaf broadcasts an acknowledgment (that can only be received by the internal node of the ray). (iv) The internal node changes state when it receives the acknowledgement and declares itself ready for the next step. Notice that the internal node and the leaf may choose to ignore performing steps (ii) or (iv). In such a case we say that the ray has "failed", 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 takes places: (i) It changes local state to $\mathcal{T}(q^{init}_{ctrl})$ 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 $\mathcal{T}(q^{init}_{proc})$. The rest of the nodes will remain passive throughout the rest of the simulation. Now all the nodes are ready: the root of $\mathcal{T}$ 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 $\mathcal{N}$, and all clocks have values equal to 0.

Simulating Discrete Transitions Below, we show how $\mathcal{T}$ simulates a rule of the form $[q_0 \to q'_0]\,[q_1; g_1 \to R_1; q'_1]\ \cdots\ [q_n; g_n \to R_n; q'_n]$. The root of $\mathcal{T}$ is in

the state $\mathcal{T}(q_0)$. First, the root resets its clock to 0 (this is done so that it can later make sure that the simulation of the rule has not taken 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. An (internal) node that receives the broadcast 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 has guessed (perhaps wrongly) that its clocks satisfy the values specified by the guard. If the node has guessed wrongly 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 each of x_1 and x_2 separately. 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 itself. 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 $\mathcal{T}(q_1')$. (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 to play the role of the first process, the root enters en error state if it happens to receive acknowledgements from several internal nodes. 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 $\mathcal{T}(q_0')$ (ii) It sends a final boraodcast where the node ready for simulating the i^{th} process will now move to $\mathcal{T}(q_i')$ 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).

The protocol associated to each phase is detailed in the appendix.

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

Cliques We show that the reachability problem for the clique topology is undecidable if the nodes have two clocks.

Theorem 4. $\texttt{TAHN}-\texttt{Reach}(\texttt{CLIQUE}, 2)$ *is undecidable.*

We will build a protocol P with $\#P = 2$ which will simulate $\mathcal{N}$ on 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 $\mathcal{T}(q_{ctrl}^{init})$, while all the other nodes will move to $\mathcal{T}(q_{proc}^{init})$.

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 reset its clock to 0. The simulation again consists of different phases, where in each phase the controller node tries to identify a ray 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 to try to 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 $\mathcal{T}(q_1')$. To prevent multiple nodes to play the role of the first process, the controller node enters en error state if it happens to receive acknowledgements from several nodes. 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.

Bounded Path Topologies Using the result of Theorem 3 we can show that 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 (number of vertices) simple paths can have 5 vertices 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 an arbitrary graph in paths are allowed to have (at least) five nodes. The discovery protocol first selects root, internal and leaf candidates and then verify that they are connected in the desired way by sending all other nodes in their vicinities to a special null state. The complete discovery protocol is detailed

in the appendix. Combining the discovery protocol and the undecidability results for two-star topology we obtain the following theorem.

Theorem 5. $\texttt{TAHN}-\texttt{Reach}\,(\texttt{BOUNDED}(5), 1)$ *is undecidable.*

6 Decidability with Dense Time

In the previous section we showed that $\texttt{TAHN}-\texttt{Reach}\,(\texttt{STAR}(2), 1)$ is undecidable. In this section we consider two other classes of topologies for which reachability becomes decidable when nodes have a single clock, namely the class $\texttt{STAR}(1)$ and $\texttt{CLIQUE}$. A convenient way to prove these results is to resort to an extension of Timed Networks with transfers for which control state reachability is still decidable. When executed together with a rendez-vous, a transfer action from q to q' forces all processes in state q to move to state q' (as in transfer arcs for Petri nets). We give next a more detailed definition of the extended TN model.

Timed Networks with Transfers (TNT) A rule of a *timed network with transfer* (TNT) $\mathcal{N}$ combines rendez-vous synchronization and transfer actions. Namely, a TNT rule has the following form:

$$\begin{array}{l}[q_0 \to q_0']\ [q_1; g_1 \to R_1; q_1']\ \cdots\ [q_n; g_n \to R_n; q_n'] \\ \qquad [p_1; g_1' \to_t R_1'; p_1'] \cdots [p_\ell; g_\ell' \to_t R_\ell'; p_\ell']\end{array}$$

where actions with $\to$ denote rendez-vous communication, whereas actions with $\to_t$ denote transfers. For $i : 1 \leq i \leq \ell$, $g_i \to_t R_i$ is a guarded transfer where g_i is a guard, R_i is a reset, $p_i, p_i' \in Q^{proc}$. We assume that there are no $i : 1 \leq i \leq \ell$ and $j : 1 \leq j \leq n$ such that $p_i = q_j$; and assume that $p_i \neq p_j$ if $i \neq j$. A TNT rule is enabled 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 executed by simultaneously changing the state of the controller to q_0' and together with the states of selected processes $p_1, \cdots, p_\ell$. Furthermore, each process in state p_i for $i : 1 \leq i \leq \ell$ whose clock value satisfies the guard g_i' moves to p_i' while its clocks are reset according to R_i'. Note that for a rule to be enabled there is no requirement on the presence or absence of processes in states $p_1, \cdots, p_\ell$.

A formal account of the semantics of TNT rules is given in appendix. The (controller state) reachability problem $\texttt{TNT}-\texttt{Reach}\,(K)$ for processes with K clocks is defined by replacing the TN transition relation with that for TNT. The following result then holds.

Theorem 6. $\texttt{TNT}-\texttt{Reach}\,(1)$ *is decidable.*

Sketch of proof. In [4], it is proved that $\texttt{TN}-\texttt{Reach}\,(1)$ is decidable. The proof is based on the general results in [1] based on a well quasi orderings of TN configurations under which the transition relation is monotonic. Monotonicity of the transition relation of a TNT still holds for the same ordering used in [4].

Furthermore, the algorithm used for computing predecessors can be extended in order to cope with transfer action. This extension is similar to that used for Data nets [9] or, for processes without clocks, to that used for Petri nets with transfer arcs. We prove next that in TAHNs with one clock restricted to the clique topology, the reachability problem is decidable.

