Performance Analysis of Probabilistic Timed Automata using Digital Clocks*

Marta Kwiatkowska[1], Gethin Norman[1], David Parker[1], and Jeremy Sproston[2]

[1] School of Computer Science, University of Birmingham,
Birmingham B15 2TT, United Kingdom.
{M.Z.Kwiatkowska,G.Norman,D.A.Parker}@cs.bham.ac.uk
[2] 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 specification formalism suitable for describing both nondeterministic and probabilistic aspects of real-time systems, and is amenable to model checking against probabilistic timed temporal logic 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 integer-time model, inducing a notion of *digital clocks*, as opposed to the standard 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. We extend previous results concerning the integer-time semantics of an important subclass of probabilistic timed automata to consider the computation of *expected costs* or *rewards*. We illustrate this approach through the analysis of the dynamic configuration protocol for IPv4 link-local addresses.

1 Introduction

Network protocols increasingly often rely on the use of randomness and timing delays, for example exponential back-off in Ethernet and IEEE 802.11, and IEEE 1394 FireWire root contention. Since these protocols execute in a distributed environment, it is important to also consider nondeterminism when modelling their behaviour. For example, we may wish to model a system for which the likelihood of a certain event occurring changes with respect to the amount of time elapsed. A natural model for systems that exhibit nondeterminism, probability and real-time, called *probabilistic timed automata* – a probabilistic extension of timed automata [1] – has been proposed in [19]. In the probabilistic timed automata model 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 [19] model checking algorithms for verifying the likelihood of certain temporal properties being satisfied by such system models are

* Supported in part by the EPSRC grants GR/N22960 and GR/S11107.

introduced. However, these model checking algorithms are either based on *region equivalence* [1], and hence suffer from the state-space explosion problem, or on *forwards reachability*, which leads to approximate results [19, 11]. An alternative approach, based on *backwards reachability*, is given in [20].

When modelling real-time systems, there is often a trade-off between the expressiveness of the model and the complexity of the associated solution algorithms. A *dense*-time model is more expressive than an *integer*-time model; however, it is often the case that an integer-time model is easier to verify, since it can lead to a finite-state system and allows one to apply efficient symbolic methods developed for untimed systems. We refer to the clocks of an integer-time model as *digital clocks*. Henzinger et al. [15] study the question of which real-time properties can be verified by considering system behaviours featuring only integer durations. These results are applied to timed automata in [9, 24], and it is shown that an approach using digital clocks is applicable to the verification of *closed*, *diagonal-free* timed automata; intuitively, these are automata whose constraints do not compare clocks or use strict comparison operators.

We have previously shown that probabilistic reachability properties, such as 'with probability 0.05 or less, the system aborts', of closed, diagonal-free probabilistic timed automata can be analysed faithfully using digital clocks [22]. The main contribution of this paper is to extend this research by showing that digital clocks are also sufficient for verifying *expected reachability* properties such as 'the expected time until a data packet is delivered is at most 0.05 seconds', or 'the expected cost of a host choosing an IP address is at most 40'.

In [12], de Alfaro presents a model-checking algorithm for verifying probabilistic and expected reachability properties of finite-state models. We implemented the algorithms of de Alfaro in the probabilistic model checking tool PRISM [17, 25], allowing us to automatically verify expected-cost properties of interest for integer-time models.

The paper proceeds by revisiting the definition of probabilistic timed automata in the next section. Expected reachability properties for probabilistic timed automata are presented in Section 3, and the correctness of the digital clock interpretation of probabilistic timed automata with respect to these properties is stated. In Section 4, we present a case study, in which PRISM is used to analyse the performance of the dynamic configuration protocol for IPv4 link-local addresses. Finally, in Section 5, we conclude the paper.

2 Probabilistic Timed Automata

Time, clocks, zones and distributions. 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 point $v \in \mathbb{T}^{|\mathcal{X}|}$ 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* of values in v by t. 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.

Let $Zones(\mathcal{X})$ be the set of *zones* over $\mathcal{X}$, which are conjunctions of atomic constraints of the form $x \sim c$ for $x \in \mathcal{X}$, $\sim \in \{\leq, =, \geq\}$, and $c \in \mathbb{N}$. The clock valuation v *satisfies* the zone ζ, written $v \models \zeta$, 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) for $x, y \in \mathcal{X}$, $c \in \mathbb{N}$.

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 $\mu_q \in \mathsf{Dist}(Q)$ be the distribution which assigns probability 1 to q.

Syntax of probabilistic timed automata. We review the definition of probabilistic timed automata [19].

