Vérification de propriétés quantitatives et fonctionnelles
Analysis and VERification for the Reliability Of Embedded Systems

Lot 4.2

Technologie de modélisation

Probabilités

Probabilistic Model Checking of the CSMA/CD protocol using PRISM and APMC

Description :	Le protocole CSMA/CD, plus connu sur le nom de protocole Ethernet, a déjà été vérifié plusieurs fois de manière qualitatitave avec des model-checkers temporisés. Nous avons ici, pour la première fois trouvé des propriétés quantitatives (des probabilités) grace aux model checker probabiliste PRISM et APMC. Cette version subsume les études réalisées dans la fourniture 4.2.1
Auteur(s) :	Marie Duflot, Laurent Fribourg, Thomas Herault, Richard Lassaigne, Frederic Magniette, Stephane Messika, Sylvain Peyronnet, Claudine Picaronny
Référence :	Averroes / Lot 4.2 / Fourniture 2 / V1.0
Date :	14 juin 2004
Statut :	Version à valider
Version :	1.0

Réseau National des Technologies Logicielles
Projet subventionné par le Ministère de la Recherche et des Nouvelles Technologies

CRIL Technology, France Télécom R&D, INRIA-Futurs, LaBRI (Univ. de Bordeaux – CNRS), LIX (École Polytechnique, CNRS) LORIA, LRI (Univ. de Paris Sud – CNRS), LSV (ENS de Cachan – CNRS)

Résumé

Carrier Sense Multiple Access/Collision Detection (CSMA/CD) is the protocol for carrier transmission access in Ethernet networks (international standard IEEE 802.3). On Ethernet, any Network Interface Card (NIC) can try to send a packet in a channel at any time. If another NIC tries to send a packet at the same time, a collision is said to occur and the packets are discarded. The CSMA/CD protocol was designed to avoid this problem, more precisely to allow a NIC to send its packet without collision. This is done by way of a randomized exponential backoff process.

In this paper, we analyse the correctness of the CSMA/CD protocol, using techniques from probabilistic model checking and approximate probabilistic model checking. The tools that we use are PRISM (developped at Birmingham University) and APMC (developped at Paris XI University). Moreover, we provide a quantitative analysis of some interesting CSMA/CD properties.

1 Introduction

The Carrier Sense Multiple Access/Collision Detection (CSMA/CD) protocol is a fundamental distributed protocol for networks. Indeed, it is one of the most important part of the widely used IEEE 802.3 international standard (Ethernet Network Communication protocol). In Ethernet, multiple Network Interface Cards (NIC) may be connected via the same channel. Since two NICs may send packets simultaneously, collisions may occur, thus discarding both packets. Both the NICs will detect this collision, but cannot re-send the packets at once, since it would induce a new collision. So, when a collision happens, the CSMA/CD protocol forces each NIC to pick at random an integer-valued delay from a bounded interval, and to wait for a length of time proportional to this integer-valued delay before re-sending the packet.

This paper considers an application of probabilistic model checking techniques to the IEEE 802.3 CSMA/CD protocol. Here, we are interested in establishing quantitative properties of the protocol, such a computing the probability that a given event occurs before a certain deadline. Other values are also computed, like the maximum expected time needed to send a packet.

Following [21, 23], we model the protocol under study in the framework of *probabilistic timed automata*. Probabilistic timed automata [24] are extensions of timed automata [1] which incorporate probability distributions of discrete transitions. A probabilistic timed automaton has an infinite number of states due to the presence of real-valued clock variables. However, for the class of reachability properties that we consider here, one can always derive an equivalent *finite-state* transition system (see [23]). We adopt here a method (referred to as "integer semantics" method, in [8, 23, 22]), where clocks are viewed as counters storing non-negative integer values, which increment as time goes. The probabilistic timed automaton modelling the system then reduces to a finite-state *Markov decision process* [10].

We then use the model-checking tool PRISM [30] in order to analyse the resulting Markov decision process for the CSMA/CD protocol. However, the original constants used by the protocol lead to a model of prohibitively large size. Therefore, the verification with PRISM is done only with smaller constants. A way to partially alleviate this limitation consists in removing the sources of nondeterminism, replacing nondeterministic choices (originating from the timed transitions and the asynchrounous product of components of the system) by probabilistic distributions. The underlying Markov decision process then becomes a "fully probabilistic system" (or, in other terms, a Markov chain), and can then be analysed via the tool APMC [13]. The same input format (Reactives Modules, [2]) is used for processing the model in both tools.

The interest of using these tools together is twofold :

1. the two tools have complementary advantages : PRISM allows to verify models with nondeterministic choices, but of smaller size (due to the state space explosion phenomenon), while APMC allows the verification of larger models (viz actual values of the CSMA/CD protocol), but only on *fully* probabilistic systems (after replacement of nondeterministic choices by probability distributions).
2. Using two different tools based on different approach allows to give more confidence on the experimental results given by both tools.

Structure of the paper After a description of the related work (sections 2), the paper proceeds by giving an informal presentation of the CSMA/CD protocol followed by the modelling of CSMA/CD in the framework of Markov decision processes (sections 3 and 4). In the rest of the paper, sections 5 and 6, we present the two tools dedicated to the model checking of probabilistic systems together with a brief description of their theoretical framework and the results of several experiments run with these tools. We discuss these results, validating again the CSMA/CD protocol. This is the first time this is done using probabilistic and approximate model checking techniques.

2 Related Work

CSMA/CD is a widely studied protocol using various techniques. We focus here on techniques related to model checking and approximation.

Previous studies of CSMA/CD LAN have mainly concerned performance evaluation by using two approaches : analytical models [12, 26] or simulation [29]. Several models were developped to analyse both throughput link and packet delay : from simple traditional model [26] to more complex models [6, 18]. Other authors based performance studies on detailed simulation and measurement to avoid some of the simplifying assumptions that analytical models employ [9].

Very few papers consider automatic verification of temporal and probabilistic specifications over a timed model. [14] give temporal constraints for this protocol in some discrete time model. In [31], the behavior of the system is described by a product timed automaton. Then, the timed automata model checking tool KRONOS is used to verify properties such as :
- a collision is detected whenever the two senders are simultaneously transmitting,
- a collision is detected in a given bounded delay,
- when one of the senders begins transmitting, there must exist an execution leading to a successfull transmission.

Timed automata model checking tools as UPPAAL [27] and KRONOS [11] do not allow to verify the satisfaction of probabilistic specifications.

In [21], the probabilistic model checking tool PRISM [30, 20] is used to verify probabilistic properties of Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) mechanism for Wireless Local Area Networks (international standard IEEE 802.11). For example, it is possible to compute the minimal probability of both stations eventually sending their packet correctly and the minimum probability of a station delivering a packet within some deadline.

3 Sketch of the CSMA/CD protocol

The CSMA/CD protocol (Carrier Sense Multiple Access with Collision Detection) is a network arbitration protocol which regulates communication between several agents who communicates by a unique channel. The IEEE 802.3 [5] standard describes precisely the differents aspects of the protocol. We focus on the *half duplex* version of the protocol, which means only one message can be carried at a time (in opposition to *full duplex*).
Non-relevant details, such as the message structure, are voluntarily omitted.

Emission All agents are equal in their ability to send messages onto the network (Multiple Access). Roughly speaking, each agent must sense the channel (Carrier Sense), and wait for the absence of signal before starting an emission. As signals take a bounded amount of time (denoted here σ) to travel, two agents may both sense the channel as free, thus starting to emit (almost) simultaneously, which yields a subsequent message collision. The detection of collision thus takes at most 2σ μs (see Fig. 1) After this amount of time, if no signal of collision has been detected, the sender can safely complete the emission. The time for emitting completely a message is assumed here to be a constant number of μs, denoted λ. (For example, for a communication over Ethernet 10 Mbps with $\sigma = 24\mu$s, we have a time of transit of around 780μs for a message of size 1024 bytes.)

