Under consideration for publication in Formal Aspects of Computing

Probabilistic Model Checking of Deadline Properties in the IEEE 1394 FireWire Root Contention Protocol [1]

Marta Kwiatkowska[a], Gethin Norman[a], and Jeremy Sproston[b]

[a]School of Computer Science, University of Birmingham, Birmingham B15 2TT, United Kingdom
[b]Dipartimento di Informatica, Università di Torino, 10149 Torino, Italy

Abstract. The interplay of real-time and probability is crucial to the correctness of the IEEE 1394 FireWire root contention protocol. We present a formal verification of the protocol using probabilistic model checking. Rather than analyze the functional aspects of the protocol, by asking such questions as "will a leader be elected?", we focus on the protocol's performance, by asking the question "how certain are we that a leader will be elected sufficiently quickly?" Probabilistic timed automata are used to formally model and verify the protocol against properties which require that a leader is elected before a deadline with a certain probability. We use techniques such as abstraction, reachability analysis, and integer-time semantics to aid the model-checking process, and the efficacy of these techniques is compared.

Keywords: Probabilistic model checking, timed automata, IEEE standard, FireWire.

1. Introduction

The increasing dependence of businesses on distributed architectures and computer networking places heavy demands on the speed and reliability of data exchange, leading to the emergence of sophisticated protocols which involve both real-time and randomization, for example those used in the FireWire IEEE 1394 standard. This paper considers an application of model-checking techniques to the FireWire IEEE 1394 root contention protocol, in which the interplay of timed and probabilistic aspects is used to break the symmetry which may arise during the leader election process. Here, we are interested in establishing properties concerning the election of a leader *within a certain deadline, and with a certain probability or greater*.

Automatic verification techniques, including model checking, have been adapted to *probabilistic timed* systems in [ACD91, dA98, BKH99, KNSS02]. In order to model the FireWire IEEE 1394 root contention

[1] Supported in part by the EPSRC grants GR/M04617 and GR/N22960, and by the EU within the DepAuDE project IST-2001-25434.
Correspondence and offprint requests to: Marta Kwiatkowska, School of Computer Science, University of Birmingham, Birmingham B15 2TT, United Kingdom. E-mail: M.Z.Kwiatkowska@cs.bham.ac.uk

protocol, we use the specification formalism of *probabilistic timed automata* [KNSS02], a variant of timed automata [AD94] extended with discrete probability distributions. The formalism is both sufficiently expressive to describe formally both the timed and probabilistic aspects of the protocol, and is amenable to model checking against probabilistic timed temporal logic properties. In contrast to the models based on stochastic processes of [ACD91, BKH99], probabilistic timed automata exhibit nondeterminism, which can be used to represent the unknown timing delays of the protocol. Furthermore, unlike the model of [dA98], probabilistic timed automata can be verified against deadline properties.

A probabilistic timed automaton comprises a finitary, probabilistic transition system equipped with a finite set of real-valued variables which increase at the same rate as real-time. The presence of such *clock* variables means that, in a continuous-time semantic interpretation, the underlying state space of a probabilistic timed automaton is infinite. Hence, approaches for obtaining finite-state representations of such models which are faithful with respect to the validity of a class of properties are necessary. We make use of three such approaches in the context of verifying the IEEE 1394 root contention protocol against deadline properties: the first uses an algorithm to conduct a *forward search* through the state space of the probabilistic timed automaton, the second employs "region equivalence" [AD94] to *partition* the state space, and the third uses an *integer-time* semantic interpretation of the model. In each of the three cases, we use the probabilistic model-checking tool PRISM [KNP02, Pri] in the final step of establishing or refuting the relevant probabilistic deadline property.

Related work. Formal verification of the IEEE 1394 root contention protocol has been performed in a number of previous works. A probabilistic guarded command language is used in [FS01] to model the protocol, following which *manual* proof methods are used to obtain a relationship between the number of attempts to resolve root contention and the probability of successful leader election. The approach of [SV99] is to consider a probabilistic timed model of the protocol, which is then verified manually through a process of stepwise abstraction. Such a refinement process is repeated in [SS01], in which the model takes the form of a non-probabilistic timed automaton, with the real-time model checker Uppaal [LPY97] being used to establish automatically abstraction/refinement relations. Parametric real-time model checking is performed in [HRSV01, CS01], and also by [BST00], in which probabilistic behaviour is modelled by fairness requirements. Our aims are different to these case studies: they do not consider deadline properties, with their emphasis concerning traditional temporal logic properties rather than probabilistic properties. In this sense, our approach has more in common with the discrete-event simulation performed on a stochastic process algebra model of the protocol in [D'A99].

All three verification techniques featured in this paper have precedents from the field of non-probabilistic timed automata. The first decidability results for timed automata were obtained using region equivalence [AD94], which we also use in this paper. Forward reachability algorithms for the verification of timed automata are implemented in the tools Uppaal [LPY97] and Kronos [DOTY96, DT98]. Furthermore, building on the theoretical basis of [HMP92, AMP98], several case studies have been successfully verified using integer semantics [BMT99, Bey01].

Plan of the paper. In Section 2, we introduce probabilistic timed automata, and show how this formalism can be used to model the root contention protocol in Section 3. Section 4 presents a number of methods to verify the protocol, and Section 5 summarizes the results and offers some directions for future research.

2. Probabilistic Timed Automata

In this section, we introduce probabilistic timed automata [KNSS02] as our model for the IEEE 1394 root contention protocol. Probabilistic timed automata are an extension of timed automata [AD94] (henceforth referred to as *classical timed automata*) with the ability to express relative likelihoods of state transitions. A classical timed automaton consists of a finitary directed control graph, the nodes of which are called *locations*, equipped with a finite set of real-valued variables called clocks, which are interpreted as increasing at the same rate as real-time. The edges of the control graph are enabled or forced to be taken depending on whether certain constraints on the clocks are satisfied. Furthermore, a set of clocks may be reset when an edge is executed. Probabilistic timed automata are timed automata for which discrete probability distributions range over the edges of the control graph.

2.1. Syntax of probabilistic timed automata

Time, clocks and zones. Let $\mathbb{T} \in \{\mathbb{R}, \mathbb{N}\}$ be the *time domain* of either the non-negative reals $\mathbb{R}$ or the naturals $\mathbb{N}$. Let $\mathcal{X}$ be a finite set of variables called *clocks* which take values from the time domain $\mathbb{T}$. A point $v \in \mathbb{T}^{|\mathcal{X}|}$ is referred to as a *clock valuation*. We use $\mathbf{0} \in \mathbb{T}^{|\mathcal{X}|}$ to denote the clock valuation which assigns 0 to all clocks in $\mathcal{X}$. Let $v \in \mathbb{T}^{|\mathcal{X}|}$ be a clock valuation, and let $t \in \mathbb{T}$ be a time duration; then the clock valuation $v \oplus t$ denotes the *time increment* for v and t (we present two alternatives for $\oplus$ in Section 2.2, one of which is standard addition $+$). We use $v[X := 0]$ to denote the clock valuation obtained from the clock valuation $v \in \mathbb{T}^{|\mathcal{X}|}$ by resetting all of the clocks in $X \subseteq \mathcal{X}$ to 0, and leaving the values of all other clocks unchanged.

Let $\mathit{Zones}(\mathcal{X})$ be the set of *zones* over $\mathcal{X}$, which are conjunctions of atomic constraints of the form $x \sim c$ and $x - y \sim c$, for clocks $x, y \in \mathcal{X}$, comparison operator $\sim \in \{<, \leq, \geq, >\}$, and naturals $c \in \mathbb{N}$. A zone ζ is *diagonal-free* if it does not feature a conjunct of the form $x - y \sim c$, and is *closed* if it does not feature a conjunct of the form $x - y \sim c$ or $x \sim c$ for $\sim \in \{<, >\}$. The clock valuation v *satisfies* the zone ζ, written $v \triangleleft \zeta$, if and only if ζ resolves to true after replacing each clock $x \in \mathcal{X}$ with the corresponding clock value v_x from v.

Probability distributions. A discrete probability *distribution* over a countable set Q is a function $\mu : Q \to [0, 1]$ such that $\sum_{q \in Q} \mu(q) = 1$. For a possibly uncountable set Q', let $\mathsf{Dist}(Q')$ be the set of distributions over countable subsets of Q'. For any element $q \in Q$, let $\mu_q \in \mathsf{Dist}(Q)$ be the distribution which assigns probability 1 to q.

Definition 2.1. (Probabilistic timed automata.) A *probabilistic timed automaton* is a tuple $\mathsf{PTA} = (L, \mathcal{X}, \Sigma, \mathit{inv}, \mathit{prob})$ where:

- L is a finite set of *locations*;
- Σ is a finite set of *events*;
- the function $\mathit{inv} : L \to \mathit{Zones}(\mathcal{X})$ is the *invariant condition*;
- the finite set $\mathit{prob} \subseteq L \times \mathit{Zones}(\mathcal{X}) \times \Sigma \times \mathsf{Dist}(2^{\mathcal{X}} \times L)$ is the *probabilistic edge relation*. A probabilistic edge takes the form of a tuple $(l, g, \sigma, p) \in \mathit{prob}$, where l is the source location of the probabilistic edge, g is its *enabling condition*, σ is its event, and $p \in \mathsf{Dist}(2^{\mathcal{X}} \times L)$ is its *edge distribution*.

A probabilistic timed automaton is *diagonal-free* (*closed*) if all the zones used in its description are diagonal-free (closed).

A state of a probabilistic timed automaton PTA is a pair (l, v) where $l \in L$ and $v \in \mathbb{T}^{|\mathcal{X}|}$ such that $v \triangleleft \mathit{inv}(l)$. If the current state is (l, v), there is a nondeterministic choice of either letting *time pass* while satisfying continuously the invariant condition $\mathit{inv}(l)$, or making a *discrete* transition according to any probabilistic edge in prob with source location l and whose enabling condition g is satisfied by the current clock valuation v. If the probabilistic edge (l, g, σ, p) is chosen, then the probability of moving to the location l' and resetting all of the clocks in the set X to 0 is given by $p(X, l')$. The semantics of probabilistic timed automata are presented formally in the next section.

Note that we sometimes identify a designated *initial location* $\bar{l} \in L$, with the intuition that the behaviour of the model commences in $\bar{l}$ with all clocks set to 0. The *initial state* $(\bar{l}, \mathbf{0})$, in which the value of all of the clocks is 0, can then be used when considering reachability properties, for example "is the probability of reaching a certain set of states from the initial state greater than λ?"

Higher-level modelling. To aid higher-level modelling, we can designate certain locations as being *urgent*; once an urgent location is entered, it must be left immediately, without time passing. The notion of urgency for locations is closely related to the concept of urgent transitions [HHWT95, DY95] (an urgent location is a location for which all outgoing discrete transitions are urgent). Urgent locations can be represented syntactically using the framework given in Definition 2.1 using an additional clock, combined with altered clock resets and invariant conditions.

It is often convenient to define systems as the *parallel composition* of a number of interacting sub-components. For example, in the case of the IEEE 1394 root contention protocol, it suffices to construct models for each of the two contending nodes, and for the two wires along which they communicate, given that the manner in which they interact is defined explicitly. Using ideas from the theory of (untimed) probabilistic systems [SL95] and classical timed automata [AD94], the parallel composition of two probabilistic timed

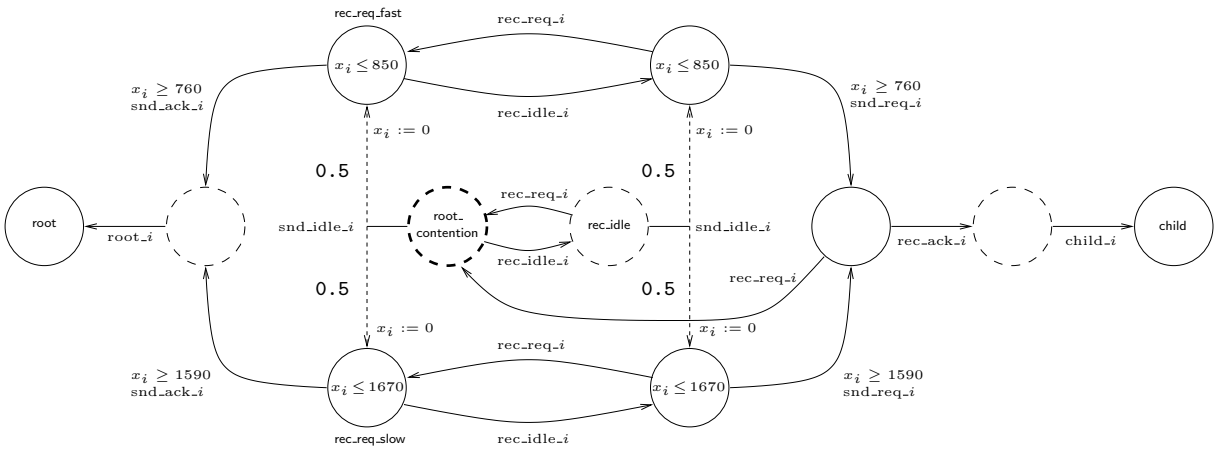

Fig. 1. The probabilistic timed automaton $\texttt{Node}_i^{\texttt{p}}$.

automata, which interact by synchronizing on common events, is defined in the following way. Let $\mathsf{PTA}_i = (L_i, \mathcal{X}_i, \Sigma_i, inv_i, prob_i)$ for $i \in \{1, 2\}$.

Definition 2.2. The *parallel composition* of two probabilistic timed automata PTA_1 and PTA_2, where $\mathcal{X}_1 \cap \mathcal{X}_2 = \emptyset$, is the probabilistic timed automaton $\mathsf{PTA}_1 \| \mathsf{PTA}_2 = (L_1 \times L_2, \mathcal{X}_1 \cup \mathcal{X}_2, \Sigma_1 \cup \Sigma_2, inv, prob)$ where $inv(l, l') = inv_1(l) \wedge inv_2(l')$ for all $(l, l') \in L_1 \times L_2$ and $((l_1, l_2), g, \sigma, p) \in prob$ if and only if one of the following conditions holds:

- $\sigma \in \Sigma_1 \setminus \Sigma_2$ and there exists $(l_1, g, \sigma, p_1) \in prob_1$ such that $p = p_1 \otimes \mu_{(\emptyset, l_2)}$;
- $\sigma \in \Sigma_2 \setminus \Sigma_1$ and there exists $(l_2, g, \sigma, p_2) \in prob_2$ such that $p = \mu_{(\emptyset, l_1)} \otimes p_2$;
- $\sigma \in \Sigma_1 \cap \Sigma_2$ and there exists $(l_1, g_1, \sigma, p_1) \in prob_1$ and $(l_2, g_2, \sigma, p_2) \in prob_2$ such that $g = g_1 \wedge g_2$ and $p = p_1 \otimes p_2$

where for any $l_1 \in L_1$, $l_2 \in L_2$, $X_1 \subseteq \mathcal{X}_1$ and $X_2 \subseteq \mathcal{X}_2$, we let $p_1 \otimes p_2(X_1 \cup X_2, (l_1, l_2)) = p_1(X_1, l_1) \cdot p_2(X_2, l_2)$.

Given the initial locations $\bar{l}_i \in L_i$ of PTA_i, for $i \in \{1, 2\}$, we let $(\bar{l}_1, \bar{l}_2)$ be the initial location of $\mathsf{PTA}_1 \| \mathsf{PTA}_2$.