Definition 1. *A* probabilistic timed automaton *is a tuple* $(L, \bar{l}, \mathcal{X}, \Sigma, I, prob)$ *where: L is a finite set of* locations *including the* initial location $\bar{l}$*; $\mathcal{X}$ is a set of* clocks*; Σ is a finite set of* events*; the function $I : L \to Zones(\mathcal{X})$ is the* invariant condition*; and the finite set $prob \subseteq L \times Zones(\mathcal{X}) \times \Sigma \times \mathsf{Dist}(2^{\mathcal{X}} \times L)$ is the* probabilistic edge relation.

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 $v \models I(l)$. Informally, the behaviour of a probabilistic timed automaton can be understood as follows. The model starts in the state $(\bar{l}, \mathbf{0})$; that is, in the initial location $\bar{l}$ with all clocks set to 0. In any 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 $(l, g, \sigma, p) \in prob$ 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 $I(l)$ is satisfied while time elapses. Note that we often refer to the model presented above as *closed, diagonal-free probabilistic timed automata*, in order to distinguish the zones used with those in previous work [19].

Semantics of probabilistic timed automata. The semantics of probabilistic timed automata is defined in terms of *timed probabilistic systems*, which exhibit timed, nondeterministic and probabilistic behaviour. They are a variant of Markov decision processes [14] and Segala's probabilistic timed automata [26].

Definition 2. *A* timed probabilistic system $\mathsf{PS} = (S, \bar{s}, Act, \mathbb{T}, Steps)$ *consists of a set S of* states*, an* initial state $\bar{s} \in S$*, a set Act of* actions*, a* time domain $\mathbb{T}$*, and a* probabilistic transition relation $Steps \subseteq S \times (Act \cup \mathbb{T}) \times \mathsf{Dist}(S)$.

A *probabilistic transition* $s \xrightarrow{a,\mu} s'$ is made from a state $s \in S$ by first nondeterministically selecting an action-distribution 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 consider two ways in which a timed probabilistic system's computation may be represented. A *path* represents a particular resolution of both nondeterminism *and* probability. Formally, a path of a timed probabilistic system is a finite or infinite sequence of probabilistic transitions $\omega = s_0 \xrightarrow{a_0,\mu_0} s_1 \xrightarrow{a_1,\mu_1} \cdots$. A path ω is *initialised in* s if $s_0 = s$. We denote by $\omega(i)$ the $(i{+}1)$th state of ω, $\mathit{last}(\omega)$ the last state of ω if ω is finite, and $\mathit{step}(\omega, i)$ the action associated with the i-th step. If ω is infinite, the duration up to the $(n+1)$th state of ω is defined by $\mathcal{D}_\omega(n{+}1) \stackrel{\text{def}}{=} \sum\{\!|a_i \mid 0 \leq i \leq n \wedge a_i \in \mathbb{T}|\!\}$. Let $\mathit{Path}_{\mathit{ful}}(s)$ be the set of infinite paths initialised in s.

The second notion of a timed probabilistic system's computations is that of an *adversary*, which represents a particular resolution of nondeterminism *only*. Formally, an adversary is a function A mapping every finite path ω to a pair (a, μ) such that $(\mathit{last}(\omega), a, \mu) \in \mathit{Steps}$ [28]. For any adversary A, let $\mathit{Path}^A_{\mathit{ful}}(s)$ denote the set of infinite paths initialised in s associated with A. Then, we define the probability measure Prob^A_s over $\mathit{Path}^A_{\mathit{ful}}(s)$ by classical techniques [16].

We restrict our attention to *time-divergent adversaries*; a common restriction imposed in real-time systems so that unrealisable behaviour (corresponding to time not advancing beyond a bound) is disregarded during analysis. We say that a path ω is *divergent* if for any $t \in \mathbb{R}$, there exists $j \in \mathbb{N}$ such that $\mathcal{D}_\omega(j) > t$.

Definition 3. *An adversary A of a timed probabilistic system* PS *is* divergent *if and only if for each state s the probability Prob^A_s of the divergent paths of $\mathit{Path}^A_{\mathit{ful}}(s)$ is 1. Furthermore, let $\mathit{Adv}_{\mathsf{PS}}$ be the set of divergent adversaries of* PS.

We now define the semantics of probabilistic timed automata defined in terms of timed probabilistic systems. Observe that the definition is parameterized both by a time domain $\mathbb{T}$ and time increment $\oplus$, and that the summation in the definition of discrete transitions is required for the cases in which multiple clock resets result in the same target location.