Theorem 7. $\texttt{TAHN}-\texttt{Reach}\,(\texttt{CLIQUE}, 1)$ *is decidable.*

Proof. We reduce $\texttt{TAHN}-\texttt{Reach}\,(\texttt{CLIQUE}, 1)$ to $\texttt{TNT}-\texttt{Reach}\,(1)$. The reduction works as follows. Since in a clique graph all nodes are connected to each other, a broadcast message sent by a node is always received by all other nodes. In other words working with a clique is like working with a multiset of processes as in a TNT. Broadcast communication can then be simulated by using TNT rules in which the sender perfoms an individual step and reception of a message is simulated by transfer actions, one for each state in which the message can be received. Furthermore, we can insert a special rule to synchronize the controller with the local state we want to reach in the TAHN. Local state reachability corresponds then to control state reachability in the resulting TNT. The claim then follows from Theorem 6.

A similar positive result can be obtained for TAHN with 1 clock restricted to the star topology of depth 1. The main difference compared to the previous result is that in a star of depth 1 we have to distinguish the root (the central node) from the leaves. When the root performs a broadcast, it is transmitted to all the leaf nodes, but when a leaf performs one, only the root can receive it.

Theorem 8. $\texttt{TAHN}-\texttt{Reach}\,(\texttt{STAR}(1), 1)$ *is decidable.*

Proof. We reduce $\texttt{TAHN}-\texttt{Reach}\,(\texttt{STAR}(1), 1)$ to $\texttt{TNT}-\texttt{Reach}\,(1)$. Let $\mathcal{T}$ be a TAHN with star topology of depth 1. We construct a TNT N that simulates $\mathcal{T}$ as follows. Initially all TNT processes are in state q^i. We first define an initialization step in which the controller non-deterministically selects one of the processes in state q^i and elects it as the root of the simulated TAHN. All the remaining processes in state q^i are then used to simulate the leaves of the star. After this step, the simulation of each TAHN rule r is split in two TNT rules: one for the root process and one for a leaf process. A local rule is simulated by a rendez-vous (with no transfer) for the root process and by a rendez-vous for a leaf process. A broadcast rule executed by the root node is simulated with a TNT transfer action involving the root process and all the leaf processes, whereas a broadcast executed by a leaf is translated into a rendez-vous between a leaf and the root process only. As for cliques, we can add rules for the controller to observe when the root of a leaf process has reached a given local state. The claim then follows from Theorem 6.

7 Decidability with Discrete Time

In this section we study the reachability problem for Discrete Time Ad Hoc Networks (DTAHN). In this model clocks range over the natural numbers instead

of the reals. Let μ' be the maximum constant used in the protocol rules and let $\mu = \mu' + 1$. When using discrete time, it is enough to restrict the evaluation of clocks to the finite range $I = [0, \mu]$. This follows from the fact that guards of the form $x > c$ remain enabled when time passes once the clock associated to variable x reaches a value greater or equal to μ; while guards of the form $x \leq c$ remain disabled. Therefore, beyond μ we need not distinguish between different values for the same clock (see e.g. [2]). For every DTAHN $\mathcal{T}$, we can then define a finite-state process that describes the behavior of a node. We use $\mathcal{C}_{\mu,K}$ to denote configurations over undirected graphs with labels in $Q \times I^K$, where Q is the set of local states of a process, I is the interval $[0, \mu]$, and K is the number of clocks in each process.

The transition relation $\rightharpoonup_{\mathcal{T}}$ is obtained from that of TAHN by assuming that the evaluation of clock variables is a function $\mathcal{X} : V \mapsto [X \to I]$ and by replacing the time step by the discrete time step defined as follows.
Discrete Time Step For configurations $\gamma = (\mathcal{Q}, \mathcal{X})$ and $\gamma' = (\mathcal{Q}', \mathcal{X}')$, we write $\gamma \rightharpoonup_{\mathcal{T}} \gamma'$ if, 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.

For a topology class *Top* and $K \geq 0$, the control state reachability problem $\texttt{DTAHN}-\texttt{Reach}\,(\mathit{Top}, K)$ is the natural reformulation of the one defined for TAHN.

We show next that reachability is decidable when restricting the topology to the class of bounded path graphs $\texttt{BOUNDED}(N)$ for any fixed $N > 1$. The decision procedure is obtained by resorting to the theory of well-structured transition systems [1]. The procedure is based on a symbolic backward exploration algorithm in which we use *constraints* to finitely represent sets of configurations of *variable size* taken from the class $\texttt{BOUNDED}(N)$ (the configurations may potentially belong to different TAHNs).

Ordering We first introduce the following ordering between configurations of variable size. Given configurations $\gamma = (\mathcal{Q}, \mathcal{X})$ defined over $G = (V, E)$ and $\gamma' = (\mathcal{Q}', \mathcal{X}')$ defined over $G' = (V', E')$, $\gamma \preceq \gamma'$ 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 $(h(u), h(u')) \in E'$;
- $\forall u \in V$, $\mathcal{Q}(u) = \mathcal{Q}'(h(u))$ and $\mathcal{X}(u) = \mathcal{X}'(h(u))$.

An upward closed set U satisfies the property that $U = \{\gamma' \mid \gamma \preceq \gamma',\ \gamma \in U\}$. The following property then holds.

Proposition 1. *If U is an upward closed set of configurations (of variable size), then $Pre(U) = \{\gamma \mid \gamma \rightharpoonup_{\mathcal{T}} \gamma',\ \gamma' \in U\}$ is still upward closed wrt. $\preceq$.*

Proof. In [5] it has been proven, that for untimed AHN, selective broadcast is monotone w.r.t. $\preceq$. We observe that DTAHN restricted to configurations in $\mathcal{C}_{\mu,K}$

can be viewed as untimed AHN extended with a time step transition. Thus, we just have to show monotonicity wrt. a time step $\gamma \rightharpoonup_{\mathcal{T}} \gamma'$. Since time can always proceed, for every $\beta \succeq \gamma$ there is a configuration β' such that $\beta \rightharpoonup_{\mathcal{T}} \beta'$ with a time step. Now since h is label preserving both time steps will have the same effect on nodes identified by h and thus $\gamma' \preceq \beta'$.

