Performance Analysis of Probabilistic Timed Automata using Digital Clocks [*]

Marta Kwiatkowska, Gethin Norman and David Parker
School of Computer Science, University of Birmingham, Edgbaston, Birmingham B15 2TT, United Kingdom ({mzk,gxn,dxp}@cs.bham.ac.uk)

Jeremy Sproston
Dipartimento di Informatica, Università di Torino, 10149 Torino, Italy (sproston@di.unito.it)

Abstract. Probabilistic timed automata, a variant of timed automata extended with discrete probability distributions, is a modelling formalism suitable for describing formally both nondeterministic and probabilistic aspects of real-time systems, and is amenable to model checking against probabilistic timed temporal logic properties. However, the previously developed verification algorithms either suffer from high complexity, give only approximate results, or are restricted to a limited class of properties. In the case of classical (non-probabilistic) timed automata it has been shown that for a large class of real-time verification problems correctness can be established using an integral model of time (digital clocks) as opposed to a dense model of time. Based on these results we address the question of under what conditions digital clocks are sufficient for the performance analysis of probabilistic timed automata and show that this reduction is possible for an important class of systems and properties including probabilistic reachability and expected reachability. We demonstrate the utility of this approach by applying the method to the performance analysis of three probabilistic real-time protocols: the dynamic configuration protocol for IPv4 link-local addresses, the IEEE 802.11 wireless local area network protocol and the IEEE 1394 FireWire root contention protocol.

Keywords: Probabilistic model checking, timed automata, digital clocks

1. Introduction

Network protocols increasingly rely on the use of randomness and timing delays, for example the exponential back-off in Ethernet and IEEE 802.11. Since these protocols execute in a distributed environment, it is important to also consider nondeterminism when modelling their behaviour. A natural model for systems that exhibit nondeterminism, probability and real time, called *probabilistic timed automata* – a probabilistic extension of timed automata [2] – has been proposed in [38]. In probabilistic timed automata, real-valued clocks measure the passage of time, and transitions can be probabilistic, that is, be expressed as a discrete probability distribution on the set of target states. In [38] model-checking algorithms for verifying the likelihood of certain temporal properties being satisfied by such system models

[*] Supported in part by the EPSRC grants GR/N22960, GR/S11107 and GR/S46727, FORWARD and MIUR-FIRB Perf.

are introduced. These model checking algorithms are either based on *region equivalence* [2], which results in prohibitively large state spaces for realistic systems, or on *forwards reachability*, which leads to approximate results [38, 20]. An alternative approach, based on *backwards reachability*, is given in [39, 42]; while this can be more efficient than the region equivalence approach and leads to exact results, the approach has been applied only to probabilistic temporal logics, and not to other classes of performance properties such as expected-time or expected-cost.

When modelling real-time systems there is a trade off between expressiveness and complexity. For example, a *dense* time model is more expressive than an *integral* time model. However, it is generally the case that an integral time model is easier to verify, since it leads to a finite-state system and allows one to apply the efficient symbolic methods developed for untimed systems. Henzinger et al. [30] study the question of when real-time properties can be verified using only integral durations (digital clocks), and show that such a reduction is possible for a large class of systems and properties, such as time-bounded invariance and response. Other related work includes [7], where it was observed that to perform reachability analysis of certain classes of timed automata one need only consider integer clock values, and [13, 14] which show that using BDDs and integral durations can lead to efficient methods for performing reachability analysis of timed automata. We also mention [25] and [45] which investigate the power of digital clocks.

The main contribution of this paper is to extend this direction of research to the domain of probabilistic timed automata by showing that digital clocks are sufficient for analysing a large class of probabilistic real-time systems and performance measures. The models that can be considered are those which can be represented by *closed*, *diagonal-free* probabilistic timed automata, intuitively automata whose clock constraints do not compare the values of clocks with one another or contain strict comparisons with constants. The performance measures include *probabilistic reachability* properties, which for example allow us to check the correctness of the following statements: ‘with probability 0.05 or less, the system aborts’ and ‘with probability 0.99 or greater, a data packet will be delivered within 5 time units’. Additionally, *expected reachability* properties can be verified using digital clocks, which enable us to validate statements such as: ‘the expected time until a data packet is delivered is at most 20ms’, ‘the expected number of packets sent before failure is at least 100’ and ‘the expected number of retransmissions before a packet is sent is at most 5’.

We then demonstrate the applicability of this approach on three case studies using the probabilistic symbolic model checker PRISM [35, 47] to perform the analysis. In each case study the interplay between real-time, nondeterminism and probabilistic behaviour is critical and each can be modelled naturally as a (closed, diagonal-free) probabilistic timed automaton. The first concerns the ZeroConf dynamic configuration protocol for IPv4 link-local addresses [18] (preliminary results concerning this case study can be found in [37]). The second extends the results of [40], and investigates the performance of the contention resolution protocol of the IEEE 802.11 wireless local area network standard [33]. In the third case study we consider the root contention protocol of the IEEE 1394 FireWire standard [32], previously studied using probabilistic timed automata in [41, 20]. In the latter two publications, the analysis was with respect to probabilistic reachability properties, whereas in this paper we study expected reachability properties.

Outline. The next section introduces preliminary concepts that we will use in the remainder of the paper. In Section 3 we introduce probabilistic timed automata, their semantics for both the dense and integral models of time, two corresponding performance measures (probabilistic and expected reachability) and model checking techniques to compute these measures. Section 4 shows that, for closed diagonal-free probabilistic timed automata, computation of these measures can be performed using digital clocks (integral semantics). In Section 5 we address the limitations of digital clocks for analysing probabilistic timed automata; that is, we identify a class of properties which cannot be verified with digital clocks. In Section 6 we present three probabilistic timed automata case studies and give some experimental results to compare the performance of the techniques described in this paper with alternative approaches from the literature. Finally, in Section 7, we conclude the paper.

2. Preliminaries

2.1. Probability and Measure Theory

We assume some familiarity with probability and measure theory, see e.g. [26]. Consider a set Ω. A *σ-field* on Ω, denoted $\mathcal{F}$, is a family of subsets of Ω that contains Ω, and is closed under complementation and countable union. The elements of a σ-field are called *measurable sets*, and $(\Omega, \mathcal{F})$ is called a *measurable space*.

Definition 1. *Let $(\Omega, \mathcal{F})$ be a measurable space. A function $P : \mathcal{F} \to [0, 1]$ is a* probability measure *on $(\Omega, \mathcal{F})$, and $(\Omega, \mathcal{F}, P)$ is a* probability space, *if P satisfies the following properties:*

1. *$P(\Omega) = 1$;*
2. *if $A_1, A_2, \ldots$ is a disjoint sequence of elements of $\mathcal{F}$, then $P(\cup_i A_i) = \sum_i P(A_i)$.*

The measure P is also referred to as a *probability distribution*. The set Ω is called the sample space, and the elements of $\mathcal{F}$ are called events.

Definition 2. *Let $(\Omega, \mathcal{F})$ and $(\Omega', \mathcal{F}')$ be two measurable spaces. A function $f : \Omega \to \Omega'$ is said to be a* measurable function *from $(\Omega, \mathcal{F})$ to $(\Omega', \mathcal{F}')$ if $f^{-1}(A') \in \mathcal{F}$ for all $A' \in \mathcal{F}'$.*

Theorem 3 ([16]). *Let $(\Omega, \mathcal{F})$ and $(\Omega', \mathcal{F}')$ be measurable spaces, and suppose that P is a measure on $(\Omega, \mathcal{F})$ and the function $T : \Omega \to \Omega'$ is measurable. If f is a real non-negative measurable function on $(\Omega', \mathcal{F}')$, then:*

$$\int_{\omega \in \Omega} f(T\omega)\, \mathrm{d}P = \int_{\omega' \in \Omega'} f(\omega')\, \mathrm{d}PT^{-1} .$$

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 $q \in Q$, let μ_q be the *point distribution* at q which assigns probability 1 to q.

2.2. Discrete-time Markov chains

We now introduce discrete-time Markov chains (DTMCs), a widely-studied stochastic process.

Definition 4. *A* DTMC *is a tuple $\mathcal{D} = (S, \bar{s}, \mathbf{P})$ where:*

– *S is a set of* states, *including the initial state $\bar{s}$;*
– *$\mathbf{P} : S \times S \to [0,1]$ is a* transition probability matrix, *such that for any $s \in S : \sum_{s' \in S} \mathbf{P}(s, s') = 1$.*

Each element $\mathbf{P}(s, s')$ of the transition probability matrix gives the probability of making a transition from state s to state s'. An execution of a system which is being modelled by a DTMC is represented by a *path*. Formally, a path ω is a non-empty finite or infinite sequence of states. In the case of a finite path $s_0 s_1 \cdots s_n$, we require $\mathbf{P}(s_i, s_{i+1}) > 0$ for all $0 \leqslant i < n$, whereas, for an infinite path $s_0 s_1 s_2 \cdots$, we require $\mathbf{P}(s_i, s_{i+1}) > 0$ for all $i \geqslant 0$. We denote by $\omega(i)$ the $(i{+}1)$th state of a path ω, $|\omega|$ the length of ω (number of transitions), and for a finite path ω, the last state by $last(\omega)$. Observe that a path can comprise a single state, in which case its number of transitions is zero. We say that a finite path ω of length n is a *prefix* of the infinite path ω' if $\omega(i) = \omega'(i)$ for $0 \leqslant i \leqslant n$. Also, we use $\omega^{(k)}$ to denote the prefix of length k of ω. The sets of all finite and infinite paths starting in state s are denoted $Path_{\mathit{fin}}(s)$ and $Path_{\mathit{ful}}(s)$, respectively.

In order to reason about the probabilistic behaviour of the DTMC, we need to be able to determine the probability that certain paths are taken. This is achieved by defining, for each state $s \in S$, a probability measure $Prob_s$ over $Path_{\mathit{ful}}(s)$. Below, we give an outline of this construction. For further details, see [34]. The probability measure is induced by the transition probability matrix $\mathbf{P}$ as follows. First, for any finite path $\omega \in \mathit{Path}_{\mathit{fin}}(s)$ of length n, we define the probability $\mathbf{P}_s(\omega)$:

$$\mathbf{P}_s(\omega) \stackrel{\text{def}}{=} \begin{cases} 1 & \text{if } n = 0 \\ \mathbf{P}(\omega(0), \omega(1)) \cdots \mathbf{P}(\omega(n-1), \omega(n)) & \text{otherwise}. \end{cases}$$

Next, we define the *cylinder set* $\mathit{cyl}(\omega)$ as:

$$\mathit{cyl}(\omega) \stackrel{\text{def}}{=} \{\omega' \in \mathit{Path}_{\mathit{ful}}(s) \,|\, \omega \text{ is a prefix of } \omega'\},$$

that is, the cylinder set $\mathit{cyl}(\omega)$ is the set of all infinite paths with prefix ω. Then let Σ_s be the smallest σ-algebra on $\mathit{Path}_{\mathit{ful}}(s)$ which contains all the sets $\mathit{cyl}(\omega)$, where ω ranges over paths in $\mathit{Path}_{\mathit{fin}}(s)$. We define the probability measure Prob_s on Σ_s as the unique measure such that: $\mathit{Prob}_s(\mathit{cyl}(\omega)) = \mathbf{P}_s(\omega)$ for all $\omega \in \mathit{Path}_{\mathit{fin}}(s)$.

2.3. Timed probabilistic systems

We now introduce *timed probabilistic systems* which extend DTMCs by allowing both non-deterministic and probabilistic behaviour and in which transitions are labelled with either a duration taken from a time domain or an action. Timed probabilistic systems are an extension of Markov decision processes [24] and a variant of Segala's probabilistic timed automata [50].

Definition 5. *A* timed probabilistic system *is a tuple* $\mathsf{TPS} = (S, \bar{s}, \mathcal{A}ct, \mathbb{T}, \mathit{Steps})$ *where:*

– S *is a set of* states*, including an* initial state $\bar{s} \in S$*;*
– $\mathcal{A}ct$ *is a finite set of* actions *such that* $\mathcal{A}ct \cap \mathbb{R} = \emptyset$*;*
– $\mathbb{T} \subseteq \mathbb{R}$ *is a set of* durations*, taken from the set of non-negative reals;*
– $\mathit{Steps} \subseteq S \times (\mathcal{A}ct \cup \mathbb{T}) \times \mathsf{Dist}(S)$ *is a* probabilistic transition relation, *such that, if* $(s, a, \mu) \in \mathit{Steps}$ *and* $a \in \mathbb{T}$*, then* μ *is a point distribution.*

A *probabilistic transition* $s \xrightarrow{a,\mu} s'$ is made from a state $s \in S$ by first nondeterministically selecting an action-distribution or duration-distribution pair (a, μ) such that $(s, a, \mu) \in \mathit{Steps}$, and second by making a probabilistic choice of target state s' according to the distribution μ, such that $\mu(s') > 0$. We require that only action-distributions can be probabilistic, that is, duration-distribution pairs always comprise a point distribution.

We consider two ways in which a timed probabilistic system's computation may be represented: using paths and adversaries. A *path* represents a particular resolution of both nondeterminism *and* probability. Formally, a path of a timed probabilistic system is a non-empty finite or infinite sequence of probabilistic transitions

$$\omega = s_0 \xrightarrow{a_0,\mu_0} s_1 \xrightarrow{a_1,\mu_1} s_2 \xrightarrow{a_2,\mu_2} \cdots .$$

We denote by $\omega(i)$ the $(i{+}1)$th state of ω, $last(\omega)$ the last state of ω if ω is finite and $step(\omega, i)$ the action or duration associated with the $(i{+}1)$-th transition (that is, $step(\omega, i) = a_i$). By abuse of notation, we say that a single state s is a path of length 0. The set of finite and infinite paths starting in the state s is denoted by $Path_{fin}(s)$ and $Path_{ful}(s)$, respectively. For any infinite path $\omega = s_0 \xrightarrow{a_0,\mu_0} s_1 \xrightarrow{a_1,\mu_1} \cdots$, the accumulated duration up to the $(n{+}1)$-th state of ω is defined by:

$$dur(\omega, n{+}1) \overset{\text{def}}{=} \sum\{\!|\, a_i \mid 0 \leqslant i \leqslant n \wedge a_i \in \mathbb{T} \,|\!\} \,.$$

In contrast to a path, an *adversary* (or scheduler) represents a particular resolution of nondeterminism *only*. More precisely, an adversary is a function which chooses an outgoing distribution in the last state of a path. Formally, we have the following definition.

Definition 6. *Let* $\mathsf{TPS} = (S, \bar{s}, \mathcal{A}ct, \mathbb{T}, Steps)$ *be a timed probabilistic system. An adversary* A *of* TPS *is a function mapping every finite path* ω *of* TPS *to a pair* (a, μ) *such that* $(last(\omega), a, \mu)$ *is an element of* $Steps$. *For any state* $s \in S$, *let* $Path^{A}_{fin}(s)$ *and* $Path^{A}_{ful}(s)$ *denote the subsets of* $Path_{fin}(s)$ *and* $Path_{ful}(s)$ *which correspond to* A.

The behaviour of a timed probabilistic system $\mathsf{TPS} = (S, \bar{s}, \mathcal{A}ct, \mathbb{T}, Steps)$ under a given adversary A is purely probabilistic. More precisely, for a state $s \in S$, the behaviour from state s can be described by the infinite-state DTMC $\mathcal{D}^A_s = (S^A_s, s, \mathbf{P}^A_s)$, where:

- $S^A_s = Path^A_{fin}(s)$;
- for any two finite paths $\omega, \omega' \in S^A_s$:

$$\mathbf{P}^A_s(\omega, \omega') = \begin{cases} \mu(s') & \text{if } \omega' \text{ is of the form } \omega \xrightarrow{a,\mu} s' \text{ and } A(\omega) = (a, \mu) \\ 0 & \text{otherwise.} \end{cases}$$

There is a one-to-one correspondence between the paths of $\mathcal{D}^A$ and the set of paths $Path^A_{ful}(s)$ in the timed probabilistic system. Hence, using the probability measure over DTMCs given in Section 2.2 we can define a probability measure $Prob^A_s$ over the set of paths $Path^A_{ful}(s)$.

To simplify proofs, we also use *randomized adversaries*, where a randomized adversary B is a function B mapping every finite path ω to a distribution over $\{(a, \mu) \mid (last(\pi), a, \mu) \in Steps\}$. Similarly to the above, we can associate with any randomized adversary a probability measure over the set of paths of the adversary (see, for example, [22, 50]).

We restrict our attention to *time-divergent adversaries*, a common restriction imposed in real-time systems so that unrealisable behaviour (i.e. corresponding to time not advancing beyond a time bound) is disregarded during analysis. We say that an infinite path ω is *divergent* if for any $t \in \mathbb{R}$, there exists $j \in \mathbb{N}$ such that $dur(\omega, j) > t$. The following lemma shows that for any adversary A and state s of a timed probabilistic system the set of divergent paths with initial state s is measurable under any adversary identified in the previous paragraph.

Lemma 7. *Let* $\mathsf{TPS} = (S, \bar{s}, \mathcal{A}ct, \mathbb{T}, Steps)$ *be a timed probabilistic system. For any adversary A of* TPS *and $s \in S$ the set of divergent paths of A is measurable.*

Proof. For any $n, m \in \mathbb{N}$, let $C_n^m = \{\omega \in Path_{ful}^A(s) \mid dur(\omega, m) > n\}$, then

$$\{\omega \in Path_{ful}^A(s) \mid \omega \text{ is divergent }\} = \bigcap_{n \in \mathbb{N}} \left(\bigcup_{m \in \mathbb{N}} C_n^m \right)$$

and hence measurable. □

Definition 8. *An adversary A for a timed probabilistic system* TPS *is* divergent *if and only if, for each state s of* TPS*, the probability $Prob_s^A$ assigned to the divergent paths of $Path_{ful}^A(s)$ is 1. Furthermore, let Adv_{TPS} be the set of divergent adversaries of* TPS*.*

For motivation on why we consider such *probabilistic* divergent adversaries, as opposed to a stronger notion in which an adversary is divergent if and only if all its paths are divergent, see [38].

3. Probabilistic Timed Automata

In this section we review the definition of probabilistic timed automata [38], a modelling framework for timed probabilistic systems. The formalism is derived from classical timed automata [1, 2] extended with discrete probability distributions over edges.

3.1. Time, clocks and clock constraints

Let $\mathbb{T} \in \{\mathbb{R}, \mathbb{N}\}$ be the *time domain* of either the non-negative reals or naturals. Let $\mathcal{X}$ be a finite set of variables called *clocks* which take values from the time domain $\mathbb{T}$. A function $v \in \mathbb{T}^{\mathcal{X}}$ is referred to as a *clock valuation*. Let $\mathbf{0} \in \mathbb{T}^{\mathcal{X}}$ be the clock valuation which assigns 0 to all clocks in $\mathcal{X}$. For any $v \in \mathbb{T}^{\mathcal{X}}$ and $t \in \mathbb{T}$, the clock valuation $v \oplus t$ denotes the *time increment* for v with t (we present two alternatives for $\oplus$ in Section 3.3; for the time domain $\mathbb{R}$ it is standard addition $+$). We use $v[X{:=}0]$ to denote the clock valuation obtained from v by resetting all of the clocks in $X \subseteq \mathcal{X}$ to 0, and leaving the values of all other clocks unchanged.

Let $CC(\mathcal{X})$ be the set of *clock constraints* over $\mathcal{X}$, which are conjunctions of atomic constraints of the form $x \sim c$ for $x \in \mathcal{X}$, $\sim \in \{\leqslant, =, \geqslant\}$, and $c \in \mathbb{N}$. The clock valuation v *satisfies* the zone ζ, written $v \triangleleft \zeta$, if and only if ζ resolves to true after substituting each clock $x \in \mathcal{X}$ with the corresponding clock value from v. Readers familiar with timed automata will note that we consider the syntax of *closed, diagonal-free zones*, which *do not* feature atomic constraints of the form $x > c$ or $x < c$ (closed) or $x - y \sim c$ (diagonal free).

3.2. Syntax of probabilistic timed automata

We now introduce formally probabilistic timed automata. Observe that we extend the original definition of [38] with urgent events, a well-established concept for classical timed automata [29, 21].