Definition 4. *Let* $\mathsf{PTA} = (L, \bar{l}, \mathcal{X}, \Sigma, I, \mathit{prob})$ *be a probabilistic timed automaton. The* semantics of PTA with respect to the time domain $\mathbb{T}$ and the time increment $\oplus$ *is the timed probabilistic system* $[\![\mathsf{PTA}]\!]^{\oplus}_{\mathbb{T}} = (S, \bar{s}, \Sigma, \mathbb{T}, \mathit{Steps})$ *where: $S \subseteq L \times \mathbb{T}^{|\mathcal{X}|}$ and $(l, v) \in S$ if and only if $v \models I(l)$; $\bar{s} = (\bar{l}, \mathbf{0})$; and $((l, v), a, \mu) \in \mathit{Steps}$ if and only if one of the following conditions holds:*

Time transitions. *$a \in \mathbb{T}$, $\mu = \mu_{(l, v \oplus a)}$ and $v \oplus t \models I(l)$ for all $0 \leq t \leq a$;*
Discrete transitions. *$a \in \Sigma$ and there exists $(l, g, \sigma, p) \in \mathit{prob}$ such that $v \models g$ and for any $(l', v') \in S$, we have $\mu(l', v') = \sum_{X \subseteq \mathcal{X} \,\&\, v' = v[X:=0]} p(X, l')$.*

Traditionally, the semantics of probabilistic timed automata assumes that the reals form the underlying model of time, paired with a time increment which is standard addition. The continuous semantics of a probabilistic timed automaton is a timed probabilistic system with generally uncountably many states.

Definition 5. *The* continuous semantics *of a probabilistic timed automaton* PTA *is defined as* $[\![\mathsf{PTA}]\!]^{+}_{\mathbb{R}}$*; that is,* $\mathbb{T} = \mathbb{R}$ *and* $\oplus = +$.

Higher-level modelling. To aid modelling, probabilistic timed automata can be *composed in parallel* [22], and can feature *integer variables*, *urgent locations and events*, and *committed locations* (as in UPPAAL timed automata [3]). The techniques of [27] can be adapted to represent, syntactically, integer variables and committed locations within our definition of probabilistic timed automata; urgent events require a minor adjustment to the semantics of probabilistic timed automata [21].

3 Performance Measures

In this section, we consider two performance measures for probabilistic timed automata. The first is *probabilistic reachability*, namely the maximal and minimal probability of reaching, from the initial state, a certain set of goal or target states. For a timed probabilistic system $\mathsf{PS} = (S, \bar{s}, Act, \mathbb{T}, Steps)$, set $F \subseteq S$ of target states, and adversary $A \in Adv_{\mathsf{PS}}$, let:

$$p_{\bar{s}}^{A}(F) \stackrel{\text{def}}{=} Prob_{\bar{s}}^{A}\{\omega \in Path_{ful}^{A}(\bar{s}) \mid \exists i \in \mathbb{N} . \, \omega(i) \in F\} \, .$$

Definition 6. *The* maximal and minimal reachability probabilities *of reaching the set of states F of the timed probabilistic system* PS *are defined as follows:*

$$p_{\mathsf{PS}}^{\max}(F) \stackrel{\text{def}}{=} \sup_{A \in Adv_{\mathsf{PS}}} p_{\bar{s}}^{A}(F) \quad \textit{and} \quad p_{\mathsf{PS}}^{\min}(F) \stackrel{\text{def}}{=} \inf_{A \in Adv_{\mathsf{PS}}} p_{\bar{s}}^{A}(F) \, .$$

This performance measure has been studied in the context of probabilistic timed automata by Kwiatkowska et al. [19, 22].

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 cost function mapping actions and durations to real values, as well as a set $F \subseteq S$ of target states, and 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 $\mathsf{PS} = (S, \bar{s}, Act, \mathbb{T}, Steps)$, cost function $c : Act \cup \mathbb{T} \to \mathbb{R}$, set $F \subseteq S$ of target states, and adversary $A \in Adv_{\mathsf{PS}}$, let $e_{\bar{s}}^{A}(cost(c, F))$ denote the usual expectation with respect to the measure $Prob_{\bar{s}}^{A}$ over $Path_{ful}^{A}(\bar{s})$, where for any $\omega \in Path_{ful}^{A}(\bar{s})$:

$$cost(c, F)(\omega) = \begin{cases} \sum_{i=1}^{\min\{j \mid \omega(j) \in F\}} c(step(\omega, i{-}11)) & \text{if } \exists j \in \mathbb{N}. \ \omega(j) \in F \\ \infty & \text{otherwise.} \end{cases}$$

The value of $cost(c, F)(\omega)$ equals the total cost, with respect to the cost function c, accumulated until a state in F is reached along the path ω. Note that 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 $c(a) = 0$ for all $a \in Act$ and $c(t) = t$ for all $t \in \mathbb{T}$.