Example. Figure 1 shows the probabilistic timed automaton $\texttt{Node}_i^{\texttt{p}}$ for a contending node of the network involved in the IEEE 1394 FireWire root contention protocol. The model $\texttt{Node}_i^{\texttt{p}}$ is a minor, probabilistic extension of the classical timed automaton node model from [SS01]. The usual conventions for the graphical representation of classical timed automata are used (note that we omit invariant conditions or guards of the form `true`, and some location names, for simplicity). Figure 1 also features bifurcating edges leaving the locations root_contention (the initial location, as denoted by the bold node) and rec_idle, which correspond to probabilistic transitions. For example, the left-hand edges leaving root_contention correspond to a probabilistic choice of taking a transition to either of the target locations, rec_req_fast and rec_req_slow, each with probability 0.5 (while resetting the clock x_i). For simplicity, we omit the probability labels from edges corresponding to probability 1. Urgent locations are indicated by the dashed locations.

The behaviour of the model commences in location root_contention, which models the situation in which the node has detected root contention. Because the location root_contention is urgent, $\texttt{Node}_i^{\texttt{p}}$ is forced to select an outgoing probabilistic edge instantly. Consider the bifurcating probabilistic edge labelled by snd_idle_i, which corresponds to the node flipping a coin in order to determine whether it should wait for a short or long time. The snd_idle_i event is sent by $\texttt{Node}_i^{\texttt{p}}$ to its communication medium, referring to a transmission of an idle signal across the node's wire to the other node. In both of the two locations rec_req_fast and rec_req_slow which may be reached after taking the probabilistic edge, the passage of time may mean that the value of the clock x_i can reach a value enabling the left-pointing edges, in turn meaning that an acknowledgement is sent (event snd_ack_i), and the node then declares itself to be leader (the event root_i which labels the subsequent left-pointing edge to the location root). In contrast, if the node receives an idle signal from the other contending node (event rec_idle_i) before sending an acknowledgement, it is forced to move to the right via an edge labelled with the event rec_idle_i. In this case, after a certain amount of time elapses,

$\texttt{Node}_i^{\texttt{P}}$ can send a request to the other node to be its parent by sending the event snd_req_i to its wire. If the node then subsequently detects a parent request from the other node (event rec_req_i), it returns to the location root_contention, and restarts the root contention process. If, on the other hand, the node detects an acknowledgement from the other node (event rec_ack_i), it proceeds to declare itself as the child by sending a child_i event.

2.2. Semantics of probabilistic timed automata

2.2.1. Probabilistic systems

The semantics of probabilistic timed automata is defined in terms of transition systems exhibiting both nondeterministic and probabilistic choice. We call such models *probabilistic systems*, noting that they are essentially equivalent to Markov decision processes [Der70], the simple probabilistic automata of [SL95], and the probabilistic-nondeterministic systems of [BdA95].

Definition 2.3. (Probabilistic systems.) A *probabilistic system* $\mathsf{PS} = (S, Act, Steps)$ consists of a set S of *states*, a set Act of *actions*, and a *probabilistic transition relation* $Steps \subseteq S \times Act \times \mathsf{Dist}(S)$.

A *probabilistic transition* $s \xrightarrow{a,\mu} s'$ is made from a state $s \in S$ by first nondeterministically selecting an action-distribution pair (a, μ) such that $(s, a, \mu) \in Steps$, and second by making a probabilistic choice of target state s' according to μ, such that $\mu(s') > 0$. In the sequel, probabilistic systems may be infinite-state.

We now give the semantics of probabilistic timed automata defined in terms of probabilistic systems. As with classical timed automata, transitions consist of two types: *time transitions*, which correspond to the passage of time while the current location remains constant, and *discrete transitions*, which correspond to a probabilistic edge being taken. The definition is parameterized both by a time domain $\mathbb{T}$ and a time increment $\oplus$.

Definition 2.4. (Semantics of probabilistic timed automata.) The *semantics of a probabilistic timed automaton* $\mathsf{PTA} = (L, \mathcal{X}, \Sigma, inv, prob)$ *with respect to the time domain* $\mathbb{T}$ *and the time increment* $\oplus$ is the probabilistic system $[\![\mathsf{PTA}]\!]_{\mathbb{T}}^{\oplus} = (S, Act, Steps)$ defined by the following.

States. Let $S \subseteq L \times \mathbb{T}^{|\mathcal{X}|}$ such that $(l, v) \in S$ if and only if $v \triangleleft inv(l)$.

Actions. Let $Act = \mathbb{T} \cup \Sigma$.

Probabilistic transitions. Let *Steps* be the least set of probabilistic transitions defined as follows. For each state $(l, v) \in S$:

- *Time transitions.* For each duration $t \in \mathbb{T}$, let $((l, v), t, \mu) \in Steps$ if and only if (1) $v \oplus t' \triangleleft inv(l)$ for each $0 \leq t' \leq t$, and (2) $\mu(l, v \oplus t) = 1$.
- *Discrete transitions.* For each probabilistic edge $(l, g, \sigma, p) \in prob$, let $((l, v), \sigma, \mu) \in Steps$ if and only if $v \triangleleft g$, for each state $(l', v') \in S$:

$$\mu(l', v') = \sum_{X \subseteq \mathcal{X} \,\&\, v' = v[X:=0]} p(X, l') \ ,$$

  and for each $(X, l') \in 2^{\mathcal{X}} \times L$ such that $p(X, l') > 0$, we have $v[X := 0] \triangleleft inv(l')$.

The summation in the definition of discrete transitions is required for the cases in which multiple clock resets result in the same target state (l', v'). Furthermore, the final clause is required to preclude the pathological situation in which an invariant of a location is not satisfied directly after a probabilistic transition.

The semantics falls into two classes, depending on whether the underlying model of time is the positive reals or the naturals. In the sequel, we always let $\oplus$ equal $+$ if $\mathbb{T} = \mathbb{R}$. We refer to $[\![\mathsf{PTA}]\!]_{\mathbb{R}}^{+}$ as the *continuous semantics* of the probabilistic timed automaton PTA. In contrast, if $\mathbb{T} = \mathbb{N}$, we always let $\oplus$ equal $\oplus_{\mathbb{N}}$, which is defined as follows. Let PTA be a probabilistic automaton; for any $x \in \mathcal{X}$, let $\mathbf{k}_x$ denote the greatest constant the clock x is compared to in the zones of PTA. Then, for any clock valuation $v \in \mathbb{N}^{|\mathcal{X}|}$ and time duration $t \in \mathbb{N}$, let $v \oplus_{\mathbb{N}} t$ be the clock valuation of $\mathcal{X}$ which assigns the value $\min\{v_x + t, \mathbf{k}_x + 1\}$ to all clocks $x \in \mathcal{X}$ (although the operator $\oplus_{\mathbb{N}}$ is dependent on PTA, we elide a sub- or superscript indicating this for clarity). Then we refer to $[\![\mathsf{PTA}]\!]_{\mathbb{N}}^{\oplus_{\mathbb{N}}}$ as the *integer semantics* of PTA. The definition of integer semantics

for probabilistic timed automata is a generalization of the analogous definition for the classical model in [Bey01], which in turn follows the methodology of [HMP92, AMP98, BMT99]. As we always use the same definition of time increment for a particular choice of time domain, we omit the $+$ and $\oplus_{\mathbb{N}}$ superscripts from the notation for the continuous and discrete semantics for simplicity.

Note that the fact that the integer semantics of a probabilistic timed automaton is finite, and the continuous semantics of probabilistic timed automaton is generally infinite, can be derived from the definitions. In both cases, all states have at least one available transition (namely, the time transition $(s, 0, \mu_s)$).

It is not difficult to check that the semantics of the parallel composition of two probabilistic timed automata corresponds to the semantics of the parallel composition of their individual semantic probabilistic systems. Formally, we overload the parallel composition operator $\|$ such that $\mathsf{PS}_1 \| \mathsf{PS}_2$ denotes the probabilistic system obtained from the parallel composition of the probabilistic systems PS_1 and PS_2 in the standard manner [SL95]. Two probabilistic systems $\mathsf{PS}_1 = (S_1, Act, Steps_1)$ and $\mathsf{PS}_2 = (S_2, Act, Steps_2)$ are *isomorphic* if there exists a bijection $f : S_1 \to S_2$ such that $(s_1, a, \mu) \in Steps_1$ if and only if $(f(s_1), a, f(\mu)) \in Steps_2$, where $f(\mu) \in \mathsf{Dist}(S_2)$ is the distribution defined by $f(\mu)(s_2) = \mu(f^{-1}(s_2))$ for each $s_2 \in S_2$. For the probabilistic timed automata PTA_1 and PTA_2 with disjoint clock sets, $[\![\mathsf{PTA}_1 \| \mathsf{PTA}_2]\!]_{\mathbb{T}}$ and $[\![\mathsf{PTA}_1]\!]_{\mathbb{T}} \| [\![\mathsf{PTA}_2]\!]_{\mathbb{T}}$ are isomorphic, both for the continuous and integer semantics.

2.2.2. Behaviour of probabilistic timed automata

The behaviour of a probabilistic timed automaton is described in terms of the behaviour of its (continuous or integer) semantic probabilistic system. We consider two ways in which a probabilistic system's computation may be represented, followed by definitions of probability measures of interest.

Paths. A *path* represents a particular resolution of both nondeterminism *and* probability of a probabilistic system. Formally, a path of a probabilistic system $\mathsf{PS} = (S, Act, Steps)$ is a non-empty finite or infinite sequence of transitions $\omega = s_0 \xrightarrow{a_0,\mu_0} s_1 \xrightarrow{a_1,\mu_1} \cdots$. We use $Path^{\mathsf{PS}}_{fin}$ to denote the set of finite paths of PS, and $Path^{\mathsf{PS}}_{ful}$ the set of infinite paths of PS.

For a path ω and $i \in \mathbb{N}$, we denote by $\omega(i)$ the $(i+1)$th state of ω; if ω is finite, we denote by $last(\omega)$ the last state of ω. We abuse notation by letting a state $s \in S$ to denote a path consisting of no transitions, and by using $\omega \xrightarrow{a,\mu} s$ to refer to a path comprising the sequence of transitions of ω followed by the transition $last(\omega) \xrightarrow{a,\mu} s$.

Adversaries. An *adversary* represents a particular resolution of nondeterminism *only*. Formally, an adversary of a probabilistic system PS is a function A mapping every finite path $\omega \in Path^{\mathsf{PS}}_{fin}$ to a pair $(a, \mu) \in Act \times \mathsf{Dist}(S)$ such that $(last(\omega), a, \mu) \in Steps$ [Var85]. Let Adv_{PS} be the set of adversaries of PS.

Probability measures. For any adversary $A \in Adv_{\mathsf{PS}}$, let $Path^A_{fin}$ and $Path^A_{ful}$ denote the set of finite and infinite paths associated with A (more precisely, the paths resulting from the choices of action-distribution pairs of A). Then we define the probability measure $Prob^A$ over $Path^A_{ful}$ in the following, standard way [KSK76]. First we define the function $\mathsf{A} : Path^A_{fin} \times Path^A_{fin} \to [0, 1]$, such that:

$$\mathsf{A}(\omega, \omega') = \begin{cases} \mu(s) & \text{if } A(\omega) = (last(\omega), a, \mu) \text{ for some } a \in Act, \text{ and } \omega' = \omega \xrightarrow{a,\mu} s \\ 0 & \text{otherwise.} \end{cases}$$

Intuitively, $\mathsf{A}(\omega, \omega')$ refers to the probability of obtaining the finite path ω' when extending the finite path ω with one transition under the control of the adversary A. Next, the function $Prob^A_{fin} : Path^A_{fin} \to [0, 1]$ is defined inductively along the length of finite paths of A as follows:

- let $Prob^A_{fin}(s) = 1$ for all states $s \in S$ (paths of length 0);
- if $(\omega \xrightarrow{a,\mu} s) \in Path^A_{fin}$, then $Prob^A_{fin}(\omega \xrightarrow{a,\mu} s) = Prob^A_{fin}(\omega) \cdot \mathsf{A}(\omega, \omega \xrightarrow{a,\mu} s)$.

For a finite path $\omega \in Path^A_{fin}$, let $\mathsf{Cone}(\omega)$ be the *cone* generated by ω, defined as the set $\mathsf{Cone}(\omega) = \{\omega' \in Path^A_{ful} \mid \omega \text{ is a prefix of } \omega'\}$ of infinite paths. We then define $Prob^A$ by $Prob^A(\mathsf{Cone}(\omega)) = Prob^A_{fin}(\omega)$, which

can then be uniquely extended to a probability measure on the sigma-algebra generated by the cones of A.

Reachability probabilities. The *maximal (minimal) reachability probability* is the maximum (minimum) probability with which a given set of states of a probabilistic system can be reached from a particular state. Formally, for the probabilistic system $\mathsf{PS} = (S, Act, Steps)$, state $s \in S$, set $F \subseteq S$ of target states, and adversary $A \in Adv_{\mathsf{PS}}$, the probability of reaching F from s under adversary A is given by:

$$ProbReach^A(s, F) \stackrel{def}{=} Prob^A\{\omega \in Path^A_{ful} \mid \omega(0) = s \;\&\; \exists i \in \mathbb{N} \,.\, \omega(i) \in F\} \,.$$

Then the maximal and minimal reachability probabilities $MaxProbReach_{\mathsf{PS}}(s, F)$ and $MinProbReach_{\mathsf{PS}}(s, F)$ are defined as:

$$MaxProbReach_{\mathsf{PS}}(s, F) \stackrel{def}{=} \sup_{A \in Adv_{\mathsf{PS}}} ProbReach^A(s, F) \;\;,\;\; MinProbReach_{\mathsf{PS}}(s, F) \stackrel{def}{=} \inf_{A \in Adv_{\mathsf{PS}}} ProbReach^A(s, F) \,.$$

The following theorem is key to establishing the correctness of the integer semantics with regard to the probability of reaching a certain set of target locations of a *closed, diagonal-free* probabilistic timed automaton. The theorem states that both the maximal and minimal probabilities of reaching a target location are equal in the continuous and integer semantics for this class of model, and is a probabilistic extension of a similar result established in [Bey01]. Let $L' \subseteq L$ be a set of target locations of a probabilistic timed automaton PTA, and let the set of all states corresponding to locations in L' be denoted by $F^{L'}_{\mathbb{T}} = \{(l, v) \mid l \in L',\; v \in \mathbb{T}^{|\mathcal{X}|} \;\&\; v \triangleleft inv(l)\}$.

Theorem 2.5. For every closed, diagonal-free probabilistic timed automata $\mathsf{PTA} = (L, \mathcal{X}, \Sigma, inv, prob)$, initial location $\bar{l} \in L$, and target set $L' \subseteq L$ of locations:

$$MaxProbReach_{[\![\mathsf{PTA}]\!]_{\mathbb{R}}}((\bar{l}, \mathbf{0}), F^{L'}_{\mathbb{R}}) = MaxProbReach_{[\![\mathsf{PTA}]\!]_{\mathbb{N}}}((\bar{l}, \mathbf{0}), F^{L'}_{\mathbb{N}})$$

$$MinProbReach_{[\![\mathsf{PTA}]\!]_{\mathbb{R}}}((\bar{l}, \mathbf{0}), F^{L'}_{\mathbb{R}}) = MinProbReach_{[\![\mathsf{PTA}]\!]_{\mathbb{N}}}((\bar{l}, \mathbf{0}), F^{L'}_{\mathbb{N}}).$$

Proof. See Appendix A. □

2.3. Time-progress in probabilistic timed automata

As with the case of classical timed automata [AD94], some paths of a probabilistic timed automaton may correspond to pathological situations in which time does not *progress* beyond some bound; therefore, such computations will not be exhibited by any real-life system. The solution for classical timed automata is to design model-checking methods which disregard such *Zeno* paths. Following precedents from the probabilistic real-time systems literature [dA97, Seg95], the lifting of the concept of time progress from classical to probabilistic timed automata involves characterizing *adversaries*, rather than paths, which exhibit "non-Zenoness", and consequently disregarding all other adversaries during model checking [KNSS02].