Definition 9. *A* probabilistic timed automaton PTA *is a tuple of the form* $(L, \bar{l}, \mathcal{X}, \Sigma, inv, prob)$ *where:*

- L *is a finite set of* locations*;*
- $\bar{l} \in L$ *is the* initial location*;*
- $\mathcal{X}$ *is a finite set of* clocks*;*
- Σ *is a finite set of* events*, of which* $\Sigma_u \subseteq \Sigma$ *are declared as being* urgent*;*
- *the function* $inv : L \rightarrow CC(\mathcal{X})$ *is the* invariant condition*;*
- *the finite set* $prob \subseteq L \times CC(\mathcal{X}) \times \Sigma \times \mathsf{Dist}(2^{\mathcal{X}} \times L)$ *is the* probabilistic edge relation*.*

Note that we often refer to the model presented above as *closed, diagonal-free probabilistic timed automata*, in order to distinguish the clock constraints used with those in previous work [38].

A *state* of a probabilistic timed automaton is a pair (l, v) where $l \in L$ and $v \in \mathbb{T}^{\mathcal{X}}$ are such that $v \triangleleft inv(l)$. Informally, the behaviour of a probabilistic timed automaton can be understood as follows. The model starts in the initial location $\bar{l}$ with all clocks set to 0, that is, in the state $(\bar{l}, \mathbf{0})$. In this, and any other state (l, v), there is a nondeterministic choice of either (1) making a *discrete transition* or (2) letting *time pass*. In case (1), a discrete transition can be made according to any probabilistic edge $(l, g, \sigma, p) \in prob$ with source location l which is *enabled*; that is, the zone g is satisfied by the current clock valuation v. Then the probability of moving to the location l' and resetting all of the clocks in X to 0 is given by $p(X, l')$. In case (2), the option of letting time pass is available only if the invariant condition $inv(l)$ is satisfied while time elapses and there does not exist an enabled probabilistic edge with an urgent event.

Note that a *timed automaton* [2] is a probabilistic timed automaton for which every probabilistic edge (l, g, σ, p) is such that $p = \mu_{(X,l')}$ (the point distribution assigning probability 1 to (X, l')) for some $(X, l') \in 2^{\mathcal{X}} \times L$.

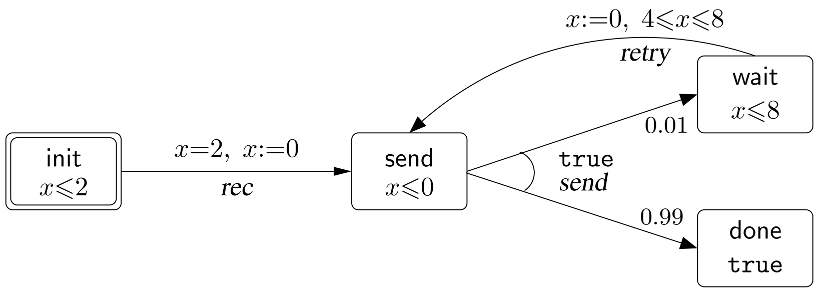


Figure 1. A probabilistic timed automaton modelling a simple communication protocol.

Example 10. *Consider the probabilistic timed automaton modelling a simple probabilistic communication protocol given in Figure 1. The protocol starts in location* init *(the double border indicates* init *is the initial location). After exactly 2 time units, the process receives data to send and moves to the location* send. *Before any time elapses the process attempts to send the data and, with probability 0.99, the data is sent correctly (location* done *is reached) while, with probability 0.01, the data is sent incorrectly (location* wait *is reached). In the latter case, the process waits between 4 and 8 time units before attempting to resend the data (returns to location* send*).*

In Figure 2 we present an example of the behaviour for the timed probabilistic system which underlies the probabilistic timed automata of Figure 1.

Higher-level modelling. To aid higher-level modelling, it is often useful to define complex systems as the *parallel composition* of a number of interacting sub-components. The definition of the parallel composition operator $\|$ of probabilistic timed automata uses ideas from the theory of (untimed) probabilistic systems [51] and classical timed automata [2]. Let $\mathsf{PTA}_i = (L_i, \bar{l}_i, \mathcal{X}_i, \Sigma_i, inv_i, prob_i)$ for $i \in \{1, 2\}$ and assume that $\mathcal{X}_1 \cap \mathcal{X}_2 = \emptyset$.

Definition 11. *The* parallel composition *of two probabilistic timed automata* PTA_1 *and* PTA_2 *is the probabilistic timed automaton*

$$\mathsf{PTA}_1 \| \mathsf{PTA}_2 = (L_1 \times L_2, (\bar{l}_1, \bar{l}_2), \mathcal{X}_1 \cup \mathcal{X}_2, \Sigma_1 \cup \Sigma_2, inv, prob)$$

such that

– $\sigma \in \Sigma_1 \cup \Sigma_2$ *is declared as urgent if and only if it is declared urgent in at least one of* PTA_1 *and* PTA_2*;*
– $inv(l, l') = inv_1(l) \wedge inv_2(l')$ *for all* $(l, l') \in L_1 \times L_2$*;*
– $((l_1, l_2), g, \sigma, p) \in prob$ *if and only if one of the following conditions holds:*

1. $\sigma \in \Sigma_1 \setminus \Sigma_2$ *and there exists* $(l_1, g, \sigma, p_1) \in prob_1$ *such that* $p = p_1 \otimes \mu_{(\emptyset, l_2)}$;

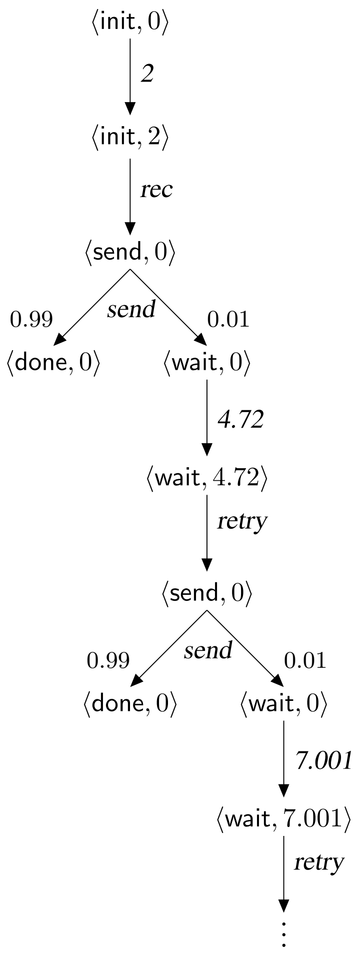


Figure 2. An example of behaviour for the simple communication protocol given in Figure 1.

2. $\sigma \in \Sigma_2 \setminus \Sigma_1$ *and there exists* $(l_2, g, \sigma, p_2) \in prob_2$ *such that* $p = \mu_{(\emptyset,l_1)} \otimes p_2$;
3. $\sigma \in \Sigma_1 \cap \Sigma_2$ *and there exists* $(l_i, g_i, \sigma, p_i) \in prob_i$ *for* $i = 1, 2$ *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* $X_2 \subseteq \mathcal{X}_2$*:*

$$p_1 \otimes p_2(X_1 \cup X_2, (l_1, l_2)) = p_1(X_1, l_1) \cdot p_2(X_2, l_2)\,.$$

In addition to parallel composition, features which are present in the UPPAAL input syntax [48, 44], such as urgent locations, committed locations and integer variables can be added to the probabilistic timed automaton framework.

3.3. Semantics of probabilistic timed automata

We now give the semantics of probabilistic timed automata defined in terms of timed probabilistic systems. Observe that the definition is parameterized by both the time domain $\mathbb{T}$ and the time increment operator $\oplus$. The time increment operator is a binary operator which takes a clock valuation $v \in \mathbb{T}^{\mathcal{X}}$ and a time duration $t \in \mathbb{T}$, and returns a clock valuation $v \oplus t \in \mathbb{T}^{\mathcal{X}}$ which represents, intuitively, the clock valuation obtained from v after t time units have elapsed.

Definition 12. *Let* $\mathsf{PTA} = (L, \bar{l}, \mathcal{X}, \Sigma, inv, prob)$ *be a probabilistic timed automaton. The* semantics of PTA with respect to the time domain $\mathbb{T}$ and time increment $\oplus$ *is the timed probabilistic system* $[\![\mathsf{PTA}]\!]_{\mathbb{T}}^{\oplus} = (S, \bar{s}, \Sigma, \mathbb{T}, Steps)$ *such that:*

– $S \subseteq L \times \mathbb{T}^{\mathcal{X}}$ *where* $(l, v) \in S$ *if and only if* $v \triangleleft inv(l)$*;*
– $\bar{s} = (\bar{l}, \mathbf{0})$*;*
– $((l, v), a, \mu) \in Steps$ *if and only if one of the following conditions holds:*

Time transitions. $a = t \in \mathbb{T}$ *and* $\mu = \mu_{(l, v \oplus t)}$ *such that:*

1. $v \oplus t' \triangleleft inv(l)$ *for all* $0 \leqslant t' \leqslant t$*;*
2. *for all probabilistic edges of the form* $(l, g, \sigma, -) \in prob$*, if* $v \oplus t' \triangleleft g$ *for some* $0 \leqslant t' \leqslant t$*, then* $\sigma \notin \Sigma_u$ *(no urgent transitions are enabled);*

Discrete transitions. $a = \sigma \in \Sigma$ *and there exists* $(l, g, \sigma, p) \in prob$ *such that* $v \triangleleft g$ *and for any* $(l', v') \in S$*:*

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

The summation in the definition of discrete transitions is required for the cases in which multiple clock resets result in the same target state.

In our setting, the semantics falls into two classes, depending on whether the underlying model of time is the positive reals or the naturals. If $\mathbb{T} = \mathbb{R}$ we let $\oplus$ equal $+$ (standard addition) and refer to $[\![\mathsf{PTA}]\!]_{\mathbb{R}}^{+}$ as the *dense-time semantics* of the probabilistic timed automaton PTA. In contrast, if $\mathbb{T} = \mathbb{N}$, we let $\oplus$ equal $\oplus_{\mathbb{N}}$ which is defined below, and refer to $[\![\mathsf{PTA}]\!]_{\mathbb{N}}^{\oplus_{\mathbb{N}}}$ as the *integral semantics* of PTA. To define $\oplus_{\mathbb{N}}$, first, for any $x \in \mathcal{X}$, let $\mathbf{k}_x$ denote the greatest constant that the clock x is compared to in the clock constraints of PTA. If the value of the clock x exceeds $\mathbf{k}_x$, then its exact value is not relevant when deciding which probabilistic edges are enabled. This means that $\mathbf{k}_x + 1$ is the maximum value of clock x that needs to be represented, because we can interpret this value as corresponding to all clock values greater than $\mathbf{k}_x$, and leads us to the following definition of $\oplus_{\mathbb{N}}$. 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).

Note that the definition of integral semantics for probabilistic timed automata is a generalization of the analogous definition for the classical model in [13]. As we always use the same type of time increment for a particular choice of time domain, we henceforth omit the $+$ and $\oplus_{\mathbb{N}}$ superscripts from the notation, and write $[\![\mathsf{PTA}]\!]_{\mathbb{R}}$ and $[\![\mathsf{PTA}]\!]_{\mathbb{N}}$, respectively. The fact that the integral semantics of a probabilistic timed automaton is finite, and the dense-time semantics of probabilistic timed automaton is generally uncountable, can be derived from the definitions.

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 timed probabilistic systems. Formally, we overload the parallel composition operator $\|$ such that $\mathsf{TPS}_1 \| \mathsf{TPS}_2$ denotes the timed probabilistic system obtained from the parallel composition of the timed probabilistic systems TPS_1 and TPS_2 in the standard manner [51] and defined below.

Definition 13. *The* parallel composition *of two timed probabilistic systems* TPS_1 *and* TPS_2 *with the same domain* $\mathbb{T}$ *is the timed probabilistic system*

$$\mathsf{TPS}_1 \| \mathsf{TPS}_2 = (S_1 \times S_2, (\bar{s}_1, \bar{s}_2), \mathcal{A}ct_1 \cup \mathcal{A}ct_2, \mathbb{T}, Steps)$$

such that $((s_1, s_2), a, \mu) \in Steps$ *if and only if one of the following conditions holds:*

1. $a \in \mathcal{A}ct_1 \setminus \mathcal{A}ct_2$ *and there exists* $(s_1, a, \mu_1) \in Steps_1$ *such that* $\mu = \mu_1 \otimes \mu_{s_2}$;
2. $a \in \mathcal{A}ct_2 \setminus \mathcal{A}ct_1$ *and there exists* $(s_2, a, \mu_2) \in Steps_2$ *such that* $\mu = \mu_{s_1} \otimes \mu_2$;
3. $a \in \mathcal{A}ct_1 \cap \mathcal{A}ct_2$ *and there exists* $(s_i, a, \mu_i) \in Steps_i$ *for* $i = 1, 2$ *such that* $\mu = \mu_1 \otimes \mu_2$;
4. $a \in \mathbb{T}$ *and there exists* $(s_i, a, \mu_i) \in Steps_i$ *for* $i = 1, 2$ *such that* $\mu = \mu_1 \otimes \mu_2$;

Two timed probabilistic systems $\mathsf{TPS}_1 = (S_1, \bar{s}_1, \mathcal{A}ct, \mathbb{T}, Steps_1)$ and $\mathsf{TPS}_2 = (S_2, \bar{s}_2, \mathcal{A}ct, \mathbb{T}, Steps_2)$ 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 integral and dense-time semantics.

3.4. Probabilistic and Expected Reachability

In this section, we consider two performance measures for probabilistic timed automata. In fact these measures are defined at the level of timed probabilistic systems, in terms of which the semantics of probabilistic timed automata are defined. The first measure is *probabilistic reachability*, namely the maximal and minimal probability of reaching, from the initial state, a certain set of target states. For a timed probabilistic system $\mathsf{TPS} = (S, \bar{s}, \mathcal{A}ct, \mathbb{T}, \mathit{Steps})$, set $F \subseteq S$ of target states, and adversary $A \in \mathit{Adv}_{\mathsf{TPS}}$, let:

$$p^A_{\bar{s}}(F) \overset{\text{def}}{=} \mathit{Prob}^A_{\bar{s}}\{\omega \in \mathit{Path}^A_{\mathit{ful}}(\bar{s}) \mid \exists i \in \mathbb{N} \,.\, \omega(i) \in F\} \;.$$

Definition 14. *The* maximal and minimal reachability probabilities *of reaching the set of states F of the timed probabilistic system* TPS *are defined as follows:*

$$p^{\max}_{\mathsf{TPS}}(F) = \sup_{A \in \mathit{Adv}_{\mathsf{TPS}}} p^A_{\bar{s}}(F) \qquad \textit{and} \qquad p^{\min}_{\mathsf{TPS}}(F) = \inf_{A \in \mathit{Adv}_{\mathsf{TPS}}} p^A_{\bar{s}}(F) \,.$$

The second measure we consider is *expected reachability*, which allows us to compute the expected cost (or reward) accumulated before reaching a certain set of states. Expected reachability is defined with respect to a set $F \subseteq S$ of target states and a cost function mapping state-action and state-duration pairs to real values (the cost of performing an action or letting a certain amount of time pass in the corresponding state, respectively). This measure corresponds to the expected cost (with respect to the given cost function) of reaching a state in F. More formally, for a timed probabilistic system $\mathsf{TPS} = (S, \bar{s}, \mathcal{A}ct, \mathbb{T}, \mathit{Steps})$, cost function $\mathcal{C} : S \times (\mathcal{A}ct \cup \mathbb{T}) \to \mathbb{R}$ (recall, $\mathbb{R}$ denotes the non-negative reals), set $F \subseteq S$ of target states, and adversary $A \in \mathit{Adv}_{\mathsf{TPS}}$, let $e^A_{\bar{s}}(\mathit{cost}(\mathcal{C}, F))$ denote the usual expectation of the function $\mathit{cost}(\mathcal{C}, F)$ (which returns, for a given path ω, the total cost accumulated until a state in F is reached along ω) with respect to the measure $\mathit{Prob}^A_{\bar{s}}$ over $\mathit{Path}^A_{\mathit{ful}}(\bar{s})$. That is:

$$e^A_{\bar{s}}(\mathit{cost}(\mathcal{C}, F)) = \int_{\omega \in \mathit{Path}^A_{\mathit{ful}}(\bar{s})} \mathit{cost}(\mathcal{C}, F)(\omega)\, d\mathit{Prob}^A_{\bar{s}}$$

where for any $\omega \in \mathit{Path}^A_{\mathit{ful}}(\bar{s})$:

$$\mathit{cost}(\mathcal{C}, F)(\omega) \overset{\text{def}}{=} \sum_{i=1}^{\min\{j \,|\, \omega(j) \in F\}} \mathcal{C}(\omega(i{-}1), \mathit{step}(\omega, i{-}1))$$

if there exists $j \in \mathbb{N}$ such that $\omega(j) \in F$, and $cost(\mathcal{C}, F)(\omega) \overset{\text{def}}{=} \infty$ otherwise.

Note that, for simplicity, we define the cost of a path which does not reach F to be ∞, even though the total cost of the path may not be infinite. Hence, the expected cost of reaching F from s is finite if and only if a state in F is reached from s with probability 1. *Expected time reachability* (the expected time with which a given set of states can be reached) is a special case of expected reachability, corresponding to the case when $\mathcal{C}(s, a) = 0$ for all $s \in S$ and $a \in \mathcal{A}ct$ and $\mathcal{C}(s, t) = t$ for all $s \in S$ and $t \in \mathbb{T}$.

Definition 15. *The* maximal and minimal expected costs *of reaching a set of states F under the cost function $\mathcal{C}$ in the timed probabilistic system* TPS *are defined as follows:*

$$\begin{aligned} e^{\max}_{\mathsf{TPS}}(\mathcal{C}, F) &= \sup_{A \in Adv_{\mathsf{TPS}}} e^A_{\bar{s}}(cost(\mathcal{C}, F)) \\ e^{\min}_{\mathsf{TPS}}(\mathcal{C}, F) &= \inf_{A \in Adv_{\mathsf{TPS}}} e^A_{\bar{s}}(cost(\mathcal{C}, F)) \,. \end{aligned}$$

We note that calculating expected reachability is equivalent to the *stochastic shortest path problem* for Markov decision processes; see for example [12, 23].

In practice, cost functions are defined not at the level of timed probabilistic systems, but in terms of probabilistic timed automata. At this level cost functions are often defined using a pair $(c_\Sigma, \mathbf{r})$, where $c_\Sigma : L \times \Sigma \to \mathbb{R}$ is a function assigning the cost, in each location, of executing each event in Σ, and $\mathbf{r} \in \mathbb{R}$ gives the rate at which cost is accumulated as time passes (independent of the current location). The associated cost function $\mathcal{C}_{c_\Sigma,\mathbf{r}}$ is defined as follows, for each $(l, v) \in L \times \mathbb{R}^{\mathcal{X}}$ and $a \in \Sigma \cup \mathbb{T}$:

$$\mathcal{C}_{c_\Sigma,\mathbf{r}}((l, v), a) \overset{\text{def}}{=} \begin{cases} c_\Sigma(l, a) & \text{if } a \in \Sigma \\ a \cdot \mathbf{r} & \text{otherwise.} \end{cases}$$

A probabilistic timed automaton equipped with a pair $(c_\Sigma, \mathbf{r})$ is a probabilistic generalisation of uniformly priced timed automata [9]. In Section 4.3 we will restrict attention to such cost functions, while in Section 4.4 we will consider more general cost functions where the cost per unit of time can vary depending on the current location, which can be considered as a probabilistic extension of linearly priced timed automata [10]. More precisely, cost functions of the form $\mathcal{C}_{c_\Sigma,r}$ where $r : L \to \mathbb{R}$ is a function assigning to each location the rate at which costs are accumulated as time passes in that location and for any $(l, v) \in L \times \mathbb{R}^{\mathcal{X}}$ and $a \in \Sigma \cup \mathbb{R}$:

$$\mathcal{C}_{c_\Sigma,r}((l, v), a) \overset{\text{def}}{=} \begin{cases} c_\Sigma(l, a) & \text{if } a \in \Sigma \\ a \cdot r(l) & \text{otherwise.} \end{cases}$$

Note that we only consider non-negative cost functions ($\mathbb{R}$ is the set of non-negative reals). However, all the results presented also hold for the corresponding non-positive cost functions.