Constraints Assume a given process definition P. A constraint is a tuple $\Phi = (G, \mathcal{Q}, \mathcal{X})$, where $G = (V, E)$ is a graph in $\texttt{BOUNDED}(N)$, and $\mathcal{Q}$ and $\mathcal{X}$ are defined on the set of vertices V. We call γ_Φ the associated configuration $(\mathcal{Q}, \mathcal{X})$ over the graph G. The denotation of a constraint Φ is defined as the infinite set of configurations $[\![\Phi]\!] = \{\gamma' \mid \gamma_\Phi \preceq \gamma'\}$ with variable size and topology. The following proposition then holds.

Proposition 2. *Given a constraint Φ, there exists an algorithm to compute a finite set of constraints whose denotation corresponds to $Pre([\![\Phi]\!])$.*

Proof. We can extend the symbolic predecessor operator defined for untimed AHN in [5] by adding some new conditions and steps. In order to correctly obtain predecessors of γ, rules of the form $\left(q, g \xrightarrow{e} R, q'\right)$ must be applied only to nodes in state q' which satisfy the post-condition R. When R is not empty, i.e. some clocks have been reset, we do not know which was their values before the step. In this case it is necessary to add a predecessor to the set $Pre(\gamma)$ for every possible combination of $i \in I$ for those clocks – keeping the old values of the node from configuration γ otherwise.
As an addition to regular protocol rules, we also have to consider time steps. Thus, for every configuration γ that does not have 0-valued clocks, we have to add to its set of predecessors every configuration γ' such that γ' can make a time step into γ.

Theorem 9. $\texttt{DTAHN}-\texttt{Reach}(\texttt{BOUNDED}(N), K)$ *is decidable for any $N \geq 1$ and $K \geq 0$.*

Proof. From propositions 1 and 2, we can apply the general results in [1] to define a backward search algorithm working on upward closed sets of extended configurations represented by their finite basis. The finite set of constraints (G, Q, X) in which G is a single node v, $Q(v) = q$, and $X(v) \in I$ can be used as finite representation of all configurations containing the control state q. Furthermore, for a fixed $N \geq 1$ the induced subgraph relation is a well-quasi-ordering on the class of N-bounded path graphs [7]. This implies that for any sequence of constraints $\Phi_0\Phi_1 \ldots$ there exist $i < j$ s.t. $\Phi_i \preceq \Phi_j$. This property ensures the termination of the symbolic backward reachability algorithm.

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. 1 shows a summary of our analysis. We also mention decidability

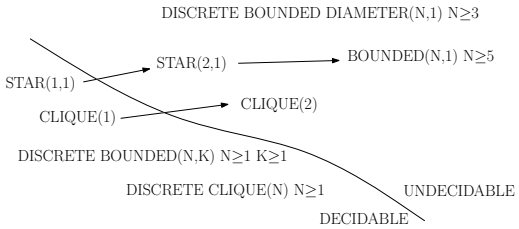

Fig. 1. Decidability and undecidability results.

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 [6].

References

1. P. Abdulla, K. Cerans, B. Jonsson, and Y. Tsay. General decidability theorems for infinite-state systems. In *LICS'96*, pages 313–321. IEEE Computer Society, 1996.
2. P. Abdulla, J. Deneux, and P. Mahata. Multi-clock timed networks. In *LICS'04*, pages 345–354. IEEE Computer Society, 2004.
3. P. Abdulla and A. Nylén. Timed petri nets and bqos. In *ICATPN'01*, volume 2075 of *LNCS*, pages 53–70. Springer, 2001.
4. P. A. Abdulla and B. Jonsson. Model checking of systems with many identical timed processes. *TCS*, 290(1):241–264, 2003.
5. G. Delzanno, A. Sangnier, and G. Zavattaro. Parameterized verification of ad hoc networks. In *CONCUR'10*, volume 6269 of *LNCS*. Springer, 2010.
6. G. Delzanno, A. Sangnier, and G. Zavattaro. On the power of cliques in the parameterized verification of ad hoc networks. In *FoSSaCS'11*, volume 6604 of *LNCS*, pages 441–455. Springer, 2011.
7. G. Ding. Subgraphs and well quasi ordering. *J. of Graph Theory*, 16(5):489–502, 1992.
8. A. Fehnker, L. van Hoesel, and A. Mader. Modelling and verification of the LMAC protocol for wireless sensor networks. In *IFM'07*, volume 4591 of *LNCS*, pages 253–272. Springer, 2007.
9. R. Lazic, T. Newcomb, J. Ouaknine, A. Roscoe, and J. Worrell. Nets with tokens which carry data. *Fund. Inf.*, 88(3):251–274, 2008.
10. M. Merro, F. F. Ballardin, and E. Sibilio. A timed calculus for wireless systems. In *Proc. of the 3rd Conference on Fundamentals of Software Engineering (FSEN'09)*, volume 5961 of *LNCS*, pages 228–243. Springer, 2010.
11. M. Saksena, O. Wibling, and B. Jonsson. Graph grammar modeling and verification of Ad Hoc Routing Protocols. In *TACAS'08*, volume 4963 of *LNCS*, pages 18–32. Springer, 2008.
12. A. Singh, C. R. Ramakrishnan, and S. A. Smolka. Query-based model checking of ad hoc network protocols. In *CONCUR'09*, volume 5710 of *LNCS*, pages 603–619. Springer, 2009.
13. UPPAAL. http://www.uppaal.com/.