Conflict If a collision occurs, the messages get lost and agents are informed of the event, upon reception of a garbled signal. They choose then independently a random waiting time before attempting to transmit again. To minimize the chance of another collision, the waiting time is chosen uniformly in an interval $[0, 2^m]$, where m denotes the minimum between α, a given constant,

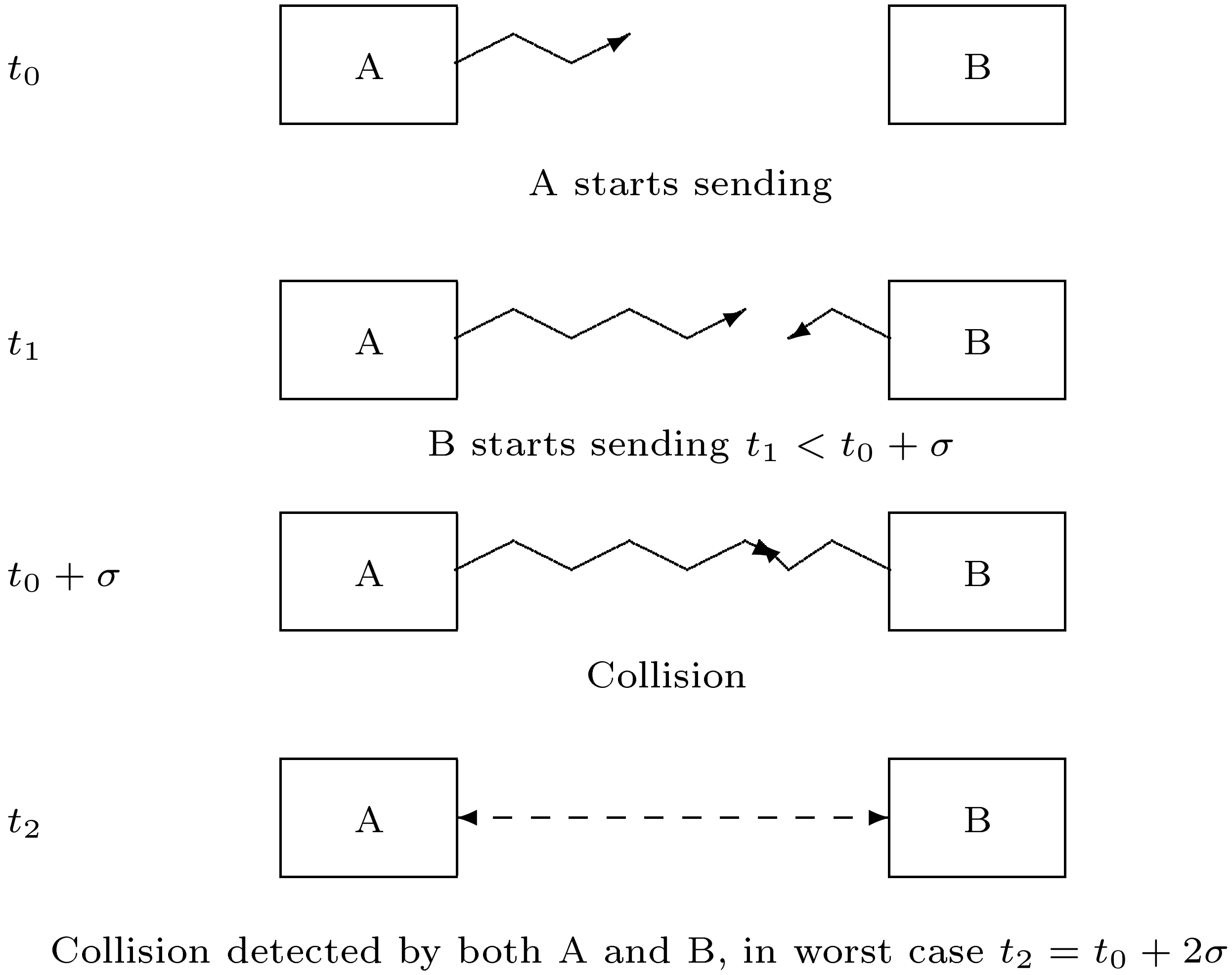


FIG. 1 – Scheme of collision.

and the number of collisions since the last good transmission. Thus, the higher the number of collisions is, the longer this interval becomes, and the more the chance of a new collision decreases.

4 Modelling

Following [21, 23], we use the framework of *probabilistic timed automata* for modelling the protocol under study. Probabilistic timed automata are extensions of classical timed automata [1] with the ability to express relative likelihoods of state transitions under the form of probability distributions [24]. For the sake of self-containment, we give in Appendix the formal definitions of probabilistic timed automata.

The model of the CSMA/CD protocol consists of three components operating in parallel, namely $Sender_1$, $Sender_2$ (sending stations) and $Chan$ (the channel). In the synchronized product $Sender_1 \| Sender_2 \| Chan$ the set $\mathcal{X}$ of clocks is a triple $\{x_1, x_2, y\}$ where x_1 represents the clock of $Sender_1$, x_2 the clock of $Sender_2$, and y a clock, which is never reset, used for measuring time bounds on reachability properties. A state s of the system is a pair (l, v) with $l = (l_1, l_2, c)$, where l_i is the location of $Sender_i$ of the form (s_i, col_i) (for $i = 1, 2$), c the location of $Chan$, and v is a valuation of $\{x_1, x_2, y\}$. In the location $l_i = (s_i, col_i)$ of $Sender_i$ the component col_i stores

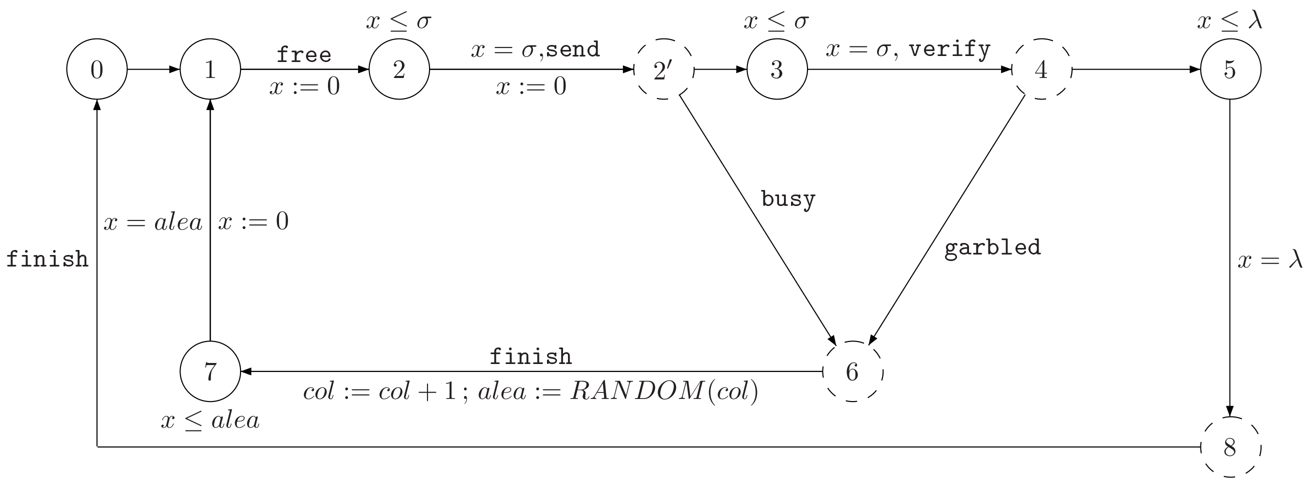


FIG. 2 – Template for the sender stations.

the number of collisions which have occurred for $Sender_i$ since the last correct emission. In the following, we assume familiarity with the graphical representation of timed automata. The presentation is much inspired from [21].

Senders Automata These automata describe the behavior of each sender. It contains a clock (x) and a variable storing the random waiting time ($alea$). The template for the senders is shown in Figure 2.

There are eight locations, among which, locations $2', 4, 6$ and 8 (represented as dashed circles) are urgent intermediate locations, where no time can pass. Note that the events `busy`, `free` and `garbled` are the urgent events of the sender. The initial location is 0. When the sender wants to emit a message, the automaton goes from location 0 to 1. This transition is preceded by a certain amount of time in location 0, which corresponds to one source of nondeterminism. If the channel is free, then the sender goes to location 2 where it waits for σ μs before going to the urgent location $2'$ where it tests the channel. If the channel is busy, the sender enters the backoff procedure (transition from location 2 to 6). Otherwise, the sender goes from location $2'$ to 3, starting to send the packet (transition labelled 'send'). In location 3, the sender waits again for σ time units, and goes to the urgent location 4, where it tests the channel (transition labelled 'verify'). If the channel is garbled, the sender enters the backoff procedure (going to location 6). Otherwise, the sender goes to location 5 where it completes the message emission (going to the urgent location 8), then returns to the initial location 0 (transition labelled 'finish'). The backoff procedure consists in setting the backoff value according to the random assignment $alea := RANDOM(col)$ (transition from 6 to 7). Here, col represents the number of collisions since the last successful emission, and $RANDOM(col)$ is a number chosen uniformly between 0 and $2^m - 1$ with $m = min(col, \alpha)$. When the value of x reaches $alea$, the sender starts re-sending its packet (transition from 7 to 1).