Definition 2.6. (Non-Zenoness.) Let PTA be a probabilistic timed automaton with time domain $\mathbb{T}$, and let $\omega = (l_0, v_0) \xrightarrow{a_0, \mu_0} (l_1, v_1) \xrightarrow{a_1, \mu_1} \cdots$ be a path of $[\![\mathsf{PTA}]\!]_{\mathbb{T}}$. The *elapsed time along* ω *at step* $i \in \mathbb{N}$ is the sum of the durations of the time transitions of ω taken before the ith transition, and is formally defined as:

$$\mathsf{time}(\omega, i) = \sum \{a_j \mid 0 \leq j < i \text{ and } a_j \in \mathbb{T}\} \,.$$

A path ω of $[\![\mathsf{PTA}]\!]_{\mathbb{T}}$ is *non-Zeno* if and only if, for all $t \in \mathbb{T}$, there exists $i \in \mathbb{N}$ such that $\mathsf{time}(\omega, i) > t$. An adversary A of $[\![\mathsf{PTA}]\!]_{\mathbb{T}}$ is *non-Zeno* if and only if $Prob^A\{\omega \in Path^A_{ful} \mid \omega \text{ is non-Zeno }\} = 1$.

For convenience, we also refer to paths and adversaries which are not non-Zeno as *Zeno*.

3. Modelling the Root Contention Protocol

3.1. The probabilistic timed automaton models

We now explain how the IEEE 1394 root contention protocol can be specified and verified formally within the framework of probabilistic timed automata. The classical timed automaton models of [SS01] are taken as the

basis for the probabilistic models introduced here. As we have seen in Section 2.1, the model $\mathtt{Node}_i^{\mathtt{p}}, i \in \{1,2\}$, of the ith node of the protocol can be extended with probability in a straightforward manner; simply let the two edges corresponding to the outcome of a coin flip each be assigned probability 0.5 by a distribution. All other edges in the node model are taken with probability 1. The communication medium between the nodes, which assumes that signals are driven continuously across wires which comprise of two-place buffers, is then represented by the models $\mathtt{Wire}_i$, for $i \in \{1,2\}$. The classical timed automaton models for the wires in [SS01] are adopted directly (the interested reader is invited to refer to [SS01]), with the interpretation that all transitions of the wire models are made with probability 1. The parallel composition $\mathtt{Impl}^{\mathtt{p}} = \mathtt{Node}_1^{\mathtt{p}} \| \mathtt{Wire}_1 \| \mathtt{Wire}_2 \| \mathtt{Node}_2^{\mathtt{p}}$ of the resulting probabilistic timed automata is then defined using Definition 2.2.

As will be explained in Section 4, during initial verification experiments, it became clear that analysis of a model as complex as the composed probabilistic timed automaton is not feasible given our current implementation of the forward reachability algorithm of [KNSS02]. Therefore, we also study the smaller, more abstract, probabilistic timed automaton $\mathtt{I}_1^{\mathtt{p}}$ of the root contention protocol. The model for $\mathtt{I}_1^{\mathtt{p}}$ is shown in Figure 2, and is a probabilistic extension of the classical timed automaton $\mathtt{I}_1$ used in the stepwise refinement process of [SS01]. The constants used in the enabling and invariant conditions are derived from those given in the IEEE 1394a standard when the communication delay between the nodes is taken to be 360 ns. Each instance of bifurcating edges corresponds to a coin being flipped; for example, in the location start_start, there is a nondeterministic choice between node 1 (re)starting the root contention protocol and flipping its coin, leading with probability 0.5 to each of slow_start and fast_start (we omit all probability labels from the diagram for simplicity), and node 2 restarting the protocol and flipping its coin. The location done represents the election of a leader. Although it is possible for a leader to be elected regardless of the outcome of the probabilistic choices, as there are edges from each of the locations fast_fast, fast_slow, slow_fast and slow_slow to the location done, note that it is also possible to return to start_start, corresponding to protocol restart, from the locations fast_fast and slow_slow, in which the nodes coin flips are identical. Note that the timing constant for the enabling condition of the edge from fast_fast to done is obtained from 760 ns, the minimal waiting time if the "fast" side of the coin is obtained, minus 360 ns, the wire propagation delay; similarly, the enabling conditions of the other edges to done are obtained from 1590 ns, the minimal waiting time if the "slow" side of the coin is obtained, minus 360 ns.

The timing constraints used in $\mathtt{I}_1^{\mathtt{p}}$ correspond to those specified in the updated standard IEEE 1394a. In the figure, the maximum transmission delay equals 360 nanoseconds (ns), which represents the assumption that the contending nodes are separated by a distance close to the maximum required for the correctness of the protocol (from the analysis of [SS01]). However, it is straightforward to change this value and re-run the experiments, if required; we also consider a delay of 30 ns, which corresponds more closely to the maximum separation of nodes specified in the IEEE 1394a standard. This maximum separation results in a maximum transmission delay of 22.725ns, and hence the choice of 30ns is an overapproximation of the delay, which results in upper bounds on the clock x being higher in our model (for example, we have invariant conditions of the form $x \leq 30$ rather than $x \leq 22.725$). Dually, lower bounds are lower in our model (for example, we have enabling conditions of the form $x \geq 1560$ rather than $x \geq 1567.275$) than in a faithful representation of the standard. The choice of a transmission delay of 30ns is made for efficiency reasons, as it allows us to use a time granularity of 10ns when we consider probabilistic model checking based on the region graph and on integer semantics in Section 4. Finally, to measure the time elapsed since the start of system execution, we augment the probabilistic timed automaton $\mathtt{I}_1^{\mathtt{p}}$ with an additional clock y, which can be referred to in the formalization of our deadline property.

The model $\mathtt{I}_1^{\mathtt{p}}$ represents an *abstraction* of the root contention protocol, in the sense that it may exhibit a superset of adversaries of a more refined protocol model, such as $\mathtt{Impl}^{\mathtt{p}}$. In addition, the raising of upper bounds and the decreasing of lower bounds on clocks described in the previous paragraph is another source of abstraction of this form; for more information we refer the reader to [AIKY95, KNS02]. This means that the most *unfavourable* adversary with respect to the satisfaction of a property of the protocol must also be exhibited by $\mathtt{I}_1^{\mathtt{p}}$. Hence, as the property of interest concerns whether a leader will be elected with a certain probability or greater, for all adversaries, the minimal probability of property satisfaction that is computed for $\mathtt{I}_1^{\mathtt{p}}$ will form a *lower bound* on the probability in the protocol model $\mathtt{Impl}^{\mathtt{p}}$. However, as will be seen in Section 4, the probabilities computed for the verification of the abstract model $\mathtt{I}_1^{\mathtt{p}}$ and the full model $\mathtt{Impl}^{\mathtt{p}}$ for a number of deadlines agree, suggesting (but not confirming) that $\mathtt{I}_1^{\mathtt{p}}$ does not abstract from information of $\mathtt{Impl}^{\mathtt{p}}$ relevant to the probabilistic deadline property.

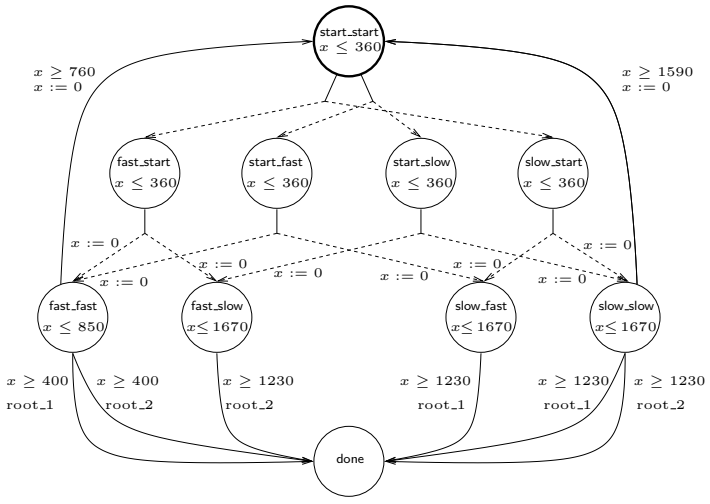

Fig. 2. The probabilistic timed automaton $\mathtt{I}_1^{\mathtt{P}}$.

3.2. Establishing the abstraction

We now describe the method which is used to verify that $\mathtt{I}_1^{\mathtt{P}}$ is indeed a probabilistic, timed abstraction of the probabilistic timed automaton $\mathtt{Impl}^{\mathtt{P}}$ using "trace refinement" or "language containment" verification on non-probabilistic structures. This class of verification method establishes that a non-probabilistic transition system TS_1 is an abstraction of another transition system TS_2 (dually, that TS_2 refines TS_1) if *every linear sequence of actions of* TS_2 *is also exhibited by* TS_1. This technique is used to prove abstraction relations between the classical timed automata modelling the root contention protocol in [SS01]. We now sketch how trace refinement verification on non-probabilistic, classical timed automata versions $\mathsf{TA}_{\mathtt{Impl}^{\mathtt{P}}}$ and $\mathsf{TA}_{\mathtt{I}_1^{\mathtt{P}}}$ of $\mathtt{Impl}^{\mathtt{P}}$ and $\mathtt{I}_1^{\mathtt{P}}$ respectively (which broadly correspond to the classical timed automata models of [SS01]) can be used to infer probabilistic refinement between $\mathtt{Impl}^{\mathtt{P}}$ and $\mathtt{I}_1^{\mathtt{P}}$. This is achieved by using traces of the non-probabilistic models to *reconstruct* paths of the probabilistic models. A more formal description of a similar procedure is given in [KNS02].

The method is crucially reliant on how much information from the probabilistic timed automata is encoded in the *actions* of the classical timed automata $\mathsf{TA}_{\mathtt{Impl}^{\mathtt{P}}}$ and $\mathsf{TA}_{\mathtt{I}_1^{\mathtt{P}}}$ (the location set and invariant conditions of the classical model equal those of the probabilistic one). An extreme case is that in which each transition $s \xrightarrow{a,\mu} s'$ of the (continuous semantics) probabilistic system $[\![\mathtt{Impl}^{\mathtt{P}}]\!]_{\mathbb{R}}$ is encoded by a transition $s \xrightarrow{\langle\!\langle s,a,\mu,s' \rangle\!\rangle} s'$ of the (continuous semantics) transition system $[\![\mathsf{TA}_{\mathtt{Impl}^{\mathtt{P}}}]\!]_{\mathbb{R}}$, thus inducing a natural one-to-one relationship between the sequence of actions (trace) $\langle\!\langle s_0, a_0, \mu_0, s_1 \rangle\!\rangle \langle\!\langle s_1, a_1, \mu_1, s_2 \rangle\!\rangle \cdots$ of the transition system and the path $s_0 \xrightarrow{a_0,\mu_0} s_1 \xrightarrow{a_1,\mu_1} s_2 \cdots$ of the probabilistic system. We then perform a similar construction for $[\![\mathtt{I}_1^{\mathtt{P}}]\!]_{\mathbb{R}}$ and $[\![\mathsf{TA}_{\mathtt{I}_1^{\mathtt{P}}}]\!]_{\mathbb{R}}$ which differs from the one given above in the sense that the labels include information about the states of $[\![\mathtt{Impl}^{\mathtt{P}}]\!]_{\mathbb{R}}$ and $[\![\mathsf{TA}_{\mathtt{Impl}^{\mathtt{P}}}]\!]_{\mathbb{R}}$. This is necessary to establish a meaningful notion of trace refinement in our context; otherwise we would be unable to match any action of $[\![\mathsf{TA}_{\mathtt{Impl}^{\mathtt{P}}}]\!]_{\mathbb{R}}$ with any action of $[\![\mathsf{TA}_{\mathtt{I}_1^{\mathtt{P}}}]\!]_{\mathbb{R}}$, and therefore we would also be unable to match traces of $[\![\mathsf{TA}_{\mathtt{Impl}^{\mathtt{P}}}]\!]_{\mathbb{R}}$ and $[\![\mathsf{TA}_{\mathtt{I}_1^{\mathtt{P}}}]\!]_{\mathbb{R}}$. Technically, this is achieved using a *step refinement* function f [SV99, SS01], which associates each state of $[\![\mathsf{TA}_{\mathtt{I}_1^{\mathtt{P}}}]\!]_{\mathbb{R}}$ with a state of $[\![\mathsf{TA}_{\mathtt{Impl}^{\mathtt{P}}}]\!]_{\mathbb{R}}$, and which is defined manually using knowledge concerning the way in which $\mathtt{I}_1^{\mathtt{P}}$ is an abstraction of $\mathtt{Impl}^{\mathtt{P}}$. For example, the states with the location component $\mathsf{start\_start}$ of $\mathtt{I}_1^{\mathtt{P}}$ could be expected to correspond to the states of $\mathtt{Impl}^{\mathtt{P}}$ with the location components $\mathsf{root\_contention}$ and $\mathsf{rec\_idle}$; therefore, these states can be associated using a step refinement function.

However, as we describe systems at the level of probabilistic and classical timed automata, we cannot implement such action-renaming strategies at the level of their (infinite-state) semantic transition systems. Therefore, we choose an encoding which involves the renaming of the events labelling the *probabilistic edges* of the probabilistic timed automata models. For example, we replace the probabilistic, bi-

furcating edge $(\mathsf{root\_contention}, \mathtt{true}, \mathrm{snd\_idle}\_i, p)$ of $\mathtt{Node}_i^{\mathtt{p}}$ with two edges, one labelled with the event $\langle\langle \mathsf{root\_contention}, \mathrm{snd\_idle}\_i, p, \mathsf{rec\_req\_fast}\rangle\rangle$, the other with $\langle\langle \mathsf{root\_contention}, \mathrm{snd\_idle}\_i, p, \mathsf{rec\_req\_slow}\rangle\rangle$. These events record the actual distribution used to make the transition (p) and its outcome ($\mathsf{rec\_req\_fast}$ or $\mathsf{rec\_req\_slow}$). This process is also repeated in the abstract model $\mathtt{I}_1^{\mathtt{p}}$, given that each location of $\mathtt{I}_1^{\mathtt{p}}$ is associated with certain locations of $\mathtt{Impl}^{\mathtt{p}}$ via a step refinement function. That this encoding of events suffices follows from the fact that the model $\mathsf{TA}_{\mathtt{Impl}^{\mathtt{p}}}$ has the following property: given a source state and an action, the target state is uniquely or probabilistically determined, implying that, for a given initial state of the continuous semantics $[\![\mathsf{TA}_{\mathtt{Impl}^{\mathtt{p}}}]\!]_{\mathbb{R}}$ and a finite sequence of actions of $[\![\mathsf{TA}_{\mathtt{Impl}^{\mathtt{p}}}]\!]_{\mathbb{R}}$, the state reached after executing actions in the sequence is unique. This property induces a one-to-one relationship between traces of $[\![\mathsf{TA}_{\mathtt{Impl}^{\mathtt{p}}}]\!]_{\mathbb{R}}$ and paths of $[\![\mathtt{Impl}^{\mathtt{p}}]\!]_{\mathbb{R}}$, as required.