A Encoding a TN in a Two-Star Topology

In this section we present in more detail states and rules that must be added to an initially empty TAHN in order to encode a TN protocol. As a convention, we omit state labels every time we want fresh ones. Fig. 2 shows the protocol that initializes the simulation. Every node in the system starts in state q^{init} and tries to count the number of neighbors. The result of this operation is a differentiation of the roles – root, internal and leaf nodes will act according to their duties. Since leaf nodes are interesting only because of their clock, they don't need any particular state information to work properly. Once correctly initialized, before and after each TN step simulation their state remains q^{ok}. In the diagram of Fig. 2 the leftmost vertical branch correspond to selection of leaf nodes, the branch in the middle to selection of internal nodes, and he rightmost one to selection of the root node.

Now let us consider a TN rule r like the one below.

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

Remember that we are working with nodes with a single clock with variable named x. A condition g on the original TN with two clocks x_1 and x_2 must be checked both on the internal node and on the leaf node of the ray. Therefore, we write $g(x_j \leftarrow x)$ as a shorthand to mean the guard obtained by first projecting g on the constraints involving only the variable x_j, and then by replacing x_j with x in the resulting formula. For instance, if $g = x_1 \geq 2, x_2 = 4$, then $g(x_1 \leftarrow x) = x \geq 2$ and $g(x_2 \leftarrow x) = x = 4$ Furthermore, for a reset R we define $R(x_j \leftarrow x) = \{x\}$ if $x_j \in R$, and $= \emptyset$ otherwise, i.e., we map a reset on x_j to a reset on clock variable x. For instance, if $R = \{x_1\}$, then $R(x_1 \leftarrow x) = \{x\}$ and $R(x_2 \leftarrow x) = \emptyset$. The timed automaton for each of them will have instances of protocols from figures 3, 4 and 5 composed as described next. The root node first executes a step that resets its clock from $\mathcal{T}(q_0)$ to q_1^r and then executes the first instance of ray selection as in figure 3. Auxiliary states and messsages are indexed by the simulated rule r and by the current simulated action i. Given $0 < i \leq n$, the simulation goes on from q_i^r to q_{i+1}^r until q_{n+1}^r is reached, meaning that a complete matching of rule r has been found. At this point the root makes a last broadcast step $!!done$ and goes to state $\mathcal{T}(q_0')$ if and only if its clock is still 0, forcing every selected ray to switch to their respective final states.

B Proof of Theorem 5: Discovery Protocol

In this section we define a discovery protocol that can be used to identify a star topology with rays of depth two in any graph in which paths can have length five (five vertices). We define the protocol $P = (Q, \mathcal{R})$ with $\#P = 1$ with $q^{init} \in Q$ a initial states. We denote by x the clock used by P. The description of the protocol

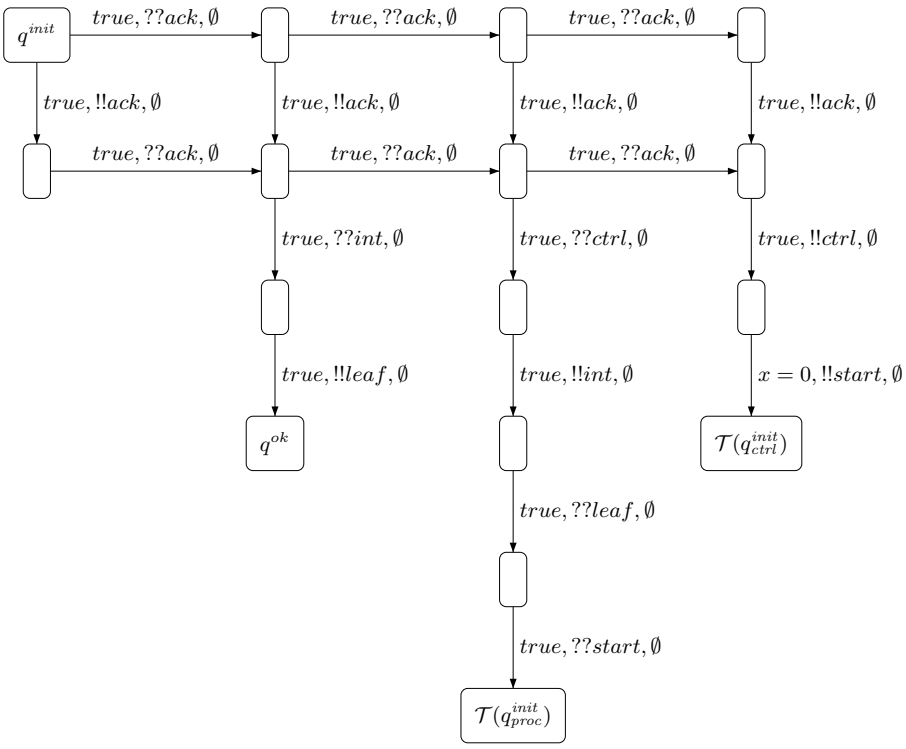

Fig. 2. Initializating the simulation

is given by the Figures 6. We assume that from the initial state P has three transitions labelled with empty event which allows to choose non-deterministically for each node whether it is going to be the root (control state q_0), a ray (control state r_0 or a leaf (control state s_0) of the star topology. These three rules have then following form: $\Big(q^{init}, x = 0 \xrightarrow{\tau} \emptyset, q_0\Big), \Big(q^{init}, x = 0 \xrightarrow{\tau} \emptyset, r_0\Big), \Big(q^{init}, x = 0 \xrightarrow{\tau} \emptyset, s_0\Big)$.

The behavior of the node chosen to be the root which is in state q_0 is given by Figure 6. This node will first send a broadcast of a message *root* and if it receives any messages before, then it goes in the error state. This will ensure that all the other nodes connected to it which were also in state q_0 will be in the error state, and it will signal to the nodes connected to it which are in state r_0 and which are choose to be rays of the star that the building of the star has begun.