As a recapitulation, the meaning of the locations can be summarized as follows :

0 Idle
1 Wait for the channel to be free
2 Wait a time σ before testing the channel
2' Test of the channel (busy or not)
3 Wait a time σ
4 Test of the channel (garbled or not)

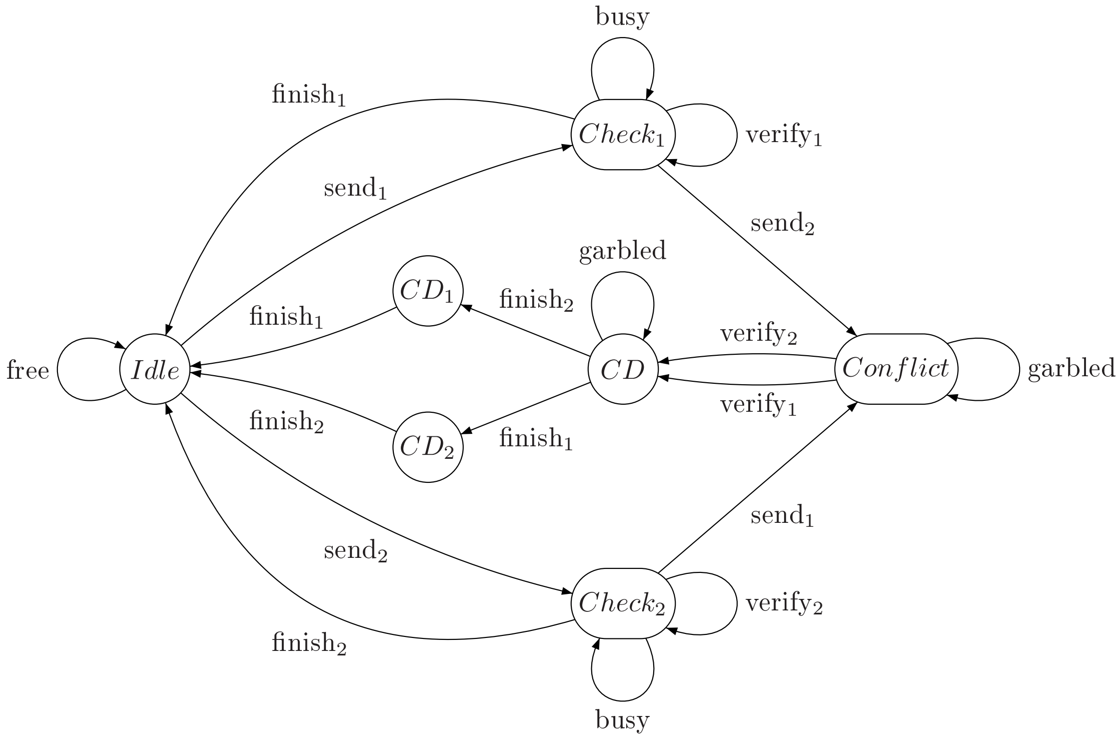


FIG. 3 – Template for the channel.

5 Completion of emission
6 Collision detected
7 Wait a random time before reemission
8 Message correctly sent

Channel Automaton The probabilistic automaton *Chan*, which represents the channel, is shown in Figure 3. The initial location *Idle* corresponds to the case where the channel is free. From this location, receipt of a message (event $\texttt{send}_1$, sent by $Sender_1$) triggers the transition to location $Check_1$; then this message can either finish successfully (event $\texttt{verify}_1$ followed by $\texttt{finish}_1$) making the channel return to location *Idle*, or collide with a message from $Sender_2$ (event $\texttt{send}_2$) making the channel proceed to location $Conflict$. Once the collision has been detected (event $\texttt{verify}_1$), the channel goes to location CD where a garbled signal is propagated (event `garbled`), and returns to location *Idle* after reception of the `finish` events from the senders.

Properties We are interested in the verification of differents kinds of properties
- Minimum and maximum probabilistic reachability properties over probabilistic timed automata :
  - The minimum probability that a sender sends correctly a message within a deadline of d μs :
    $P_{min}[\overline{s} \to^* \{s \mid (x_1 = 8 \vee\ x_2 = 8)\ \wedge\ y \leq d\}]$
  - The minimum and maximum probabilities that at least N collisions occur after d μs :
    $P_{min}[\overline{s} \to^* \{s \mid col_1 \geq N\ \wedge\ y \leq d\}]$
    $P_{max}[\overline{s} \to^* \{s \mid col_1 \geq N\ \wedge\ y \leq d\}]$

  Informally, given a set F of target states, $P_{\max}(\overline{s} \to^* F)$ (resp. $P_{\min}(\overline{s} \to^* F)$) represents the probability that the probabilistic system reaches F, when all nondeterministic choices are as favorable (resp. unfavorable) as possible.
- Time bounded probabilistic reachability properties over fully probabilistic systems :
  - The probability that a sender sends correctly a message within a deadline of d μs :

$Prob[\overline{s} \rightarrow^* \{s \mid (x_1 = 8 \vee \ x_2 = 8) \ \wedge \ y \leq d\}]$
- The probability that at least N collisions occur after d μs :
$Prob[\overline{s} \rightarrow^* \{s \mid col_1 \geq N \ \wedge \ y \leq d\}]$

Informally, given a set F of target states, $Prob[\overline{s} \rightarrow^* F]$ is the probability that the fully probabilistic system reaches F.

5 Verification using the PRISM tool

For verifying probabilistic reachability properties, we must derive an equivalent *finite-state* (probabilistic) system. In the non-probabilistic framework, possible methods of reduction are : “region equivalence” [1], “forward exploration” [11, 27], “integer semantics” [8]. These methods have been extended to the probabilistic framework [21, 23]. As explained in [23] (Sec. 3.2), the two first methods require in practice the preliminary construction of an abstraction of the original probabilistic timed automaton. For the sake of simplicity, we chose the third approach (integer semantics), which allows us to work directly at the level of the original probabilistic timed automaton (see [23, 22]). In such a method, clocks are viewed as counters storing non-negative integer values, which increment as time goes. The probabilistic timed automaton modelling the system can then be seen as a finite-state *Markov decision process* [10].

5.1 Theoretical foundations of PRISM

Markov Decision Processes (MDP) (also called Probabilistic Nondeterministic Systems (PNS) [7]) allow accurate modelling of systems which exhibit both probabilistic and nondeterministic behavior. A common example of this is the interleaved parallel composition of several probabilistic processes, but nondeterminism can also be useful to leave parts of a system underspecified and to model interaction with an unknown environment. Formally :

Definition 1 *A* Markov decision process *(MDP) is a tuple* $(S, \overline{s}, Act, Steps)$ *where :*
- S *is a finite set of states*
- $\overline{s}$ *is the initial state*
- Act *is a set of actions*
- $Steps \subseteq S \times Act \times Dist(S)$ *is a probabilistic transition relation.*

An MDP transition $s \xrightarrow{a,\mu} s'$ is made from a state $s \in S$ first by nondeterministically selecting an action-distribution pair (a, μ) such that $(s, a, \mu) \in steps$, and second by making a probabilistic choice of the target state s' according to the distribution μ, such that $\mu(s') > 0$.

A *path* represents a particular resolution of both nondeterminism *and* probability : it is a non-empty finite or infinite sequence of probabilistic transitions $\pi = s_0 \xrightarrow{a_0,\mu_0} s_1 \xrightarrow{a_1,\mu_1} \cdots$ such that $s_0 = \overline{s}$. We denote by $\pi(i)$ the $(i+1)$-th state of π and $last(\pi)$ the last state of π if π is finite. An *adversary* represents a particular resolution of nondeterminism *only*. Formally, an adversary of a Markov decision process is a function A mapping every finite path π to a pair (a, μ) such that $(last(\pi), a, \mu) \in Steps$.

The evolution of the MDP according to a particular adversary A is a measurable set of infinite paths associated with A, which can classically be provided with a probability measure. Given a set $F \subseteq S$ of target states, let : $P(\overline{s} \xrightarrow{A}{}^{*} F)$ denote the probability of reaching F starting from the initial state under A.

The *maximal reachability probability* $P_{max}(\overline{s} \rightarrow^* F)$, (resp. *minimal reachability probability* $P_{min}(\overline{s} \rightarrow^* F)$), is the maximum (resp. minimum) probability over all the adversaries, with which a given set of states can be reached from the initial state :

$$\begin{aligned} P_{max}(\overline{s} \rightarrow^* F) &= \max_A\{P(\overline{s} \xrightarrow{A}{}^{*} F)\} \\ P_{min}(\overline{s} \rightarrow^* F) &= \min_A\{P(\overline{s} \xrightarrow{A}{}^{*} F)\} \end{aligned}$$

Informally, $P_{\max}(\overline{s} \rightarrow^* F)$ (resp. $P_{\min}(\overline{s} \rightarrow^* F)$) represents the probability that the MDP reaches F, when all non deterministic choices are as favorable (resp. unfavorable) as possible.