For both probabilistic and expected reachability, we can consider reaching a state satisfying a formula ϕ which is a conjunction of propositions identifying locations and clock constraints (that is, constraints of the form $x \sim c$ for $x \in \mathcal{X}$, $\sim \in \{\leqslant, =, \geqslant\}$ and $c \in \mathbb{N}$). Instead of considering these cases separately, we just note that such reachability problems can be reduced to those referring to locations only by modifying syntactically the probabilistic timed automaton of interest (see [38]). Note that, if all the clock constraints appearing in the PTA and present in the formula ϕ are closed and diagonal-free, then all the clock constraints appearing in the modified PTA are also closed and diagonal-free.

In the case of probabilistic reachability the types of properties which can be expressed can be classified as follows:

Probabilistic reachability: The system reaches a certain set of states with a given maximal or minimal probability. For example, 'with probability at least 0.999, a data packet is correctly delivered'.

Probabilistic time-bounded reachability: The system reaches a certain set of states within a certain time deadline and probability threshold. For example, 'with probability 0.01 or less, a data packet is lost within 5 time units'.

Probabilistic cost-bounded reachability: The system reaches a certain set of states within a certain cost and probability bound. For example, 'with probability 0.75 or greater, a data packet is correctly delivered with at most 4 retransmissions'.

Invariance: The system does not leave a certain set of states with a given probability. For example, 'with probability 0.875 or greater, the system never aborts'.

Bounded response: The system inevitably reaches a certain set of states within a certain time deadline with a given probability. For example, 'with probability 0.99 or greater, a data packet will always be delivered within 5 time units'.

On the other hand, expected time reachability allows us to express, for example, 'the (maximum) expected time until a data packet is delivered is at most 20ms' and 'the (minimum) expected time until a packet collision occurs is at least 100 seconds'. In general, expected reachability allows us to validate properties including: 'the expected number of retransmissions before the message is correctly delivered is less than 3', 'the expected number of packets sent before failure is at least 300' and 'the expected number of lost messages within the first 200 seconds is at most 10'.

We illustrate the expected reachability approach using the final property as an example. We would first need to modify the probabilistic timed automaton under study by adding a distinct clock (to represent the elapsed time) and location such that, from all locations, once this clock has reached 200 seconds the only transition is to this new location. The set of target states would then be the set containing only the new location. The cost function would equal 0 on all time transitions and events except the event(s) corresponding to a message being lost, whose cost would be set to 1.

Finally, using the more general cost functions $\mathcal{C}_{c_\Sigma,r}$, we can consider performance measures such as:

– the expected time the channel is free before N messages are sent (by setting $r(l)$ to be 1 if location l corresponds to a state in which the channel is free, and 0 otherwise);
– the expected time a sender spends waiting for an acknowledgement (by setting $r(l)$ to be 1 if location l corresponds to a state in which the sender is waiting for an acknowledgement, and 0 otherwise);
– the expected energy consumption within the first $T(\in \mathbb{N})$ seconds (by setting $r(l)$ to the power usage (Watts) of the location $l \in L$ and $c_\Sigma(l, \sigma)$ to be the energy consumption associated with performing the event σ in location l).

3.5. Model Checking

To apply model checking methods we must first ensure that the system we consider has only finitely many states and is finitely branching. From the construction, the integral semantics has only finitely many states. However, to ensure finite branching, we must restrict the delays in the integral semantic models from $\mathbb{N}$ to some finite set. For example, since we have not permitted probabilistic choices over delays, we can restrict delays to have duration 1 only and, since any time transition of duration in $\mathbb{N}$ can be modelled by a sequence of time transitions of duration 1 and we restrict attention to divergent adversaries, nothing is lost by omitting delays of duration greater than 1 or of duration 0.

The model checking algorithms for both probabilistic and expected reachability are available in the literature; for probabilistic reachability see [15, 8], and for expected reachability see [22, 23]. In both cases verification reduces to solving a linear optimization problem for which one can apply iterative methods. We also note that, in [22], algorithms for checking for the presence of divergent adversaries are given.

The integral semantic model can suffer from the state space explosion problem; in particular, the size of the models is exponential in the number of clocks and the largest constant that the clocks are compared to. An abstraction technique which can be used to reduce the size of the model under study is that of changing the *time scale*, since this can reduce the constants that clocks are compared to. More formally, one can increase the time unit and then round upper bounds on the values of the constraints up, lower bounds down. For (non-probabilistic) timed automata, it is established in [4] that the trace set of the timed automaton after such a transformation includes that of the original model. It follows that, in the probabilistic setting, carrying out our model-checking on the transformed automaton gives bounds on the performance indices of the original automaton. More precisely, the computed *maximum* probabilistic and expected reachability measures on the transformed model are *upper* bounds and the minimum probabilistic and expected reachability measures are lower bounds on those that would be obtained for the original automaton.

3.5.1. *The probabilistic model checker PRISM*

PRISM [35, 47] is a probabilistic model checker developed at the University of Birmingham. The current implementation of PRISM supports the analysis of *finite*-state probabilistic models of the following three types: discrete-time Markov chains, Markov decision processes and continuous-time Markov chains. Discrete-time Markov chains are defined in Section 2.2, while Markov decision processes extend DTMCs by allowing both probabilistic and nondeterministic behaviour. Continuous-time Markov chains allow transitions to occur in real-time as opposed to discrete steps, but differ from probabilistic timed automata in that delays are represented by exponential distributions. Furthermore, there is no nondeterminism in continuous-time Markov chains.

Models in PRISM are described in a high-level language, a variant of reactive modules [3] based on guarded commands. The basic components of the language are *modules* and *variables*. A system is constructed as a number of modules which can interact with each other. A module contains a number of variables which express the state of the module, and its behaviour is given by a set of guarded commands of the form:

```
[<action>] <guard> → <updates>;
```

The guard is a predicate over the variables of the system and the updates describe transitions which the module can make if the guard is true (using primed variables to denote the next values of variables). Updates are specified as follows:

```
<prob> : <atomic_update> + ··· + <prob> : <atomic_update>
```

PRISM accepts specifications in probabilistic temporal logics. This allows us to express various probabilistic properties such as ‘event E happens with probability 1’, and ‘the probability of cost exceeding C is 95%’. The model checker then analyses the model and checks if the property holds in each state. In the case of Markov decision processes, specifications are written in the logic PCTL, and for the analysis PRISM implements the algorithms of [27, 15, 8]. The tool also supports verification of expected reachability properties using the algorithms of [22, 23].

The underlying data structures used in PRISM are BDDs (binary decision diagrams) and MTBDDs (multi-terminal BDDs) [19]. Model construction is always performed using MTBDDs and BDDs. For numerical computation, PRISM includes three separate *engines*. The first uses MTBDDs to store the model and iteration vector, while the second uses conventional data structures for numerical analysis: sparse matrices and arrays. The latter nearly always provides faster numerical computation than its MTBDD counterpart, but sacrifices the ability to conserve memory by exploiting structure. The third, *hybrid*, engine provides a compromise by storing the models in an MTBDD-like structure, which is adapted so that numerical computation can be carried out in combination with array-based storage for iteration vectors. This hybrid approach is generally faster than MTBDDs, while handling larger systems than sparse matrices. For further details and comparisons between the engines see [36, 46].

We note that, by using integral semantics and PRISM, and hence MTBDDs, we see similar advantages to those reported in [13, 14] for modelling and verifying classical timed automata using integral semantics and BDDs. In particular, by using only MTBDDs we are able to model and verify large and complex probabilistic real-time systems.

3.5.2. *Modelling probabilistic timed automata in PRISM*

We now explain the techniques used for modelling (the integral semantic models of) probabilistic timed automata as Markov decision processes in PRISM. First, due to the compositionality of the integral semantic model, if the system under study is a parallel composition of a number of probabilistic timed automata, then the integral semantics of each automaton can be modelled by a PRISM module and the system can be defined as the parallel composition of the modules. The only complication in this approach is representing passage of time; this is accomplished by including a distinct action, `time`, and then labelling the transitions of each module which correspond to time passing with this action. Hence, when a time action is executed, all modules must synchronize on this action.

Since, in the integral semantic model, the possible values of any clock x are in the range $\{0, 1, 2, \ldots, \mathbf{k}_x, \mathbf{k}_x + 1\}$, we can model each clock as a bounded integer-valued variable. Furthermore, we can use bounded integer-valued variables to model the locations of an automaton. The possible transitions of an automaton are then defined by a guarded command expressed in terms of these variables. In the case of time transitions, supposing in location l the invariant is $(y \leqslant 4)$ and the automaton has two clocks x and y, then the time passage transitions (of duration 1) in location l can be modelled by the guarded command:

```
[time] l=1&y<4 → (x'=min(x+1,kx+1))&(y'=min(y+1,ky+1));
```

Note that the non-strict upper bound in the invariant condition is made strict, because, if the clock x equals 4, no more time can pass (if one more time unit could elapse, then the value of y would become 5 and the invariant would be false).

In the case of discrete transitions, if in location l there is a discrete transition which has the enabling condition $4 \leqslant x \leqslant 6$, performs the event a, and moves to location 2 with probability 0.25, and to location 3 while resetting the clocks x and y with probability 0.75, then this transition can be modelled by the following guarded command:

```
[a] l=1&x>=4&x<=6 → 0.25:(l'=2) + 0.75:(l'=3)&(x'=0)&(y'=0);
```

4. Correctness of the Integral Semantics

In this section we show, under the restriction that the probabilistic timed automaton under study is closed and diagonal-free, that probabilistic and expected reachability values are the same in the integral and dense-time semantics. Therefore, for this class of probabilistic timed automata, it suffices to study the integral semantic model.

Let $\mathsf{PTA} = (L, \bar{l}, \mathcal{X}, \Sigma, inv, prob)$ be a probabilistic timed automaton. For any set of locations $F \subseteq L$, we denote by $F_{\mathbb{T}}$ the set of all states of $[\![\mathsf{PTA}]\!]_{\mathbb{T}}^{\oplus}$ which correspond to these locations; that is

$$F_{\mathbb{T}} = \{(l, v) \mid l \in F,\ v \in \mathbb{T}^{\mathcal{X}} \text{ and } v \triangleleft inv(l)\}\,.$$

4.1. ε-DIGITIZATION

In this section we extend the techniques developed in the classical timed automata case. We begin with the following definition and lemma which are taken from [28, 30].

Definition 16. *For any $t \in \mathbb{R}$ and $\varepsilon \in [0,1]$ let:*

$$[t]_\varepsilon = \begin{cases} \lfloor t \rfloor & \textit{if } t \leqslant \lfloor t \rfloor + \varepsilon \\ \lceil t \rceil & \textit{otherwise}. \end{cases}$$

Note that, from Definition 16, it trivially follows that:

$$[t]_1 \leqslant t \leqslant [t]_0 \quad \text{for all } t \in \mathbb{R}. \tag{1}$$

Lemma 17. *For any $t, t' \in \mathbb{R}$, $c \in \mathbb{N}$ and $\sim \in \{\leqslant, =, \geqslant\}$, if $t - t' \sim c$ then $[t]_\varepsilon - [t']_\varepsilon \sim c$ for all $\varepsilon \in [0,1]$.*

Next, we introduce the following property on paths of probabilistic timed automata.

Lemma 18. *For any path $\omega = (l_0, v_0) \xrightarrow{a_0,\mu_0} (l_1, v_1) \xrightarrow{a_1,\mu_1} \cdots$, $x \in \mathcal{X}$ and $i \in \mathbb{N}$, there exists $j \leqslant i$ such that $v_i(x) = dur(\omega, i) - dur(\omega, j)$.*

Proof. The proof follows from choosing $j \leqslant i$ such that (l_j, v_j) is the most recent state where the clock x was reset. □

Using the above, we now define the *ε-digitization* of a path [30, 28].

Definition 19 (ε-digitization). *For any path:*

$$\omega = (\bar{l}, \mathbf{0}) \xrightarrow{a_0,\mu_0} (l_1, v_1) \xrightarrow{a_1,\mu_1} \cdots$$

of $[\![\mathsf{PTA}]\!]_\mathbb{R}$, its ε-digitization is the path

$$[\omega]_\varepsilon = (\bar{l}, \mathbf{0}) \xrightarrow{a_0',\mu_0'} (l_1, [v_1]_\varepsilon) \xrightarrow{a_1',\mu_1'} \cdots$$

of $[\![\mathsf{PTA}]\!]_\mathbb{N}$ where for any $i \in \mathbb{N}$ and $x \in \mathcal{X}$:

- *$[v_i]_\varepsilon(x) = \min([dur(\omega,i)]_\varepsilon - [dur(\omega,j)]_\varepsilon, \mathbf{k}_x + 1)$ and $j \leqslant i$ such that $v_i(x) = dur(\omega,i) - dur(\omega,j)$ which exists by Lemma 18;*
- *if $a_i \in \Sigma$ and μ_i is constructed from a probabilistic edge of PTA which is of the form $(-,-,a_i,p_i)$, then $a_i' = a_i$ and for any $(l', v') \in L \times \mathbb{N}^{\mathcal{X}}$:*

$$\mu_i'(l', v') = \sum_{\substack{X \subseteq \mathcal{X}\ \& \\ v' = [v_i]_\varepsilon[X:=0]}} p_i(X, l') \;;$$

- *if $a_i \in \mathbb{R}$, then $a_i' = [dur(\omega, i{+}1)]_\varepsilon - [dur(\omega,i)]_\varepsilon$ and $\mu_i' = \mu_{(l_i, [v_i]_\varepsilon \oplus a_i')}$.*

The well-definedness of this construction – that is, the fact that $[\omega]_\varepsilon$ is a path of $[\![\mathsf{PTA}]\!]_\mathbb{N}$ – follows from Lemma 17, Lemma 18, and the fact that the clock constraints appearing in PTA are closed and diagonal-free. For example, for any $x \in \mathcal{X}$ and $i \geqslant 0$ such that $a_i \in \mathbb{R}$, by Definition 19 there exists $j \leqslant i$ such that:

$$\begin{aligned}
\min([v_i]_\varepsilon(x) + a_i', \mathbf{k}_x{+}1) &= \min([dur(\omega,i)]_\varepsilon - [dur(\omega,j)]_\varepsilon + a_i', \mathbf{k}_x{+}1) \\
&= \min([dur(\omega,i)]_\varepsilon - [dur(\omega,j)]_\varepsilon + [dur(\omega,i{+}1)]_\varepsilon - [dur(\omega,i)]_\varepsilon, \mathbf{k}_x{+}1) \\
&\qquad \text{by construction} \\
&= \min([dur(\omega,i{+}1)]_\varepsilon - [dur(\omega,j)]_\varepsilon, \mathbf{k}_x{+}1) \qquad \text{rearranging} \\
&= [v_{i+1}]_\varepsilon(x) \qquad \text{by Definition 19.}
\end{aligned}$$

We now introduce the following lemmas which relate the time and cost of a path with those of its digitization.

Lemma 20. *For any path $\omega \in Path_{ful}(\bar{s})$, $\varepsilon \in [0,1]$ and $i \in \mathbb{N}$:*

$$dur([\omega]_\varepsilon, i) = [dur(\omega, i)]_\varepsilon \, .$$

Proof. Consider any path $\omega = (\bar{l}, \mathbf{0}) \xrightarrow{a_0,\mu_0} (l_1, v_1) \xrightarrow{a_1,\mu_1} \cdots \in Path_{ful}(\bar{s})$. We prove the lemma by induction on $i \in \mathbb{N}$. If $i = 0$, then by definition $dur([\omega]_\varepsilon, i) = [dur(\omega, i)]_\varepsilon = 0$ as required.

Now suppose that the lemma holds for some $i \in \mathbb{N}$. We have two cases to consider.

- If $a_i \in \Sigma$, then $[dur(\omega, i{+}1)]_\varepsilon = [dur(\omega, i)]_\varepsilon$ and from Definition 19 we have $dur([\omega]_\varepsilon, i{+}1) = dur([\omega]_\varepsilon, i)$, and hence the lemma holds by induction.
- If $a_i \in \mathbb{R}$, then by Definition 19 we have

$$\begin{aligned}
&dur([\omega]_\varepsilon, i{+}1) = dur([\omega]_\varepsilon, i) + ([dur(\omega, i{+}1)]_\varepsilon - [dur(\omega, i)]_\varepsilon) \\
&= \ [dur(\omega, i)]_\varepsilon + ([dur(\omega, i{+}1)]_\varepsilon - [dur(\omega, i)]_\varepsilon) \quad \text{by induction} \\
&= \ [dur(\omega, i{+}1)]_\varepsilon \qquad\qquad\qquad\qquad\qquad\qquad \text{as required.}
\end{aligned}$$

Since these are the only cases to consider, the lemma holds by induction on $i \in \mathbb{N}$. □

Lemma 21. *For any path $\omega \in Path_{ful}(\bar{s})$, set of locations $F \subseteq L$ and cost function $\mathcal{C}_{c_\Sigma,\mathbf{r}}$ we have:*

$$cost(\mathcal{C}_{c_\Sigma,\mathbf{r}}, F_\mathbb{N})([\omega]_1) \leqslant cost(\mathcal{C}_{c_\Sigma,\mathbf{r}}, F_\mathbb{R})(\omega) \leqslant cost(\mathcal{C}_{c_\Sigma,\mathbf{r}}, F_\mathbb{N})([\omega]_0) \, .$$

Proof. Consider any path $\omega \in Path_{ful}(\bar{s})$, set of locations $F \subseteq L$ and cost function $\mathcal{C}_{c_\Sigma,\mathbf{r}}$.

- If there does not exist $i \in \mathbb{N}$ such that $\omega(i) \in F_\mathbb{R}$, then, for any $\varepsilon \in [0,1]$, from Definition 19, there does not exist $i \in \mathbb{N}$ such that $[\omega(i)]_\varepsilon \in F_\mathbb{N}$. Therefore, by definition of $cost(\mathcal{C}_{c_\Sigma,\mathbf{r}}, F_\mathbb{T})$, we have:

$$cost(\mathcal{C}_{c_\Sigma,\mathbf{r}}, F_\mathbb{N})([\omega]_\varepsilon) = cost(\mathcal{C}_{c_\Sigma,\mathbf{r}}, F_\mathbb{R})(\omega) = \infty$$

  for all $\varepsilon \in [0,1]$, and hence the lemma holds in this case.
- If there exists $i \in \mathbb{N}$ such that $\omega(i) \in F_\mathbb{R}$, then, by definition of $dur(\omega, \cdot)$ and $cost(\mathcal{C}_{c_\Sigma,\mathbf{r}}, F_\mathbb{R})$, it follows that:

$$cost(\mathcal{C}_{c_\Sigma,\mathbf{r}}, F_\mathbb{R})(\omega) = cost(\mathcal{C}_{c_\Sigma,0}, F_\mathbb{R})(\omega) + \mathbf{r} \cdot dur(\omega, i_F) \, . \qquad (2.1)$$

  where $i_F = \min\{j \mid \omega(j) \in F_\mathbb{R}\}$. Next, from Definition 19 we have, for any $\varepsilon \in [0,1]$, $i \in \mathbb{N}$ and $\sigma \in \Sigma$:

  - $\omega(i) \in F_\mathbb{R}$ if and only if $[\omega]_\varepsilon(i) \in F_\mathbb{N}$;

- $step(\omega, i) = \sigma$ if and only if $step([\omega]_\varepsilon, i) = \sigma$.

It then follows that:

$$cost(\mathcal{C}_{c_\Sigma,0}, F_\mathbb{N})([\omega]_\varepsilon) = cost(\mathcal{C}_{c_\Sigma,0}, F_\mathbb{R})(\omega) \quad \text{for all } \varepsilon \in [0,1]. \tag{2.2}$$

Furthermore, using Lemma 20 and (1) (see page 19) we have:

$$dur([\omega]_1, i_F) \leqslant dur(\omega, i_F) \leqslant dur([\omega]_0, i_F)\,. \tag{2.3}$$

Finally, combining (2.1)-(2.3) and using the fact that $\mathbf{r} \geqslant 0$ we have

$$cost(\mathcal{C}_{c_\Sigma,\mathbf{r}}, F_\mathbb{N})([\omega]_1) \leqslant cost(\mathcal{C}_{c_\Sigma,\mathbf{r}}, F_\mathbb{R})(\omega) \leqslant cost(\mathcal{C}_{c_\Sigma,\mathbf{r}}, F_\mathbb{N})([\omega]_0)$$

as required.