Hence, if $[\![\mathsf{TA}_{\mathtt{I}_1^{\mathtt{p}}}]\!]_{\mathbb{R}}$ is proved to be an abstraction of $[\![\mathsf{TA}_{\mathtt{Impl}^{\mathtt{p}}}]\!]_{\mathbb{R}}$ via trace refinement, then we know that every path of $[\![\mathtt{I}_1^{\mathtt{p}}]\!]_{\mathbb{R}}$ can be matched, modulo the step refinement function f, with a path of $[\![\mathtt{Impl}^{\mathtt{p}}]\!]_{\mathbb{R}}$. Now note that, in the construction of $[\![\mathsf{TA}_{\mathtt{I}_1^{\mathtt{p}}}]\!]_{\mathbb{R}}$ and $[\![\mathsf{TA}_{\mathtt{Impl}^{\mathtt{p}}}]\!]_{\mathbb{R}}$, we replaced probabilistic branching with nondeterministic branching; more precisely, the alternatives available through probabilistic choice in any given state of the probabilistic system are available through nondeterministic choice in the corresponding state of the transition system. This permits us to reconstruct probabilistic branching in the following way: if, in a state s of the transition system, there are the actions $\langle\langle s \xrightarrow{a,\mu} s' \rangle\rangle$ and $\langle\langle s \xrightarrow{a,\mu} s'' \rangle\rangle$ available, then we know that we can reconstruct the transitions $s \xrightarrow{a,\mu} s'$ and $s \xrightarrow{a,\mu} s''$ of the probabilistic system, and that the probability of the first transition is $\mu(s')$, while the probability of the second transition is $\mu(s'')$. This permits us to construct adversaries of $[\![\mathtt{Impl}^{\mathtt{p}}]\!]_{\mathbb{R}}$ and $[\![\mathtt{I}_1^{\mathtt{p}}]\!]_{\mathbb{R}}$ from sets of traces of $[\![\mathsf{TA}_{\mathtt{Impl}^{\mathtt{p}}}]\!]_{\mathbb{R}}$ and $[\![\mathsf{TA}_{\mathtt{I}_1^{\mathtt{p}}}]\!]_{\mathbb{R}}$, along with their associated probability measures. Then, again modulo the step refinement function f, we can identify adversaries of $[\![\mathtt{Impl}^{\mathtt{p}}]\!]_{\mathbb{R}}$ and $[\![\mathtt{I}_1^{\mathtt{p}}]\!]_{\mathbb{R}}$ with one another. Indeed, if $[\![\mathsf{TA}_{\mathtt{I}_1^{\mathtt{p}}}]\!]_{\mathbb{R}}$ is an abstraction of $[\![\mathsf{TA}_{\mathtt{Impl}^{\mathtt{p}}}]\!]_{\mathbb{R}}$, every adversary of $[\![\mathtt{Impl}^{\mathtt{p}}]\!]_{\mathbb{R}}$ can be matched via f with an adversary of $[\![\mathtt{I}_1^{\mathtt{p}}]\!]_{\mathbb{R}}$ (for those familiar with Segala's theory of probabilistic systems, such adversaries will have the same trace distributions [Seg95]). Such matching allows us to reason about properties referring to the probability of observing sequences of actions, including our deadline property. More precisely, for every adversary of $[\![\mathtt{Impl}^{\mathtt{p}}]\!]_{\mathbb{R}}$, there exists an adversary of $[\![\mathtt{I}_1^{\mathtt{p}}]\!]_{\mathbb{R}}$ for which the probability of a root_i event occurring within *deadline* time units is the same. As the converse does not necessarily hold, we can conclude that the minimal and maximal probabilities of electing a leader within the deadline obtained by analysis of $\mathtt{I}_1^{\mathtt{p}}$ form lower and upper bounds, respectively, on the probabilities of electing a leader within the deadline for $\mathtt{Impl}^{\mathtt{p}}$.

After constructing the classical timed automata $\mathsf{TA}_{\mathtt{Impl}^{\mathtt{p}}}$ and $\mathsf{TA}_{\mathtt{I}_1^{\mathtt{p}}}$ from $\mathtt{Impl}^{\mathtt{p}}$ and $\mathtt{I}_1^{\mathtt{p}}$, we then used the methodology of [SS01] to test that $\mathsf{TA}_{\mathtt{Impl}^{\mathtt{p}}}$ trace refines $\mathsf{TA}_{\mathtt{I}_1^{\mathtt{p}}}$. This result was established by automating the refinement proof using the classical timed automaton model-checking tool UPPAAL. Hence, we conclude that every adversary of $\mathtt{Impl}^{\mathtt{p}}$ can be matched via f with an adversary of $\mathtt{I}_1^{\mathtt{p}}$, and that the minimum and maximum probabilities of satisfying the deadline property in $\mathtt{I}_1^{\mathtt{p}}$ form bounds on the corresponding probabilities obtained from $\mathtt{Impl}^{\mathtt{p}}$.

4. Verification and Analysis

Model-checking strategies for classical and probabilistic timed automata are crucially reliant on assumptions concerning the underlying model of time. Traditionally, timed automata are presented as a dense-time model (for example, with $\mathbb{T} = \mathbb{R}$), in which case the set of possible values that clocks take is infinite, and therefore the semantic probabilistic system is infinite-state. However, for a rich class of properties, including the reachability properties that we consider in this paper, a *finite-state* transition system to be model checked can be derived. A number of methods for obtaining such a finite-state system from a timed automaton operating in continuous time exist: one concerns partitioning the state space according to the well-established *region equivalence* [AD94]; another concerns the verification of timed reachability properties by *forward exploration* through the state space by iterating successively transition successor relations from state sets [YPD94, DOTY96]. Another alternative, in which the *integer semantics* of the model is verified, is also available, given the stronger assumption that the timed automaton is closed and diagonal-free.

We consider adaptations of all three methods to probabilistic timed automata. The rest of this section is divided into three parts: the first explores the verification of the abstract probabilistic timed automaton model $\mathtt{I}_1^{\mathtt{p}}$ shown in Figure 2 using all three methods mentioned in the previous paragraph, whereas the

second part considers the verification of the integer semantics of the full protocol model $\mathtt{Impl}^{\mathtt{p}}$. Finally, the statistics obtained from the probabilistic model-checking tool PRISM are presented.

In order to establish a technical basis for the following results, we use the probabilistic temporal logic PCTL [HJ94, BdA95] as a formal language for the deadline property. This logic is obtained from the standard temporal logic CTL [CE81] by replacing the standard path quantifiers $\forall$ and $\exists$ by a *probabilistic quantifier* Pr. For example, the PCTL formula $\Pr_{\geq\lambda}(\Diamond\Phi)$ is true in a state of a probabilistic system if all adversaries assign probability of at least λ to paths which reach a state in the future for which the subformula Φ is true. Formally, the semantics of PCTL formulae with a $\Diamond$ modality can be expressed using the maximal and minimal reachability probability notation in the following way. We defined the satisfaction relation of PCTL with respect to a certain subset $\mathcal{A} \subseteq Adv$ of adversaries in the manner of [BK98]. Let $s \in S$ be a state of a probabilistic system PS, $\sqsupseteq \in \{\geq, >\}$, $\sqsubseteq \in \{\leq, <\}$, $\lambda \in [0,1]$, and $[\![\Phi]\!]_{\mathcal{A}}$ be the set of states of PS that satisfy the PCTL formula Φ when the satisfaction relation is defined with respect to the set $\mathcal{A}$ of adversaries. Then the satisfaction relation $\models_{\mathcal{A}}$ for PCTL formulae with a $\Diamond$ modality, with respect to the set of adversaries $\mathcal{A} \subseteq Adv$, is defined by:

$$s \models_{\mathcal{A}} \Pr_{\sqsupseteq\lambda}(\Diamond\Phi) \Leftrightarrow \mathit{MinProbReach}_{\mathsf{PS}}(s, [\![\Phi]\!]_{\mathcal{A}}) \sqsupseteq \lambda\ , \quad s \models_{\mathcal{A}} \Pr_{\sqsubseteq\lambda}(\Diamond\Phi) \Leftrightarrow \mathit{MaxProbReach}_{\mathsf{PS}}(s, [\![\Phi]\!]_{\mathcal{A}}) \sqsubseteq \lambda\ .$$

Note that we use location names such as done, and constraints on clocks, such as $y \leq \mathit{deadline}$, as "atomic propositions" that are true or false in a state of a probabilistic system. For a more formal description of PCTL, refer to [HJ94, BdA95, BK98].

4.1. Abstract model

4.1.1. Forward reachability and symbolic states

The forward exploration method for the verification of probabilistic timed automata proceeds by a graph-theoretic search through the location space of the model, using edges, zones (of the enabling and invariant conditions) and clock resets to compute sets of reachable states of the continuous semantics. A set of states computed at any point during the algorithm is a pair comprising of location and a zone, called a *symbolic state*; for any classical or probabilistic timed automaton, the number of symbolic states is finite [DT98]. *Probabilistic timed reachability properties*, such as "with probability 0.99 or greater, the system reaches a leader-elected state within 100,000 ns", can be verified by defining a "symbolic state probabilistic system", the states of which are the generated symbolic states, and the transitions of which are derived from the probabilistic timed automaton's distributions and edges [KNSS02].

We now report on the forward reachability model construction and verification of a probabilistic deadline property of the root contention protocol within a continuous time semantics, using the probabilistic model-checking tool PRISM in conjunction with the tool HyTech [HHWT97]. The system model is taken to be the probabilistic timed automaton $\mathtt{I}_1^{\mathtt{p}}$ introduced in the previous section. We proceed according to the following two steps: first we describe how the automaton $\mathtt{I}_1^{\mathtt{p}}$ is decorated with additional locations and transitions to result in a new probabilistic timed automaton $\mathtt{I}_1^{\mathtt{p}+}$ for which the deadline property on $\mathtt{I}_1^{\mathtt{p}}$ is reduced to a probabilistic reachability property on $\mathtt{I}_1^{\mathtt{p}+}$. Secondly, we briefly describe the state-space analysis algorithm of [KNSS02], which establishes probabilistic reachability properties of probabilistic timed automata. We also describe its implementation in the scripting language of the tool HyTech, its application to the probabilistic timed automaton obtained in the first step, and the way in which the results obtained are subsequently used as input to PRISM.

Step 1: Decorating the probabilistic timed automaton. The property of interest requires that the system elect a leader before a certain deadline with a certain probability or greater, *for all* adversaries. In contrast, the forward reachability procedure of [KNSS02] analyzes properties considering the *existence* of an adversary in which a certain location is reached with a given probability or greater. The former type of property can be represented in terms of the latter given adjustments to the structure of the model $\mathtt{I}_1^{\mathtt{p}}$ (following precedents for classical timed automata such as [LPY98]). We add the new location $\mathsf{deadline\_exceeded}$ (which has no outgoing edges), and, from all locations apart from done, we add a distribution over a single edge with the target location $\mathsf{deadline\_exceeded}$. Recall that y is the clock of $\mathtt{I}_1^{\mathtt{p}}$ which measures the time elapsed since the start of system execution. The enabling condition of each edge takes the form $y \geq \mathit{deadline}$,

where *deadline* refers to the value of the time bound considered, and the invariant condition of every location except deadline_exceeded and done is taken in conjunction with the constraint $y \leq \mathit{deadline}$. The result of these changes is that, if the value of y reached deadline in a location other than done, then the model must make a transition to deadline_exceeded (or done, if an enabling condition of an edge to done is satisfied) before letting time pass. We denote the new probabilistic timed automaton resulting from these changes by $\mathtt{I}_1^{\mathtt{p}+}$. Then the target location of the forward reachability property is taken to be deadline_exceeded. If the maximal probability with which deadline_exceeded can *possibly* (existentially) be reached is λ, then $1-\lambda$ will be the maximal probability with which deadline_exceeded will *inevitably* (universally) be avoided.

We now establish the correctness of this transformation by formalizing the concepts using PCTL. Note that the forward reachability procedure can verify generic PCTL properties of the form $\Pr_{\sqsubseteq\lambda}(\Diamond\Phi)$ (or $\neg\Pr_{\sqsubseteq\lambda}(\Diamond\Phi)$), where $\sqsubseteq \in \{<, \leq\}$, $\lambda \in [0,1]$, and Φ is a PCTL formula which does not have a subformula featuring a probabilistic quantifier Pr. We can represent the universal and existential properties described above in the syntax of PCTL: the property "for all adversaries, the probability of reaching the location done within *deadline* nanoseconds is λ_1 or greater" is written as the PCTL formula $\Pr_{\geq\lambda_1}(\Diamond(\mathsf{done} \wedge y \leq \mathit{deadline}))$. Similarly, the property "there exists an adversary such that the probability of reaching the location deadline_exceeded is λ_2 or greater" is written as $\neg\Pr_{<\lambda_2}(\Diamond\mathsf{deadline\_exceeded})$ (paraphrased as "it is not true that all adversaries reach deadline_exceeded with probability strictly less than λ_2").

For an arbitrary PCTL formula Φ and $\lambda \in [0,1]$, we have $\Pr_{\geq\lambda}(\Diamond\Phi) \equiv \neg\Pr_{<1-\lambda}(\Box\neg\Phi)$ [BK98]. Applied to the first PCTL formula given in the previous paragraph, this gives us the equivalence:

$$\Pr_{\geq\lambda_1}(\Diamond(\mathsf{done} \wedge y \leq \mathit{deadline})) \equiv \neg\Pr_{<1-\lambda_1}(\Box\neg(\mathsf{done} \wedge y \leq \mathit{deadline})) \ .$$

Although the probabilistic quantifier $\Pr_{<1-\lambda_1}$ is of the correct form for our probabilistic timed automata verification approach, the path formula which it quantifies over uses a $\Box$ temporal modality, rather than a $\Diamond$ modality, as required. However, on non-Zeno paths of $\mathtt{I}_1^{\mathtt{p}+}$, the formulas $\Box\neg(\mathsf{done} \wedge y \leq \mathit{deadline})$ and $\Diamond\mathsf{deadline\_exceeded}$ are equivalent.

Lemma 4.1. Let ω be a non-Zeno path of $[\![\mathtt{I}_1^{\mathtt{p}+}]\!]_{\mathbb{R}}$, and let $\mathcal{A} \subseteq \mathit{Adv}$. Then $\omega \models_{\mathcal{A}} \Box\neg(\mathsf{done} \wedge y \leq \mathit{deadline})$ if and only if $\omega \models_{\mathcal{A}} \Diamond\mathsf{deadline\_exceeded}$.

Proof. See Appendix B. □

Corollary 4.2. Let $\mathcal{A}_{nZ} \subseteq \mathit{Adv}$ be the set of non-Zeno adversaries of $[\![\mathtt{I}_1^{\mathtt{p}+}]\!]_{\mathbb{R}}$. Then $(\mathsf{start\_start}, \mathbf{0}) \models_{\mathcal{A}_{nZ}} \Pr_{<1-\lambda_1}(\Box\neg(\mathsf{done} \wedge y \leq \mathit{deadline}))$ if and only if $(\mathsf{start\_start}, \mathbf{0}) \models_{\mathcal{A}_{nZ}} \Pr_{<1-\lambda_1}(\Diamond\mathsf{deadline\_exceeded})$.