Specification of reachability properties to be checked on a MDP may be expressed in PCTL, which is a probabilistic extension of the popular temporal logic CTL. In [7] the authors provide an algorithm to enrich the usual algorithm to check a CTL formula on a finite state system, with the computation of maximal and minimal probabilities. This can be done in polynomial time by solving linear programming systems.

PRISM [20] is a *probabilistic model checker* which provides support for analysis of Markov decision processes and performs verification of PCTL formulae for MDPs, using the model checking algorithm of [7]. The most expensive part of this is the computation of reachability probabilities. For this there are two options, either solution of a linear optimization problem or iterative numerical solution techniques (based on dynamic programming). PRISM uses the second of these. Each iteration computes new values for the reachability probabilities, tending towards the exact solution. The computation is terminated when it has converged to within the desired precision (parameter ε specified by the user).

To analyse an MDP, PRISM has to construct the full reachable state space and the transition matrix which represents it. However, the tool can often handle very large models because it uses *symbolic model checking* techniques. It uses BDD (binary decision diagram) based data structures, in particular MTBDDs (multi-terminal BDDs). These can provide compact storage of large probabilistic models by exploiting their high-level structure. For more details, see e.g. [15].

In this paper, we have made use of a prototype extension of PRISM which provides support for analysis of properties based on costs (or conversely, rewards) [22]. Each state or transition of the model can be assigned a cost. PRISM allows computation of, for example, the *expected* amount of cost cumulated before a certain set of states is reached. Because we use MDPs, we must compute either the minimum or maximum expected cost (over all resolutions of nondeterminism). In PRISM, these properties are expressed as follows :

1. $R_{min}[true\ U goal]$
2. $R_{max}[true\ U goal]$

PRISM uses the algorithms of [3] to perform model checking of these formulas.

5.2 Experiments

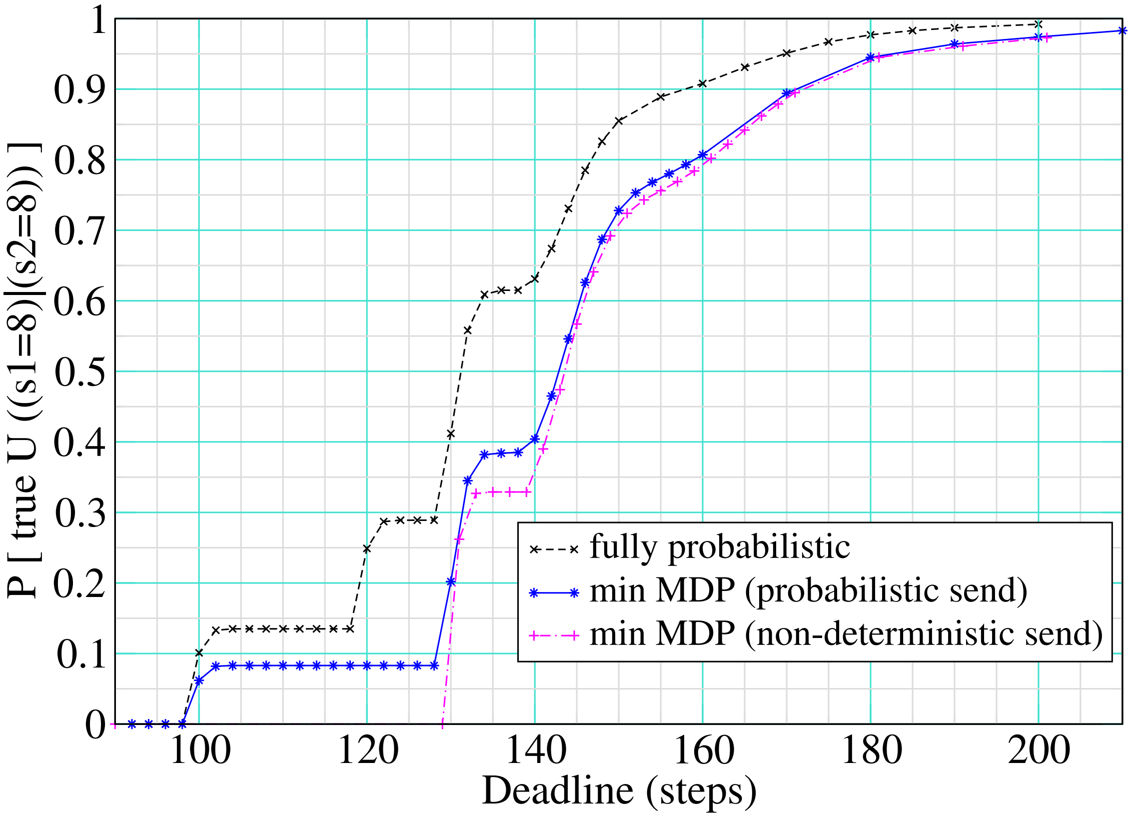
All experiments were run using a Pentium IV 2.80GHz with 1 Gb of RAM. Due to the size of the models (up to dozens of millions states), we have used the MTBDD engine of PRISM which is usually more efficient for larger systems. We set the approximation parameter epsilon to 10^{-6}. The main reason why the models become so large is the variable needed to count the time (for bounded probabilistic reachability). For the expected time, since the costs are not stored in the states of the model, the state space is reasonably small.

The main difference with APMC (see section 6) is that PRISM can handle the inherent nondeterminism of the model. By effectively building the model it can then choose the "best" or "worst" resolutions of the nondeterminism, and give a realistic analysis of the worst and best cases, for example here with the probability to have at least N collisions. It is also possible to model the case in which the choice to send or not a message is made nondeterministically, as we will see in the following.

Verification of probabilistic reachability properties

due to the size of the model, it was impossible to verify it with PRISM using the real values of the constants λ and σ. Nevertheless, we have preserved the ratio of these two constants. We have also set the maximum number of collisions α (used to compute the random delay after before sending a new message after a collision) to 6.

The three main properties we have verified are the following :



($\lambda = 96$, $\sigma = 3$, $\alpha = 6$)
FIG. 4 – Probability of emission as function of the deadline.

- Property 1 represents the minimum probability to reach a state in which at least one of the senders has successfully sent a message before deadline d. This is written in the syntax of PCTL :

$$P_{min}[true\ U((s_1 = 8|s_2 = 8)\ \&\ y \leq d)]^1$$

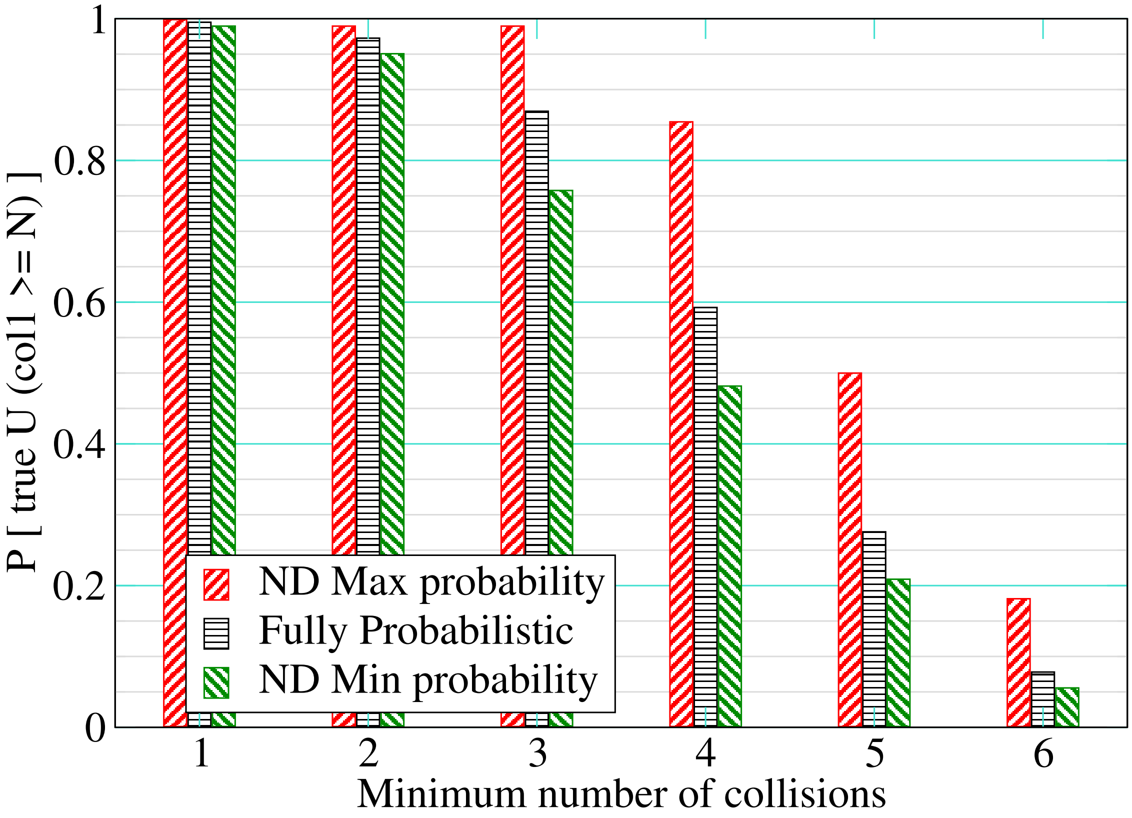
- Properties 2-3 represent respectively the minimum and maximum probability to reach a state where at least N collisions (with $1 \leq N \leq \alpha$) have occurred for sender 1 before deadline d[2]. This writes :