Since these are the only cases to consider the lemma holds. □

We now extend the notion of digitization from paths to adversaries. Note that we extend the digitization notation $[\cdot]_\varepsilon$ to *sets* of paths: for a set Ω of infinite paths, let $[\Omega]_\varepsilon = \{[\omega]_\varepsilon \mid \omega \in \Omega\}$. To simplify the presentation, when considering a fixed adversary $A \in Adv_{[\![\mathsf{PTA}]\!]_\mathbb{R}}$, we suppose that the domain of the mapping $[\cdot]_\varepsilon$ is *restricted* to the sets of paths $Path^A_{\mathit{fin}}(\bar{s})$ and $Path^A_{\mathit{ful}}(\bar{s})$. More precisely, for any path π of $[\![\mathsf{PTA}]\!]_\mathbb{N}$:

$$[\pi]_\varepsilon^{-1} = \begin{cases} \{\omega \in Path^A_{\mathit{fin}}(\bar{s}) \mid [\omega]_\varepsilon = \pi\} & \text{if } \pi \text{ is finite} \\ \{\omega \in Path^A_{\mathit{ful}}(\bar{s}) \mid [\omega]_\varepsilon = \pi\} & \text{otherwise} \end{cases} \tag{3}$$

Using this interpretation we have the following result.

Lemma 22. *For any adversary $A \in Adv_{[\![\mathsf{PTA}]\!]_\mathbb{R}}$, path π of $[\![\mathsf{PTA}]\!]_\mathbb{N}$ and $\varepsilon \in [0,1]$, the set of paths $[\pi]_\varepsilon^{-1} \subseteq Path^A_{\mathit{fin}}(\bar{s})$ is finite.*

Proof. Consider any adversary $A \in Adv_{[\![\mathsf{PTA}]\!]_\mathbb{R}}$, path π of $[\![\mathsf{PTA}]\!]_\mathbb{N}$ and $\varepsilon \in [0,1]$, then the result is a direct consequence of the following two facts:

– for any $n \in \mathbb{N}$ the set of paths $\{\omega \in Path^A_{\mathit{fin}}(\bar{s}) \mid |\omega|{=}n\}$ is finite;

– $|\omega| = |[\omega]_\varepsilon|$ for all $\omega \in Path^A_{\mathit{fin}}(\bar{s})$. □

We now extend the notion of digitization to adversaries through the following proposition in which we use $\mathcal{F}_{Path^A_{\mathit{ful}}(\bar{s})}$ to denote the σ-algebra generated by the adversary A over the set of infinite paths $Path^A_{\mathit{ful}}(\bar{s})$.

Proposition 23. *For any adversary $A \in Adv_{[\![\mathsf{PTA}]\!]_\mathbb{R}}$ and $\varepsilon \in [0,1]$, there exists a (randomized) adversary $B^\varepsilon \in Adv_{[\![\mathsf{PTA}]\!]_\mathbb{N}}$ such that: $Prob^{B^\varepsilon}_{\bar{s}}(\Pi) = Prob^A_{\bar{s}}([\Pi]_\varepsilon^{-1})$ for all $\Pi \in \mathcal{F}_{Path^{B^\varepsilon}_{\mathit{ful}}(\bar{s})}$.*

Proof. Consider any adversary A of $[\![\mathsf{PTA}]\!]_{\mathbb{R}}$. First, to ease notation, for any set of finite paths $\Omega \subseteq Path^A_{fin}(\bar{s})$ let:

$$Prob^A_{\bar{s}}(\Omega) = Prob^A_{\bar{s}}\{\omega' \in Path^A_{ful}(\bar{s}) \,|\, \omega \leqslant \omega' \text{ for some } \omega \in \Omega\}$$

and for any finite path $\pi \in Path^A_{fin}(\bar{s})$ and $(a,\mu) \in Steps(last(\pi))$ let:

$$(\pi \xrightarrow{a,\mu}) \stackrel{\text{def}}{=} \{\pi' \in Path^A_{fin}(\bar{s}) \,|\, \exists s \in S.\ \pi' = \pi \xrightarrow{a,\mu} s\}\,.$$

We define the adversary B^ε as follows. The set of paths of B^ε is given by $\{[\omega]_\varepsilon \,|\, \omega \in Path^A_{ful}(\bar{s})\}$, and, as usual, let $Path^{B^\varepsilon}_{fin}(\bar{s})$ be the set of finite prefixes of these paths. For any path $\pi \in Path^{B^\varepsilon}_{fin}(\bar{s})$ and $(a,\mu) \in Steps(last(\pi))$, the probability of B^ε choosing (a,μ) after π has been performed is given by:

$$B^\varepsilon(\pi)(a,\mu) \stackrel{\text{def}}{=} \frac{Prob^A_{\bar{s}}([\pi \xrightarrow{a,\mu}]^{-1}_\varepsilon)}{Prob^A_{\bar{s}}([\pi]^{-1}_\varepsilon)}\ .$$

Note that the above probabilities are well defined since, from (3), the sets of paths $[\pi \xrightarrow{a,\mu}]^{-1}_\varepsilon$ and $[\pi]^{-1}_\varepsilon$ are both finite. We are now in a position, using the cylinder construction (see Section 2.2), to define the probability measure $Prob^{B^\varepsilon}_{\bar{s}}$ over $Path^{B^\varepsilon}_{ful}(\bar{s})$.

To complete the proof it remains to show that $Prob^{B^\varepsilon}_{\bar{s}}(\Pi) = Prob^A_{\bar{s}}([\Pi]^{-1}_\varepsilon)$ for all $\Pi \in \mathcal{F}_{Path^{B^\varepsilon}_{ful}(\bar{s})}$. Now, from the cylinder construction (see Section 2.2), it follows that it is sufficient to show that:

$$Prob^{B^\varepsilon}_{\bar{s}}(\pi) = Prob^A_{\bar{s}}([\pi]^{-1}_\varepsilon) \ \text{ for all } \pi \in Path^{B^\varepsilon}_{fin}(\bar{s}) \tag{4}$$

which we now prove by induction on the length of π. Note that, again, it follows from (3) that the set of paths $[\pi]^{-1}_\varepsilon$ is finite.

Therefore, consider any path $\pi \in Path^{B^\varepsilon}_{fin}(\bar{s})$. If $|\pi| = 0$ then $Prob^{B^\varepsilon}_{\bar{s}}(\pi) = 1 = Prob^A_{\bar{s}}([\pi]^{-1}_\varepsilon)$ as required.

Next, suppose by induction the lemma holds for all paths of length n and π is of length $n+1$. Now, π is of the form $\pi' \xrightarrow{a,\mu} s'$ for some path π' of length n such that $(a,\mu) \in Steps(last(\pi'))$ and $s' \in S$, and therefore:

$$\begin{aligned}
&Prob^{B^\varepsilon}_{\bar{s}}(\pi) = Prob^{B^\varepsilon}_{\bar{s}}(\pi') \cdot B^\varepsilon(\pi')(a,\mu) \cdot \mu(s') \\
&= Prob^A_{\bar{s}}([\pi']^{-1}_\varepsilon) \cdot B^\varepsilon(\pi')(a,\mu) \cdot \mu(s') && \text{by induction} \\
&= Prob^A_{\bar{s}}([\pi']^{-1}_\varepsilon) \cdot \frac{Prob^A_{\bar{s}}([\pi' \xrightarrow{a,\mu}]^{-1}_\varepsilon)}{Prob^A_{\bar{s}}([\pi']^{-1}_\varepsilon)} \cdot \mu(s') && \text{by definition of } B^\varepsilon \\
&= Prob^A_{\bar{s}}([\pi' \xrightarrow{a,\mu}]^{-1}_\varepsilon) \cdot \mu(s') && \text{rearranging} \\
&= Prob^A_{\bar{s}}\{\omega \in Path^A_{fin}(\bar{s}) \,|\, [\omega]_\varepsilon = \pi' \xrightarrow{a,\mu} s'\} && \text{by definition of } Prob^A_{\bar{s}} \\
&= Prob^A_{\bar{s}}([\pi]^{-1}_\varepsilon) && \text{by construction of } \pi
\end{aligned}$$

and hence (4) holds by induction.

Finally, to show that $B^\varepsilon \in Adv_{[\![\mathsf{PTA}]\!]_\mathbb{N}}$, we must show that B^ε is divergent. This result follows from (4), the fact that A is divergent, and since from Lemma 20 any (infinite) path is divergent if and only if its ε-digitization is divergent. □

4.2. Probabilistic Reachability

In this section we show that it is sufficient to consider the integral semantics when computing probabilistic reachability properties. Note that an alternative proof of Theorem 24 appears in [41].

Theorem 24. *For any (closed, diagonal-free) probabilistic timed automaton* PTA *and set of locations $F \subseteq L$:*

$$\begin{aligned} p^{\max}_{[\![\mathsf{PTA}]\!]_\mathbb{R}}(F_\mathbb{R}) &= p^{\max}_{[\![\mathsf{PTA}]\!]_\mathbb{N}}(F_\mathbb{N}) \\ p^{\min}_{[\![\mathsf{PTA}]\!]_\mathbb{R}}(F_\mathbb{R}) &= p^{\min}_{[\![\mathsf{PTA}]\!]_\mathbb{N}}(F_\mathbb{N}) \,. \end{aligned}$$

Proof. First, from Definition 19, we have that $\omega(i) \in F_\mathbb{R}$ if and only if $[\omega]_\varepsilon(i) \in F_\mathbb{N}$ for any path ω of $[\![\mathsf{PTA}]\!]_\mathbb{R}$. We have seen in Proposition 23 that for any adversary $A \in Adv_{[\![\mathsf{PTA}]\!]_\mathbb{R}}$ and $\varepsilon \in [0,1]$, we can define an adversary $B^\varepsilon \in Adv_{[\![\mathsf{PTA}]\!]_\mathbb{N}}$ such that: $Prob^{B^\varepsilon}_{\bar{s}}(\Pi) = Prob^{A}_{\bar{s}}([\Pi]^{-1}_\varepsilon)$ for all $\Pi \in \mathcal{F}_{Path^{B^\varepsilon}_{ful}(\bar{s})}$. Combining these results it follows that, for any adversary $A \in Adv_{[\![\mathsf{PTA}]\!]_\mathbb{R}}$ and $\varepsilon \in [0,1]$, we can construct an adversary $B^\varepsilon \in Adv_{[\![\mathsf{PTA}]\!]_\mathbb{N}}$ such that:

$$p^A_{\bar{s}}(F_\mathbb{R}) = p^{B^\varepsilon}_{\bar{s}}(F_\mathbb{N}) \,,$$

and hence:

$$\inf_{A \in Adv_{[\![\mathsf{PTA}]\!]_\mathbb{R}}} p^A_{\bar{s}}(F_\mathbb{R}) \geqslant \inf_{B \in Adv_{[\![\mathsf{PTA}]\!]_\mathbb{N}}} p^B_{\bar{s}}(F_\mathbb{N})$$

and

$$\sup_{A \in Adv_{[\![\mathsf{PTA}]\!]_\mathbb{R}}} p^A_{\bar{s}}(F_\mathbb{R}) \leqslant \sup_{B \in Adv_{[\![\mathsf{PTA}]\!]_\mathbb{N}}} p^B_{\bar{s}}(F_\mathbb{N}) \,.$$

On the other hand, by definition of the integral and dense-time semantics, for any adversary $B \in Adv_{[\![\mathsf{PTA}]\!]_\mathbb{N}}$, there exists an adversary $A \in Adv_{[\![\mathsf{PTA}]\!]_\mathbb{R}}$ such that $p^A_{\bar{s}}(F_\mathbb{R}) = p^B_{\bar{s}}(F_\mathbb{N})$. Intuitively, the adversary A behaves like the integral semantic adversary B; that is, it chooses to make timed transitions of only integral duration. The result of Theorem 24 then follows. □

4.3. Expected Reachability and Uniform Cost Functions

In this section we consider expected reachability while restricting attention to cost functions defined by pairs $(c_\Sigma, \mathbf{r})$, where $c_\Sigma : L \times \Sigma \to \mathbb{R}$ and $\mathbf{r} \in \mathbb{R}$. Note that an alternative characterization of the cost functions we consider is those that satisfy the following property:

– $\mathcal{C}(s,t) = \mathcal{C}(s',t)$ for all $s, s' \in S$ and $t \in \mathbb{R}$;

– $\mathcal{C}(s,t+t') = \mathcal{C}(s,t) + \mathcal{C}(s,t')$ for all $s \in S$ and $t, t' \in \mathbb{R}$.

Before we give the proof of correctness under these assumptions, we require the following lemma and proposition. Recall the definition of a measurable function (Definition 2).

Lemma 25. *For any adversary $A \in Adv_{[\![\mathsf{PTA}]\!]_\mathbb{R}}$, the mapping $[\cdot]_\varepsilon$ is a measurable function from $(Path^A_{ful}(\bar{s}), \mathcal{F}_{Path^A_{ful}(\bar{s})})$ to the measurable space induced from the set of paths $\{[\omega]_\varepsilon \,|\, \omega \in Path^A_{ful}(\bar{s})\}$.*

Proof. Recall, from Section 2.2, that for any finite path ω, the cylinder set of ω is given by $cyl(\omega)$. The proof follows from the fact that for any finite path $\pi \in \{[\omega]_\varepsilon \,|\, \omega \in Path^A_{fin}(\bar{s})\}$:

$$[cyl(\pi)]^{-1}_\varepsilon = \bigcup \{ cyl(\omega) \,|\, \omega \in Path^A_{fin}(\bar{s}) \wedge [\omega]_\varepsilon = \pi \} \ ,$$

and the set of paths $\{\omega \in Path^A_{fin}(\bar{s}) \,|\, [\omega]_\varepsilon = \pi\}$ is finite. That is, the set $[cyl(\pi)]^{-1}_\varepsilon$ is the finite union of measurable sets of $Path^A_{ful}(\bar{s})$, and hence a measurable set. □

Proposition 26. *For any set of locations $F \subseteq L$, adversary $A \in Adv_{[\![\mathsf{PTA}]\!]_\mathbb{R}}$ and cost function $\mathcal{C}_{c_\Sigma,\mathbf{r}}$, if the adversaries $B^0, B^1 \in Adv_{[\![\mathsf{PTA}]\!]_\mathbb{N}}$ are constructed following Proposition 23, then:*

$$e^{B^1}_{\bar{s}}(cost(\mathcal{C}_{c_\Sigma,\mathbf{r}}, F_\mathbb{N})) \leqslant e^A_s(cost(\mathcal{C}_{c_\Sigma,\mathbf{r}}, F_\mathbb{R})) \leqslant e^{B^0}_{\bar{s}}(cost(\mathcal{C}_{c_\Sigma,\mathbf{r}}, F_\mathbb{N}))$$

Proof. Consider any adversary $A \in Adv_{[\![\mathsf{PTA}]\!]_\mathbb{R}}$ and cost function $\mathcal{C}_{c_\Sigma,\mathbf{r}}$. First, for any $\varepsilon \in [0,1]$, since $\mathcal{C}_{c_\Sigma,\mathbf{r}}$ is non-negative, $cost(\mathcal{C}_{c_\Sigma,\mathbf{r}}, F_\mathbb{N})$ is a real non-negative function on $(Path^{B^\varepsilon}_{ful}(\bar{s}), \mathcal{F}_{Path^{B^\varepsilon}_{ful}(\bar{s})})$, and hence using Proposition 23 and Lemma 25 we can apply Theorem 3 which gives:

$$\begin{aligned} &\int_{\omega \in Path^A_{ful}(\bar{s})} cost(\mathcal{C}_{c_\Sigma,\mathbf{r}}, F_\mathbb{N})([\omega]_\varepsilon) \,\mathrm{d}Prob^A_{\bar{s}} \\ &\quad = \int_{\pi \in Path^{B^\varepsilon}_{ful}(\bar{s})} cost(\mathcal{C}_{c_\Sigma,\mathbf{r}}, F_\mathbb{N})(\pi) \,\mathrm{d}Prob^{B^\varepsilon}_{\bar{s}} \ . \end{aligned} \tag{5}$$

Now by definition of $e^A_{\bar{s}}$:

$$\begin{aligned}
e^A_{\bar{s}}(\mathit{cost}(\mathcal{C}_{c_\Sigma,\mathbf{r}}, F_{\mathbb{R}})) &= \int_{\omega \in \mathit{Path}^A_{\mathit{ful}}(\bar{s})} \mathit{cost}(\mathcal{C}_{c_\Sigma,\mathbf{r}}, F_{\mathbb{R}})(\omega)\, \mathrm{d}\mathit{Prob}^A_{\bar{s}} \\
&\leqslant \int_{\omega \in \mathit{Path}^A_{\mathit{ful}}(\bar{s})} \mathit{cost}(\mathcal{C}_{c_\Sigma,\mathbf{r}}, F_{\mathbb{N}})([\omega]_0)\, \mathrm{d}\mathit{Prob}^A_{\bar{s}} \qquad \text{by Lemma 21} \\
&= \int_{\pi \in \mathit{Path}^{B^0}_{\mathit{ful}}(\bar{s})} \mathit{cost}(\mathcal{C}_{c_\Sigma,\mathbf{r}}, F_{\mathbb{N}})(\pi)\, \mathrm{d}\mathit{Prob}^{B^0}_{\bar{s}} \qquad \text{by (5)} \\
&= e^{B^0}_{\bar{s}}(\mathit{cost}(\mathcal{C}_{c_\Sigma,\mathbf{r}}, F_{\mathbb{N}})) \qquad \text{by definition.}
\end{aligned}$$

Similarly, we can show $e^{B^1}_{s}(\mathit{cost}(\mathcal{C}_{c_\Sigma,\mathbf{r}}, F_{\mathbb{N}})) \leqslant e^A_s(\mathit{cost}(\mathcal{C}_{c_\Sigma,\mathbf{r}}, F_{\mathbb{R}}))$. □

Theorem 27. *For any (closed, diagonal-free) probabilistic timed automaton* PTA*, set of locations* $F \subseteq L$ *and cost function* $\mathcal{C}_{c_\Sigma,\mathbf{r}}$*:*

$$\begin{aligned}
e^{\max}_{[\![\mathsf{PTA}]\!]_{\mathbb{R}}}(\mathcal{C}_{c_\Sigma,\mathbf{r}}, F_{\mathbb{R}}) &= e^{\max}_{[\![\mathsf{PTA}]\!]_{\mathbb{N}}}(\mathcal{C}_{c_\Sigma,\mathbf{r}}, F_{\mathbb{N}}) \\
e^{\min}_{[\![\mathsf{PTA}]\!]_{\mathbb{R}}}(\mathcal{C}_{c_\Sigma,\mathbf{r}}, F_{\mathbb{R}}) &= e^{\min}_{[\![\mathsf{PTA}]\!]_{\mathbb{N}}}(\mathcal{C}_{c_\Sigma,\mathbf{r}}, F_{\mathbb{N}})\ .
\end{aligned}$$

Proof. Consider any set of locations $F \subseteq L$ and cost function $\mathcal{C}_{c_\Sigma,\mathbf{r}}$. We have seen in Proposition 23 and Proposition 26 that for any adversary $A \in \mathit{Adv}_{[\![\mathsf{PTA}]\!]_{\mathbb{R}}}$ we can construct adversaries $B^0, B^1 \in \mathit{Adv}_{[\![\mathsf{PTA}]\!]_{\mathbb{N}}}$ such that:

$$e^{B^1}_{\bar{s}}(\mathit{cost}(\mathcal{C}_{c_\Sigma,\mathbf{r}}, F_{\mathbb{N}})) \leqslant e^A_s(\mathit{cost}(\mathcal{C}_{c_\Sigma,\mathbf{r}}, F_{\mathbb{R}})) \leqslant e^{B^0}_{\bar{s}}(\mathit{cost}(\mathcal{C}_{c_\Sigma,\mathbf{r}}, F_{\mathbb{N}}))\ ,$$

and hence it follows that:

$$\inf_{A \in \mathit{Adv}_{[\![\mathsf{PTA}]\!]_{\mathbb{R}}}} e^A_{\bar{s}}(\mathit{cost}(\mathcal{C}_{c_\Sigma,\mathbf{r}}, F_{\mathbb{R}})) \geqslant \inf_{B \in \mathit{Adv}_{[\![\mathsf{PTA}]\!]_{\mathbb{N}}}} e^B_{\bar{s}}(\mathit{cost}(\mathcal{C}_{c_\Sigma,\mathbf{r}}, F_{\mathbb{N}}))$$

and

$$\sup_{A \in \mathit{Adv}_{[\![\mathsf{PTA}]\!]_{\mathbb{R}}}} e^A_{\bar{s}}(\mathit{cost}(\mathcal{C}_{c_\Sigma,\mathbf{r}}, F_{\mathbb{R}})) \leqslant \sup_{B \in \mathit{Adv}_{[\![\mathsf{PTA}]\!]_{\mathbb{N}}}} e^B_{\bar{s}}(\mathit{cost}(\mathcal{C}_{c_\Sigma,\mathbf{r}}, F_{\mathbb{N}}))\ .$$