Definition 7. *The* maximal and minimal expected costs *of reaching a set of states F under the cost function c in the timed probabilistic system* PS *are defined as follows:*

$$e^{\max}_{\mathsf{PS}}(c,F) = \sup_{A\in Adv_{\mathsf{PS}}} e^A_{\bar{s}}(cost(c,F)) \quad and \quad e^{\min}_{\mathsf{PS}}(c,F) = \inf_{A\in Adv_{\mathsf{PS}}} e^A_{\bar{s}}(cost(c,F))\,.$$

We note that calculating expected reachability is equivalent to the *stochastic shortest path problem* for Markov decision processes; see for example [6].

At the level of probabilistic timed automata, one can define a cost function using a pair (r, c_Σ), where $r \in \mathbb{R}$ gives the rate at which cost is accumulated as time passes, and $c_\Sigma : \Sigma \to \mathbb{R}$ is a function assigning the cost of executing each event in Σ. The associated cost function c_{r,c_Σ} is defined by $c_{r,c_\Sigma}(t) = t \cdot r$ for all $t \in \mathbb{T}$, and $c_{r,c_\Sigma}(\sigma) = c_\Sigma(\sigma)$ for all $\sigma \in \Sigma$. A probabilistic timed automaton equipped with a pair (r, c_Σ) is a probabilistic generalisation of uniformly priced timed automata [4].

For both probabilistic and expected reachability, we can consider reaching a state satisfying a formula which is a conjunction of propositions identifying locations and clock constraints of the form $x \sim c$ for $x \in \mathcal{X}$, $\sim\in \{\leq,=,\geq\}$ 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 [19]).

For examples of the types of properties of probabilistic timed automata which can be expressed using expected reachability, consider the following: 'the expected time until a host can use an IP address is at most 0.05 seconds', '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'. In the case of the third example, we would first need to modify the probabilistic timed automaton under study by adding a distinct clock (to represent global time) and a location such that, from all locations, once the global 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 and the cost function would equal 0 on all time transitions and events except those events corresponding to a message being lost; the costs for those actions would be set to 1.

Performance measures and digital clocks. We now show, under the restriction that the probabilistic timed automaton under study is diagonal-free and closed, that it suffices just to consider the integer-time semantics when verifying expected reachability properties.

Definition 8. *For any $x \in \mathcal{X}$, let $\mathbf{k}_x$ denote the greatest constant that the clock x is compared to in the zones of* PTA. *Define $\oplus_\mathbb{N}$ such that, for any clock valuation $v \in \mathbb{N}^{|\mathcal{X}|}$ and time duration $t \in \mathbb{N}$, the clock valuation $v \oplus_\mathbb{N} t$ assigns the value $\min\{v_x + t, \mathbf{k}_x + 1\}$ to all clocks $x \in \mathcal{X}$. The* integer-time semantics *of* PTA *is then defined as $[\![\mathsf{PTA}]\!]^{\oplus_\mathbb{N}}_\mathbb{N}$; that is, $\mathbb{T} = \mathbb{N}$ and $\oplus = \oplus_\mathbb{N}$.*

Let $\mathsf{PTA} = (L, \bar{l}, \mathcal{X}, \Sigma, I, prob)$ be a (closed, diagonal-free) probabilistic timed automaton. For any set of locations $L' \subseteq L$, we denote by $F^{L'}_\mathbb{T}$ the set of all states of $[\![\mathsf{PTA}]\!]_{\mathbb{T}}^{\oplus}$ which correspond to these locations; that is $F_{\mathbb{T}}^{L'} = \{(l,v) \mid l \in L',\ v \in \mathbb{T}^{|\mathcal{X}|} \ \wedge\ v \models I(l)\}$.

Theorem 1. *For any (closed, diagonal-free) probabilistic timed automaton* PTA*, set of locations* $L' \subseteq L$ *and cost function* $c : \Sigma \cup \mathbb{R} \to \mathbb{R}$ *which satisfies* $c(t+t') = c(t) + c(t')$ *for all* $t, t' \in \mathbb{R}$*:*

$$e^{\max}_{[\![\mathsf{PTA}]\!]_{\mathbb{R}}^{+}}(c, F_{\mathbb{R}}^{L'}) = e^{\max}_{[\![\mathsf{PTA}]\!]_{\mathbb{N}}^{\oplus_{\mathbb{N}}}}(c, F_{\mathbb{N}}^{L'}) \ \ \textit{and} \ \ e^{\min}_{[\![\mathsf{PTA}]\!]_{\mathbb{R}}^{+}}(c, F_{\mathbb{R}}^{L'}) = e^{\min}_{[\![\mathsf{PTA}]\!]_{\mathbb{N}}^{\oplus_{\mathbb{N}}}}(c, F_{\mathbb{N}}^{L'})\,.$$