The nodes in control state r_0 on reception of the message *root* will run the protocol depicted on Figure 7. Basically the main aim of this protocol is to ensure

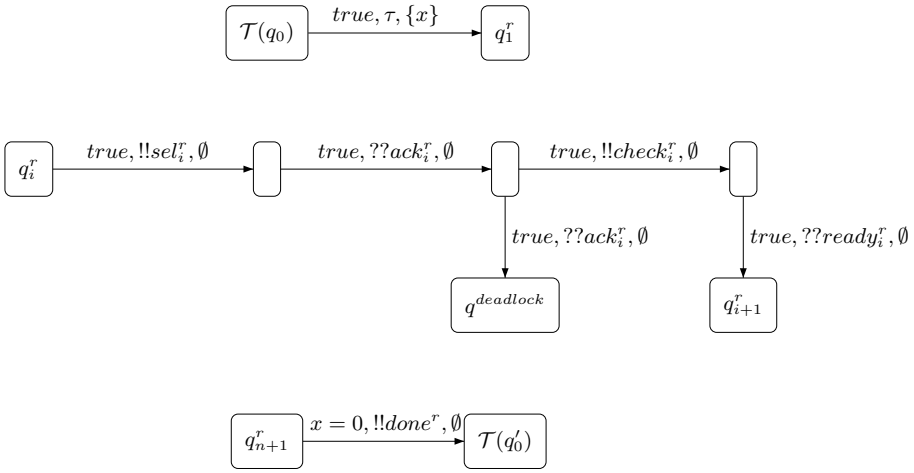

Fig. 3. Ray selection: root node

that when the node choosen as the root node send the message *endroot*, then all the nodes to which it was connected which were in state r_0 before are either in state *null* or in state r_0' and furhtermore none of the state in r_0' is connected to another state which was initially in r_0. This can be ensured by the fact that each node receive a message *root* has to sent a message *ackroot* to arrive in state r_0', this message *ackroot* is not received the root node, but is used to bring adjacent nodes in state r_0 to the null 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 useful to ensure that a ray node is node connected to two different nodes chosen as root node. So when reaching the state r_0' is connected to at most one node which was in state q_0 and wich is not in state *null* and it is not connected to any node which was previously in state r_0 and which is not in state *null*.

Finally, the Figures 8 and 9 shows the dialog between the nodes chosen for the leaves and the one chosen for the rays. A node which in state s_0 will at some point send a message $leaf$ to its adjacent node, it has to do it when the other node is in r_0' otherwise, the adjacent node will go to the state *null* by applying the rule of Figure 7. Then the protocol will ensure that when a ray node reaches r_F it is connected exactly to one node in state s_F and also (by the previous explanation) to one node in state q_F and furthermore each node in state s_F is connected to exactly one in state q_F.

Finally, we have that if $\mathcal{T} = (P, G)$ is a TAHN such that $G \in \texttt{BOUNDED}(5)$, then all configurations $\gamma = (\mathcal{Q}, \mathcal{X})$ reachable in $\mathcal{T}$ satisfy the following properties:

- for all vertices $v \in V$ 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 vertices $v_1, v_2 \in V$ 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 vertices $v' \in V \setminus \{v_1, v_2\}$, $v' \sim v$ 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 $\mathcal{Q}(v') = r_F$.

Basically, this means that if γ is a configuration reachable in $\mathcal{T}$ then all the nodes in the state q_F can be seen as the root node of a star of depth 2 where the rays are in state r_F and the leaves in state q_F and all the other nodes connected to these nodes are in state *null* and will not take part to the further communication. Hence from q_F using the protocol proposed in the proof of Theorem 3, we can now simulate the behavior of a $\mathcal{N}$ as if we were in a star of depth 2.

We have then that given a $\mathcal{N}$ and a final control state, if the associated control state is reachable for some $\mathcal{T}$ restricted to topologies in $\texttt{BOUNDED}(5)$, then it has to be reachable for some $\mathcal{T}'$ restricted to topologies in $\texttt{STAR}(2)$ equipped with the protocol of Theorem 3. This holds because, the first part of our protocol will build the star topology and then simulate $\mathcal{N}$ in this star. On the other hand, assume the final control state is reachable in $\mathcal{N}$, then using the proof of Theorem 3, we know there exists $\mathcal{T}'$ restricted to topologies in $\texttt{STAR}(2)$ equipped with the protocol of Theorem 3 for which the associated control state is reachable, and since any star of depth 2 can be obtained with our pre-protcol, we deduce that the associated control state will also be reachable in a $\mathcal{T}$ restricted to topologies in $\texttt{BOUNDED}(5)$ equipped with the protocol we just defined.

C Timed Networks with Transfers (TNT)

A TNT $\mathcal{N}$ is is a pair $(Q, \mathcal{R})$, where Q is a finite set of *states*, partitioned into a set Q^{ctrl} of *controller states*, and a set Q^{proc} of *process states*; and $\mathcal{R}$ is a finite set of *rules* where each rule is of the form

$$\begin{bmatrix} q_0 \\ \rightarrow \\ q_0' \end{bmatrix} \quad \begin{bmatrix} q_1 \\ g_1 \rightarrow R_1 \\ q_1' \end{bmatrix} \quad \cdots \quad \begin{bmatrix} q_n \\ g_n \rightarrow R_n \\ q_n' \end{bmatrix} \quad \begin{bmatrix} p_1 \\ g_1' \rightarrow_t R_1' \\ p_1' \end{bmatrix} \quad \cdots \quad \begin{bmatrix} p_\ell \\ g_\ell' \rightarrow_t R_\ell' \\ p_\ell' \end{bmatrix}$$

where the first part is exactly as for the rules for Timed Networks and for all $i : 1 \leq i \leq \ell$ we have that $g_i \rightarrow_t R_i$ is a guarded transfer command where g_i is a guard and R_i is a reset and $p_i, p_i' \in Q^{proc}$ and there does not exists $j : 1 \leq j \leq n$ such that $p_i = q_j$ neither $j : 1 \leq j \leq \ell$ such that $i \neq j$ and $p_i = p_j$. A rule of the above form is enabled 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 executed by simultaneously changing the state of the controller to q_0' and the states of the n processes as in Timed Networks and furthermore each process in state p_i for $i : 1 \leq i \leq \ell$ whose clock value satisfies the guard g_i' will have its state changed into p_i' and its clock will be reset according to R_i'. Note that for a rule to be enabled there is no requirement on the presence or not of process in states $p_1, \cdots, p_\ell$. A TNT $\mathcal{N}$ induces a transition relation $\longrightarrow_{\mathcal{T}}$ on the set of configurations (that have the same form as TN configurations). The relation $\longrightarrow_{\mathcal{T}}$ is the union of a *discrete* transition relation $\longrightarrow_D$, representing transitions induced by the rules, and a *timed* transition relation $\longrightarrow_T$ which