$$P_{min}[true\ U(col_1 = N\ \&\ y \leq d)]$$

$$P_{max}[true\ U(col_1 = N\ \&\ y \leq d)]$$

Figure 4 shows the probability to satisfy property 1 for different values of the deadline d. There are three curves, corresponding to the degree of nondeterminism of the modelled system. There are indeed two main sources of nondeterminism in our model. The first source originates from the unbounded waiting period in location 0 of $Sender_1$ (resp. $Sender_2$) : when a sender is at location 0, it decides nondeterministically every time unit whether it will start the emission or not[3]. The other source of nondeterminism originates from the interleaving of asynchronous actions of $Sender_1$ and $Sender_2$ such as $send_1$ and $send_2$. The first curve corresponds to the original nondeterministic model. The second curve has been obtained by replacing the nondeterministic branch at location 0

[1] We focus on P_{min} because P_{max} converges quickly towards 1
[2] Since the system is symmetric, this probability is the same for sender 1 and sender 2.
[3] In this case, we assume that at least one sender tries to send a message, because, if nobody tries to send a message, then the probability for at least one message to be received is always 0 !



($deadline = 210$, $\lambda = 96$, $\sigma = 3$, $\alpha = 6$)
Fig. 5 – Minimum and maximum probability to get N collisions.

of each sender by a uniform probabilistic transition. The third one has been obtained by replacing every nondeterministic choice of the composed system by a uniform probabilistic transition. (This corresponds to a fully probabilistic option provided by PRISM.) Note that the latter curve is the same as the approximated one obtained with APMC, as shown in Figure 6.

Properties 2-3 are shown on figure 5 for constants $\lambda = 96$, $\alpha = 6$ and $\sigma = 3$ and for deadline $d = 210$. We have set the deadline to 210 because at that time, the probability for a message to be delivered is very high (greater than 0.98). The probability is very high to have at least three collisions, but quickly decreases afterwards until the maximum number of collisions α we have considered, which is 6. These curves illustrate the influence of nondeterminism on the possible number of collisions.

Expected time

using the prototype extension of PRISM which enables the computation of expected costs, we have computed (for a maximum number of collisions of 6) the maximum expected time to send a message :

$R_{max}[true\ U((s_1 = 8)|(s_2 = 8))]$

The results are 144.4 for the model with probabilistic sending, and 147.0 for the one with nondeterministic sending. There is not much difference between these two results.

Concerning expected time, we have also considered the maximum expected time to reach a state in which a given sender has successfully sent a message (since the system is symmetric, this probability is the same for both senders) :

$R_{max}[true\ U(s_1 = 8)]$

In this case the effect of choosing nondeterministic or probabilistic sending is crucial. The results are 707 time units for probabilistic sending and 5750 for nondeterministic sending. This shows that, by trying repeatedly to send messages (as long as it is allowed to do so by the protocol), one sender can delay the other one for quite a long time. This phenomenon is due to the fact that if a sender success to transmit a message, its counter of collision is resetted, although the counter of the other is not. Thus, it is easier to a sender who already success, to transmit again.
This last result illustrates the importance to be able to really model the nondeterminism. For this last property, the "worst case" expected time is very far from the expected time associated with the probabilistic approximation we had made.

Figure 4 demonstrates that the fully probabilistic and the nondeterministic models have a similar general behavior and justifies the approach followed in section 6.

6 Verification using the APMC tool

APMC is an approximate probabilistic model checker dedicated to the verification of quantitative properties over Discrete Time Markov Chains (DTMCs, that is fully probabilistic systems). It uses the same input language as PRISM (reactive modules). In this section, we first recall the theoretical framework of APMC, then after a description of the tool, we present the experiments done on the CSMA/CD Model.

6.1 Theoretical foundations of APMC

The APMC approach [13] uses an efficient Monte-Carlo method to approximate satisfaction probabilities of monotone properties over fully probabilistic transitions systems. Properties to be checked are expressed in LTL : Linear Temporal Logic.

6.1.1 APMC method

LTL formulas are built over a set of atomic propositions labeling states.

Definition 2 *A fully probabilistic transition system (PTS or DTMC) is a tuple $\mathcal{M} = (S, \overline{s}, P)$ where S is a set of states, $\overline{s}$ is the initial state, and P is a transition probability function.*

We denote by $Path(s)$ the set of paths whose first state is s. The length of a path π is the number of states in the path and is denoted by $|\pi|$, this length can be infinite.

The probability measure $Prob$ over the set $Path(s)$ is defined in a classical way [19].

We denote by $Prob[\phi]$ the measure of the set of paths $\{\pi \mid \pi(0) = s \text{ and } \mathcal{M}, \pi \models \phi\}$ (see [28] for details). Let $Path_k(s)$ be the set of all paths of length $k > 0$ starting at s in a PTS. The probability of an LTL formula ϕ on $Path_k(s)$ is the measure of paths satisfying ϕ in $Path_k(s)$ and is denoted by $Prob_k[\phi]$.

Definition 3 *An LTL formula ϕ is said to be* monotone *if and only if for all $k > 0$, for all paths π of length k, $\mathcal{M}, \pi \models \phi \implies \mathcal{M}, \pi^+ \models \phi$, where π^+ is any path of which π is a prefix.*

A basic property of monotone formulas is the following one : if ϕ is a monotone formula, $0 < b \leq 1$ and if there exists some $k \in \mathbb{N}^*$ such that $Prob_k[\phi] \geq b$, then $Prob[\phi] \geq b$.

In order to verify some probabilistic specification $Prob[\phi] \geq b$, we choose a first value of $k = O(log|S|)$, then we approximate the probability $Prob_k[\phi]$ and test if the result is greater than b. If $Prob_k[\phi] \geq b$ is true, then the monotonicity of the property guarantees that $Prob[\phi] \geq b$ is true. Otherwise, we increment the value of k and approximate again $Prob_k[\phi]$. We iterate this procedure within a certain bound which, in many cases, is logarithmic in the number of states. In the worst case, this bound is strongly related to the rapid mixing rate of the underlying Markov chain [25]. If the results of all tests $Prob_k[\psi] \geq b$ are negative, then we can conclude that $Prob[\psi] \not\geq b$.

If we are interested only with probabilistic time bounded properties, as here, we can set k to the maximum time bound in subformulas of the specification.

In the next paragraph, we briefly recall how approximate efficiently the probability $Prob_k[\phi]$.

6.1.2 Randomized approximation scheme

In order to estimate the probabilities of monotone properties with a simple randomized algorithm, we generate random paths in the probabilistic space underlying the DTMC structure of depth k and compute a random variable A/N which estimates $Prob_k[\psi]$. To verify a statement $Prob_k[\psi] \geq b$, we test whether $A/N > b - \varepsilon$. Our decision is correct with confidence $(1 - \delta)$ after a number of samples polynomial in $\frac{1}{\varepsilon}$ and $\log \frac{1}{\delta}$. The main advantage of the method is that we can proceed with just a succinct representation of the transition graph, that is a succinct description in an input language, which is the same in PRISM [2].

Our approximation problem is defined by giving as input x a succinct representation of a Markov decision process, a formula and a positive integer k. The succinct representation is used to generate a set of execution paths of length k.

A randomized approximation scheme is a randomized algorithm which computes with high confidence a good approximation of the probability measure $\mu(x)$ of the formula ϕ over the set of execution paths.