The proof of the correctness of Theorem 1 can be found in [18]. Note that any cost functions defined by a pair (r, c_Σ), where $r \in \mathbb{R}$ and $c_\Sigma : \Sigma \to \mathbb{R}$, will satisfy the condition $c(t+t') = c(t) + c(t')$ for all $t, t' \in \mathbb{R}$. The analogous result for probabilistic reachability is proved in [22] and states:

$$p^{\max}_{[\![\mathsf{PTA}]\!]_{\mathbb{R}}^{+}}(F_{\mathbb{R}}^{L'}) = p^{\max}_{[\![\mathsf{PTA}]\!]_{\mathbb{N}}^{\oplus_{\mathbb{N}}}}(F_{\mathbb{N}}^{L'}) \quad \text{and} \quad p^{\min}_{[\![\mathsf{PTA}]\!]_{\mathbb{R}}^{+}}(F_{\mathbb{R}}^{L'}) = p^{\min}_{[\![\mathsf{PTA}]\!]_{\mathbb{N}}^{\oplus_{\mathbb{N}}}}(F_{\mathbb{N}}^{L'})\,.$$

4 Case study: Dynamic configuration of link-local addresses in IPv4

In this section, we illustrate the utility of the integer-time semantics of probabilistic timed automata with an analysis of the dynamic configuration protocol for IPv4 link-local addresses [10].

The dynamic configuration protocol for IPv4 addresses offers a distributed 'plug-and-play' solution in which IP address configuration is managed by individual devices connected to a local network. Upon connecting to the network, a device, henceforth called a *host*, first randomly chooses an IP address from a pool of 65024 available (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 sending 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 its chosen address and sending new probes. If the host sends four probes without receiving an ARP reply packet, then it commences to use the chosen IP address. The host then sends confirmations of this fact to the other hosts of the network by means of two *gratuitous* ARPs, also 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 the sending an ARP packet, thereby indicating that it is using 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 [29], 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 of Bohnenkamp et al. [8], we model the possibility that a device could surrender an IP address that it is using to another host; and in contrast to timed-automata-based analysis of Zhang and Vaandrager [29], 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 [10], 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 (choses a new IP address). However, when the host does reconfigure, unless it picks the same IP address, which happens with the 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 does not do anything about these messages (no_reset) and another where the host clears its buffer (removes the messages) when it is about to choose a new IP address (reset).

4.1 Modelling the dynamic configuration protocol

Preliminaries. We consider in detail one *concrete host*, which is attempting to configure an IP address for a network in which, as in [8], there are 1000 *abstract hosts* (they are called abstract because we do not study their behaviour in depth) which have already configured IP addresses. Therefore, when the concrete host picks an address, the probability of this address being *fresh* (not in use by an abstract host) is 64024/65024. We also assume that the concrete host never picks the same IP address twice, as this happens only with a very small probability.

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.

As in the standard [10], we suppose that it takes between 0 and 1 second to send a packet between hosts (where the choice of the exact time delay is nondeterministic). Since the abstract hosts have already picked their IP address, by

variable	description	range
coll	the number of address 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 otherwise)	$0 \dots 1$
probes	the number of probes/ARPs sent by the concrete host	$0 \dots N$
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	$0 \dots 1$
m_0	the number of packets containing an IP address of type 0 in all of the buffers of the abstract hosts	$0 \dots 20$
m_1	the number of packets containing an IP address of type 1 in all of the buffers of the abstract hosts	$0 \dots 8$

Table 1. Integer variables used in the probabilistic timed automata

supposing that they always defend their addresses, the concrete host will never receive probes. It then follows that 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 [29], 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.

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).

Probabilistic timed automata for the protocol. In the following, we describe the modelling of the reset version of the protocol only. We use two probabilistic timed automata, one to model the concrete host and one to model the environment (the abstract hosts and the output buffers of *all* hosts).

The model for the concrete host is shown in Figure 1. The host commences in the location RECONF (the double border indicates it is the initial location); this 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 CHOOSE_WAIT 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:=RAND$(1,2)$, which corresponds to the host randomly selecting an IP address (using the probabilities given at the start of this section). The assignment to the clock x (a uniform choice between $\{0,1,2\}$) approximates