represents passage of time. The discrete relation $\longrightarrow_D$ is the union $\bigcup_{r\in\mathcal{R}} \longrightarrow_r$, where $\longrightarrow_r$ represents a transition performed according to rule r. Let r be a rule of the form described in the above definition of timed networks. Consider two configurations $\gamma = (I, q, \mathcal{Q}, \mathcal{X})$ and $\gamma' = (I, q', \mathcal{Q}', \mathcal{X}')$. We use $\gamma \longrightarrow_r \gamma'$ to denote that there is an injection $h : \{1, \ldots, n\} \to I$ such that for each $i : 1 \leq i \leq n$ and $k : 1 \leq k \leq K$ we have:

1. $q = q_0$, $\mathcal{Q}(h(i)) = q_i$, and $g_i(\mathcal{X}_1(h(i)), \ldots, \mathcal{X}_K(h(i)))$ holds. That is, the rule r is enabled.
2. $q' = q'_0$, and $\mathcal{Q}'(h(i)) = q'_i$. The states are changed according to r.
3. If $x_k \in R_i$ then $\mathcal{X}'_k(h(i)) = 0$, while if $x_k \notin R_i$ then $\mathcal{X}'_k(h(i)) = \mathcal{X}_k(h(i))$.

and for all $j \in I \setminus \text{range}(h)$

1. If $\mathcal{Q}(j) = p_i$ for some $1 \leq i \leq \ell$ and if $g'_i(\mathcal{X}_1(j), \ldots, \mathcal{X}_K(j))$ holds, then $\mathcal{Q}'(j) = p'_i$ and $\mathcal{X}'_k(j) = 0$ for each $x_k \in R'_i$ and $\mathcal{X}'_k(j) = \mathcal{X}_k(j)$ for each $x_k \notin R'_i$.
2. Otherwise $\mathcal{Q}'(j) = \mathcal{Q}(j)$ and $\mathcal{X}'_k(j) = \mathcal{X}_k(j)$.

The timed transition relation is then the same as for timed networks. An the (controller state) reachability problem $\texttt{TNT}-\texttt{Reach}\,(K)$ for timed networks with transfer working over K clocks is defined as for timed networks.

D Proof of Theorem 7

Let $P = (Q, \mathcal{R})$ be a protocol with $\#P = 1$ and let $q^{init}, q_F \in Q$. Without loss of generality we assume that if $\left(q_1, g_1 \xrightarrow{??a} R_1, q'_1\right)$ and $\left(q_2, g_2 \xrightarrow{??a} R_2, q'_2\right)$ are two different rules in $\mathcal{R}$ then $q_1 \neq q_2$ (in other words from each local state there is at most one rule per message a labelled with $??a$). Note that this restriction can easily be obtained by adding empty event rules to the model. We now show how to build a timed network with transfert $\mathcal{N} = (Q', \mathcal{R}')$ with 1 clock which will simulate the behavior of P. First the set of controller states Q^{ctrl} is equal to $\{q^c_0, q^c_1\}$ (with $\{q^c_0, q^c_1\} \cap Q = \emptyset$) and the set of process states Q^{proc} is equal to Q. We assume that q^c_0 is the initial controller state and q^{init} is the initial process state. The finite set of rules $\mathcal{R}'$ is the smallest set satisfying the following conditions:

– For each rule $\left(q, g \xrightarrow{\tau} R, q'\right)$ in $\mathcal{R}$ there is a rule in $\mathcal{R}'$ of the form:

$$\begin{bmatrix} q^c_0 \\ \to \\ q^c_0 \end{bmatrix} \quad \begin{bmatrix} q \\ g \to R \\ q' \end{bmatrix}$$

– For each rule $\left(q, g \xrightarrow{!!a} R, q'\right)$ in $\mathcal{R}$, there is a rule in $\mathcal{R}'$ of the form:

$$\begin{bmatrix} q^c_0 \\ \to \\ q^c_0 \end{bmatrix} \quad \begin{bmatrix} q \\ g \to R \\ q' \end{bmatrix} \quad \begin{bmatrix} p_1 \\ g_1 \to_t R_1 \\ p'_1 \end{bmatrix} \quad \cdots \quad \begin{bmatrix} p_\ell \\ g_\ell \to_t R_\ell \\ p'_\ell \end{bmatrix}$$

such that the set $\{\left(p_i, g_i \xrightarrow{??a} R_i, p_i'\right) \mid 1 \leq i \leq \ell\}$ is exactly the subset of $\mathcal{R}$ containing all the rules of the form $\left(q_0, g' \xrightarrow{??a} R', q_0'\right)$. Intuitively, this rule select non deterministically a node which will perform a broadcast and then simulate the broadcast using the transfert.

– There is a rule in $\mathcal{R}'$ of the form:

$$\begin{bmatrix} q_0^c \\ \rightarrow \\ q_1^c \end{bmatrix} \quad \begin{bmatrix} q_F \\ (0 \leq x) \rightarrow \emptyset \\ q_F \end{bmatrix}$$

Intuitively this rule allows the controller to detect that one of the process has reached the state q_F.

With this construction and using the semantics of both TAHN and TN with transfert nodes, one can easily prove that there is a TAHN $\mathcal{T} = (P, G)$ such that $G \in \texttt{CLIQUE}$ and q_F is reachable in $\mathcal{T}$ iff the controller state q_1^c is reachable in $\mathcal{N}$. Hence since $\mathcal{N}$ works over one clock, using Theorem 6 we obtain the desired result.