Definition 4 *A fully polynomial randomized approximation scheme (FPRAS) for a probability problem is a randomized algorithm $\mathcal{A}$ that takes an input x, two real numbers $0 < \varepsilon, \delta < 1$ and produces a value $A(x, \varepsilon, \delta)$ such that :*

$$Prob\big[|A(x, \varepsilon, \delta) - \mu(x)| \leq \varepsilon\big] \;\geq\; 1 - \delta.$$

The running time of $\mathcal{A}$ is polynomial in $|x|$, $\frac{1}{\varepsilon}$ and $\log \frac{1}{\delta}$.

The probability is taken over the random choices of the algorithm. We call ε the *approximation parameter* and δ the *confidence parameter*.

The APMC approximation algorithm consists in generating $O(\frac{1}{\varepsilon^2}.\log \frac{1}{\delta})$ paths, verifying the formula ϕ on each path and computing the fraction of satisfying paths.

Theorem 1 *The APMC approximation algorithm is a fully randomized approximation scheme for the probability $p = Prob_k[\psi]$ of an LTL formula ψ if $p \in]0, 1[$.*

This result is obtained by using Chernoff-Hoeffding bounds [16] on the tail of the distribution of a sum of independent random variables.

The complexity of the algorithm depends on $\log(1/\delta)$, this allows us to set δ to very small values. The dependence in ε is much more crucial, since the complexity is quadratic in $1/\varepsilon$.

6.2 Experiments

We used APMC on the same model as in the PRISM experiment. APMC is a distributed approximate model checker [13] implemented in ANSI C and freely available under the GPL License. It uses a client/server computation model to distribute path generation and verification on a cluster of machines. The model, formula and other parameters are entered by the user via the Graphical User Interface which runs on the server (master). Both the model and formula are translated into C source code, compiled and sent to clients (the workers) when they request a job. Regularly, workers send current verification results, receiving an acknowledgment from the master, to know whether they have to continue or stop the computation. Since the workers only need memory to store the generated code and one path, the verification requires very little memory. Furthermore, since each path is verified independently, there is no problem of load balancing.

Experimental conditions all experiments were run using a cluster of 75 Pentium IV 2GHz with 512 Mb of RAM under Linux 2.4.23. We set the approximation parameter $\varepsilon = 10^{-2}$ because this value allows the best tradeoff between verification quality and time, and the confidence parameter $\delta = 10^{-10}$.

The pseudo-random generator for paths was provided by the Gnu libc library 2.3.2. APMC provides three different compilation strategies. For these experiments, we used the most efficient strategy : synchronization at compile-time. This strategy uses more memory (still avoiding computing the whole model) but is much more efficient in term of speed.

The model for CSMA/CD is parameterized by 3 variables : λ, σ and α. We ran several experiments, for each of them we fixed different values of these three parameters.

In the following paragraphs, we present our experiments. For each verified property, we provide two results : one with the same parameters as used when verifying with PRISM (see section 5.2), the other one with the actual values of the CSMA/CD protocol (that is $\lambda = 780$, $\sigma = 24$, and $\alpha = 10$).

Verification of probabilistic reachability properties here, we approximate the probability of either sender correctly delivering its packet before a given deadline (property 1 of section 5.2). As a validation of the approximate method, we first compute this probability for $\lambda = 96$, $\sigma = 3$ and $\alpha = 6$. The results are presented in figure 6.

As expected, the results obtained with APMC are exactly the same than the results of figure 4. Figure 7 presents the results of the verification of the same property, using the actual values of the protocol. The curve of this figure is exact, to avoid false interpretations due to interpolation, we computed the values of the probability for each value of the deadline between 1000 and 2010 (this represents two computation days for 75 workers).

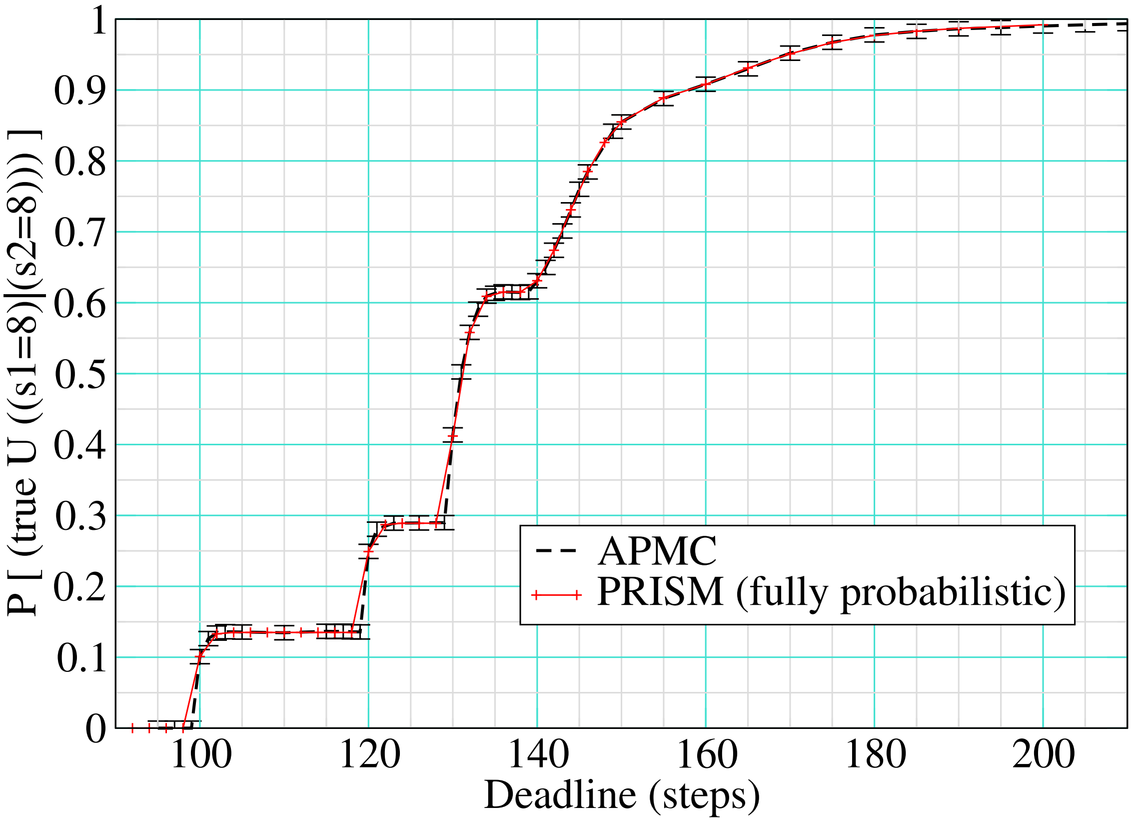
Time bounded probability of collisions We approximate the probability to get at least N collisions in an execution of the protocol, for different values of N (properties 2 and 3 of section 5.2) before a given deadline. As a validation of the approximate method, we first compute this probability for $\lambda = 96$, $\sigma = 3$, $\alpha = 6$ and $N \in [1; 6]$. We used a value of 210 for the deadline. The results are presented in figure 8.

As expected, this figure demonstrates that the probability computed by APMC is lower and upper bounded by the minimum and maximum probabilities computed by PRISM in properties 2 and 3.

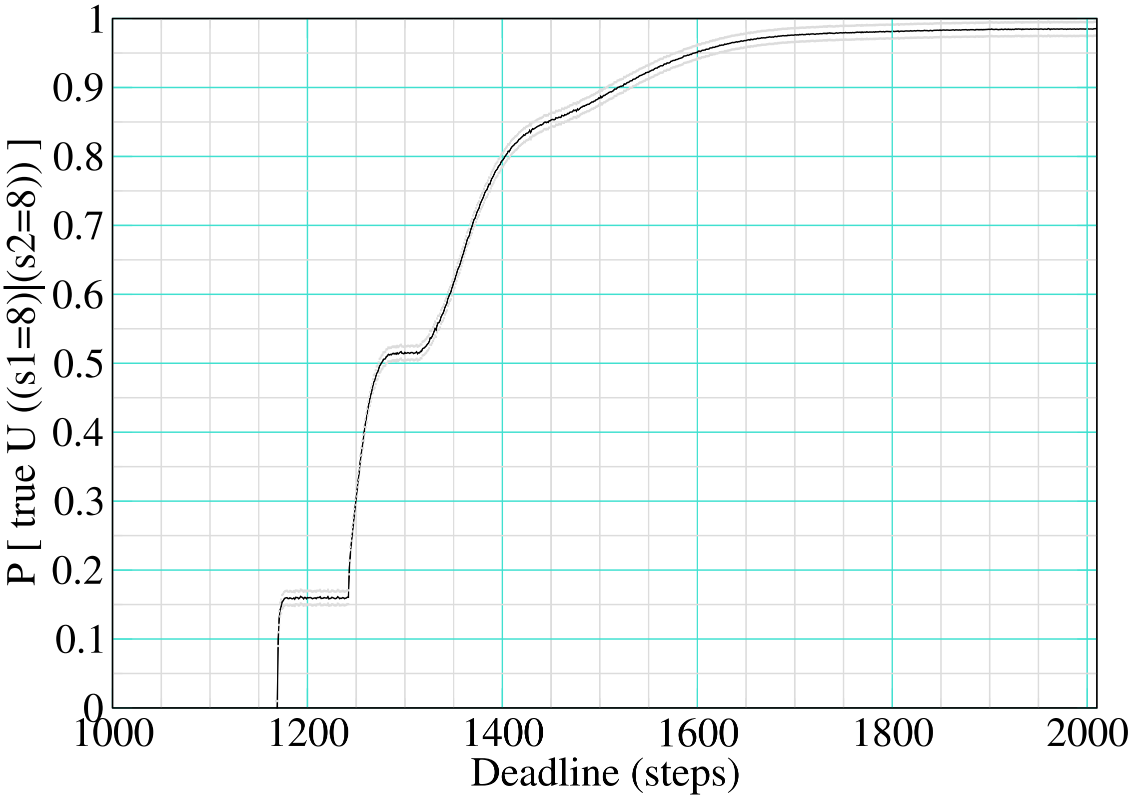
Figure 9 presents the same measure for the actual values, with a deadline of 2000. We ran a set of experiments with lesser values for the deadline, and the results are not surprising and show that the number of collisions decreases when the deadline decrease.