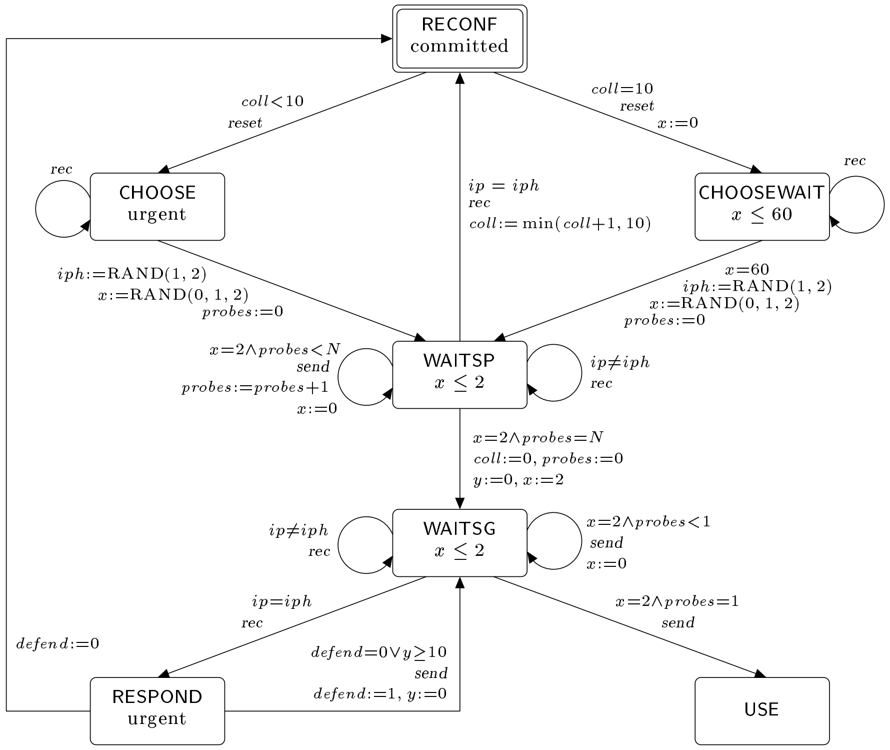

Fig. 1. Probabilistic timed automaton for the concrete host

the random delay of between 0 and 2 made by the host before sending the first probe. Note that, in CHOOSEWAIT, since the host has already experienced at least ten address collisions, it waits 60 seconds before choosing a new address.

In the location WAITSP the host sends N probes at 2 second intervals (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 it has the same address, the host immediately reconfigures (moves to RECONF). When sending the Nth probe, the host proceeds to location WAITSG, waits 2 seconds and then sends two gratuitous ARPs (re-using the variable *probes* 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 ($y{\geq}10$). This defence takes the form of sending of a defending packet, as denoted by the send labelled transition from RESPOND to WAITSG.

The model for the environment is shown in Figure 2. The dotted box labelled with three transitions which surrounds the model denotes that these transitions

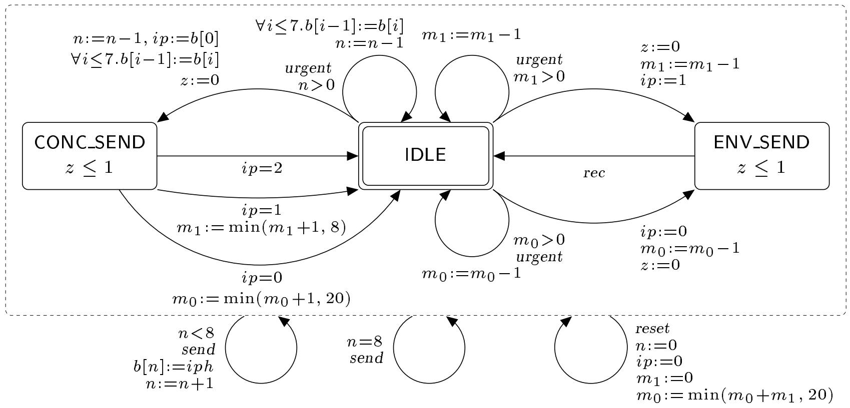

Fig. 2. Probabilistic timed automaton for the environment

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 ($n < 8$), the corresponding packet is added to the buffer (otherwise it is lost); 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, sets 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 environment sending 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 something 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 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 CONC_SEND or 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.