E Proof of Theorem 8

Let $P = (Q, \mathcal{R})$ be a protocol with $\#P = 1$ and let $q^{init}, q_F \in Q$. As for the proof of Theorem 7 we assume that from each local state there is at most one rule per message a labelled with $??a$. We now show how to build a timed network with transfert $\mathcal{N} = (Q', \mathcal{R}')$ with 1 clock which will simulate the behavior of P. First the set of controller states Q^{ctrl} is equal to $\{q_0^c, q_1^c, q_2^c\}$ (with $\{q_0^c, q_1^c\} \cap Q = \emptyset$) and the set of process states Q^{proc} is equal to $Q \cup q_r$ where $Q_r = \{q_r \mid q \in Q\}$. The set q^r will be used to distinguish the root from the leaves. We assume that q_0^c is the initial controller state and q^{init} is the initial process state. The finite set of rules $\mathcal{R}'$ is the smallest set satisfying the following conditions:

– There is a rule in $\mathcal{R}'$ of the form:

$$\begin{bmatrix} q_0^c \\ \rightarrow \\ q_1^c \end{bmatrix} \quad \begin{bmatrix} q^{init} \\ (0 = x) \rightarrow \emptyset \\ q_r^{init} \end{bmatrix}$$

Intuitively with this rule the timed network with broadcast selects non deterministically a node which will be the root.

– For each rule $\left(q, g \xrightarrow{\tau} R, q'\right)$ in $\mathcal{R}$ there is a rule in $\mathcal{R}'$ of the form (local action of a leaf):

$$\begin{bmatrix} q_1^c \\ \rightarrow \\ q_1^c \end{bmatrix} \quad \begin{bmatrix} q \\ g \rightarrow R \\ q' \end{bmatrix}$$

and a rule of the form (local action of the root):

$$\begin{bmatrix} q_1^c \\ \to \\ q_1^c \end{bmatrix} \quad \begin{bmatrix} q_r \\ g \to R \\ q'_r \end{bmatrix}$$

– For each rule $\left(q, g \xrightarrow{!!a} R, q'\right)$ in $\mathcal{R}$, there is a rule in $\mathcal{R}'$ of the form (broadcast from the root):

$$\begin{bmatrix} q_1^c \\ \to \\ q_1^c \end{bmatrix} \quad \begin{bmatrix} q_r \\ g \to R \\ q'_r \end{bmatrix} \quad \begin{bmatrix} p_1 \\ g_1 \to_t R_1 \\ p'_1 \end{bmatrix} \quad \cdots \quad \begin{bmatrix} p_\ell \\ g_\ell \to_t R_\ell \\ p'_\ell \end{bmatrix}$$

such that the set $\{\left(p_i, g_i \xrightarrow{??a} R_i, p'_i\right) \mid 1 \leq i \leq \ell\}$ is exactly the subset of $\mathcal{R}$ containing all the rules of the form $\left(q_0, g' \xrightarrow{??a} R', q'_0\right)$.

– For each rule $\left(q, g \xrightarrow{!!a} R, q'\right)$ in $\mathcal{R}$, there is a rule in $\mathcal{R}'$ of the form (broadcast from the root):

$$\begin{bmatrix} q_1^c \\ \to \\ q_1^c \end{bmatrix} \quad \begin{bmatrix} q \\ g \to R \\ q' \end{bmatrix} \quad \begin{bmatrix} p_{1,r} \\ g_1 \to_t R_1 \\ p'_{1,r} \end{bmatrix} \quad \cdots \quad \begin{bmatrix} p_{\ell,r} \\ g_\ell \to_t R_\ell \\ p'_{\ell,r} \end{bmatrix}$$

such that the set $\{\left(p_i, g_i \xrightarrow{??a} R_i, p'_i\right) \mid 1 \leq i \leq \ell\}$ is exactly the subset of $\mathcal{R}$ containing all the rules of the form $\left(q_0, g' \xrightarrow{??a} R', q'_0\right)$. Note that in this case only the root will receive the broadcast and we still use the transfert in order to be sure that if it can receive it, it will change its state. However only one of the transfert will be done since there is only note labelled by a state in Q_r.

– There is a rule in $\mathcal{R}'$ of the form (a leaf reaches state q_F):

$$\begin{bmatrix} q_1^c \\ \to \\ q_2^c \end{bmatrix} \quad \begin{bmatrix} q_F \\ (0 \leq x) \to \emptyset \\ q_F \end{bmatrix}$$

and a rule of the form (the root reaches q_F):

$$\begin{bmatrix} q_1^c \\ \to \\ q_2^c \end{bmatrix} \quad \begin{bmatrix} q_{F,r} \\ (0 \leq x) \to \emptyset \\ q_{F,r} \end{bmatrix}$$

Intuitively this rule allows the controller to detect that one of the process has reached the state q_F.

With this construction and using the semantics of both TAHN and TN with transfert nodes, one can easily prove that there is a TAHN $\mathcal{T} = (P, G)$ such that $G \in \texttt{STAR}(1)$ and q_F is reachable in $\mathcal{T}$ iff the controller state q_2^c is reachable in $\mathcal{N}$. Hence since $\mathcal{N}$ works over one clock, using Theorem 6 we obtain the desired result.