6.3 Analysis

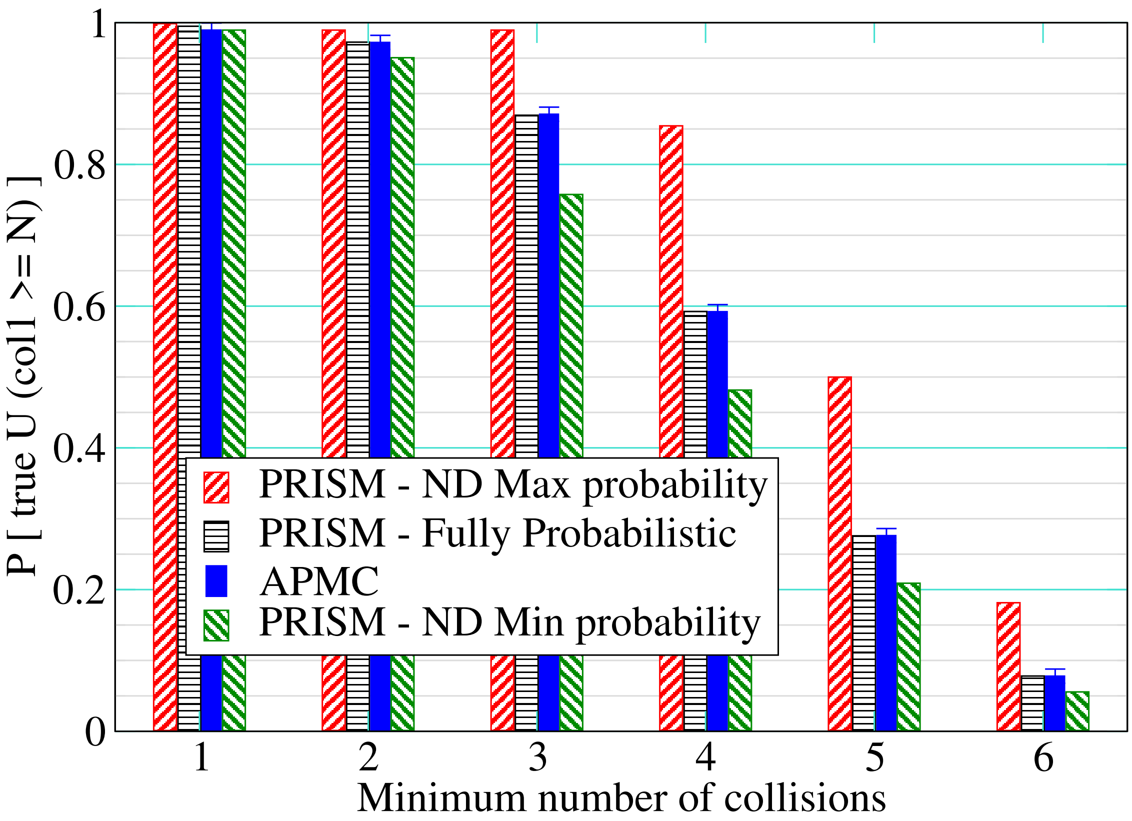
Figure 9 analyses the probability of having at least N collisions ($1 \leq N \leq 10$), for the actual values of the CSMA/CD protocol, with two senders emitting on a single channel. We can see that while $2^N \leq \sigma$, the probability of having N collisions is almost 1. As soon as $2^N > \sigma$, this probability decreases quickly (this is also shown in figure 5 for $\sigma = 3$). Since the two senders begin in same state, there is a first collision with great probability. They have the same initial value of *col*, so they pick at random a backoff in the same interval. For small values of N, the interval is not large enough to guarantee that the time difference between the two backoff is larger than σ. Now, when this difference is smaller than σ, none of the senders detects that the channel is busy, so they produce a new collision.



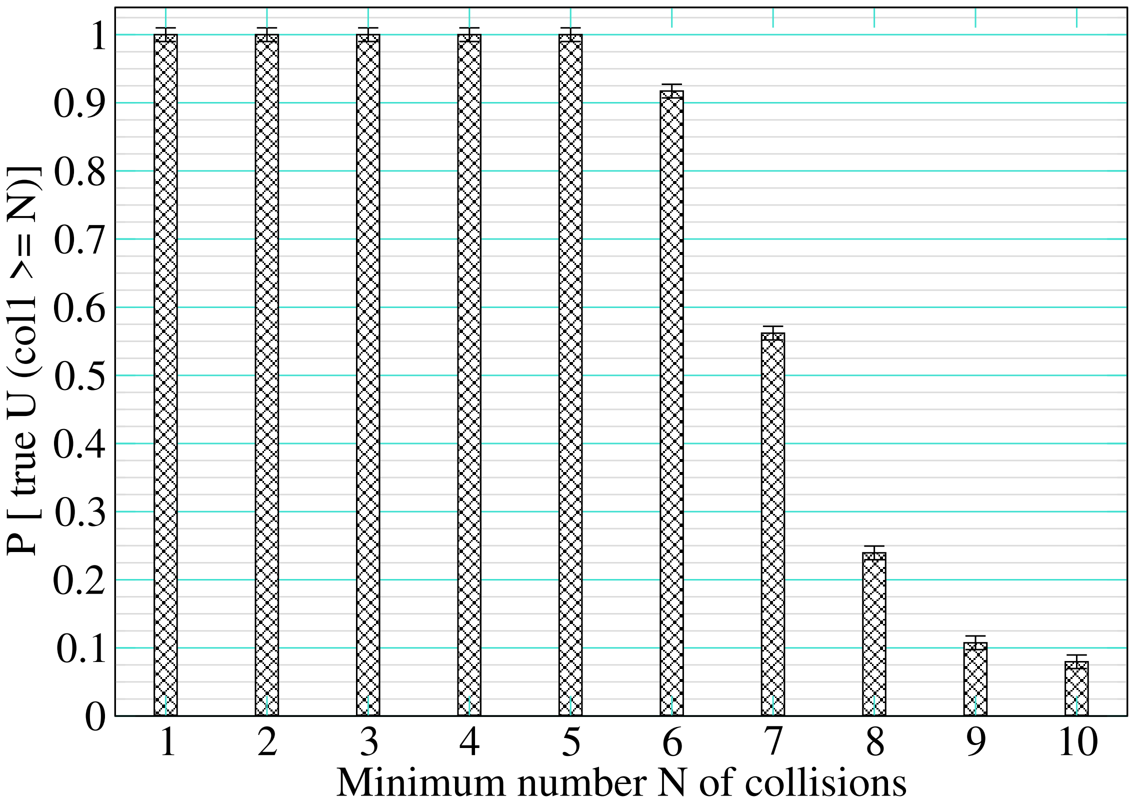
($\lambda = 96$, $\sigma = 3$, $\alpha = 6$)
Fig. 6 – Probability of emission as function of the deadline.



($\lambda = 780$, $\sigma = 24$, $\alpha = 10$)
FIG. 7 – Probability of emission as function of the deadline.



($\lambda = 96$, $\sigma = 3$, $\alpha = 6$, $deadline = 210$)
FIG. 8 – Probability of having more than N collisions, as function of N.