The corollary follows because, although a non-Zeno adversary may exhibit Zeno paths (for which the equivalence of Lemma 4.1 does not hold), the probability measure of such paths must be 0. Hence, we have reduced the PCTL formula $\Pr_{\geq\lambda_1}(\Diamond(\mathsf{done} \wedge y \leq \mathit{deadline}))$ to $\neg\Pr_{<1-\lambda_1}(\Diamond\mathsf{deadline\_exceeded})$ on the non-Zeno adversaries of $\mathtt{I}_1^{\mathtt{p}+}$. That is, we have converted a formula concerning the reachability of a certain set of states with a certain probability or greater into a formula concerning the reachability of a different set of states with a certain probability or less.

Not all adversaries of $\mathtt{I}_1^{\mathtt{p}+}$ (and indeed $\mathtt{I}_1^{\mathtt{p}}$) are non-Zeno, because an adversary may choose to advance time in a Zeno, or convergent manner (for example, letting 0 nanoseconds elapse in all time transitions performed from some point onwards). This would be problematic for our analysis if the set of adversaries which reach the target location deadline_exceeded with the maximal probability are all Zeno. The following lemma shows that this is not the case: more precisely, we show that, for any Zeno adversary of $\mathtt{I}_1^{\mathtt{p}+}$, there exists a non-Zeno adversary which can reach the location deadline_exceeded with an equal or greater probability.

Lemma 4.3. Let A be a Zeno adversary of $[\![\mathtt{I}_1^{\mathtt{p}+}]\!]_{\mathbb{R}}$. Then there exists a non-Zeno adversary A_{nZ} of $[\![\mathtt{I}_1^{\mathtt{p}+}]\!]_{\mathbb{R}}$ such that:

$$\mathit{ProbReach}^{A_{nZ}}((\mathsf{start\_start}, \mathbf{0}), \mathsf{deadline\_exceeded}) \geq \mathit{ProbReach}^{A}((\mathsf{start\_start}, \mathbf{0}), \mathsf{deadline\_exceeded}) \ .$$

Proof. See Appendix C. □

Step 2: Applying forward reachability to probabilistic timed automata. The second step involves the computation of the relevant symbolic states, together with information concerning transitions made between such symbolic states, using the tool HyTech. As this tool is equipped with a simple programming

language in which the symbolic states of a classical timed automaton serve as the basic data structure, it permits us to implement the forward reachability algorithm of [KNSS02], given that a number of auxiliary, discrete, integer-valued variables are added to a classical timed automaton representation of $\mathtt{I}_1^{\mathtt{p}+}$ so that sufficient information concerning the probabilistic transition relation between the symbolic states is maintained. Firstly, the additional variable `edge` is used to record the edge of the classical timed automaton that was last traversed; this is achieved by assigning to each edge of the model a different value, and then setting the variable `edge` to this value upon traversal of the relevant edge. Secondly, the variable `number` is used to uniquely number the symbolic states that are generated by the program; this is to distinguish sets of states which may have the same location, but, because they belong to different symbolic states, may correspond to different probabilities of reaching the target location deadline_exceeded. Finally, when a new symbolic state is generated (recall that this occurs when the effect of a discrete transition is applied to a previously generated symbolic state), the variable `source` is set to the value of `number` corresponding to the source symbolic state. Note that a symbolic state may have more than one value of `source`, meaning that it may be reached from more than one symbolic state in a single discrete transition.

We then perform probabilistic model checking on the resulting "symbolic state probabilistic system", the states of which correspond to the symbolic states generated by the previous step, and the transitions of which are derived from the values of `edge` and `source` in each symbolic state, against a PCTL property which refers to the maximum probability of reaching any symbolic state with the location component deadline_exceeded.

Although it is established in [KNSS02] that, in general, the forward reachability process described above is only guaranteed to compute an *upper* bound on the reachability probability in the probabilistic timed automaton, for certain classes of systems, the technique will result in the computation of the *exact* probability; indeed, the probabilistic timed automaton $\mathtt{I}_1^{\mathtt{p}+}$ falls into this class. Rather than present a formal proof, we provide an experimental validation of this result by verifying that the probabilities obtained using the forward reachability method agree with those obtained on the region graph and the probabilistic timed automaton with integer semantics.

4.1.2. Region graph

The second approach taken for the verification of the abstract probabilistic timed automaton model $\mathtt{I}_1^{\mathtt{p}}$ involves the definition of a finite state *region graph*. Such a graph is obtained from a finite partition of the clock valuation space first defined in the context of classical timed automata in [AD94]. The region graph is defined using the combination of this partition with the locations and transitions of the model. It is shown in [KNSS02] that the region graph of a probabilistic timed automaton takes the form of a probabilistic system which can be used to verify PCTL properties. Hence, we can use the region graph of $\mathtt{I}_1^{\mathtt{p}}$ for probabilistic model checking of the property $\mathrm{Pr}_{\geq \lambda_1}(\Diamond(\mathsf{done} \wedge y \leq \mathit{deadline}))$ directly. For efficiency, we combine all states corresponding to the bound *deadline* being exceeded by the clock y into a single state.

This region graph is then encoded directly into the system description language of PRISM in anticipation of probabilistic model checking. The description language is state-based, where a state of the model corresponds to a valuation of a finite set of finite-domain variables, and where the state transitions are obtained by guarded commands which establish a relationship between the values of the state variables before and after a transition. We use six variables to encode the region graph: one for the location component of the region, two for the integer values of each of the clocks x and y, two to record whether the fractional part of the clocks is 0 or not, and one to record the order on the fractional parts of the two clocks.

4.1.3. Integer semantics

The final technique for the verification of the abstract probabilistic timed automaton $\mathtt{I}_1^{\mathtt{p}}$ concerns the use of its integer semantics, defined in Definition 2.4. As in the previous two subsections, we obtain a finite state probabilistic system which can be model checked using PRISM. The encoding of the integer-semantics probabilistic system into the system description language of PRISM involves the use of three variables: one to denote the location component of the state, and two to denote the integer values of the clocks.

As the probabilistic timed automaton $\mathtt{I}_1^{\mathtt{p}}$ is closed and diagonal-free, Theorem 2.5 holds, and therefore the minimum probability of reaching a state with location component done before the deadline is the same in the continuous and integer semantics.

Table 1. Verification results for the abstract model $\mathtt{I}_1^{\mathtt{P}}$ with the wire delay set to 360 ns.

deadline (ns)	forward reachability			region graph			integer semantics			probability
	states	iters.	time (s)	states	iters.	time (s)	states	iters.	time (s)	
2,000	48	13	0.06	423,016	670	13.2	68,185	169	1.31	0
2,500	68	26	0.07	652,890	743	18.9	104,333	207	4.08	0.5
3,000	72	26	0.09	900,390	1,157	50.2	143,133	293	3.86	0.5
3,500	103	31	0.11	1,147,890	1,230	68.2	181,933	331	5.43	0.625
4,000	137	33	0.13	1,395,390	1,485	119	220,733	375	7.72	0.625
4,500	142	38	0.16	1,642,890	1,717	139	259,533	455	11.7	0.78125
5,000	183	38	0.19	1,890,390	1,972	175	298,333	499	16.1	0.78125
5,500	218	40	0.22	2,137,890	2,131	228	337,133	541	24.9	0.84375
6,000	234	45	0.24	2,385,390	2,300	265	375,933	581	22.5	0.851562
7,000	–	–	–	2,880,390	2,787	374	453,533	705	29.8	0.908203
8,000	–	–	–	3,375,390	3,115	519	531,133	789	39.6	0.939453
9,000	–	–	–	3,870,390	3,602	659	608,733	913	51.5	0.961914
10,000	–	–	–	4,365,390	3,930	843	686,333	995	62.1	0.974731
100,000	–	–	–	48,915,390	12,229	6,007	7,670,333	3,097	599	0.999996
1,000,000	–	–	–	–	–	–	77,510,333	3,097	589	0.999996
∞	–	–	–	1,542	1,492	1.25	776	752	0.51	1

Table 2. Verification results for the abstract model $\mathtt{I}_1^{\mathtt{P}}$ with the wire delay set to 30 ns.

deadline (ns)	forwards reachability			region graph			integer semantics			probability
	states	iters.	time (s)	states	iters.	time (s)	states	iters.	time (s)	
1,500	32	8	0.02	57,903	1	0.04	7,235	1	0.02	0
2,000	51	11	0.04	85,638	677	2.31	14,868	174	0.36	0.5
2,500	71	11	0.05	144,450	684	5.09	24,389	174	0.81	0.5
3,000	71	14	0.07	222,090	1,032	9.94	36,410	265	0.94	0.625
3,500	101	14	0.11	319,852	1,387	29.8	52,108	356	4.09	0.78125
4,000	131	17	0.15	433,018	1,387	35.2	69,791	356	4.69	0.78125
4,500	131	20	0.16	563,570	1,742	67.8	90,212	447	8.80	0.851563
5,000	171	20	0.17	715,390	1,742	92.5	114,001	447	12.3	0.851563
5,500	202	23	0.19	880,754	2,097	122	139,817	538	22.0	0.908203
6,000	211	23	0.21	1,065,154	2,398	193	168,567	611	27.2	0.931641
7,000	–	–	–	1,455,737	2,807	248	229,085	720	40.6	0.962036
8,000	–	–	–	1,851,737	3,162	293	290,185	811	56.8	0.975494
9,000	–	–	–	2,247,737	3,517	439	351,285	902	76.7	0.984383
10,000	–	–	–	2,643,737	3,872	612	412,385	993	103	0.989970
100,000	–	–	–	38,283,737	9,985	1,059	5,911,385	2,536	736	0.999996
1,000,000	–	–	–	–	–	–	60,901,385	2,536	757	0.999996
∞	–	–	–	1,212	1,360	1.11	611	686	0.52	1

4.1.4. Results

The results obtained from verifying the abstract model are shown in Tables 1 and 2. We include the verification results for the case in which the communication delay along the wires between the two nodes is 360 ns in Table 1, and 30 ns in Table 2. In both tables, the left-most column shows the deadline used in the property, and the right-most column shows the minimum probability with which the system reaches a leader-elected state before the deadline (the same probability was computed by all three methods). The results reflect the obvious fact that increasing the value of the deadline has the effect of increasing the probability of a timely leader election; intuitively, the greater the deadline, the "more chances" the system has of flipping different results for each node (therefore electing a leader). Observe that the rate of increase both in the number of computed symbolic states and the probability proceeds in "jumps" as the deadline increases. In particular, the probability of satisfying the required property is dependent on how many protocol restarts (and therefore coin flips) the system can perform before expiration of the deadline. Although the generation of probabilities for low values of the deadline (particularly 3,000 or less) is trivial, the probability computation becomes more involved for greater deadlines.

For all three techniques (forward reachability, region graph, and integer semantics), the sub-columns marked "states" give the number of states of the probabilistic system which is taken as input to PRISM, the sub-columns marked "iters." give the number of iterations taken by PRISM to compute the probability, and the sub-columns marked "time" give the time in seconds taken by PRISM for the computation. Observe that the probability shown in the right-most column of the table is actually one minus the maximum of the existential reachability property computed by PRISM using the forward reachability method (as we verify

the dual of the actual formula we are interested in). As expected, the probabilities computed for all three methods agree. A dash − denotes that verification was not attempted.

The symbolic states generated by the forward reachability algorithm, as implemented in HyTech, form the states of the probabilistic system taken as input to PRISM. We do not include time and memory statistics for HyTech as, in general, the execution of the forward reachability algorithm implementation periodically required termination and subsequent re-execution in order to avoid excessive memory consumption (for higher deadlines, this was achieved using a shell script). As a guide, the symbolic states for the deadline of 6,000 ns took approximately 24 hours to obtain. This performance bottleneck made the verification of greater deadlines impractical. In contrast to this poor performance, observe that the state spaces which are generated by the forward reachability method are significantly smaller (by an order of magnitude of 10^4 compared to the region graph technique, and by an order of 10^3 compared to the integer semantics technique). Deadlines of up to 100,000 ns and 1,000,000 ns are within reach of the capabilities of PRISM using the region graph and the integer semantics, respectively, although it should be noted that PRISM is designed to handle large state spaces using efficient data structures such as Multi-Terminal Binary Decision Diagrams (MTBDDs) [CFM+93]. Using the integer semantics of $\mathtt{I}_1^{\mathtt{p}}$ results in performance gains over the region-based approach (the state spaces generated are just over 6 times smaller than those obtained from the region graph), which is explained by observing that the information encoded into the integer semantics is a strict subset of that encoded into the region graph.

In addition to the verification times given above, the tool PRISM also involves a model construction phase, during which the system description is parsed, converted into an MTBDD representation, and subjected to an efficient, BDD-based state-space exploration procedure to find the reachable states of the system [dAKN+00]. As a guide, for a deadline of 100,000, the model construction time for the region graph is approximately 20 minutes, compared to 14 minutes for the integer semantics model. For more information, including model construction times for all deadlines, see the web-site [Pri].

As PRISM computes the probability of property satisfaction using an iterative technique, a termination criterion is given by a sufficiently small ϵ, where the iteration terminates if the relative difference of successively computed probabilities is less than ϵ. For this case study, we set $\epsilon = 10^{-6}$. Hence, although the probabilities computed using the integer semantics for deadlines of 100,000 and 1,000,000 ns are identical, this should not be interpreted as indicating that the probabilities have converged. Indeed, the final row of the table gives the performance statistics of the verification of the property $\Pr_{\geq\lambda_3}(\Diamond(\mathsf{done}) \wedge y \leq \infty) \equiv \Pr_{\geq\lambda_3}(\Diamond(\mathsf{done}))$ on the region graph and integer semantics (such a property is out of the scope of the forward reachability procedure). We perform verification on a simplified model in which the clock y, used to measure global time, is not represented. Use of PRISM establishes that this property is satisfied even for $\lambda_3 = 1$, and therefore we conclude that the probability of electing a leader converges to 1 as the deadline approaches ∞. The efficiency of PRISM for this property is due to the fact that a graph-theoretic pre-computation procedure which calculates the states satisfying formulae with probability 1, is used in the model-checking phase.

4.2. Full model

In this section, we report on the application of the most efficient verification method employed in the previous section, that concerning integer semantics, to the *full* probabilistic timed automaton $\mathtt{Impl}^{\mathtt{p}}$ of the root contention protocol. As with the case of $\mathtt{I}_1^{\mathtt{p}}$ in the previous section, verification on the integer semantics is sufficient because $\mathtt{Impl}^{\mathtt{p}}$ is a closed, diagonal-free probabilistic timed automaton, and, by Theorem 2.5, the minimum probability of reaching a state with a leader-elected location before the deadline is the same for the continuous and integer semantics of $\mathtt{Impl}^{\mathtt{p}}$. We do not attempt the region graph technique, as the state space generated by this method is strictly greater than that for the integer semantics.