On the other hand, as in the proof of Theorem 24, for any adversary $B \in \mathit{Adv}_{[\![\mathsf{PTA}]\!]_{\mathbb{N}}}$, there exists an adversary $A \in \mathit{Adv}_{[\![\mathsf{PTA}]\!]_{\mathbb{R}}}$ such that:

$$e^A_{\bar{s}}(\mathit{cost}(\mathcal{C}_{c_\Sigma,\mathbf{r}}, F_{\mathbb{R}})) = e^B_{\bar{s}}(\mathit{cost}(\mathcal{C}_{c_\Sigma,\mathbf{r}}, F_{\mathbb{N}})). \quad \square$$

4.4. Expected Reachability and Variable Cost Functions

In this section we extend our results on expected reachability to the case when, as in (non-probabilistic) linearly priced timed automata [10], the costs associated with the time spent in locations can vary between locations. More precisely, we consider cost functions of the form $\mathcal{C}_{c_\Sigma,r}$ where:

- $c_\Sigma : L \times \Sigma \to \mathbb{R}$ is a function assigning the cost of executing events;
- $r : L \to \mathbb{R}$ is a function assigning to each location the rate at which costs are accumulated as time passes in that location;
- for any $(l, v) \in L \times \mathbb{R}^{\mathcal{X}}$ and $a \in \Sigma \cup \mathbb{R}$:

$$\mathcal{C}_{c_\Sigma,r}((l,v),a) \overset{\text{def}}{=} \begin{cases} c_\Sigma(l,a) & \text{if } a \in \Sigma \\ a \cdot r(l) & \text{otherwise.} \end{cases}$$

Note that, an alternative characterization of such cost functions are those that satisfy:

$$\mathcal{C}(s, t+t') = \mathcal{C}(s,t) + \mathcal{C}(s,t') \text{ for all } s \in S \text{ and } t, t' \in \mathbb{R}.$$

As in the previous section, the proof demonstrating that digital clocks are sufficient for calculating expected reachability properties for such cost functions relies on showing that, for any fixed (dense-time semantic) adversary, there exist integral-time semantic adversaries whose expected costs of reaching a set of target states within n transitions bound that of A. First, we require the following results from linear programming.

Definition 28. *A matrix* $\mathbf{A}$ *is* totally unimodular *if each subdeterminant of* $\mathbf{A}$ *is* 0, $+1$ *or* -1.

Theorem 29 ([49] Theorem 19.3). *Let* $\mathbf{A}$ *be a matrix with entries* 0, $+1$ *and* -1. *Then the following are equivalent:*

1. $\mathbf{A}$ *is totally unimodular;*
2. *each collection of columns of* $\mathbf{A}$ *can be split into two parts so that the sum of the columns in one part minus the sum of the columns in the other part is a vector with entries only* 0, $+1$ *and* -1.

Theorem 30 ([49] Corollary 19.1.a). *Let* $\mathbf{A}$ *be a totally unimodular matrix, and let* $\mathbf{b}$ *and* $\mathbf{c}$ *be integral vectors. Then both problems in the linear programming duality equation*

$$\max\{\mathbf{cx} \,|\, \mathbf{x} \geqslant 0 \ \wedge\ \mathbf{Ax} \leqslant \mathbf{b}\} = \min\{\mathbf{yb} \,|\, \mathbf{y} \geqslant 0 \ \wedge\ \mathbf{yA} \geqslant \mathbf{c}\}$$

have integral optimum solutions.

We next define, for any adversary A, a sequence of functions $(e_n^A)_{n\in\mathbb{N}}$, where, for any state s, cost function $\mathcal{C}_{c_\Sigma,r}$ and set of target locations F, $e_n^A(\mathcal{C}_{c_\Sigma,r}, F_{\mathbb{T}}, s)$ equals the expected cost, under the adversary A, of reaching F from s within n transitions. Since adversaries can choose on the basis of history, we first define e_n^A over paths, then restrict to the case of the initial state (paths of length 0).

Definition 31. *Let* $\mathsf{PTA} = (L, \bar{l}, \mathcal{X}, \Sigma, inv, prob)$ *be a probabilistic timed automaton and* $\mathsf{TPS} = (S, \bar{s}, \mathcal{A}ct, \mathbb{T}, Steps)$ *be its semantics for the time domain* $\mathbb{T}$*. For any subset of target locations* $F \subseteq L$*, cost function* $\mathcal{C}_{c_\Sigma,r}$*, adversary* $A \in Adv_{\mathsf{TPS}}$ *and* $\omega \in Path^A_{fin}$*, if* $last(\omega) = (l, v)$ *and* $A(\omega) = (a, \mu)$*, let* $e^A_0(\mathcal{C}_{c_\Sigma,r}, F_\mathbb{T}, \omega) = 0$ *and for any* $n \geqslant 0$ *:*

$$e^A_{n+1}(\mathcal{C}_{c_\Sigma,r}, F_\mathbb{T}, \omega) = \begin{cases} 0 & \textit{if } (l,v) \in F_\mathbb{T} \\ \mathcal{C}_{c_\Sigma,r}(l,a) + \sum_{s' \in S} \mu(s') \cdot e^A_n(\mathcal{C}_{c_\Sigma,r}, F_\mathbb{T}, \omega \xrightarrow{a,\mu} s') & \textit{otherwise.} \end{cases}$$

Lemma 32. *Let* $\mathsf{PTA} = (L, \bar{l}, \mathcal{X}, \Sigma, inv, prob)$ *be a probabilistic timed automaton and* $\mathsf{TPS} = (S, \bar{s}, \mathcal{A}ct, \mathbb{T}, Steps)$ *be its semantics for the time domain* $\mathbb{T}$*. For any subset of target locations* F*, cost function* $\mathcal{C}_{c_\Sigma,r}$ *and adversary* $A \in Adv_{\mathsf{TPS}}$*,* $\langle e^A_n(\mathcal{C}_{c_\Sigma,r}, F_\mathbb{T}, \bar{s}) \rangle_{n \in \mathbb{N}}$ *is a non-decreasing sequence converging to* $e^A_{\bar{s}}(cost(\mathcal{C}_{c_\Sigma,r}, F_\mathbb{T}))$*.*

Lemma 33. *Let* $\mathsf{PTA} = (L, \bar{l}, \mathcal{X}, \Sigma, inv, prob)$ *be a probabilistic timed automaton and* $\mathcal{C}_{c_\Sigma,r}$ *a (non-negative) cost function with rational coefficients. For any adversary* $A \in Adv_{[\![\mathsf{PTA}]\!]_\mathbb{R}}$*, set of target locations* F *and* $n \in \mathbb{N}$*, there exist adversaries* $B, C \in Adv_{[\![\mathsf{PTA}]\!]_\mathbb{N}}$ *such that:*

$$e^B_n(\mathcal{C}_{c_\Sigma,r}, F_\mathbb{N}, \bar{s}) \leqslant e^A_n(\mathcal{C}_{c_\Sigma,r}, F_\mathbb{R}, \bar{s}) \leqslant e^C_n(\mathcal{C}_{c_\Sigma,r}, F_\mathbb{N}, \bar{s}) \, .$$

Proof. The first step in the proof involves constructing a set of constraints on the time steps of the adversaries which follow the same choices as A (except in the actual duration of time transitions) up until the nth discrete transition. Using these constraints we then formulate a linear programming problem, whose objective is either to maximize or minimize the expected cost of reaching a set of target states within n transitions. The result then follows from showing that there exist integer solutions which achieve the maximum and minimum. Below, we consider only the construction of the adversary C (the construction of B follows similarly).

We therefore begin by constructing a set of linear constraints from which we can derive a set of adversaries that behave 'almost' the same as A. More precisely, we consider any sequence of real values $\underline{t} = \langle t_\omega \rangle_{\omega \in Path^A_{fin}}$ which satisfy, for any $\omega \in Path^A_{fin}$, the following constraints:

– if $|\omega| = 0$, then

$$t_\omega \geqslant 0 \qquad (6.1)$$

$$-t_\omega \geqslant 0 \qquad (6.2)$$

– if $|\omega| > 0$, then

$$t_\omega - t_{\omega(k)} \geqslant \lfloor dur(\omega, |\omega|) - dur(\omega, k) \rfloor \quad \text{for all } k < |\omega| \qquad (6.3)$$

$$-t_\omega + t_{\omega(k)} \geqslant -\lceil dur(\omega, |\omega|) + dur(\omega, k) \rceil \quad \text{for all } k < |\omega|. \qquad (6.4)$$

Note that there exists a sequence of values $\underline{t}$ which satisfy these constraints; for example, letting t_ω equal $dur(\omega, |\omega|)$ or $[dur(\omega, |\omega|)]_\varepsilon$ for any $\varepsilon \in [0,1]$ gives one possible solution.

Now suppose that we fix a sequence of real values $\underline{t}$ which satisfy the above constraints. From these values we can construct an adversary $C_{\underline{t}}$ which 'almost' matches the behaviour of A. The set of finite paths of $C_{\underline{t}}$ is given by $\{[\omega]_{\underline{t}} \,|\, \omega \in Path^A_{\mathit{fin}}\}$, which we define inductively as follows: if $\omega = (\bar{l}, \mathbf{0})$, then $[\omega]_{\underline{t}} \overset{\text{def}}{=} \omega$, and if ω is of the form $\omega' \xrightarrow{a,\mu} (l,v)$, then:

$$[\omega]_{\underline{t}} \overset{\text{def}}{=} [\omega']_{\underline{t}} \xrightarrow{a',\mu'} (l, v')$$

where

– $v'(x) = t_\omega - t_{\omega(j)}$ and $j \leqslant |\omega|$ such that $v(x) = dur(\omega, |\omega|) - dur(\omega, j)$, which exists by Lemma 18;
– if $a \in \Sigma$, $last([\omega']_{\underline{t}}) = (\tilde{l}, \tilde{v})$ and μ' is constructed from a probabilistic edge of PTA which is of the form $(\cdot, \cdot, a, p)$, then $a' = a$ and for any $(l'', v'') \in L \times \mathbb{N}^{\mathcal{X}}$:

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

– if $a \in \mathbb{R}$, then $a' = t_\omega - t_{\omega'}$ and $\mu' = \mu_{(l,v')}$.

Following the approach used in Proposition 23, we can then use these paths to construct an adversary $C_{\underline{t}}$. The fact that $C_{\underline{t}}$ is an adversary of $[\![\mathsf{PTA}]\!]_{\mathbb{N}}$ follows from Lemma 18, equations (6.1)–(6.4) and since we restrict attention to closed, diagonal-free probabilistic timed automata. For any state $(l, v) \in S$, let $r(l, v) = r(l)$. Now, from the construction of $C_{\underline{t}}$ and Definition 31, it follows that $e_n^{C_{\underline{t}}}(\mathcal{C}_{c_\Sigma, r}, F_{\mathbb{R}}, \bar{s})$ equals:

$$\begin{aligned} &\sum_{\substack{\omega \in Path^A_{\mathit{fin}} \wedge |\omega| \leqslant n \\ \wedge \forall i \leqslant |\omega| .\, \omega(i) \notin F_{\mathbb{R}}}} Prob^{C_{\underline{t}}}(\omega) \cdot (t_\omega - t_{\omega^{|\omega|-1}}) \cdot r(last(\omega)) + e_n^{C_{\underline{t}}}(\mathcal{C}_{c_\Sigma, 0}, F_{\mathbb{R}}, \bar{s}) \\ &= \sum_{\substack{\omega \in Path^A_{\mathit{fin}} \wedge |\omega| \leqslant n \\ \wedge \forall i \leqslant |\omega| .\, \omega(i) \notin F_{\mathbb{R}}}} Prob^{A}(\omega) \cdot (t_\omega - t_{\omega^{|\omega|-1}}) \cdot r(last(\omega)) + e_n^{A}(\mathcal{C}_{c_\Sigma, 0}, F_{\mathbb{R}}, \bar{s}) \qquad (7) \end{aligned}$$

since A and $C_{\underline{t}}$ make the same discrete choices.

Now, suppose we fix some $n \in \mathbb{N}$ and consider the following linear programming problem over the variables $\langle t_\omega \rangle_{\omega \in Path^A_n(\bar{s})}$, where $Path^A_n(\bar{s})$ is the set of paths of $Path^A_{\mathit{fin}}(\bar{s})$ with length at most n: maximize (7) such that the constraints given by (6.1)-(6.4) are satisfied. First note that, under the assumption that all costs and probabilities are rational, we can scale the objective function such that it contains only integer values. Furthermore, from the construction of the constraints it follows that the corresponding matrix is *totally unimodular* (using Theorem 29 and the fact that, for any collection of columns of the constraint matrix, the sum of the columns is a vector with entries only 0, $+1$ and -1). Therefore, using Theorem 30, it follows that the maximum solution is achieved by an integer vector. More precisely, there exists an adversary $C \in Adv_{[\![\mathsf{PTA}]\!]_{\mathbb{N}}}$ such that $e^A_n(\mathcal{C}_{c_\Sigma,r}, F_{\mathbb{R}}, \bar{s}) \leqslant e^C_n(\mathcal{C}_{c_\Sigma,r}, F_{\mathbb{N}}, \bar{s})$ as required. □

Theorem 34. *For any (closed, diagonal-free) probabilistic timed automaton* PTA *(where all probability values are rational), set of locations* $F \subseteq L$ *and (non-negative) cost function* $\mathcal{C}_{c_\Sigma,r}$ *with rational coefficients, we have:*

$$\begin{aligned} e^{\max}_{[\![\mathsf{PTA}]\!]_{\mathbb{R}}}(\mathcal{C}_{c_\Sigma,r}, F_{\mathbb{R}}) &= e^{\max}_{[\![\mathsf{PTA}]\!]_{\mathbb{N}}}(\mathcal{C}_{c_\Sigma,r}, F_{\mathbb{N}}) \\ e^{\min}_{[\![\mathsf{PTA}]\!]_{\mathbb{R}}}(\mathcal{C}_{c_\Sigma,r}, F_{\mathbb{R}}) &= e^{\min}_{[\![\mathsf{PTA}]\!]_{\mathbb{N}}}(\mathcal{C}_{c_\Sigma,r}, F_{\mathbb{N}}) \ . \end{aligned}$$

Proof. From Lemma 33 it follows that for any $n \in \mathbb{N}$ and adversary $A \in Adv_{[\![\mathsf{PTA}]\!]_{\mathbb{R}}}$:

$$\inf_{B \in Adv_{[\![\mathsf{PTA}]\!]_{\mathbb{N}}}} e^B_n(\mathcal{C}_{c_\Sigma,r}, F_{\mathbb{N}}, \bar{s}) \leqslant e^A_n(\mathcal{C}_{c_\Sigma,r}, F_{\mathbb{R}}, \bar{s}) \leqslant \sup_{C \in Adv_{[\![\mathsf{PTA}]\!]_{\mathbb{N}}}} e^C_n(\mathcal{C}_{c_\Sigma,r}, F_{\mathbb{N}}, \bar{s}) \, ,$$

and hence, taking limits as n tends to infinity together with Lemma 32, we have:

$$\begin{aligned} \lim_{n\to\infty} \inf_{B \in Adv_{[\![\mathsf{PTA}]\!]_{\mathbb{N}}}} e^B_n(\mathcal{C}_{c_\Sigma,r}, F_{\mathbb{N}}, \bar{s}) &\leqslant e^A_{\bar{s}}(\mathcal{C}_{c_\Sigma,r}, F_{\mathbb{R}}) \\ &\leqslant \lim_{n\to\infty} \sup_{C \in Adv_{[\![\mathsf{PTA}]\!]_{\mathbb{N}}}} e^C_n(\mathcal{C}_{c_\Sigma,r}, F_{\mathbb{N}}, \bar{s}) \, . \end{aligned}$$

Now, from [22, 23], we need only consider the (finite) subset of simple adversaries[1] when verifying expected reachability properties for $[\![\mathsf{PTA}]\!]_{\mathbb{N}}$. It then follows, again using Lemma 32, that:

$$\inf_{B \in Adv_{[\![\mathsf{PTA}]\!]_{\mathbb{N}}}} e^B_{\bar{s}}(\mathcal{C}_{c_\Sigma,r}, F_{\mathbb{N}}) \leqslant e^A_{\bar{s}}(\mathcal{C}_{c_\Sigma,r}, F_{\mathbb{R}}) \leqslant \sup_{C \in Adv_{[\![\mathsf{PTA}]\!]_{\mathbb{N}}}} e^C_{\bar{s}}(\mathcal{C}_{c_\Sigma,r}, F_{\mathbb{N}}) \, .$$

The proof then concludes similarly to that of Theorem 27. □

[1] An adversary B is simple, if for any finite paths π, π' such that $last(\pi) = last(\pi')$ we have $B(\pi) = B(\pi')$.

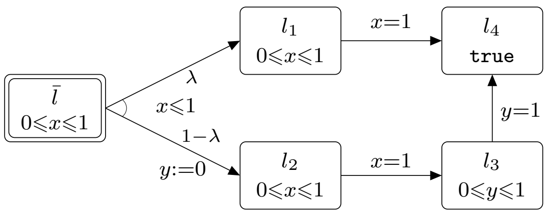

$\bar{l}$
$0\leqslant x\leqslant 1$
$x\leqslant 1$
λ
$1-\lambda$
$y:=0$
l_1
$0\leqslant x\leqslant 1$
l_2
$0\leqslant x\leqslant 1$
$x=1$
$x=1$
l_4
true
l_3
$0\leqslant y\leqslant 1$
$y=1$

Figure 3. Example demonstrating that probabilistic stopwatch-bounded reachability properties are not preserved.

5. Limitations of Digital Clocks

In this section we investigate the limitations of digital clocks when analysing probabilistic timed automata. In particular, in Section 5.1 we show that the integral-time semantics does not preserve probabilistic stopwatch-bounded reachability properties, while in Section 5.2 we show that the integral-time semantics does not preserve the satisfaction of the probabilistic timed temporal logic PTCTL [38].

5.1. Probabilistic Stopwatch-Bounded Reachability

We now show, by means of a counter-example, that the integral-time semantics does not preserve probabilistic stopwatch-bounded reachability properties; that is, properties concerned with the probability of reaching a certain set of states before the time spent in a certain set of locations reaches a bound. This means that properties such as ‘the probability that a message is correctly delivered while spending at most T time units waiting for an acknowledgement is greater than 0.9’ cannot, in general, be verified correctly using the integral-time semantics.

Consider the probabilistic timed automaton of Figure 3, and suppose that we associate a stopwatch with this automaton which increases at the same rate as real-time (‘running’) in the locations l_1 and l_3, and remains constant (‘stopped’) in all other locations. The property we consider is the minimum probability of reaching the location l_4 while the stopwatch (i.e. time spent in l_1 and l_3) remains (less than or) equal to zero. First, under the integral-time semantic model, the transition from $\bar{l}$ can be taken either when the clock x equals 0 or 1.

- If x equals 0 when the transition from $\bar{l}$ is taken, then, on the upper branch, 1 time unit is spent in location l_1 before the location l_4 is reached, while, on the lower branch, 0 time units are spent in location l_3 before the location l_4 is reached.
- If x equals 1 when the transition from $\bar{l}$ is taken, then, on the upper branch, 0 time units are spent in location l_1 before the location l_4 is

reached, while, on the lower branch, 1 time unit is spent in location l_3 before the location l_4 is reached.