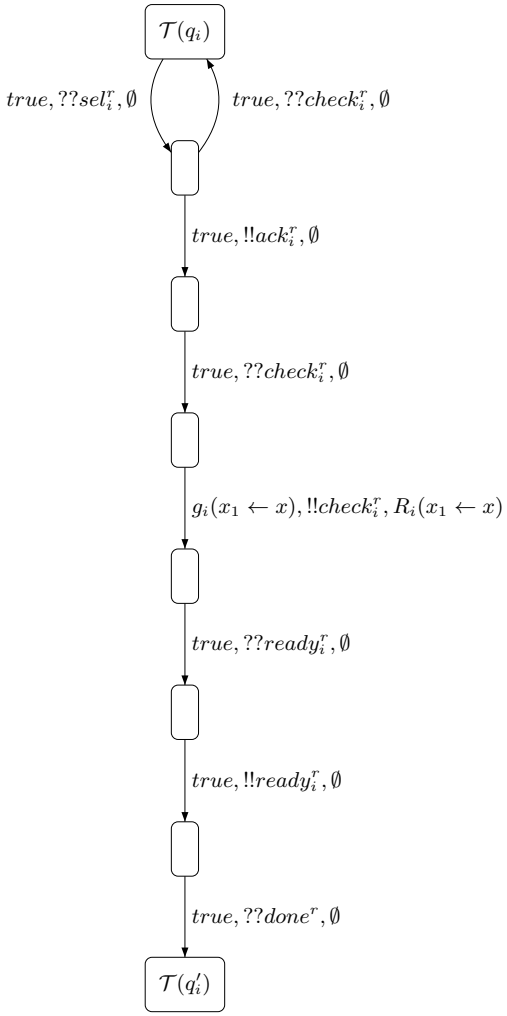

Fig. 4. Ray selection: internal node

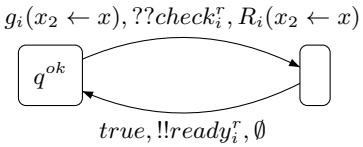

Fig. 5. Ray selection: leaf node

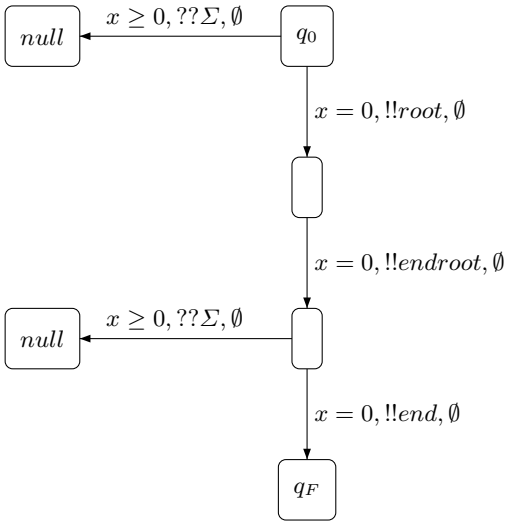

Fig. 6. The behavior of the node chosen as the root

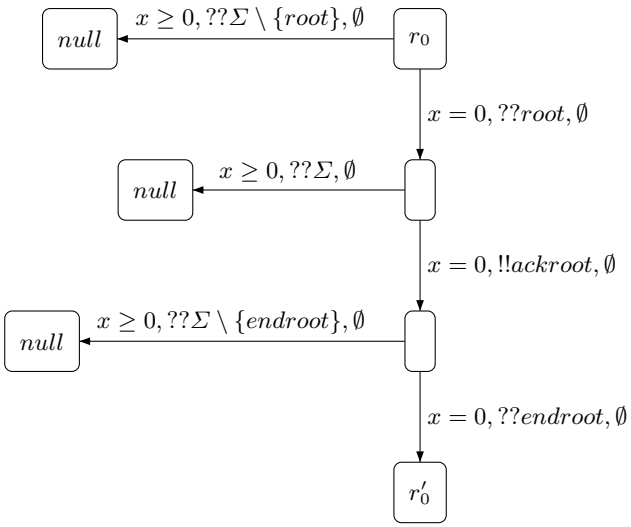

Fig. 7. The behavior of the node chosen as a ray (communication with the root)

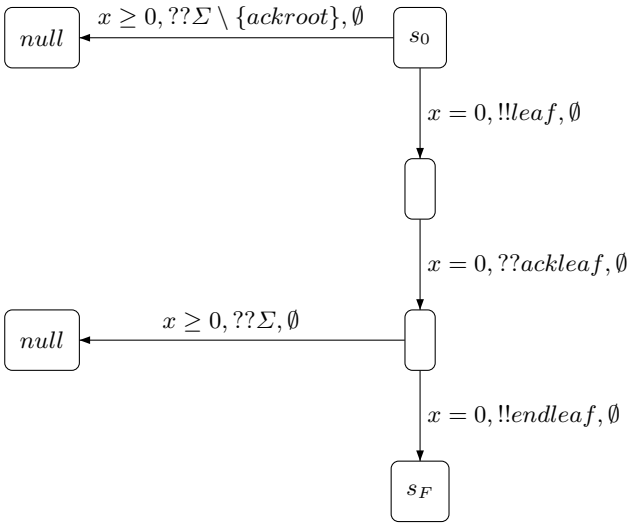

Fig. 8. The behavior of the node chosen as a leaf

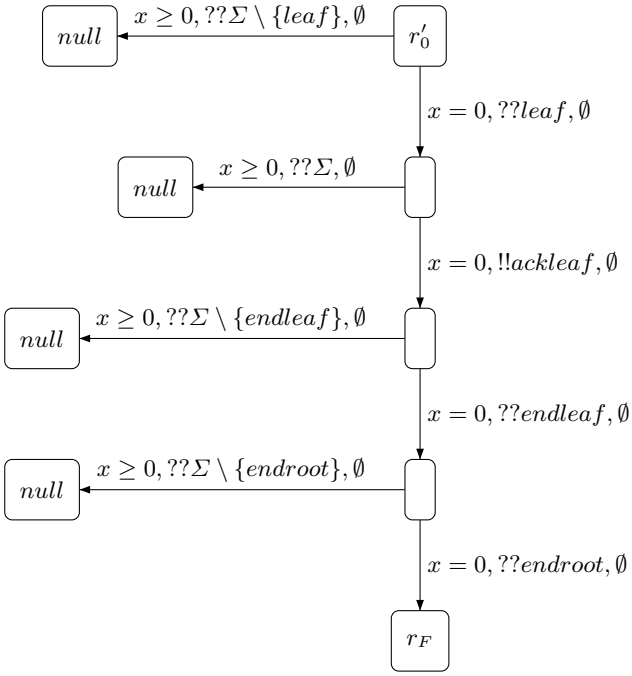

Fig. 9. The behavior of the node chosen as a ray (communication with the leaf)