Rather than construct the integer semantics $[\![\mathtt{Impl}^{\mathtt{p}}]\!]_{\mathbb{N}}$ of the composed probabilistic timed automaton $\mathtt{Impl}^{\mathtt{p}} = \mathtt{Node}_1^{\mathtt{p}} \| \mathtt{Wire}_1 \| \mathtt{Wire}_2 \| \mathtt{Node}_2^{\mathtt{p}}$, we elected to define the integer semantics of each of the four components of $\mathtt{Impl}^{\mathtt{p}}$, and then use the parallel composition facility of PRISM to generate the probabilistic system

$$[\![\mathtt{Node}_1^{\mathtt{p}}]\!]_{\mathbb{N}} \| [\![\mathtt{Wire}_1]\!]_{\mathbb{N}} \| [\![\mathtt{Wire}_2]\!]_{\mathbb{N}} \| [\![\mathtt{Node}_2^{\mathtt{p}}]\!]_{\mathbb{N}} \ .$$

As it was established that $[\![\mathtt{Impl}^{\mathtt{p}}]\!]_{\mathbb{N}}$ is isomorphic to $[\![\mathtt{Node}_1^{\mathtt{p}}]\!]_{\mathbb{N}} \| [\![\mathtt{Wire}_1]\!]_{\mathbb{N}} \| [\![\mathtt{Wire}_2]\!]_{\mathbb{N}} \| [\![\mathtt{Node}_2^{\mathtt{p}}]\!]_{\mathbb{N}}$ in Section 2.2.1, and as isomorphism is stronger than (probabilistic) bisimulation, which preserves all PCTL formulae on probabilistic systems [SL95], we conclude that the two systems satisfy the same PCTL formulae, including the formula $\Pr_{\geq\lambda}(\Diamond(\mathsf{done} \wedge y \leq \mathit{deadline}))$ requiring that "for all adversaries, the probability of electing a

Table 3. Verification results for the full model $\texttt{Impl}^{\texttt{p}}$.

deadline (ns)	wire delay of 360 ns				wire delay of 30 ns			
	states	iters.	time (s)	probability	states	iters.	time (s)	probability
1,500	2,426,116	48	6.34	0	29,631	15	0.35	0
2,000	6,719,773	50	20.2	0	80,980	179	4.23	0.5
2,500	14,089,691	212	51.7	0.5	143,394	185	7.58	0.5
3,000	23,319,148	213	287	0.5	213,805	275	12.9	0.625
3,500	33,789,641	341	621	0.625	328,547	371	22.8	0.78125
4,000	44,366,235	341	7,325	0.625	434,364	371	29.3	0.78125
4,500	54,973,279	470	11,338	0.78125	572,437	467	40.7	0.851563
5,000	65,586,679	470	24,556	0.78125	731,109	467	61.7	0.851563
5,500	76,200,079	552	23,229	0.84375	899,798	563	71.5	0.908203
6,000	86,813,479	599	35,293	0.851562	1,093,658	631	114	0.931641
7,000	108,040,279	728	51,028	0.908203	1,499,591	755	132	0.962036
8,000	129,267,079	810	58,504	0.939453	1,915,291	851	209	0.975494
9,000	150,493,879	939	73,829	0.961914	2,330,991	947	253	0.984383
10,000	171,720,679	1,021	95,000	0.974731	2,746,691	1,043	296	0.989970
∞	212,268	844	336	1	4,157	712	36.2	1

leader before the deadline is greater than λ". Each of the probabilistic systems $[\![\texttt{Node}_1^{\texttt{p}}]\!]_{\mathbb{N}}$, $[\![\texttt{Wire}_1]\!]_{\mathbb{N}}$, $[\![\texttt{Wire}_2]\!]_{\mathbb{N}}$, and $[\![\texttt{Node}_2^{\texttt{p}}]\!]_{\mathbb{N}}$ is encoded as a separate "module", with its own state variables, within the language of PRISM. For example, the module for $[\![\texttt{Node}_1^{\texttt{p}}]\!]_{\mathbb{N}}$ features two variables, one corresponding to the location component of the state, and the other recording the current value of the clock of $\texttt{Node}_1^{\texttt{p}}$. Finally, we also include a module recording global time using a single variable.

The results obtained from generating the integer semantics of the full model are given in Table 3, both for a communication delay of 360 ns and for 30 ns. Note that the automatic model construction process generally took a small number of minutes, in addition to the computation time given in the table (see the web-page [Pri] for full details). We observe that the ratio of the number of states obtained from this method to the number of states obtained from the integer semantics of the abstract model $\texttt{I}_1^{\texttt{p}}$ increases as the deadline increases (from a ratio of approximately 100 for the deadline 2,000 ns, to approximately 250 for the deadline 10,000 ns), although at a decreasing rate; therefore, the abstract model $\texttt{I}_1^{\texttt{p}}$ scales better with regard to the largest numerical constant of the system. Note also that the probabilities obtained using the full model agree with those obtained from verification on the abstract model.

Although the utility of integer semantics is limited to closed, diagonal-free classical and probabilistic timed automata, in this case such semantics have allowed us to verify high deadlines without the need for abstraction. We envisage that the combination of integer semantics and abstraction may lead to successful verification of more complex probabilistic real-time systems.

5. Conclusions

We have considered an approach to the formal specification of both timed and probabilistic aspects of the root contention protocol of IEEE 1394 using probabilistic timed automata, and have investigated the way in which the tools PRISM and HyTech may be used to automatically verify the resulting model against deadline properties. The properties that we consider are closely related to the property "a leader is eventually chosen", whereas we do not consider safety properties such as "only one leader is chosen". The verification used a variety of techniques, including high-level abstraction and methods for the construction of finite-state probabilistic systems using explorative algorithms, region equivalence, and non-standard integer semantics. The abstraction method considered was originally developed in [SV99] as one step of a manual proof; we envisage that automatic, probabilistic abstraction methods, for example [DJJL01], could also be applied to improve the efficiency of probabilistic timed automata model checking.

We consider probabilistic timed automata to be particularly appropriate for the specification of the nodes involved in the root contention protocol, together with their communication media. Probabilistic timed automata can express a variety of notions which are important in representing the nodes and wires, such as the probabilistic coin-flip, the communication between the components, and the nondeterministic choice of time delays (within given short or long waiting intervals) in a formal yet concise way. The timing parameters from the updated IEEE 1394a standard were used in the models, although observe that it is straightforward to change the values of the parameters and re-run the experiments, as we did for the wire propagation delay. Such a change of parameters results in the loss of the work performed in previous analyses,

although, because the analysis procedure is automatic, this involves no significant manual effort. We consider probabilistic model checking to be an appropriate technique for the analysis of the root contention protocol, as manual analysis is notoriously prone to human error, even for simple systems.

When evaluating the total time taken to obtain a solution via our analysis, we should consider that both the modelling and analysis of [SS01] required approximately two person months. As we use the classical timed automata models (and some of the results) of [SS01] as a basis for our probabilistic models, we should regard this as the time taken to also derive our models. Subsequent translation of these models into the HyTech and PRISM description languages required a small number of hours (with the region graph models being the most difficult to encode, as detailed understanding of its partitioning method is necessary). We note that, given knowledge of probabilistic systems and classical timed automata, the textual system description languages of PRISM and HyTech are easy to learn, requiring only a number of hours of effort. For an average programmer, or a person not literate in formal methods, the task of obtaining sufficient knowledge of the formalisms may be more challenging, but we nevertheless estimate this to be possible within in a week. In terms of analysis, as noted in Section 4, the forward reachability constructions implemented in HyTech took approximately 24 hours for the higher values of the deadline considered, whereas model checking the resulting probabilistic systems using PRISM required seconds; similarly, model checking using PRISM alone on the region graph and the integer semantics required at most 27 hours for the full model $\texttt{Impl}^{\texttt{P}}$. Finally, the proofs of correctness for the minimal and maximal reachability probabilities obtained via the integer semantics, and the results concerning non-Zenoness for the forward reachability analysis, each took a small number of days.

In terms of future work, our models could be used as a basis for the automatic computation of the *expected time* to leader election; it would therefore be of interest to formalize such a notion in the context of probabilistic timed automata, and to relate it to other precedents in the probabilistic model checking literature, for example [dA97]. One aspect that is missing from the forward reachability approach is an automatic method for verifying that the probability of the system electing a leader within a deadline of infinity is 1. This requires a thorough investigation into such a class of probabilistic "liveness" properties, which cannot be analyzed by use of the reachability algorithm of [KNSS02]. Although manual proof methods dealing with such expected-time and probabilistic liveness properties have been known for some time, in [Koz81] and [HSP83] respectively, we reiterate that such proof methods are prone to error, and that there is a chronic need for automatic, model-checking methods for such properties.

The verification of the root contention protocol was carried out as part of an ongoing activity concerning the practical implementation of model-checking methods for probabilistic timed automata. One lesson learned was that, although the size of the generated state spaces can be handled easily by the tool PRISM, the high memory usage of our algorithm implementation in HyTech proved to be the bottleneck of the analysis, and meant that we were able to perform verification of only short deadlines. This is to be expected, as HyTech is a flexible tool which can be applied to a superclass of timed automata, including hybrid automata and parameterized systems, and for which efficient data structures have not been implemented; in contrast, PRISM uses the compact data structure of MTBDDs to represent probabilistic systems. Indeed, the forward reachability algorithm has been implemented recently in the classical timed automata tool Kronos [DOTY96], for which efficient data structures to represent symbolic states have been developed, with results that far surpass those obtained with HyTech [DKN02].

References

[ACD91] R. Alur, C. Courcoubetis, and D. L. Dill. Model-checking for probabilistic real-time systems. In J. Leach Albert, B. Monien, and M. Artalejo Rodríguez, editors, *Proceedings of the 18th International Conference on Automata, Languages and Programming (ICALP'91)*, volume 510 of *LNCS*, pages 115–136. Springer-Verlag, 1991.

[AD94] R. Alur and D. L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126(2):183–235, 1994.

[AIKY95] R. Alur, A. Itai, R. Kurshan, and M. Yannakakis. Timing verification by successive approximation. *Information and Computation*, 18(1):142–157, 1995.

[AMP98] E. Asarin, O. Maler, and A. Pnueli. On discretization of delays in timed automata and digital circuits. In Sangiorgi and de Simone [SdS98], pages 470–484.

[BdA95] A. Bianco and L. de Alfaro. Model checking of probabilistic and nondeterministic systems. In P. S. Thiagarajan, editor, *Proceedings of the International Conference on the Foundations of Software Technology and Theoretical Computer Science (FSTTCS'95)*, volume 1026 of *LNCS*, pages 499–513. Springer-Verlag, 1995.

[Bey01] D. Beyer. Improvements in BDD-based reachability analysis of timed automata. In J. N. Oliveira and P. Zave,

editors, *Proceedings of the 10th International Symposium of Formal Methods Europe (FME 2001)*, volume 2021 of *LNCS*, pages 318–343. Springer-Verlag, 2001.

[BK98] C. Baier and M. Z. Kwiatkowska. Model checking for a probabilistic branching time logic with fairness. *Distributed Computing*, 11(3):125–155, 1998.

[BKH99] C. Baier, J.-P. Katoen, and H. Hermanns. Approximate symbolic model checking of continuous-time Markov chains. In J. Baeten and S. Mauw, editors, *Proceedings of the 10th International Conference on Concurrency Theory (CONCUR'99)*, volume 1664 of *LNCS*, pages 142–162. Springer-Verlag, 1999.

[BMT99] M. Bozga, O. Maler, and S. Tripakis. Efficient verification of timed automata using dense and discrete time semantics. In L. Pierre and T. Kropf, editors, *Proceedings of the 11th Advanced Research Working Conference on Correct Hardware Design and Verification Methods (CHARME'99)*, volume 1703 of *LNCS*, pages 125–141. Springer-Verlag, 1999.

[BST00] G. Bandini, R. L. Spelberg, and W. J. Toetenel. Parametric verification of the IEEE 1394a root contention protocol using LPMC. In *Proceedings of the 7th International Conference on Real-Time Computing Systems and Applications (RTCSA 2000)*, pages 207–214. IEEE Computer Society Press, 2000.

[CE81] E. M. Clarke and E. A. Emerson. Design and synthesis of synchronization skeletons using branching time temporal logic. In D. Kozen, editor, *Proceedings of the Workshop on Logics of Programs*, volume 131 of *LNCS*, pages 52–71. Springer-Verlag, 1981.

[CFM+93] E. M. Clarke, M. Fujita, P. McGeer, K. McMillan, J. Yang, and X. Zhao. Multi-Terminal Binary Decision Diagrams: An efficient data structure for matrix representation. In *Proceedings of the International Workshop on Logic Synthesis (IWLS'93)*, pages 6a:1–15, 1993. Also available in *Formal Methods in System Design*, 10(2/3):149–169, 1997.

[CS01] A. Collomb-Annichini and M. Sighireanu. Parameterized reachability analysis of the IEEE 1394 root contention protocol using TReX. In P. Pettersson and S. Yovine, editors, *Proceedings of the Workshop on Real-Time Tools (RT-TOOLS 2001)*, 2001.

[dA97] L. de Alfaro. *Formal Verification of Probabilistic Systems*. PhD thesis, Stanford University, 1997.

[dA98] L. de Alfaro. Stochastic transition systems. In Sangiorgi and de Simone [SdS98], pages 423–438.

[D'A99] P. R. D'Argenio. *Algebras and Automata for Timed and Stochastic Systems*. PhD thesis, Department of Computer Science, University of Twente, 1999.

[dAKN+00] L. de Alfaro, M. Z. Kwiatkowska, G. Norman, D. Parker, and R. Segala. Symbolic model checking of concurrent probabilistic processes using MTBDDs and the Kronecker representation. In S. Graf and M. Schwartzbach, editors, *Proceedings of the 6th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS 2000)*, volume 1785 of *LNCS*, pages 395–410. Springer-Verlag, 2000.

[Der70] C. Derman. *Finite-State Markovian Decision Processes*. Academic Press, 1970.

[DJJL01] P. R. D'Argenio, B. Jeannet, H. E. Jensen, and K. G. Larsen. Reachability analysis of probabilistic systems by successive refinements. In L. de Alfaro and S. Gilmore, editors, *Proceedings of the Joint PAPM-PROBMIV 2001 Workshop*, volume 2165 of *LNCS*, pages 39–56. Springer-Verlag, 2001.

[DKN02] C. Daws, M. Z. Kwiatkowska, and G. Norman. Automatic verification of IEEE 1394 root contention protocol with KRONOS and PRISM. In R. Cleaveland and H. Garavel, editors, *Proceedings of the 7th International Workshop on Formal Methods for Industrial Critical Systems (FMICS 2002)*, pages 107–122, 2002. To appear in Electronic Notes in Theoretical Computer Science.

[DOTY96] C. Daws, A. Olivero, S. Tripakis, and S. Yovine. The tool KRONOS. In R. Alur, T. A. Henzinger, and E. D. Sontag, editors, *Hybrid Systems III: Verification and Control*, volume 1066 of *LNCS*, pages 208–219. Springer-Verlag, 1996.

[DT98] C. Daws and S. Tripakis. Model–checking of real–time reachability properties using abstractions. In Steffen [Ste98], pages 313–329.

[DY95] C. Daws and S. Yovine. Two examples of verification of multirate timed automata with KRONOS. In A. Burns, Y.-H. Lee, and K. Ramamritham, editors, *Proceedings of the 16th IEEE Real-Time Systems Symposium (RTSS'95)*, pages 66–75. IEEE Computer Society Press, 1995.

[FS01] C. Fidge and C. Shankland. But what if I don't want to wait forever? In S. Maharaj, J. Romijn, and C. Shankland, editors, *Proceedings of the International Workshop on Application of Formal Methods to IEEE 1394 Standard*, 2001.

[HHWT95] T. A. Henzinger, P.-H. Ho, and H. Wong-Toi. A user guide to HYTECH. In E. Brinksma, W. R. Cleaveland, K. G. Larsen, T. Margaria, and B. Steffen, editors, *Proceedings of the 1st International Workshop on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'95)*, volume 1019 of *LNCS*, pages 41–71. Springer-Verlag, 1995.

[HHWT97] T. A. Henzinger, P.-H. Ho, and H. Wong-Toi. HYTECH: a model checker for hybrid systems. *Software Tools for Technology Transfer*, 1(1+2):110–122, 1997.