It then follows that, under the integral-time semantic model, the minimum probability of reaching l_4 while the stopwatch remains equal to zero is $\min(\lambda, 1-\lambda)$.

On the other hand, for the dense-time semantic model, suppose that the transition from $\bar{l}$ is taken when $x = \delta \in (0, 1)$. Then, on the upper branch, $1 - \delta(>0)$ time units are spent in location l_1 before the location l_4 is reached, while, on the lower branch, $\delta(>0)$ time units are spent in location l_3 before the location l_4 is reached. Therefore, for the dense-time semantic model, the minimum probability of reaching location l_4 while the stopwatch remains equal to zero is 0 (consider any adversary which lets some $\delta \in (0, 1)$ time units pass before taking the discrete transition from $\bar{l}$), and hence the minimum probabilities in the integral and dense-time semantic models disagree.

Note that, for the corresponding minimum and maximum expected reachability properties, i.e. the expected time spent in the locations l_1 and l_3 until the location l_4 is reached, the integral and dense-time models agree. More precisely, for the cost function $\mathcal{C}_{c_\Sigma,r}$ such that $c_\Sigma(l, a) = 0$ for all $l \in L$ and $a \in \Sigma$ and $r(l) = 1$ if $l \in \{l_1, l_3\}$ and 0 otherwise, then, for both the integral and dense-time semantics, the minimum and maximum expected cost of reaching location l_4 equal $\min(\lambda, 1-\lambda)$ and $\max(\lambda, 1-\lambda)$ respectively.

5.2. THE LOGIC PTCTL

PTCTL is a combination of two extensions of the branching temporal logic CTL, the real-time temporal logic TCTL [31] and the probabilistic temporal logic PCTL [27, 15]. The logic TCTL can express timing constraints referring to the clocks of the probabilistic timed automaton and a new set of *formula clocks*, and includes the reset quantifier $z.\phi$, used to reset the formula clock z so that ϕ is evaluated from a state at which $z = 0$. PTCTL is obtained by enhancing TCTL with the probabilistic operator $\mathcal{P}_{\sim\lambda}[\cdot]$ from PCTL. In the derived logic we can express properties such as 'with probability 0.95 or greater, the value of the system clock x does not exceed 3 before 8 time units have elapsed', which is represented as the PTCTL formula $z.\mathcal{P}_{\geqslant 0.95}[(x\leqslant 3)\ \mathcal{U}\ (z=8)]$. For details on the formal syntax and semantics of PTCTL see, for example, [38].

Note that, if we restrict attention to formulae which do not contain nested $\mathcal{P}_{\sim\lambda}[\cdot]$ operators, it is straightforward to extend our results to show that the satisfaction, in the initial state, of such formulae is preserved by the integral semantics (under the restriction that all clock constraints appearing in the formula are closed and diagonal-free). This result follows from the fact that the satisfaction of such a formula reduces to computing either the maximum or minimum reachability probability on a closed diagonal-free probabilistic

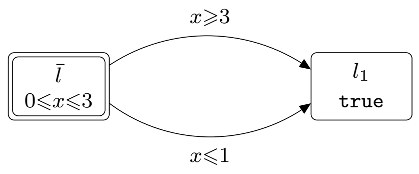


Figure 4. Example demonstrating that the satisfaction of PTCTL formulae is not preserved.

timed automaton. However, as the following example demonstrates, digital clocks do not suffice for the verification of PTCTL formulae containing *nested* $\mathcal{P}_{\sim\lambda}[\cdot]$ operators.

Consider the (probabilistic) timed automaton given in Figure 4. In the integral-time semantic model, no state of the corresponding timed probabilistic system satisfies the formula $\phi = z.\mathcal{P}_{<1}[\texttt{true}\ \mathcal{U}\ (loc_{l_1} \wedge z \leqslant 1)]$ (for all adversaries the probability of reaching the location l_1 within 1 time unit is less than 1). More precisely, if the automaton is in location $\bar{l}$ and the clock x has an integer value, then there exists an adversary such that (with probability 1) location l_1 is reached within 1 time unit. Therefore, since no state of the integral-time semantic model can reach a state satisfying ϕ, the formula $\mathcal{P}_{<1}[\texttt{true}\ \mathcal{U}\ \phi]$ (for all adversaries the probability of reaching a state satisfying the formula ϕ is less than 1) is trivially true in the initial state $(\bar{l}, \mathbf{0})$.

On the other hand, for the dense-time semantics, when the automaton is in location $\bar{l}$ and the clock x is in the interval $(1, 2)$, then l_1 cannot be reached without letting more than 1 time unit elapse. Hence, starting from the initial state $(\bar{l}, \mathbf{0})$ and letting time pass until $x \in (1, 2)$ the formula ϕ becomes true, and thus $\mathcal{P}_{<1}[\texttt{true}\ \mathcal{U}\ \phi]$ is not satisfied in the initial state.

6. Case Studies

In this section, we illustrate the utility of the integral-time semantics of probabilistic timed automata by considering three case studies, where all experiments were performed using the probabilistic symbolic model checker PRISM. Further details on the case studies, including the model checking statistics, can be found on the PRISM web page [47]. This section also includes experimental results to compare the performance of the techniques described in this paper with alternative approaches from the literature.

6.1. ZEROCONF DYNAMIC CONFIGURATION PROTOCOL FOR IPV4 LINK-LOCAL ADDRESSES

The first case study concerns the ZeroConf dynamic configuration protocol for IPv4 link-local addresses [18], which offers a distributed ‘plug-and-play’ solution in which IP address configuration is managed by individual devices connected to a local network. This work extends the results presented in [37]. The protocol has also been studied in [17, 54, 6].

The aim of the protocol is to configure an IP address for a device which newly joins the local network. The IP address is then used to facilitate local communication between the devices of the network. Henceforth, we refer to devices which partake of the protocol as *hosts*. When a host connects to the network, it first randomly selects an IP address from a pool of 65024 available addresses (the Internet Assigned Number Authority has allocated the addresses from 169.254.1.0 to 169.254.254.255 for the purpose of such link-local networks). The host waits a random time of between 0 and 2 seconds before starting to send four *Address Resolution Protocol* (ARP) packets, called *probes*, to all of the other hosts of the network. Probes contain the IP address selected by the host, operate as requests to use the address, and are sent at 2 second intervals. A host which is already using the address will respond with an ARP reply packet, asserting its claim to the address, and the original host will restart the protocol by reconfiguring, where reconfiguration involves randomly choosing a new address and sending new probes. Each time a host witnesses an ARP packet with an address which conflicts with the address that it has chosen, a counter is incremented. If the counter reaches the value 10, then the host 'backs off' and remains idle for at least one minute. If the host sends four probes without receiving an ARP reply packet, then it commences to use the chosen IP address. The host also sends confirmation of this fact to the other hosts of the network by means of two further messages, called *gratuitous* ARPs, which are also sent at 2 second intervals. The protocol has an inherent degree of redundancy, for example with regard to the number of repeated ARP packets sent, in order to cope with message loss. Indeed, message loss makes possible the undesirable situation in which two or more hosts use the same IP address simultaneously.

A host which has commenced using an IP address must reply to ARP packets containing the same IP addresses that it receives from other hosts. It continues using the address unless it receives any ARP packet other than a probe (for example, a gratuitous ARP) containing the IP address that it is using currently. In such a case, the host can either *defend* its IP address, or *defer* to the host which sent the conflicting ARP packet. The host may only defend its address if it has not received a previous conflicting ARP packet within the previous ten seconds; otherwise it is forced to defer. A defending host replies by sending an ARP packet, thereby indicating that it is using, and will continue to use, the IP address. A deferring host does not send a reply; instead, it ceases using its current IP address, and reconfigures its IP address by restarting the protocol.

As in [54], we assume a broadcast-based communication medium with no routers (for example, a single wire), in which messages arrive in the order in which they are sent. In contrast to the analytic analysis of the protocol by Bohnenkamp et al. [17], we model the possibility that a device could surrender an IP address that it is using to another host; and in contrast to the timed-automata-based analysis of Zhang and Vaandrager [54], we model some important probabilistic characteristics of the protocol, and consider parameters more faithful to the standard (such as the maximum number of times a device can witness an ARP packet with the same IP address as that which it wishes to use before 'backing off' and remaining idle for at least one minute).

In the standard [18], there is no mention of what a host should do with messages corresponding to its current IP address (i.e. the probes and gratuitous ARP packets specified in the standard) which are in its output buffer (i.e. those that have yet to be sent), when it reconfigures (chooses a new IP address). However, when the host does reconfigure, unless it picks the same IP address, which happens with a very small probability ($1/65024$), these messages are not relevant. In fact, such messages will slow down the network and may even make hosts reconfigure when they do not need to. We therefore considered two different versions of the protocol: one where the host leaves the messages within its output buffer (`NoReset`) and another where the host clears its buffer when it is about to choose a new IP address (`Reset`).

6.1.1. *Modelling the dynamic configuration protocol*

We consider in detail one *concrete host*, which is attempting to configure an IP address for a network in which there are N *abstract hosts* which have already configured IP addresses. These hosts are called abstract because we do not study their behaviour in depth. When the concrete host picks an address, the probability of this address being *fresh* (not in use by an abstract host) is $(65024{-}N)/65024$. We also assume that the concrete host never picks the same IP address twice, as this happens with a very small probability. Also, the (continuous) uniform choice over [0,2], made by the concrete host to determine the delay before it sends its first probe, is abstracted to a discrete uniform choice over $\{0,1,2\}$.

To enable the analysis of the protocol under various network scenarios we consider two different values of N corresponding to networks with a different numbers of hosts. More precisely, we consider the cases when $N{=}1000$ (the value taken in [17]) and $N{=}20$.

Following the above assumptions, we require only three *abstract* IP addresses:

0 - an address of an abstract host which the concrete host previously chose;

1 - an address of an abstract host which is the concrete host's current choice;

2 - a fresh address which is the concrete host's current choice.

Table 1. Integer variables used in the probabilistic timed automata

variable	description	range
$coll$	the number of collisions detected by the concrete host	$0 \dots 10$
iph	the current address of the concrete host	$1 \dots 2$
$defend$	equals 1 when the host is defending its address	$0 \dots 1$
$probes$	the number of probes/ARPs sent by the concrete host	$0 \dots K$
ip	the address of the ARP packet currently being sent	$0 \dots 2$
n	the number of packets in the concrete host's output buffer	$0 \dots 8$
$b[i]$	the address of packet i in the concrete host's output buffer	$0 \dots 2$
m_0	the number of packets containing an IP address of type 0 in the buffers of the abstract hosts	$0 \dots 20$
m_1	the number of packets containing an IP address of type 1 in the buffers of the abstract hosts	$0 \dots 8$

As in the standard [18], we suppose that it takes between 0 and 1 seconds to send a packet between hosts (where the choice of the exact time delay is nondeterministic).

Since we suppose that the abstract hosts always defend their addresses and have already picked their IP address, the concrete host will never receive probes. Therefore, we do not need to record the type of message being sent, but instead only the IP address in the message, and whether it is sent from the concrete host to the abstract hosts or vice versa.

As in [54], we consider the case in which hosts use output buffers to store the packets they want to send. We have chosen the size of the buffers such that the probability of any buffer becoming full is negligible. We suppose that the concrete host can send a packet to all the abstract hosts at the same time, and only one of the abstract hosts can send a packet to the concrete host at a time.

Variables. The set of variables of our probabilistic timed automata includes both clocks (x, y and z) and *integer variables* which are described in Table 1. Note that the range of the integer variable $probes$ is changed for different verification instances, and since the abstract IP address 2 corresponds to a fresh address chosen by the concrete host we need only two buffers for the abstract hosts (corresponding to addresses of type 0 and 1).

Events. We model the protocol using two probabilistic timed automata, one for the concrete host and one for the environment (which comprises both abstract hosts and the output buffer of the concrete host); these models communicate through the three events *rec*, *send* and *reset* as described in Table 2.

Table 2. Events used in the probabilistic timed automata

event	description
rec	concrete host receives an ARP packet from the environment
send	concrete host sends an ARP packet to the environment
reset	concrete host (re)configures a new IP address
urgent	the environment starts sending a packet

In Table 2 we have also included the event *urgent*, which, although it is not a communicating event, is urgent (no time can pass if it is enabled), since a packet should be sent as soon as it is possible. Note that, in the model `Reset`, we use the event *reset* to model the environment resetting the packets in the buffers of both the concrete and abstract hosts when the concrete host reconfigures.

6.1.2. *Probabilistic timed automata for the protocol*

In the following, we describe the modelling of the `Reset` version of the protocol only. Recall that we use two probabilistic timed automata, one to model the concrete host and one to model the environment (which contains the abstract hosts and the output buffers of *all* hosts). Note that, in the description below, we have omitted the labelling of the non-urgent events on which the two automata do not synchronize, that is, the non-urgent events which do not appear in the set of events of both automata.

The concrete host. The model for the concrete host is shown in Figure 5. The host commences in the location RECONF (the double border indicates the initial location). The location RECONF is a committed location, and therefore must be left immediately. In RECONF, the host chooses a new IP address by moving to the location CHOOSE if it has experienced less than ten address collisions, and to CHOOSEWAIT otherwise. These transitions are labelled with the event *reset* to inform the environment that the host's buffer is to be reset (all messages in its buffer are to be removed).

In both CHOOSE and CHOOSEWAIT, the address selection is represented by the assignment $iph := \mathbf{RAND}(1,2)$, which corresponds to the host randomly selecting an IP address. That is, with probability $N/65024$ the host sets iph equal to 1 (selects an IP address already in use) and, with probability $(65024{-}N)/65024$, sets iph equal to 2 (selects a fresh IP address). Note that, in CHOOSEWAIT, since the host has already experienced at least ten address collisions, it waits 60 seconds before choosing a new address. The assignment to the clock x (a uniform choice between $\{0,1,2\}$), which labels the transitions from CHOOSE and CHOOSEWAIT to WAITSP,

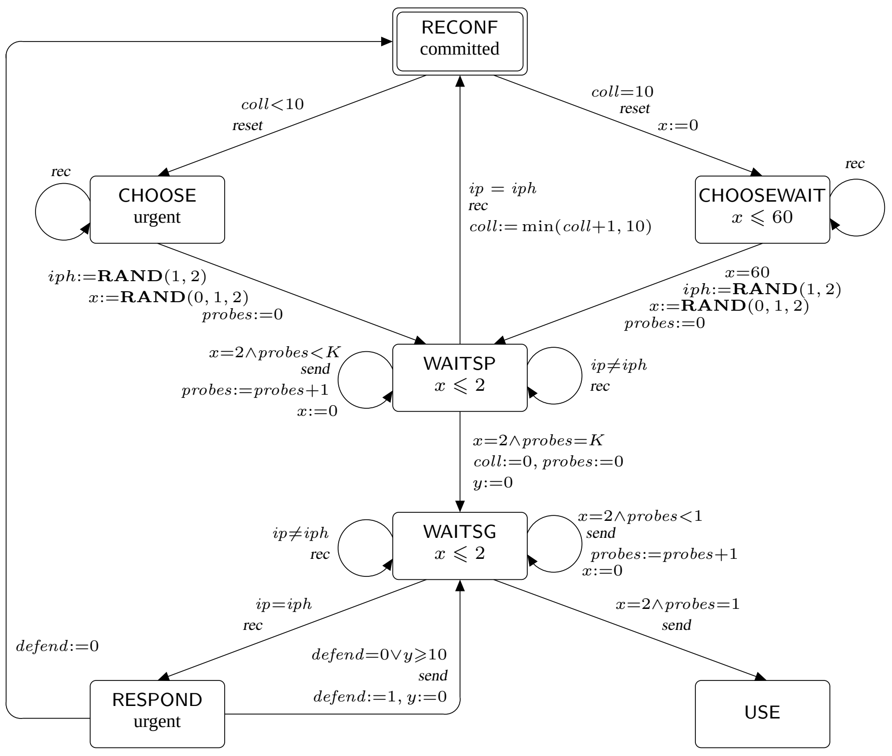


Figure 5. Probabilistic timed automaton for the concrete host (model `Reset`)

approximates the random delay of between 0 and 2 made by the host before sending the first probe.

In the location WAITSP, the host sends K probes at 2 second intervals (denoted by the self-loop labelled with *send*). The host may also receive packets by means of the event *rec*. If it receives a packet which has a different IP address $(ip \neq iph)$, then the host ignores the packet (and remains in WAITSP); however, if the packet has the same address, the host immediately reconfigures (moves to RECONF).

After sending the Kth probe, the host remains in location WAITSP for 2 seconds before moving to WAITSG. The host then sends two ARPs separated by a delay of 2 seconds. Note that the variable *probes* is used to count these ARPs. After these ARPs have been sent, the host moves to USE. However, if while in WAITSG the host receives a packet with the same IP address, it moves to RESPOND. In this location, the host can decide to reconfigure (return to RECONF), or defend its IP address (by sending an ARP packet) if it has either not yet defended the address $(defend = 0)$ or 10 seconds have passed since it previously defended the address $(y \geqslant 10)$. This defence takes the form of the sending of a defending packet, as denoted by the *send* labelled

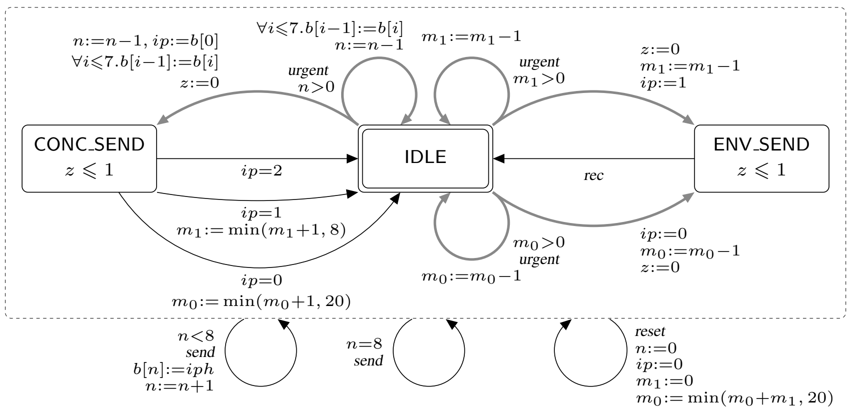


Figure 6. Probabilistic timed automaton for the environment (model `Reset`)

transition from RESPOND to WAITSG. Note that the clock y cannot actually reach 10 in the location WAITSG, and therefore this transition is only enabled if the address has not been defended, although to be faithful to the standard we have not removed this condition.

The environment. The model for the environment is shown in Figure 6. The probabilistic transitions are indicated by thicker grey arrows. The dotted box labelled with three transitions which surrounds the model denotes that these transitions are available in *all* of the locations of the model. More precisely, in all locations, the environment may receive a *send* event from the concrete host and, if the host's buffer is not full ($n{<}8$), add the corresponding packet to the buffer (otherwise it is lost). Also, in all locations, the environment may receive a *reset* event and clear the buffer of the concrete host ($n := 0$) and, since we assume that the concrete host will never choose the same IP address twice, set the IP address in any packet being sent or to be sent to type 0 (i.e. $ip := 0$, $m_1 := 0$ and $m_0 := \min(m_0{+}m_1, 20)$).