4.2 Verification using PRISM

In this section, we outline our results of using PRISM [17] to verify the integer-time model of the probabilistic timed automata of the dynamic configuration protocol given in Section 4.1. In the experiments, we fixed the number of hosts at

number of probes sent	probability of message loss							
	0		0.1		0.01		0.001	
	no_reset	reset	no_reset	reset	no_reset	reset	no_reset	reset
1	0.01538	0.01538	0.01538	0.01538	0.01538	0.01538	0 .01538	0.01538
2	8.0e-5	0	0.00298	0.00296	3.8e-4	3.1e-4	1.1e-4	3.1e-5
3	1.2e-6	0	5.6e-4	5.6e-4	7.2e-6	6.2e-6	1.3e-6	6.2e-8
4	4.2e-7	0	1.1e-4	1.1e-4	5.0e-7	1.2e-7	4.1e-7	1.2e-10
5	8.5e-9	0	2.0e-5	2.0e-5	9.8e-9	2.4e-9	8.4e-9	2.5e-13
6	2.2e-9	0	3.9e-6	3.9e-6	1.9e-9	4.9e-11	2.2e-9	4.9e-16

Table 2. Maximum probabilistic reachability results for the IPv4 protocol

1000 and varied both the number of probes a host sends (N), and the probability of message loss. Further details, including analysis for a network of 20 hosts, can be found at the PRISM web page [25]. The algorithms used by PRISM for both probabilistic and expected reachability are taken from the literature; for probabilistic reachability see [7], and for expected reachability see [12, 13].

To apply model-checking methods we must ensure that the model under study has only finitely-many states and is finitely branching. From the construction given in Section 3, the integer-time model will have finitely many states. To ensure finite branching, we restrict the delays from $\mathbb{N}$ to some finite set. More precisely, we allow delays of duration 1 only. Then, since any transition of duration $t \in \mathbb{N}$ can be modelled by a sequence of transitions of duration 1 and we restrict our attention to divergent adversaries, nothing is lost by omitting delays greater than 1 or equal to 0.

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 in between the minimum and maximum reachability probabilities computed for the model under study.

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. The results obtained in the case of maximum probabilistic reachability are given in Table 2. For results concerning minimum reachability probabilities see the PRISM web page [25].

The results obtained show the expected result: increasing the number of probes sent decreases the probability of the host using an IP address which is already in use (recall that the number specified by the standard is four). When the probability of message loss is 0, Table 2 shows that the maximum probability is 0 for the 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 no_reset (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 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 (as the probability is 0, the host will eventually get a reply).

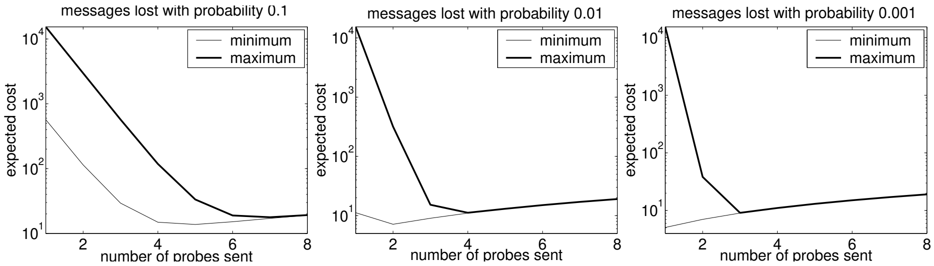

Fig. 3. Expected cost results for the IPv4 protocol

In the cases when message loss is greater than 0, the results again demonstrate that, by allowing the host to clear its buffer, the performance of the protocol improves; that is, the maximum reachability probability decreases (our experiments also show that the minimum probability increases, see [25]).

Expected Reachability. We consider the expect cost of a host choosing an IP address and using it. As in [8], the cost is defined as the time to start using an IP address plus an additional cost (10^6) associated with the host using an address which is already in use. 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 3. Note that, the model no_reset produced similar results, although the minimum costs are smaller and the maximum costs are larger (see [25] for further details). This is to be expected, since the results for probabilistic reachability show that, when the host does not clear its buffer, there is a greater chance of it using an IP address which is already in use, and hence of incurring a greater cost.

These results are similar to those of [8]: as the message loss probability increases, one must increase the number of probes sent in order to reduce the expected cost; however, by sending too many probes the expected cost may then 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 (sending more probes takes more time).

5 Conclusions

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 widely applicable, a fact which we illustrate by analysing the performance of the IPv4 dynamic configuration protocol.

Future work could consider extending the cost functions in order to vary the rate of cost accumulation in different locations, as in priced or weighted timed automata [5, 2]. There are still limitations as to the size of the models that can be considered using digital clocks. In the case of probabilistic reachability, a generally more efficient approach is to consider *zones*, and in particular the backwards reachability approach introduced in [20]. The application of zones to the verification of priced timed automata [23] may be instructive to this line of research.