[HJ94] H. Hansson and B. Jonsson. A logic for reasoning about time and reliability. *Formal Aspects of Computing*, 6(5):512–535, 1994.

[HMP92] T.A. Henzinger, Z. Manna, and A. Pnueli. What good are digital clocks? In W. Kuich, editor, *Proceedings of the 19th International Colloquium on Automata, Languages, and Programming (ICALP'92)*, volume 623 of *LNCS*, pages 545–558. Springer-Verlag, 1992.

[HRSV01] T. S. Hune, J. M. T. Romijn, M. I. A. Stoelinga, and F. W. Vaandrager. Linear parametric model checking of timed automata. In T. Margaria and W. Yi, editors, *Proceedings of the 7th Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS 2001)*, volume 2031 of *LNCS*, pages 189–203. Springer-Verlag, 2001.

[HSP83] S. Hart, M. Sharir, and A. Pnueli. Termination of probabilistic concurrent programs. *ACM Transactions on Programming Languages and Systems*, 5(3):356–380, 1983.

[KNP02] M. Z. Kwiatkowska, G. Norman, and D. Parker. PRISM: Probabilistic symbolic model checker. In T. Field, P. G. Harrison, J. Bradley, and U. Harder, editors, *Proceedings of the 12th International Conference on Computer Performance Evaluation, Modelling Techniques and Tools (TOOLS 2002)*, volume 2324 of *LNCS*, pages 200–204. Springer-Verlag, 2002.

[KNS02] M. Z. Kwiatkowska, G. Norman, and J. Sproston. Probabilistic model checking of the IEEE 802.11 wireless local area network protocol. In H. Hermanns and R. Segala, editors, *Proceedings of the Joint PAPM-PROBMIV 2002 Workshop*, volume 2399 of *LNCS*, pages 169–187. Springer-Verlag, 2002.

[KNSS02] M. Z. Kwiatkowska, G. Norman, R. Segala, and J. Sproston. Automatic verification of real-time systems with discrete probability distributions. *Theoretical Computer Science*, 282(1):101–150, 2002. A preliminary version of this paper appeared in Katoen [**?**], pages 75–95.

[Koz81] D. Kozen. Semantics of probabilistic programs. *Journal of Computer and System Sciences*, 22(3):328–350, 1981.

[KSK76] J. G. Kemeny, J. L. Snell, and A. W Knapp. *Denumerable Markov Chains*. Graduate Texts in Mathematics. Springer, 2nd edition, 1976.

[LPY97] K. G. Larsen, P. Pettersson, and W. Yi. UPPAAL in a nutshell. *Software Tools for Technology Transfer*, 1(1+2):134–152, 1997.

[LPY98] M. Lindahl, P. Pettersson, and W. Yi. Formal design and analysis of a gear-box controller. In Steffen [Ste98], pages 281–297.

[Pri] PRISM webpage: `http://www.cs.bham.ac.uk/~dxp/prism/`.

[SdS98] D. Sangiorgi and R. de Simone, editors. *Proceedings of the 9th International Conference on Concurrency Theory (CONCUR'98)*, volume 1466 of *LNCS*. Springer-Verlag, 1998.

[Seg95] R. Segala. *Modeling and Verification of Randomized Distributed Real-Time Systems*. PhD thesis, MIT, Dept. of Electrical Engineering and Computer Science, 1995.

[SL95] R. Segala and N. A. Lynch. Probabilistic simulations for probabilistic processes. *Nordic Journal of Computing*, 2(2):250–273, 1995.

[SS01] D. P. L. Simons and M. I. A. Stoelinga. Mechanical verification of the IEEE 1394a root contention protocol using UPPAAL2k. *Software Tools for Technology Transfer*, 3(4):469–485, 2001.

[Ste98] B. Steffen, editor. *Proceedings of the 4th Workshop on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'98)*, volume 1384 of *LNCS*. Springer-Verlag, 1998.

[SV99] M. I. A. Stoelinga and F.W. Vaandrager. Root contention in IEEE 1394. volume 1601 of *LNCS*, pages 53–74. Springer-Verlag, 1999.

[Var85] M. Y. Vardi. Automatic verification of probabilistic concurrent finite-state programs. In *Proceedings of the 26th IEEE Symposium on Foundations of Computer Science (FOCS'85)*, pages 327–338. IEEE Computer Society Press, 1985.

[YPD94] W. Yi, P. Pettersson, and M. Daniels. Automatic verification of real-time communicating systems by constraint-solving. In D. Hogrefe and S. Leue, editors, *Proceedings of the 7th International Conference on Formal Description Techniques (FORTE'94)*, pages 223–238. North–Holland, 1994.

A. Proof of Theorem 2.5

Before we can prove Theorem 2.5 we require the following definitions and lemmas. In the following, we suppose that the closed, diagonal-free probabilistic timed automaton $\mathsf{PTA} = (L, \mathcal{X}, \Sigma, inv, prob)$ is fixed. First, we introduce the relation $\succ \subseteq \mathbb{R}^{|\mathcal{X}|} \times \mathbb{N}^{|\mathcal{X}|}$ first presented in [Bey01], which associates with every real-valued clock valuation the set of its possible integer representatives. Recall that $\mathbf{k}_x$ denotes the greatest constant that the clock x is compared to in the zones of PTA.

Definition A.1. For $v \in \mathbb{R}^{|\mathcal{X}|}$ and $v' \in \mathbb{N}^{|\mathcal{X}|}$, $v \succ v'$ holds if and only if there exists $t \in [0, 1)$ such that for all $x \in \mathcal{X}$ one of the following conditions holds:

- $v'(x) - 1 + t < v(x) \leq v'(x) + t$
- $v'(x) - 1 + t < v(x)$ and $v'(x) = \mathbf{k}_x + 1$.

The following lemma is immediate.

Lemma A.2. For any zone ζ of PTA and clock valuations $v \in \mathbb{R}^{|\mathcal{X}|}$, $v' \in \mathbb{N}^{|\mathcal{X}|}$ such that $v \succ v'$, if $v \triangleleft \zeta$ then $v' \triangleleft \zeta$.

In particular, the lemma means that, for any state $(l, v) \in S_{\mathbb{R}}$ of the continuous semantics $[\![\mathsf{PTA}]\!]_{\mathbb{R}}$, then, for any $v' \in \mathbb{N}^{|\mathcal{X}|}$ such that $v \succ v'$, the pair (l, v') is a state of the integer semantics $[\![\mathsf{PTA}]\!]_{\mathbb{N}}$ (that is, $(l, v') \in S_{\mathbb{N}}$), because $v \triangleleft inv(l)$ implies $v' \triangleleft inv(l)$.

Next, for the probabilistic system $[\![\mathsf{PTA}]\!]_{\mathbb{T}} = (S, Act, Steps)$ and any adversary $C \in Adv_{[\![\mathsf{PTA}]\!]_{\mathbb{T}}}$, we define a sequence of functions $(\mathbf{P}^C_n)_{n \in \mathbb{N}}$. Intuitively, for any finite path $\omega \in Path^C_{fin}$ and set of locations $L' \subseteq L$, $\mathbf{P}^C_n(\omega, L')$ equals the probability, according to the adversary C, of reaching a state in $F^{\mathbb{T}}_{L'}$ in at most n transitions after performing the path ω. Formally we have the following definition.

Definition A.3. If $[\![\mathsf{PTA}]\!]_{\mathbb{T}} = (S, Act, Steps)$, then for any adversary $C \in Adv_{[\![\mathsf{PTA}]\!]_{\mathbb{T}}}$, $L' \subseteq L$ and finite path $\omega \in Path^C_{fin}$, let:

$$\mathbf{P}^C_0(\omega, L') = \begin{cases} 1 & \text{if } last(\omega) \in F^{\mathbb{T}}_{L'} \\ 0 & \text{otherwise,} \end{cases}$$

and for any $n \in \mathbb{N}$, $\omega \in Path^C_{fin}$, if $C(\omega) = (a, \mu)$ then:

$$\mathbf{P}^C_{n+1}(\omega, L') = \begin{cases} 1 & \text{if } last(\omega) \in F^{\mathbb{T}}_{L'} \\ \sum_{s' \in S} \mu(s') \cdot \mathbf{P}^C_n(\omega \xrightarrow{a,\mu} s', L') & \text{otherwise.} \end{cases}$$

We now give Lemma A.4, which follows from classical probabilistic reachability. We abuse notation by regarding a single state $s \in S$ as a path of length 0 (that is, a path without any transitions).

Lemma A.4. If $[\![\mathsf{PTA}]\!]_{\mathbb{T}} = (S, Act, Steps)$, then for any state $s \in S$ and $L' \subseteq L$, we have:

$$MaxProbReach_{[\![\mathsf{PTA}]\!]_{\mathbb{T}}}(s, F^{\mathbb{T}}_{L'}) \stackrel{def}{=} \sup_{C \in Adv_{[\![\mathsf{PTA}]\!]_{\mathbb{T}}}} \lim_{n \to \infty} \mathbf{P}^C_n(s, L')$$

$$MinProbReach_{[\![\mathsf{PTA}]\!]_{\mathbb{T}}}(s, F^{\mathbb{T}}_{L'}) \stackrel{def}{=} \inf_{C \in Adv_{[\![\mathsf{PTA}]\!]_{\mathbb{T}}}} \lim_{n \to \infty} \mathbf{P}^C_n(s, L').$$

Finally, we require the following lemma.

Lemma A.5. Let $\mathsf{PTA} = (L, \mathcal{X}, \Sigma, inv, prob)$ be a closed, diagonal-free probabilistic timed automaton, with continuous semantics $[\![\mathsf{PTA}]\!]_{\mathbb{R}} = (S_{\mathbb{R}}, \mathbb{R} \cup \Sigma, Steps_{\mathbb{R}})$ and integer semantics $[\![\mathsf{PTA}]\!]_{\mathbb{N}} = (S_{\mathbb{N}}, \mathbb{N} \cup \Sigma, Steps_{\mathbb{N}})$. For all $L' \subseteq L$, $n \in \mathbb{N}$, $(l, v) \in S_{\mathbb{R}}$, $A \in Adv_{[\![\mathsf{PTA}]\!]_{\mathbb{R}}}$ and $v' \in \mathbb{N}^{|\mathcal{X}|}$ such that $v \succ v'$, there exist adversaries $B_1, B_2 \in Adv_{[\![\mathsf{PTA}]\!]_{\mathbb{N}}}$ such that:

$$\mathbf{P}^{B_1}_n((l, v'), L') \leq \mathbf{P}^A_n((l, v), L') \leq \mathbf{P}^{B_2}_n((l, v'), L').$$

Proof. If $l \in L'$ then, by Definition A.3, for any adversary $B \in Adv_{[\![\mathsf{PTA}]\!]_{\mathbb{N}}}$ and $n \in \mathbb{N}$, we have $\mathbf{P}^A_n((l, v), L') = \mathbf{P}^B_n((l, v'), L') = 1$, and hence the lemma holds in this case.

We are left to consider the case when $l \notin L'$. We proceed by induction on $n \in \mathbb{N}$. For $n = 0$, the result follows from the fact that for any adversary $B \in Adv_{[\![\mathsf{PTA}]\!]_{\mathbb{N}}}$:

$$\mathbf{P}^A_0((l, v), L') = \mathbf{P}^B_0((l, v'), L') = 0.$$

Now suppose that the lemma holds for some $n \in \mathbb{N}$. By Definition 2.4, we have the following two cases to consider:

Time transitions. $A(l, v) = (t, \mu)$ for some $t \in \mathbb{R}$ such that $v + \tilde{t} \triangleleft inv(l)$ for all $0 \leq \tilde{t} \leq t$ and $\mu(l, v + t) = 1$.

In this case, for any $(l, v') \in S_{\mathbb{N}}$, if $v \succ v'$, then from Lemma 1 of [Bey01] there exists $t' \in \mathbb{N}$ such that:

$$v' \oplus_{\mathbb{N}} t' \triangleleft inv(l) \text{ and } v + t \succ v' \oplus_{\mathbb{N}} t'. \tag{1}$$

Let $A_{nZ} \in Adv_{[\![\mathsf{PTA}]\!]_{\mathbb{R}}}$ be an adversary such that $A_{nZ}(\omega) = A((l, v) \xrightarrow{t,\mu} \omega)$. Then, by Definition A.3 and the fact that $\mu(l, v + t) = 1$, we have $\mathbf{P}^A_{n+1}((l, v), L') = \mathbf{P}^{A_{nZ}}_n((l, v + t), L')$, and, by induction and (1), there exist $B'_1, B'_2 \in Adv_{[\![\mathsf{PTA}]\!]_{\mathbb{N}}}$ such that:

$$\mathbf{P}^{B'_1}_n((l, v' \oplus_{\mathbb{N}} t'), L') \leq \mathbf{P}^{A_{nZ}}_n((l, v + t), L') \leq \mathbf{P}^{B'_2}_n((l, v' \oplus_{\mathbb{N}} t'), L'). \tag{2}$$

Furthermore, by (1) and Definition 2.4, there exists $((l, v), t', \mu') \in Steps_{\mathbb{N}}$ with $\mu'(l, v' \oplus_{\mathbb{N}} t') = 1$. Therefore, letting B_1 and B_2 be the adversaries which choose the transition (t', μ') in (l, v') and then behave like B'_1 and B'_2 respectively, by Definition A.3 we have $\mathbf{P}^{B_1}_{n+1}((l, v), L') = \mathbf{P}^{B'_1}_n((l, v' \oplus_{\mathbb{N}} t'), L')$ and $\mathbf{P}^{B_2}_{n+1}((l, v'), L') = \mathbf{P}^{B'_2}_n((l, v' \oplus_{\mathbb{N}} t'), L')$. The required result then follows from (2).