The behaviour of the environment commences in the location IDLE. The transition which probabilistically moves to either IDLE or CONC_SEND corresponds to the sending of a packet from the concrete host's buffer. The *urgent* labelling denotes that the transition should be taken as soon as it is enabled, i.e. it should be taken as soon as there is a packet to send. Similarly, the transitions which move probabilistically to either IDLE or ENV_SEND correspond to an abstract host sending a packet, and are again urgent. There are two such transitions, since the address in the packet can either be of type 0 ($m_0{>}0$) or 1 ($m_1{>}0$). For each of these transitions, the loop (remaining in IDLE) corresponds to the packet being lost by the medium, while the other edge corresponds to the packet being sent correctly (therefore the required buffers are updated when one of these transitions is taken). Note that, since each of

Table 3. Minimum probabilistic reachability results

number of abstract hosts equals 1000						
no. of probes	message loss rate 0.1		message loss rate 0.001		message loss rate 0	
sent (K)	`NoReset`	`Reset`	`NoReset`	`Reset`	`NoReset`	`Reset`
1	5.6e-4	5.6e-4	6.2e-8	6.2e-8	0	0
2	1.1e-4	1.1e-4	1.2e-10	1.2e-10	0	0
3	2.0e-5	2.0e-5	2.5e-13	2.5e-13	0	0
4	3.9e-6	3.9e-6	5.0e-16	5.0e-16	0	0
5	7.3e-7	7.3e-7	<1.0e-16	<1.0e-16	0	0
6	1.4e-7	1.4e-7	<1.0e-16	<1.0e-16	0	0

number of abstract hosts equals 20						
no. of probes	message loss rate 0.1		message loss rate 0.001		message loss rate 0	
sent (K)	`NoReset`	`Reset`	`NoReset`	`Reset`	`NoReset`	`Reset`
1	1.1e-5	1.1e-5	1.2e-9	1.2e-9	0	0
2	2.1e-6	2.1e-6	2.4e-12	2.4e-12	0	0
3	4.0e-7	4.0e-7	4.9e-15	4.9e-15	0	0
4	7.6e-8	7.6e-8	<1.0e-16	<1.0e-16	0	0
5	1.4e-8	1.4e-8	<1.0e-16	<1.0e-16	0	0
6	2.8e-9	2.8e-9	<1.0e-16	<1.0e-16	0	0

these transitions corresponds to a message from a different host, when more than one of these transitions is enabled, there is a nondeterministic choice as to which one is taken. We vary the probability of message loss depending on the verification instance. Once in either $\mathsf{CONC\_SEND}$ or $\mathsf{ENV\_SEND}$, after a delay of between 0 and 1 seconds, the model returns to IDLE; this corresponds to the message taking between 0 and 1 seconds to send.

6.1.3. *Performance Analysis*

In this section, we outline our results of using PRISM to verify the integral-time models of the probabilistic timed automata of the dynamic configuration protocol given in Section 6.1.2. In the experiments, as explained above we consider the cases when the number of hosts (N) equals 1000 and 20, and vary both the number of probes a host sends (K) and the probability of message loss.

Note that, because we have abstracted certain aspects of the network (for example, the time taken to send a message), the presented results will give upper and lower bounds on the performance of the protocol, for example the actual reachability probability will lie between the minimum and maximum reachability probabilities computed for the model under study.

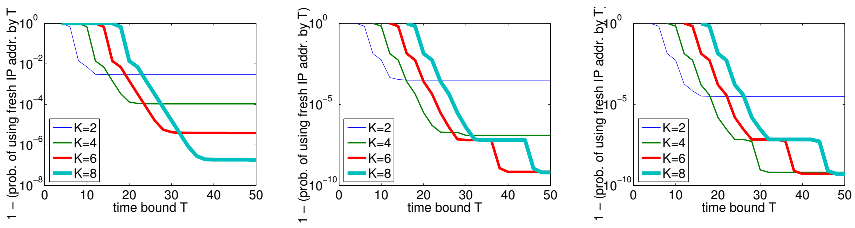
Table 4. Maximum probabilistic reachability results

number of abstract hosts equals 1000						
no. of probes	message loss rate 0.1		message loss rate 0.001		message loss rate 0	
sent (K)	`NoReset`	`Reset`	`NoReset`	`Reset`	`NoReset`	`Reset`
1	0.01538	0.0154	0.0154	0.0154	0.0154	0.0154
2	0.00304	0.00296	1.9e-4	3.1e-5	1.6e-4	0
3	6.1e-4	5.6e-4	8.0e-5	6.2e-8	8.0e-5	0
4	1.1e-4	1.1e-4	1.2e-6	1.2e-10	1.2e-6	0
5	2.0e-5	2.0e-5	4.1e-7	2.5e-13	4.2e-7	0
6	3.9e-6	3.9e-6	8.4e-9	5.0e-16	8.5e-9	0
number of abstract hosts equals 20						
no. of probes	message loss rate 0.1		message loss rate 0.001		message loss rate 0	
sent (K)	`NoReset`	`Reset`	`NoReset`	`Reset`	`NoReset`	`Reset`
1	3.1e-4	3.1e-4	3.1e-4	3.1e-4	3.1e-4	3.1e-4
2	5.8e-5	5.8e-5	6.8e-7	6.2e-7	6.3e-8	0
3	1.1e-5	1.1e-5	3.3e-8	1.2e-9	3.2e-8	0
4	2.1e-6	2.1e-6	1.2e-11	2.5e-12	9.7e-12	0
5	4.0e-7	4.0e-7	3.2e-12	4.9e-15	3.2e-12	0
6	7.6e-8	7.6e-8	1.3e-15	$<$1.0e-16	1.3e-15	0

Probabilistic reachability. The probabilistic reachability property we consider is the (minimum and maximum) probability of the host using an IP address which is already in use by another host. In Tables 3 and 4 we present the minimum and maximum probabilistic reachability results obtained in the cases when the number of abstract hosts (N) is fixed at 1000 and 20.

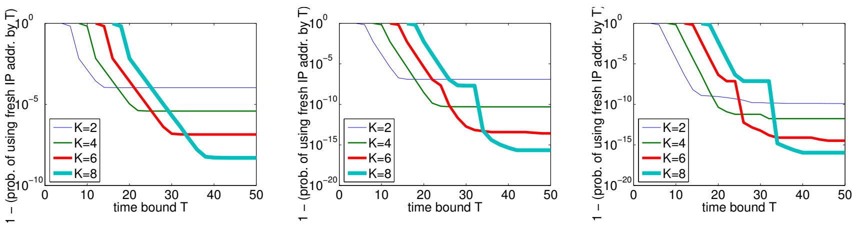
For both models the results demonstrate that the probabilities decrease as the number of probes increases, which is to be expected: if more probes are sent then there is a greater chance of receiving a reply to a probe when an IP address already in use is chosen (i.e. the chance that not all the probes and responses get lost). Furthermore, the probabilities in the case when $N = 20$ are smaller than when $N = 1000$; again this is to be expected, because when N is smaller there are fewer abstract hosts, hence fewer IP addresses in use, and therefore there is a smaller chance of the host choosing an IP address which is already in use.

When the probability of message loss is 0, Table 4 shows that the maximum probability is 0 for the model `Reset` (the model where the host clears its buffer) provided the host sends more than one probe. On the other hand, for the model `NoReset` (when the host does not clear its buffer), even if the host sends more than one probe, this maximum reachability probability is greater than 0. To understand this result, consider the fact that, if a host does not clear



(a) message loss rate 0.1 (b) message loss rate 0.01 (c) message loss rate 0.001

Figure 7. Minimum probabilistic time-bounded reachability results (N=1000)



(a) message loss rate 0.1 (b) message loss rate 0.01 (c) message loss rate 0.001

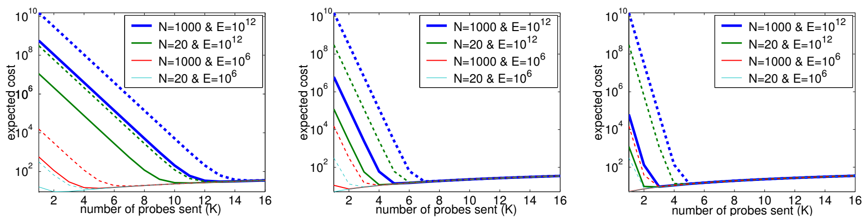
Figure 8. Maximum probabilistic time-bounded reachability results (N=1000)

its buffer, then there is a chance that the probes corresponding to its new IP address will get delayed, and hence the host will not receive a reply to these probes until after it starts using the address (since the probability of message loss is 0, the host will eventually get a reply).

In the cases where the message loss probability is greater than 0, the results presented in Tables 3 and 4 demonstrate that, by allowing the host to clear its buffer, the performance of the protocol improves; that is, the maximum reachability probabilities decrease while the minimum reachability probabilities remain the same.

Time-bounded probabilistic reachability. The time-bounded property we consider is the (minimum and maximum) probability of the host using a fresh IP address within time T. For the model `Reset` when N=1000, the minimum and maximum time-bounded reachability results are presented in Figure 7 and Figure 8, respectively. Note that the graphs use a log scale and plot 1 minus the actual probabilities under study. The time-bounded reachability results when N=20 are similar except that the probabilities are higher and the explanation for this is the same as for the probabilistic reachability case; namely, when N=20 there is a greater chance that the host will choose a fresh IP address.

The results demonstrate that, for small time bounds, the probability of the property is higher when K is smaller. This is to be expected since sending more probes takes more time. However, for larger time bounds the probability



(a) message loss rate 0.1 (b) message loss rate 0.01 (c) message loss rate 0.001

Figure 9. Maximum (dotted) and Minimum (solid) expected cost results

is larger when more probes are sent. This is due to the fact that, when more probes are sent, there is less chance of using an IP address already in use, and hence not reaching a state where the host uses a fresh IP address.

The results for the model `NoReset` are similar to those presented for the `Reset`, although, as for the maximum probabilistic reachability results, the probabilities are larger for the model `NoReset`.

Expected reachability. We consider the expected cost of a host choosing an IP address and using it. As in [17], the cost is defined as the time to start using an IP address plus an additional cost (E) associated with the host using an address which is already in use. We consider two different values for E, namely 10^6 and 10^{12}. Note that the choice of the value of this additional cost will depend on how damaging it is for two hosts to use the same IP address, which in turn depends on the network and the nature of its devices.

The results for the model `Reset` are presented in Figure 9. In each graph a log scale has been used to improve readability. The results for the model `NoReset` are similar, although both the minimum and maximum costs are larger for the model `NoReset` (see [47] for further details). This agrees with our intuition, since, as the results for probabilistic reachability demonstrate, when the host does not clear its buffer, there is a greater chance of using an IP address which is already in use, and hence of incurring a greater cost.

These results are similar to those of [17]: as the message loss probability increases, one must increase the number of probes sent in order to reduce the expected cost; however, if too many probes are sent, the expected cost may start to increase. The rationale for this is that, although increasing the number of probes sent decreases the probability of the host using an IP address which is already in use (that is, decreases the chance of incurring the additional cost), it increases the expected time to choose an IP address (because sending more probes takes more time).

Figure 9 also shows that, as the probability of message loss increases, to minimize the expected costs one must send more probes (increase the value of K). Similarly, the results presented demonstrate the fact that, when the cost of using an IP address which is already in use by another host increases, one must send more probes to minimize the expected cost.

6.2. IEEE 802.11 Wireless LAN

This case study concerns the IEEE 802.11 wireless local area network protocol [33] and focuses on the contention resolution protocol for ‘ad hoc networks’. Such networks comprise a number of stations communicating over a shared channel, in a peer-to-peer manner and without a centralized medium access control protocol to arbitrate requests to transmit on the channel. The protocol includes a randomized, slotted exponential backoff procedure, which is designed to break the symmetry between stations that are repeating previously failed transmissions, i.e. those which collided.

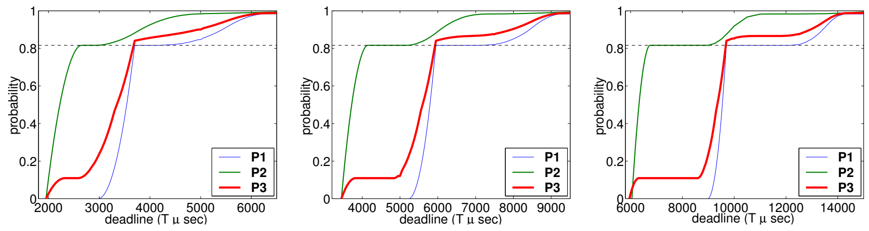
We assume a fixed network topology, consisting of two sending stations and two destination stations, and consider the scenario in which both senders are attempting to send a packet to their destinations. For detailed information on the probabilistic timed automaton model, see [40, 47]. Preliminary results concerning time-bounded probabilistic reachability can also be found in [40]. In our experiments, we investigate both the effect of changing the maximum time it takes for a station to send a packet ($TTmax$), which is specified in the standard as 15,800μsec, and the maximum value which a station's backoff counter can take ($bcmax$), which the standard specifies to be 6. Note that a station increases its backoff counter (up to the value of $bcmax$) each time a collision occurs and that the value of the backoff counter influences the range over which its next backoff is chosen. More precisely, a station chooses its next backoff value uniformly from the range $\{0, 1, \ldots, 2^{k+4}-1\}$ when its backoff counter equals k.

Probabilistic reachability. First we calculate the minimal probability of both stations eventually sending their packet. As expected, this has probability 1, regardless of the values of $TTmax$ and $bcmax$. The second probabilistic reachability property we consider is the maximum probability of the stations colliding at least k times.

The results obtained as the value of $bcmax$ varies are presented in Table 5 (changing the value of $TTmax$ does not influence the results). Observe that the greater the value of the backoff counters, the greater the number of collisions, and hence the longer it takes for a data packet to be sent correctly. We observe that the probability falls rapidly as k increases. This result is to be expected since after each collision (up to the $bcmax$th collision), the range over which each station chooses its backoff grows exponentially. Since a station's backoff counter is only incremented after a collision it follows that, if the stations collide for the ith time, where $i \geqslant 2$, their backoff counters equal

Table 5. Performance results relating to the number of of collisions.

value of	Maximum probability of at least k collisions					max expected
$bcmax$	k=2	k=4	k=6	k=7	k=8	no. of collisions
0	0.183594	0.006188	0.000209	0.000038	7.03e-6	1.224872
1	0.183594	0.001580	0.000014	1.26e-6	1.17e-7	1.202366
2	0.183594	0.000794	1.73e-6	8.05e-8	3.75e-9	1.201459
3	0.183594	0.000794	4.34e-7	1.01e-8	2.37e-10	1.201440
4	0.183594	0.000794	2.17e-7	2.54e-9	2.98e-11	1.201439
5	0.183594	0.000794	2.17e-7	1.27e-9	7.45e-12	1.201439
6	0.183594	0.000794	2.17e-7	1.27e-9	3.72e-12	1.201439



(a) $TTmax = 500$ (b) $TTmax = 1,250$ (c) $TTmax = 2,500$

Figure 10. Model checking results for the time-bounded reachability properties

$\min(i{-}2, bcmax)$ at this point, and hence for any k the results agree for all $bcmax$ such that $bcmax \geqslant k{-}2$.

Time-bounded probabilistic reachability. The time bounded probabilistic reachability properties we consider concern reaching states in which packets have been successfully delivered. More formally, we verify the following properties: the minimum probability of both stations correctly delivering their packets within time T (**P1**); the minimum probability of either station correctly delivering its packet within time T (**P2**); and the minimum probability of station 1 correctly delivering its packet within time T (**P3**).

In Figure 10 we have plotted the results for **P1**–**P3** when using the maximum backoff value taken from the standard ($bcmax$=6) in the cases when $TTmax$ equals 500, $1,250$ and $2,500$. Note that, since stations initially collide with probability 1, the probability will be zero for any deadline which does not allow the stations to collide, enter the backoff procedure, and then resend their data packets. In Figure 10 the dotted line in the graphs corresponds to the minimum probability of a station sending a packet correctly while not entering backoff more than once; the results below this line correspond to deadlines where only the first backoff procedure can influence the outcome (that is, for these deadlines there is insufficient time for a station to enter the

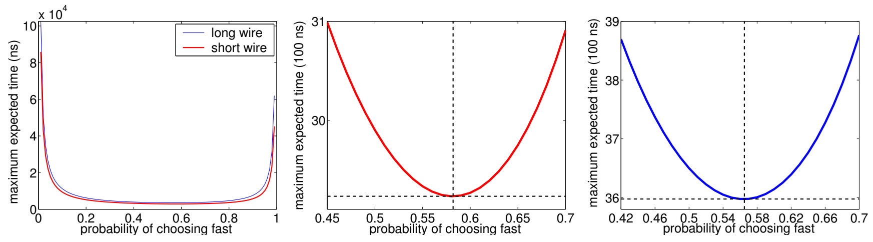
Table 6. Model checking results for the maximum expected time properties

$TTmax$ (μs)	**E1** (Maximum expected time (μ sec) until both stations correctly deliver packets)					
	$bcmax$=0	$bcmax$=1	$bcmax$=2	$bcmax$=3	$bcmax$=4	$bcmax$=6
500	3,792	3,865	3,882	3,883	3,883	3,883
10,000	34,401	34,265	34,274	34,275	34,275	34,275
15,800	52,944	52,677	52,680	52,682	52,682	52,682
$TTmax$ (μs)	**E2** (Maximum expected time (μ sec) until a station correctly delivers a packet)					
	$bcmax$=0	$bcmax$=1	$bcmax$=2	$bcmax$=3	$bcmax$=4	$bcmax$=6
500	2,525	2,551	2,558	2,559	2,559	2,559
10,000	23,636	23,451	23,450	23,451	23,451	23,451
15,800	36,429	36,113	36,107	36,108	36,108	36,108
$TTmax$ (μs)	**E3** (Maximum expected time (μ sec) until station 1 correctly delivers a packet)					
	$bcmax$=0	$bcmax$=1	$bcmax$=2	$bcmax$=3	$bcmax$=4	$bcmax$=6
500	3,322	3,352	3,359	3,360	3,360	3,360
10,000	31,899	31,685	31,679	31,680	31,680	31,680
15,800	49,200	48,839	48,826	48,826	48,826	48,826

backoff procedure more than once and send its data correctly). Furthermore, the portions of the graph where the probability does not increase correspond to deadlines which are not large enough for a station to enter backoff more than once and successfully send its data packet, but are sufficient for all cases when backoff is entered at most once.

Expected reachability. The first expected reachability property concerns the maximum expected number of collisions before both stations correctly send their packets. Similarly to the corresponding probabilistic reachability property (maximum probability of the stations colliding at least k times), the expected number of collisions decreases as the value of $bcmax$ increases.

The remaining expected reachability properties we consider are the maximum expected time until both stations correctly deliver their packets (**E1**); the maximum expected time until either station correctly delivers its packet (**E2**); and the maximum expected time until station 1 correctly delivers its packet (**E3**). In Table 6 we present the results for **E1**–**E3** as $bcmax$ varies using both the maximum transmission delay length from the standard ($15,800$) and two other smaller delays. For each of **E1-E3**, as the maximum value of the backoff counter increases, the expected time initially decreases and then increases slightly. The initial decrease is due to the fact that, as the backoff counter increases, there is less chance of the stations colliding when they attempt to retransmit, and hence the packets are sent sooner. The subsequent increase arises from the fact that for larger values of $bcmax$ the time that stations



(a) both wire lengths (b) short wire (c) long wire

Figure 11. Maximum expected time to elect a leader

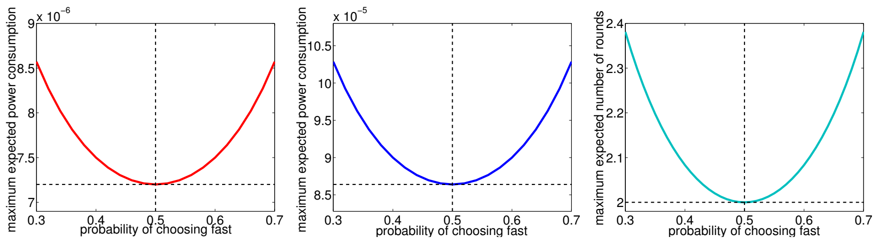
spend in backoff dominates the time that it takes for the stations to collide and retransmit their packets correctly. More precisely, as the backoff counter increases, the decrease in collisions is out-weighed by the increase in the time that each station spends in backoff. The results for the other values of $TTmax$ demonstrate the same pattern and, because for smaller values of $TTmax$ the time to send a packet decreases, the value of the maximum backoff for which the time spent in backoff out-weighs the time needed to retransmit packets becomes smaller.

6.3. IEEE 1394 FireWire Root Contention Protocol

The third case study we consider is the IEEE FireWire root contention protocol [32]. In this protocol, in order to elect a leader (the root), nodes exchange “be my parent” requests with their neighbours. However, *contention* may arise when two nodes simultaneously send “be my parent” requests to each other. The solution adopted by the standard to overcome this conflict is both probabilistic and timed: each node will flip a coin in order to decide whether to wait for a short or for a long time for a request. For details on the probabilistic timed automaton used in this case study see [41, 47].