References

1. R. Alur and D. L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126(2):183–235, 1994.
2. R. Alur, S. Torre, and G. Pappas. Optimal paths in weighted timed automata. In *Proc. HSCC'01*, volume 2034 of *LNCS*, pages 49–62. Springer, 2001.
3. G. Behrmann, A. David, K. Larsen, O. Möller, P. Pettersson, and W. Yi. UPPAAL - present and future. In *Proc.CDC'01*. IEEE Computer Society Press, 2001.
4. G. Behrmann, A. Fehnker, T. Hune, K. Larsen, P. Pettersson, and J. Romijn. Efficient guiding towards cost-optimality in UPPAAL. In *Proc. TACAS'01*, volume 2031 of *LNCS*, pages 174–188. Springer, 2001.
5. G. Behrmann, A. Fehnker, T. Hune, K. Larsen, P. Pettersson, J. Romijn, and F. Vaandrager. Minimum-cost reachability for priced timed automata. In *Proc. HSCC'01*, volume 2034 of *LNCS*, pages 147–161. Springer, 2001.
6. D. Bertsekas and J. Tsitsiklis. An analysis of stochastic shortest path problems. *Mathematics of Operations Research*, 16(3):580–595, 1991.
7. A. Bianco and L. de Alfaro. Model checking of probabilistic and nondeterministic systems. In *Proc. FSTTCS'95*, volume 1026 of *LNCS*, pages 499–513. Springer, 1995.
8. H. Bohnenkamp, P. v. d. Stok, H. Hermanns, and F. Vaandrager. Cost-optimisation of the IPv4 zeroconf protocol. In *Proc. IPDS 2003*. IEEE CS Press, 2003.
9. D. Bosnacki. Digitization of timed automata. In *Proc. FMICS'99*, pages 283–302, 1999.
10. S. Cheshire, B. Adoba, and E. Guttman. Dynamic configuration of IPv4 link-local addresses. Draft, August 2002. Available from `www.ietf.org/internet-drafts/draft-ietf-zeroconf-ipv4-linklocal-07.txt`.
11. C. Daws, M. Kwiatkowska, and G. Norman. Automatic verification of the IEEE 1394 root contention protocol with KRONOS and PRISM. In *Proc. FMICS'02*, volume 66(2) of *ENTCS*. Elsevier Science, 2002.
12. L. de Alfaro. *Formal Verification of Probabilistic Systems*. PhD thesis, Stanford University, 1997.
13. L. de Alfaro. Computing minimum and maximum reachability times in probabilistic systems. In *Proc. CONCUR'99*, volume 1664 of *LNCS*, pages 66–81. Springer, 1999.
14. C. Derman. *Finite-State Markovian Decision Processes*. Academic Press, 1970.
15. T. Henzinger, Z. Manna, and A. Pnueli. What good are digital clocks? In *Proc. ICALP'92*, volume 623 of *LNCS*, pages 545–558. Springer, 1992.
16. J. Kemeny, J. Snell, and A. Knapp. *Denumerable Markov Chains*. Graduate Texts in Mathematics. Springer, 2nd edition, 1976.
17. M. Kwiatkowska, G. Norman, and D. Parker. PRISM: Probabilistic symbolic model checker. In *Proc. TOOLS'02*, volume 2324 of *LNCS*, pages 200–204, 2002.

18. M. Kwiatkowska, G. Norman, D. Parker, and J. Sproston. Performance analysis of probabilistic timed automata using digital clocks. Technical Report CSR-03-6, School of Computer Science, University of Birmingham, 2003.
19. 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.
20. M. Kwiatkowska, G. Norman, and J. Sproston. Symbolic computation of maximal probabilistic reachability. In *Proc. CONCUR'01*, LNCS, pages 169–183. Springer, 2001.
21. M. Kwiatkowska, G. Norman, and J. Sproston. Probabilistic model checking of the IEEE 802.11 wireless local area network protocol. In *Proc. PAPM/PROBMIV'02*, volume 2399 of *LNCS*, pages 169–187. Springer, 2002.
22. 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(3):295–318, 2003.
23. 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 *Proc. CAV'01*, volume 2102 of *LNCS*, pages 493–505. Springer, 2001.
24. J. Ouaknine and J. Worrell. Universality and language inclusion for open and closed timed automata. In *Proc. HSCC'03*, volume 2623 of *LNCS*, pages 375–388. Springer, 2003.
25. PRISM web page. http://www.cs.bham.ac.uk/~dxp/prism/.
26. R. Segala. *Modeling and Verification of Randomized Distributed Real-Time Systems*. PhD thesis, Massachusetts Institute of Technology, 1995.
27. S. Tripakis. *The formal analysis of timed systems in practice*. PhD thesis, Université Joseph Fourier, 1998.
28. M. Vardi. Automatic verification of probabilistic concurrent finite state programs. In *Proc. FOCS'85*, pages 327–338. IEEE Computer Society Press, 1985.
29. M. Zhang and F. Vaandrager. Analysis of a protocol for dynamic configuration of IPv4 link local addresses using UPPAAL. Technical Report, NIII, University of Nijmegen, 2003.