Discrete transitions. $A(l, v) = (\sigma, \mu)$ for some $\sigma \in \Sigma$ such that there exists $(l, g, \sigma, p) \in prob$ with $v \triangleleft g$

and for all $(l', w) \in S_{\mathbb{R}}$:

$$\mu(l', w) = \sum_{X \subseteq \mathcal{X} \;\&\; w = v[X:=0]} p(X, l') \; . \tag{3}$$

In this case, for any $(l, v') \in S_{\mathbb{N}}$ such that $v \succ v'$, it follows from Lemma A.2 and the definition of $\succ$ that $v' \triangleleft g$. Now by Definition 2.4 there exists $((l, v'), \sigma, \mu') \in \mathit{Steps}_{\mathbb{N}}$ such that, for all $(l', w') \in S_{\mathbb{N}}$:

$$\mu'(l', w') = \sum_{X \subseteq \mathcal{X} \;\&\; w' = v'[X:=0]} p(X, l') \; . \tag{4}$$

Consider any $(l', X) \in L \times 2^{\mathcal{X}}$ such that $p(l', X) > 0$. Let $A^{l', v[X:=0]}$ be the adversary such that $A^{l', v[X:=0]}(\omega) = A((l, v) \xrightarrow{\sigma, \mu} \omega)$ for any finite path ω such that $\omega(0) = (l', v[X := 0])$. Then by Definition A.3 we have:

$$\begin{aligned}
\mathbf{P}^A_{n+1}((l, v), L') &= \sum_{(l', w) \in S_{\mathbb{R}}} \mu(l', w) \cdot \mathbf{P}^A_n((l, v) \xrightarrow{\sigma, \mu} (l', w), L') \\
&= \sum_{(l', w) \in S_{\mathbb{R}}} \left(\sum_{X \subseteq \mathcal{X} \;\&\; w = v[X:=0]} p(X, l') \right) \cdot \mathbf{P}^A_n((l, v') \xrightarrow{\sigma, \mu} (l', w), L') && \text{by (3)} \\
&= \sum_{l' \in L \;\&\; X \subseteq \mathcal{X}} p(X, l') \cdot \mathbf{P}^A_n((l, v) \xrightarrow{\sigma, \mu'} (l', v[X := 0]), L') && \text{rearranging} \\
&= \sum_{l' \in L \;\&\; X \subseteq \mathcal{X}} p(X, l') \cdot \mathbf{P}^{A^{l', v[X:=0]}}_n((l', v[X := 0]), L') && \text{by construction.}
\end{aligned}$$

Now, from Lemma 1 of [Bey01], since $v \succ v'$, for any $X \subseteq \mathcal{X}$ we have $v[X := 0] \succ v'[X := 0]$, and hence by induction there exists $B_1^{l', v[X:=0]} \in \mathit{Adv}_{[\![\mathsf{PTA}]\!]_{\mathbb{N}}}$ such that:

$$\mathbf{P}^{B_1^{l', v[X:=0]}}_n((l', v'[X := 0]), L') \leq \mathbf{P}^{A^{l', v[X:=0]}}_n((l', v[X := 0]), L'). \tag{5}$$

For any $X \subseteq \mathcal{X}$, let $B_1^{l', v'[X:=0]}$ be the adversary from the set

$$\{B_1^{l', v[X':=0]} \mid X' \subseteq \mathcal{X} \wedge v'[X := 0] = v'[X' := 0]\}$$

which minimizes the value of $\mathbf{P}^{B_1^{l', v[X:=0]}}_n((l', v'[X := 0]), L')$. Finally, letting B_1 be the adversary that chooses the transition (σ, μ') in (l, v') and then behaves like $B_1^{l', v'[X:=0]}$ in the state $(l', v'[X := 0])$, using Definition A.3 and (4) it follows that:

$$\begin{aligned}
\mathbf{P}^{B_1}_{n+1}((l, v'), L') &= \sum_{l' \in L \;\&\; X \subseteq \mathcal{X}} p(X, l') \cdot \mathbf{P}^{B_1^{l', v'[X:=0]}}_n((l', v'[X := 0]), L') \\
&\leq \sum_{l' \in L \;\&\; X \subseteq \mathcal{X}} p(X, l') \cdot \mathbf{P}^{B^{l', v[X:=0]}}_n((l', v'[X := 0]), L') && \text{by construction} \\
&\leq \sum_{l' \in L \;\&\; X \subseteq \mathcal{X}} p(X, l') \cdot \mathbf{P}^{A^{l', v[X:=0]}}_n((l', v[X := 0]), L') && \text{by (5)} \\
&= \mathbf{P}^A_{n+1}((l', v[X := 0]), L') && \text{from above.}
\end{aligned}$$

Showing that $\mathbf{P}^A_{n+1}((l, v), L') \leq \mathbf{P}^{B_2}_{n+1}((l, v'), L')$ for some $B_2 \in \mathit{Adv}_{[\![\mathsf{PTA}]\!]_{\mathbb{N}}}$ follows similarly. □

We now give the proof of Theorem 2.5, that is, for every closed, diagonal-free probabilistic timed automata $\mathsf{PTA} = (L, \mathcal{X}, \Sigma, \mathit{inv}, \mathit{prob})$, initial location $\bar{l} \in L$, and target locations $L' \subseteq L$:

$$\mathit{MaxProbReach}_{[\![\mathsf{PTA}]\!]_{\mathbb{R}}}((\bar{l}, \mathbf{0}), F^{L'}_{\mathbb{R}}) = \mathit{MaxProbReach}_{[\![\mathsf{PTA}]\!]_{\mathbb{N}}}((\bar{l}, \mathbf{0}), F^{L'}_{\mathbb{N}})$$
$$\mathit{MinProbReach}_{[\![\mathsf{PTA}]\!]_{\mathbb{R}}}((\bar{l}, \mathbf{0}), F^{L'}_{\mathbb{R}}) = \mathit{MinProbReach}_{[\![\mathsf{PTA}]\!]_{\mathbb{N}}}((\bar{l}, \mathbf{0}), F^{L'}_{\mathbb{N}}).$$

Proof of Theorem 2.5 Since $(\bar{l}, \mathbf{0}) \succ (\bar{l}, \mathbf{0})$, by Lemma A.5 and Lemma A.4 if follows that:

$$MaxProbReach_{[\![\mathsf{PTA}]\!]_{\mathbb{R}}}((\bar{l}, \mathbf{0}), F^{L'}_{\mathbb{R}}) \leq MaxProbReach_{[\![\mathsf{PTA}]\!]_{\mathbb{N}}}((\bar{l}, \mathbf{0}), F^{L'}_{\mathbb{N}})$$
$$MinProbReach_{[\![\mathsf{PTA}]\!]_{\mathbb{R}}}((\bar{l}, \mathbf{0}), F^{L'}_{\mathbb{R}}) \geq MinProbReach_{[\![\mathsf{PTA}]\!]_{\mathbb{N}}}((\bar{l}, \mathbf{0}), F^{L'}_{\mathbb{N}}).$$

On the other hand, by definition of the continuous and integer semantics of PTA for any adversary $B \in Adv_{[\![\mathsf{PTA}]\!]_{\mathbb{N}}}$ there exists an adversary $A \in Adv_{[\![\mathsf{PTA}]\!]_{\mathbb{R}}}$ such that:

$$\mathbf{P}^A_n((\bar{l}, \mathbf{0}), L') = \mathbf{P}^B_n((\bar{l}, \mathbf{0}), L')$$

for all $n \in \mathbb{N}$. Intuitively, the adversary A behaves like an integer semantics adversary; that is, it chooses to make timed transitions of only integer duration. Since this was for an arbitrary $B \in Adv_{[\![\mathsf{PTA}]\!]_{\mathbb{N}}}$, it follows that:

$$MaxProbReach_{[\![\mathsf{PTA}]\!]_{\mathbb{R}}}((\bar{l}, \mathbf{0}), F^{L'}_{\mathbb{R}}) \geq MaxProbReach_{[\![\mathsf{PTA}]\!]_{\mathbb{N}}}((\bar{l}, \mathbf{0}), F^{L'}_{\mathbb{N}})$$
$$MinProbReach_{[\![\mathsf{PTA}]\!]_{\mathbb{R}}}((\bar{l}, \mathbf{0}), F^{L'}_{\mathbb{R}}) \leq MinProbReach_{[\![\mathsf{PTA}]\!]_{\mathbb{N}}}((\bar{l}, \mathbf{0}), F^{L'}_{\mathbb{N}}).$$

Putting the above together we have

$$MaxProbReach_{[\![\mathsf{PTA}]\!]_{\mathbb{R}}}((\bar{l}, \mathbf{0}), F^{L'}_{\mathbb{R}}) = MaxProbReach_{[\![\mathsf{PTA}]\!]_{\mathbb{N}}}((\bar{l}, \mathbf{0}), F^{L'}_{\mathbb{N}})$$
$$MinProbReach_{[\![\mathsf{PTA}]\!]_{\mathbb{R}}}((\bar{l}, \mathbf{0}), F^{L'}_{\mathbb{R}}) = MinProbReach_{[\![\mathsf{PTA}]\!]_{\mathbb{N}}}((\bar{l}, \mathbf{0}), F^{L'}_{\mathbb{N}})$$

as required. □

B. Proof of Lemma 4.1

Proof. Consider the path formula $\Box\neg(\mathsf{done} \wedge y \leq \mathit{deadline})$. By the distributivity of conjunction, $\Box\neg(\mathsf{done} \wedge y \leq \mathit{deadline}) \equiv \Box((\neg\mathsf{done}) \vee y > \mathit{deadline})$. We now show that $\omega \models_{\mathcal{A}} \Box((\neg\mathsf{done}) \vee y > \mathit{deadline})$ if and only if $\omega \models_{\mathcal{A}} \Diamond\mathsf{deadline\_exceeded}$.

($\Rightarrow$) From $\omega \models_{\mathcal{A}} \Box((\neg\mathsf{done}) \vee y > \mathit{deadline})$, we have $(\neg\mathsf{done})$ continuously for all states corresponding to times in the interval $[0, \mathit{deadline}]$. Consider the final transition made from a state at a time in the interval $[0, \mathit{deadline}]$. Firstly, note that this transition exists, because the path is non-Zeno and y is never reset; secondly, observe that it must be a continuous transition, as discrete transitions are instantaneous. The location of both the source and target state of this transition must be other than done, because time transitions must have the same location in both their source and target state. From inspection of $\mathtt{I}_1^{\mathtt{p}+}$, we have $y \leq \mathit{deadline}$ in all locations apart from done and deadline_exceeded. Hence, the source and target location of the transition must be deadline_exceeded, and therefore $\Diamond\mathsf{deadline\_exceeded}$ holds for this path.

($\Leftarrow$) From inspection of $\mathtt{I}_1^{\mathtt{p}+}$, we have $\omega \models_{\mathcal{A}} \Diamond\mathsf{deadline\_exceeded}$ implies $\omega \models_{\mathcal{A}} \Box(\neg\mathsf{done})$, as it is impossible to reach deadline_exceeded from done, and vice versa. Then clearly $\Box(\neg\mathsf{done})$ implies $\Box((\neg\mathsf{done}) \vee y > \mathit{deadline})$. □

C. Proof of Lemma 4.3

Before giving the proof of Lemma 4.3, we present two auxiliary lemmas. The following lemma establishes that there does not exist a state of $\mathtt{I}_1^{\mathtt{p}+}$ from which it is impossible for some adversary to let time advance unboundedly with probability 1.

Lemma C.1. For all states $s \in S$ of $[\![\mathtt{I}_1^{\mathtt{p}+}]\!]_{\mathbb{R}}$, there exists an adversary A of $[\![\mathtt{I}_1^{\mathtt{p}+}]\!]_{\mathbb{R}}$ such that:

$$Prob^A\{\omega \in Path^A_{ful} \mid \omega(0) = s \ \ \& \ \ \omega \text{ is non-Zeno } \} = 1 \ .$$

Proof. We proceed by contradiction: assume that there exists a state of $[\![\mathtt{I}_1^{\mathtt{p}+}]\!]_{\mathbb{R}}$ such that there does not exist a "non-Zeno" adversary which meets the criteria given in the lemma. A necessary (but not sufficient) condition for the existence of such a state is that either (1) an invariant of a location imposes an upper bound on at least one clock which is greater than the upper bound on the same clock included in the

enabling conditions for all of the location's outgoing edges, or (2) a loop in the graph derived from $\mathtt{I}_1^{\mathtt{p}+}$ features locations whose invariants impose an upper bound on at least one clock which is not reset within the loop. Neither of these conditions is satisfied in $\mathtt{I}_1^{\mathtt{p}+}$, and therefore a state from which there does not exist a "non-Zeno" adversary is not possible. □

We now examine the Zeno paths of the probabilistic timed automaton $\mathtt{I}_1^{\mathtt{p}+}$. The following lemma expresses the fact that the location of $\mathtt{I}_1^{\mathtt{p}+}$ cannot change infinitely often in a finite amount of time.

Lemma C.2. Let ω be a Zeno path of $[\![\mathtt{I}_1^{\mathtt{p}+}]\!]_{\mathbb{R}}$. Then there exists an infinite suffix ω^{suf} of ω such that all of its states feature the same location.

Proof. Each loop of the graph given by the locations and edges of $\mathtt{I}_1^{\mathtt{p}+}$ features (1) an enabling condition of the form $x \geq c$ for some $c > 0$, and (2) a clock reset of x. Hence, if a path exhibits infinitely many edge traversals, then it must be non-Zeno. □

For any finite path ω and infinite path ω' such that $last(\omega) = \omega'(0)$ (i.e. the last state of ω equals the first state of ω'), let $\omega\omega'$ denote the infinite path obtained by *concatenating* ω and ω'. Moreover, for any infinite path ω, let $\omega^{(i)}$ denote the *ith prefix* of ω. We now proceed to the proof of the main lemma.

Proof of Lemma 4.3. Consider the following adversary A_{nZ} of $[\![\mathtt{I}_1^{\mathtt{p}+}]\!]_{\mathbb{R}}$, defined as behaving in the same manner as A on all finite prefixes of non-Zeno paths, and as behaving in a non-Zeno manner after some sufficiently long finite prefix of any Zeno path. Formally, let $j \in \mathbb{N}$ be some index such that the location component after the jth state of all Zeno paths of A is constant. Such an index exists by Lemma C.2. We then define the adversary A_{nZ} in the following manner:

- for any non-Zeno path ω of the adversary A, let $A_{nZ}(\omega^{(i)}) = A(\omega^{(i)})$ for all $i \in \mathbb{N}$;
- for any Zeno path ω of A, let $\tilde{A}$ be an adversary of $[\![\mathtt{I}_1^{\mathtt{p}+}]\!]_{\mathbb{R}}$ such that:

  $$Prob^{\tilde{A}}\{\tilde{\omega} \in Path_{ful}^{\tilde{A}} \mid \tilde{\omega}(0) = \omega(j) \ \ \& \ \ \tilde{\omega} \text{ is non-Zeno } \} = 1 \ .$$

  Such an adversary exists by Lemma C.1. Then let $A_{nZ}(\omega^{(i)}) = A(\omega^{(i)})$ for all $0 \leq i \leq j$, and let $A_{nZ}(\omega^{(j)}\tilde{\omega}) = \tilde{A}(\tilde{\omega})$ for all finite paths $\tilde{\omega}$ of $\tilde{A}$ such that $\tilde{\omega}(0) = \omega(j)$.

It follows that A_{nZ} is a non-Zeno adversary of $[\![\mathtt{I}_1^{\mathtt{p}+}]\!]_{\mathbb{R}}$.

The inequality of Lemma 4.3 holds for the following reasons. It suffices to consider the cones generated by all finite paths of A of length j. We consider two cases, depending on whether a cone corresponds to possibly many non-Zeno paths or to a single Zeno path.

Case: non-zeno paths. By the definition of A_{nZ}, the adversaries A and A_{nZ} agree along both the finite path generating the cone and along all of the non-Zeno paths included in the cone. Hence, the cone of A and the corresponding cone of A_{nZ} have the same probability of reaching deadline_exceeded.

Case: Zeno paths. Recall that, by Lemma C.2, after the jth state, the cone of A comprising a single infinite, Zeno path remains within the same location. We consider the following sub-cases depending on the identity of this location:

- **Sub-case: done.** As done is unreachable from deadline_exceeded, and vice versa, the probability assigned by the cones to reaching deadline_exceeded is 0 in both A and A_{nZ}.
- **Sub-case: deadline_exceeded.** The probability assigned by the cones to reaching deadline_exceeded is 1 in both A and A_{nZ}.
- **Sub-case: location $l \in L \setminus \{$done, deadline_exceeded$\}$.** The probability assigned to the property of reaching deadline_exceeded by the cone of A_{nZ} is at least the probability assigned to the same property by the cone of A, for the following reason. The finite path of A generates a cone comprising a single infinite path which remains in the current location, thus reaching deadline_exceeded with probability 0. Hence, the probability of reaching deadline_exceeded assigned by the cone of A_{nZ} is at least that for the cone of A (in fact, the location invariants, combined with the property of non-Zenoness of the cone of A_{nZ}, will ensure progress through the location of $\mathtt{I}_1^{\mathtt{p}+}$, in turn implying that the cone will feature probabilistic branching, and finally also implying that the probability of reaching deadline_exceeded is strictly greater than 0). □