In our analysis, we will consider two cases for the maximum transmission delay: 360 nanoseconds (ns) and 30ns. This models the distance between the two nodes, i.e. the length of the connecting wires. A delay of 360ns represents the assumption that the nodes are separated by a distance close to the maximum required for the correctness of the protocol (from the analysis of [52]). A delay of 30ns corresponds more closely to the maximum separation specified in the IEEE standard. In the following paragraphs, we will refer to the two cases as ‘long wire’ and ‘short wire’, respectively.

Analysis with respect to probabilistic reachability and time-bounded probabilistic reachability can be found in [41, 20]. We concentrate instead on the effect of using a biased coin with respect to expected reachability in terms of the time to elect a leader, the power consumption before a leader is elected and the number of rounds before a leader is elected. Note that we suppose the nodes in root contention use coins of the same bias. Although it is possible



(a) short wire (b) long wire (c) both wire lengths

Figure 12. Maximum expected costs before a leader is elected

to improve the performance of the protocol by supposing that the nodes' coins have different biases, this is not feasible in practice since each node follows the same procedure and it is not known in advance which nodes of the network will take part in the root contention protocol.

In Figures 11 and 12 we have plotted the effect of using a biased coin on the maximum expected time until a leader is elected, and the maximum expected number of rounds and expected power consumption (assuming that the power usage when sending messages along the wires is 30W and all remaining power consumption is negligible). To consider the expected power consumption we use a cost function which is of the form given in Section 4.4; more precisely, we set the cost (per second) to be 60 in locations where both processes are sending messages, 30 in the locations where only one process is sending a message and 0 otherwise.

The results for expected time reachability demonstrate that the (timing) performance of the root contention protocol can be improved using a biased coin which has a higher probability of flipping 'fast'. The intuition behind this result is that, although the use of such a biased coin decreases the likelihood of the nodes flipping different values, when nodes flip the same values there is a greater chance that less time passes before they flip again (i.e. when both flip 'fast') [53]. There is a compromise here, because as the coin becomes more biased towards 'fast', the probability of the nodes actually flipping different values (which is required for a leader to be elected) decreases even though the delay between coin flips will on average decrease. This decrease in probability is demonstrated in Figure 12 since the expected number of rounds increases as the bias of the coin increases. The results in Figure 11 further demonstrate that, for a shorter wire length, there is a greater advantage when using a biased coin. The reasoning behind this result is that for the short wire length there is a greater saving in time when both nodes flip fast than for a longer wire length, since the time required when both nodes flip fast is dependent on a constant delay given by the protocol specification plus a delay dependent on the wire length.

On the other hand, the expected power consumption and expected number of rounds obtain their minimum for a fair coin. This is unsurprising since, if the nodes used a biased coin, the probability of them flipping the same value would increase, and hence the expected number of rounds would also increase. Similarly, the number of messages will increase if the coins are biased leading to a higher power consumption. Furthermore, because increasing the wire length leads to an increase in the time to send a message, and hence the power consumed when sending a message, the expected power consumption is greater for the longer wire length.

6.4. Comparison with alternative approaches

In this section we compare the digital clocks approach for verifying probabilistic timed automata with techniques already present in the literature[2], namely those based on *forwards reachability* [38, 20] and *backwards reachability* [39, 42]. The following summarises the different approaches:

– the forwards reachability approach is applicable to general probabilistic timed automata, but is restricted to computing upper bounds on maximum reachability probabilities;

– the backwards reachability approach is applicable to general probabilistic timed automata and full PTCTL, but, at the time of writing, there do not exist backwards reachability methods for the computation of expected reachability values;

– as shown in this paper, the digital clocks approach is applicable to closed and diagonal free probabilistic timed automata; it can be used to compute probabilistic and expected reachability measures, but cannot be used to verify general PTCTL formulae.

There are clear advantages in using the backwards and digital clocks methods if one requires a comprehensive analysis of the automaton under study. On the other hand, as one would expect, the applicability of these approaches is restricted by both time and space requirements. It is difficult to provide a fair comparison between these methods especially as one can use the “mature” model checker PRISM for the digital clocks approach, while the other approaches have only prototype implementations. In particular, we could not provide direct comparison for all the examples presented in the previous sections as these models require complex interaction between sub-automata which is not yet available in the prototype implementations. Below we try and give some indication as to the difference in the time and space-complexity of

[2] Note that we do not include results for the *region equivalence* based approach given in [38], as this leads to prohibitively large state spaces even for relatively simple automata.

Table 7. State spaces and MTBDD sizes (KB) when verifying maximal time-bounded reachability probabilities for the FireWire case study

Time bound	backwards [42]		forwards [20]		digital clocks	
(10^3ns)	states	size (KB)	states	size (KB)	states	size (KB)
2	1,219	7.24	825	18.9	80,980	554
4	4,844	30.6	2,329	35.2	434,364	730
6	10,981	55.0	3,833	51.9	1,093,658	860
8	-	-	6,841	74.1	1,915,291	875
10	-	-	9,661	90.1	2,746,691	875
20	-	-	35,041	204	6,903,691	890

the approaches. Note that both the forwards and backwards approaches require a zone-based reachability analysis to construct a finite state probabilistic system followed by the model checking of this system.

In situations where each approach can be applied, the digital clocks approach leads to the largest state space while the forwards and backwards approaches lead to similar model sizes (sometimes the forwards approach leads to a smaller state space, while at other times a smaller state space is generated by the backwards approach). Note that the digital clocks approach is the only one where the size of the model is highly dependent on the constants appearing in the automaton under study; however, the impact of the dependence of the state space size on verification experiments is overcome partially by using the "symbolic" techniques available in PRISM, which take advantage of any regularity of the automaton under study. Note that, because such regularity is to some degree lost in the forwards and backwards approach because only a subset of the state space is constructed, symbolic techniques are not as applicable to these approaches. To illustrate this fact, in Table 7 we present the results for the FireWire case study (see Section 6.3) when computing maximum reachability probabilities. Note that the results presented in Table 7 do not take into account the space required by both the forwards and backwards approaches when performing the initial zone-based reachability analysis.

With regard to the time for model construction, the backwards reachability approach requires complex operations on zones which means that this approach is, in general, the most time intensive. The forwards approach also involves operations on zones but these operations are much simpler; the fact that the forwards approach has been implemented using an adaptation of the established tool KRONOS, whereas the implementation of the backward approach is an early prototype, is another reason why model construction is much faster for the forwards approach. The model construction with the digital clocks approach is more straightforward than the other approaches,

Table 8. Construction and model checking (m/c) times when verifying maximal time-bounded reachability probabilities for the FireWire case study

Time bound	backwards [42]		forwards [20]		digital clocks	
(10^3ns)	construction	m/c	construction	m/c	construction	m/c
2	544+33.0	0.106	0.42+ 0.69	0.383	10.2	7.87
4	26,992+753	0.345	0.94+ 2.08	0.802	38.3	43.9
6	618,493+4,388	1.31	1.64+ 3.76	1.40	85.8	145
8	-	-	2.93+ 10.3	1.60	145	228
10	-	-	4.27+ 20.3	2.54	205	335
20	-	-	18.7+ 226	5.11	549	469

again for the reason that the symbolic techniques of PRISM exploit the regularity which is present in the system. However, due to the larger state space, the construction in the digital clocks case usually takes longer than that of the forwards reachability approach (but less than that of the backwards approach). In terms of model checking times there is no significant difference between the three approaches. Note that, although the digital clocks approach generates much larger state spaces, this does not have a drastic effect on the model checking times because of the regularity in the model and the symbolic approach employed by PRISM. To illustrate these observations, in Table 8 we have presented the model construction and model checking times for the FireWire case study. The model construction times for the forwards and backwards approaches have been separated into the time required by the zone-based reachability computation and the time of the model construction in PRISM.

7. Conclusions

In this paper, we have presented results demonstrating that digital clocks are sufficient for analysing a large class of probabilistic timed automata and performance properties. Since many of today's protocols include both timing and probabilistic behaviour, this approach is applicable to a wide area, which we demonstrated by analysing the performance of three real-world protocols.

In particular, we have demonstrated that digital clocks are sufficient for the analysis of probabilistic (timed-bounded) reachability and expected reachability against closed, diagonal-free probabilistic timed automata. In the case of expected reachability, these results extend to the cases when the rate of cost accumulation varies in different locations, as in priced or weighted timed automata [10, 5]. Furthermore, we have shown the limitations of this approach:

digital clocks are not sufficient for checking stopwatch properties or general PTCTL specifications.

One possible area of future work is to apply the results presented in [13, 14], concerning the verification of classical timed automata using integral semantics and BDDs, to this setting (verifying probabilistic timed automata using integral semantics and MTBDDs) in an attempt to improve efficiency.

There are still limitations in the size of the models that can be considered using digital clocks. In the case of probabilistic reachability a more efficient approach is to consider the symbolic model checking technique for probabilistic timed automata against PTCTL introduced in [42, 39]. However, in the case of expected reachability, and in particular expected time reachability, it is not clear if there is an alternative, since by using zones the exact timing information may be lost, and hence the best one could hope for would be approximate results. The application of zones to the verification of priced timed automata [43] may be instructive to this line of research.

References

1. R. Alur, C. Courcoubetis, and D. Dill. Model-checking in dense real-time. *Information and Computation*, 104(1):2–34, 1993.
2. R. Alur and D. Dill. A theory of timed automata. *Theoretical Computer Science*, 126:183–235, 1994.
3. R. Alur and T. Henzinger. Reactive modules. *Formal Methods in System Design*, 15(1):7–48, 1999.
4. R. Alur, A. Itai, R. Kurshan, and M. Yannakakis. Timing verification by successive approximation. *Information and Computation*, 18(1):142–157, 1995.
5. R. Alur, S. La Torre, and G. Pappas. Optimal paths in weighted timed automata. In Benedetto and Sangiovanni-Vincentelli [11], pages 49–62.
6. S. Andova, H. Hermanns, and J.-P. Katoen. Discrete-time rewards model-checked. In K. Larsen and P. Niebert, editors, *Proc. Formal Modeling and Analysis of Timed Systems (FORMATS'03)*, volume 2791 of *LNCS*, pages 88–104. Springer-Verlag, 2003.
7. E. Asarin, O. Maler, and A. Pnueli. On discretization of delays in timed automata and digital circuits. In R. de Simone and D. Sangiorgi, editors, *Proc. 9th Int. Conf. Concurrency Theory (CONCUR'98)*, volume 1466 of *Lecture Notes in Computer Science*, pages 470–484. Springer-Verlag, 1998.
8. C. Baier and M. Kwiatkowska. Model checking for a probabilistic branching time logic with fairness. *Distributed Computing*, 11:125–155, 1998.
9. G. Behrmann, A. Fehnker, T. Hune, K. Larsen, P. Pettersson, and J. Romijn. Efficient guiding towards cost-optimality in UPPAAL. In T. Margaria and W. Yi, editors, *Proc. 7th Int. Conf. Tools and Algorithms for the Construction and Analysis of Systems (TACAS'01)*, volume 2031 of *Lecture Notes in Computer Science*, pages 174–188. Springer-Verlag, 2001.
10. G. Behrmann, A. Fehnker, T. Hune, K. Larsen, P. Pettersson, J. Romijn, and F. Vaandrager. Minimum-cost reachability for linearly priced timed automata. In Benedetto and Sangiovanni-Vincentelli [11], pages 147–162.

11. M. D. Benedetto and A. Sangiovanni-Vincentelli, editors. *Proc. 4th Int. Workshop on Hybrid Systems: Computation and Control (HSCC'01)*, volume 2034 of *Lecture Notes in Computer Science*. Springer-Verlag, 2001.
12. D. Bertsekas and J. Tsitsiklis. An analysis of stochastic shortest path problems. *Mathematics of Operations Research*, 16(3):580–595, 1991.
13. D. Beyer. Improvements in BDD-based reachability analysis of timed automata. In J. Oliveira and P. Zave, editors, *Proc. Symp. Formal Methods Europe (FME'01)*, volume 2021 of *Lecture Notes in Computer Science*, pages 318–343. Springer-Verlag, 2001.
14. D. Beyer and A. Noack. Efficient verification of timed automata using BDDs. In S. Gnesi and U. Ultes-Nitsche, editors, *Proc. Formal Methods for Industrial Critical Systems (FMICS'01)*, pages 95–113, 2001.
15. A. Bianco and L. de Alfaro. Model checking of probabilistic and nondeterministic systems. In P. Thiagarajan, editor, *Proc. Foundations of Software Technology and Theoretical Computer Science*, volume 1026 of *Lecture Notes in Computer Science*, pages 499–513. Springer-Verlag, 1995.
16. P. Billingsley. *Probability and Measure*. John Wiley and Sons: New York, 1979.
17. H. Bohnenkamp, P. v. d. Stok, H. Hermanns, and F. Vaandrager. Cost-optimisation of the IPv4 zeroconf protocol. In *Proc. Int. Performance and Dependability Symposium (IPDS'03)*, pages 531–540. IEEE Computer Society Press, 2003.
18. S. Cheshire, B. Adoba, and E. Guttman. Dynamic configuration of IPv4 link-local addresses (draft August 2002). Zeroconf Working Group of the Internet Engineering Task Force (www.zeroconf.org).
19. E. 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 *Proc. Int. Workshop on Logic Synthesis (IWLS'93)*, pages 1–15, 1993. Also available in *Formal Methods in System Design*, 10(2/3):149–169, 1997.
20. C. Daws, M. Kwiatkowska, and G. Norman. Automatic verification of the IEEE 1394 root contention protocol with KRONOS and PRISM. *International Journal on Software Tools for Technology Transfer (STTT)*, 5(2–3):221–236, 2004.
21. C. Daws and S. Yovine. Two examples of verification of multirate timed automata with KRONOS. In *Proc. IEEE Real-Time Systems Symposium (RTSS'95)*, pages 66–75. IEEE Computer Society Press, 1995.
22. L. de Alfaro. *Formal Verification of Probabilistic Systems*. PhD thesis, Stanford University, 1997.
23. L. de Alfaro. Computing minimum and maximum reachability times in probabilistic systems. In J. Baeten and S. Mauw, editors, *Proc. 10th Int. Conf. Concurrency Theory (CONCUR'99)*, volume 1664 of *Lecture Notes in Computer Science*, pages 66–81. Springer-Verlag, 1999.
24. C. Derman. *Finite-State Markovian Decision Processes*. Academic Press: New York, 1970.
25. A. Göllü, A. Puri, and P. Varaiya. Discretization of timed automata. In *Proc. 33rd IEEE Conf. Decision and Control*, pages 957–958. IEEE Computer Society Press, 1994.
26. P. Halmos. *Measure Theory*. Springer-Verlag: New York, 1950.
27. H. Hansson and B. Jonsson. A logic for reasoning about time and reliability. *Formal Aspects of Computing*, 6(4):512–535, 1994.
28. T. Henzinger. *The Temporal Specification and Verification of Real-time Systems*. PhD thesis, Stanford University, 1991.
29. T. Henzinger, P.-H. Ho, and H. Wong-Toi. A user guide to HYTECH. In E. Brinksma, R. Cleaveland, and K. Larsen, editors, *Proc. 1st Int. Conf. Tools and Algorithms for Construction and Analysis of Systems (TACAS'95)*, volume 1019 of *Lecture Notes in Computer Science*, pages 41–71. Springer-Verlag, 1995.

30. T. Henzinger, Z. Manna, and A. Pnueli. What good are digital clocks? In W. Kuich, editor, *Proc. 19th Int. Colloquium on Automata, Languages and Programming (ICALP'92)*, volume 623 of *Lecture Notes in Computer Science*, pages 545–558. Springer-Verlag, 1992.
31. T. Henzinger, X. Nicollin, J. Sifakis, and S. Yovine. Symbolic model checking for real-time systems. *Information and Computation*, 111(2):193–244, 1994.
32. IEEE 1394-1995. High Performance Serial Bus Standard. 1995.
33. IEEE 802.11. Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications Standard. 1997.
34. J. Kemeny, J. Snell, and A. Knapp. *Denumerable Markov Chains*. Springer-Verlag: New York, 2nd edition, 1976.
35. M. Kwiatkowska, G. Norman, and D. Parker. PRISM 2.0: A tool for probabilistic model checking. In *Proc. 1st International Conference on Quantitative Evaluation of Systems (QEST'04)*, pages 322–323. IEEE Computer Society Press, 2004.
36. M. Kwiatkowska, G. Norman, and D. Parker. Probabilistic symbolic model checking with PRISM: A hybrid approach. *International Journal on Software Tools for Technology Transfer (STTT)*, 6(2):128–142, 2004.
37. M. Kwiatkowska, G. Norman, D. Parker, and J. Sproston. Performance analysis of probabilistic timed automata using digital clocks. In K. Larsen and P. Niebert, editors, *Proc. Formal Modeling and Analysis of Timed Systems (FORMATS'03)*, volume 2791 of *Lecture Notes in Computer Science*, pages 105–120. Springer-Verlag, 2003.
38. M. Kwiatkowska, G. Norman, R. Segala, and J. Sproston. Automatic verification of real-time systems with discrete probability distributions. *Theoretical Computer Science*, 282:101–150, 2002.
39. M. Kwiatkowska, G. Norman, and J. Sproston. Symbolic computation of maximal probabilistic reachability. In K. Larsen and M. Nielsen, editors, *Proc. 13th Int. Conf. Concurrency Theory (CONCUR'01)*, volume 2154 of *Lecture Notes in Computer Science*, pages 169–183. Springer-Verlag, 2001.
40. M. 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, *Proc. 2nd Joint Int. Workshop Process Algebra and Probabilistic Methods, Performance Modeling and Verification (PAPM/PROBMIV'02)*, volume 2399 of *Lecture Notes in Computer Science*, pages 169–187. Springer-Verlag, 2002.
41. M. Kwiatkowska, G. Norman, and J. Sproston. Probabilistic model checking of deadline properties in the IEEE 1394 FireWire root contention protocol. *Formal Aspects of Computing*, 14:295–318, 2003.
42. M. Kwiatkowska, G. Norman, J. Sproston, and F. Wang. Symbolic model checking for probabilistic timed automata. In Y. Lakhnech and S. Yovine, editors, *Joint Conference on Formal Modelling and Analysis of Timed Systems (FORMATS) and Formal Techniques in Real-Time and Fault Tolerant Systems (FTRTFT)*, volume 3253 of *LNCS*, pages 293–308. Springer, 2004.
43. K. Larsen, G. Behrmann, E. Brinksma, A. Fehnker, T. Hune, P. Pettersson, and J. Romijn. As cheap as possible: Efficient cost-optimal reachability for priced timed automata. In G. Berry, H. Comon, and A. Finkel, editors, *Proc. 13th Int. Conf. Computer Aided Verification (CAV'01)*, volume 2102 of *Lecture Notes in Computer Science*, pages 493–505. Springer-Verlag, 2001.
44. K. Larsen, P. Pettersson, and W. Yi. UPPAAL in a nutshell. *Software Tools for Technology Transfer*, 1(1-2):134–152, 1997.
45. J. Ouaknine. Digitisation and full abstraction for dense-time model checking. In J.-P. Katoen and P. Stevens, editors, *Proc. 8th Int. Conf. Tools and Algorithms for the*

Construction and Analysis of Systems (TACAS'02), volume 2280 of *Lecture Notes in Computer Science*, pages 37–51. Springer-Verlag, 2002.
46. D. Parker. *Implementation of Symbolic Model Checking for Probabilistic Systems*. PhD thesis, University of Birmingham, 2002.
47. PRISM web page. www.cs.bham.ac.uk/~dxp/prism.
48. UPPAAL web page. www.uppaal.com.
49. A. Schrijver. *Theory of Linear and Integer Programming*. J. Wiley and Sons: New York, 1986.
50. R. Segala. *Modelling and Verification of Randomized Distributed Real Time Systems*. PhD thesis, Massachusetts Institute of Technology, 1995.
51. R. Segala and N. A. Lynch. Probabilistic simulations for probabilistic processes. *Nordic Journal of Computing*, 2(2):250–273, 1995.
52. D. 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.
53. M. Stoelinga. *Alea jacta est: verification of probabilistic, real-time and parametric systems*. PhD thesis, University of Nijmegen, the Netherlands, 2002.
54. M. Zhang and F. Vaandrager. Analysis of a protocol for dynamic configuration of IPv4 link local addresses using UPPAAL. Technical Report, NIII-R04XX, University of Nijmegen, 2004.