($\lambda = 780$, $\sigma = 24$, $\alpha = 10$, $deadline = 2000$)
FIG. 9 – Probability of having more than N collisions, as function of N.

One can observe that the protocol does not introduce too many collisions : the protocol is calibrated to handle at most 10 collisions, which seems sound, since in our experiment, where the senders start in the same initial state (the worst case for this measurement), we observe that the probability of having 10 collisions is very low.

Figure 6 and 7 analyse the probability of successfully sending a message from one of the two senders, as function of the deadline. We can see a gap from the zero probability to a non-zero probability of emission on both figures. The time T_0 of the first non-zero probability is given by the following formula deduced from the modelling, and confirmed by a set of complementary experiments

$$T_0 \simeq \overline{T}_{\text{Backoff}(N_{min})} + 3 \times N_{min} \times \sigma + \lambda$$

where N_{min} is the minimum number of unavoidable collisions, as given in figures 8 and 9, and $\overline{T}_{\text{Backoff}(N_{min})}$ is the average time spent during the corresponding backoff. For the actual values of the protocol, $N_{min} = 5$, and $\overline{T}_{\text{Backoff}(N_{min})} = 31$. Note that this is not a strict equality, since in our modelling, we spend a constant number of time units waiting that one of the two senders enter the sending state.

After this gap, the probability of emission increases quickly up to an asymptotic behavior, close to 1. This meets a goal of the protocol, being that the probability a message is transmitted after a short delay is high (for example, after two times its transmission delay the probability that a message is effectively transmitted is greater than 0.9).

Last, one can observe on the curve a persistent plateaux phenomenon. It is due to deadlines which are not large enough for a sender to enter the backoff process, and thus successfully send its packet. It is not a surprising result, since it was already shown in [12, 29], using other techniques, like simulation. We conducted some extra experiments (rough figures in appendix B) which suggest that the length of these plateaux is a linear function of σ and are independent of λ.

7 Discussion

In this paper we apply probabilistic and approximate model checking techniques to the verification of quantitatives properties of the CSMA/CD protocol. To our knowledge, this is the first study on the complementarity of these two techniques for such a model.

Here, we considered two different frameworks : MDPs and DTMCs. Some measurement can be achieved only on the DTMC model, which provide less accurate information. So, we first checked that the results of the DTMC model were as meaningful as the results on the MDP model. Figure 5 shows that for the experiment of the minimum number of collisions, it seems reasonable to consider fully probabilistic models. Indeed, one can see that the fully probabilistic measure follows the same tendency as the minimum probability.

Since both tools use different theories, another concern was to ensure that results from PRISM and APMC are equivalent. Figures 8 and 6 shows that the two tools obtain the same measures (up to the approximation parameter) for the same fully probabilistic models. We assumed that this property hold for other values of the parameters of the CSMA/CD protocol.

On one hand, using PRISM, we were able to verify the protocol as a probabilistic system with nondeterminism, thus modelling the asynchronous behavior in it, but with parameters smaller than the "real-life" protocol values. On the other hand, using APMC we were able to verify the protocol with the actual values, but with nondeterministic choices replaced by probabilistic choices.

Acknowledgments We would like to thank David Parker and Jeremy Sproston for helpful advice on PRISM.

Références

[1] R. Alur and D.L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126 :2, pp. 183-235, 1994.

[2] R. Alur and T. Henzinger. Reactive modules. in proceedings of *Logics In Computer Science*, 1996.

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

[4] APMC Web Page `http ://www.lri.fr/~syp/APMC`

[5] IEEE Web Page `http ://www.ieee802.org/3`

[6] M. Bernardo. Theory and application of Extended Markovian Process Algebra. PhD, University of Bologna (Italy), 1999.

[7] A. Bianco and L. de Alfaro. Model Checking of Probabilistic and Nondeterministic Systems. In *Proc. 15th Conference on Foundations of Software Technology and Theoretical Computer Science*, pp 499-513, LNCS :1026, 1995.

[8] D. Beyer. Improvements in BDD-based reachability analysis of timed automata. In *Proc. FME'01*, pp 318-343, LNCS :2021, 2001.

[9] D. R. Bogs, J.C. Mogul, and C.A. Kent. Measure capacity of an Ethernet : myths and reality. Proceedings of the SIGCOMM'88 Symposium on Communications Architectures and Protocols .

[10] C. Derman. *Finite-State Markovian Decision Processes*, academic Press, 1970.

[11] C. Daws, A. Olivero, S. Tripakis, and S. Yovine. The tool KRONOS. In *Hybrid Systems III : Verification and Control*, pp 208-219, LNCS :1066, 1996.

[12] T.A. Gonsalves and F.A.Tobagi. On the performance effects of station location and access protocol parameters in Ethernet networks. IEEE Transactions on Communications, vol.36, no 4, p. 441-449, april 1988.

[13] T. Herault, R. Lassaigne, F. Magniette and S. Peyronnet. Approximate Probabilistic Model Checking. In *Proceedings of Fifth International Conference on Verification, Model Checking and Abstract Interpretation (VMCAI'04)*, pp 73-84, LNCS :2937, January 2004.

[14] H.A. Hansson. Time and probability in formal design of distributed systems. PhD thesis, Uppsala University, Sweden, sept. 1991.

[15] H. Hermanns, M. Kwiatkowska, G. Norman, D. Parker and M. Siegle. On the use of MTBDDs for Performability Analysis and Verification of Stochastic Systems. In *Journal of Logic and Algebraic Programming : Special Issue on Probabilistic Techniques for the Design and Analysis of Systems*. pp 23-67, 56 :1-2, 2003.

[16] W. Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58 :13-30, 1963.

[17] B. Jeannet, P.R. D'Argenio and K.G. Larsen RAPTURE : A tool for verifying Markov Decision Processes. *Tools Day'02, Brno, Czech Republic, 2002.*

[18] J.P. Katoen. A semi-Markov model of a home network access protocol. Proc. of the Int. Workshop on Modeling, Analysis, and Simulation On Computer and Telecommunication Systems, p. 293-298, jan. 1993.

[19] J. Kemeny, J. Snell and A. Knapp. *Denumerable markov chains.* Springer-Verlag, 1976.

[20] M. Kwiatkowska, G. Norman and D. Parker. PRISM : Probabilistic Symbolic Model Checker. In *Proc. 12th International Conference on Modelling Techniques and Tools for Computer Performance Evaluation (TOOLS'02)*, pp 200-204, LNCS :2324, April 2002.

[21] M. Kwiatkowska, G. Norman, and J. Sproston. Probabilistic Model Checking of the IEEE 802.11 Wireless Local Area Network Protocol. Proc. of the 2nd Int. Worshop PAPM-PROBMIV 2002, LNCS no 2399, pp. 169-187, july 2002.

[22] M. Kwiatkowska, G. Norman, David Parker and J. Sproston. Performance Analysis of Probabilistic Timed Automata using Digital Clocks. *Proc. 1st International Workshop on Formal Modeling and Analysis of Timed Systems (FORMATS 2003)*, Marseille, Sept. 2003.

[23] 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, pp. 295-318, 2003.

[24] M. Kwiatkowska, G. Norman, R. Segala, and J. Sproston. Automatic verification of real-time systems with discrete probability distributions. *Theoretical Computer Science*, vol. 282, pp. 101–150, 2002.

[25] L. Lovasz and P. Winkler. Exact mixing time in an unknown markov chain. *Electronic journal of combinatorics*, 1995.

[26] H. Takagi and L. Kleinrock. Throughput analysis for persistent CSMA system. IEEE Transactions on Communications, vol.33, no 7, p. 627-638, july 1985.

[27] UPPAAL homepage. http ://www.docs.uu.se/docs/rtmv/uppaal/benchmarks/.

[28] M.Y. Vardi. Automatic verification of probabilistic concurrent finite-state programs. *Proc. 26th Annual Symposium on Foundations of Computer Science*, pages 327–338, 1985.

[29] J. Wang and S. Keshav. Efficient and accurate Ethernet simulation. Proc. of the IEEE 24th Conference on Local Computer Networks, p. 182-191, oct. 1999.

[30] PRISM homepage. http ://cs.bham.ac.uk/˜dxp/prism/.

[31] S. Yovine. Kronos : A verification tool for real-time systems. International Journal of Software Tools for Technology Transfer, Vol. 1, Issue 1/2, pages 123-133, oct. 1997, Springer-Verlag.

A : Probabilistic Timed Automata

The following material is essentially borrowed from [23]. A classical timed automaton [24] consists of a finitary directed control graph, the nodes of which are called *locations*, equipped with a finite set of real-valued variables called *clocks*, which are interpreted as increasing as the same rate as real-time. The edges of the control graph are enabled or forced to be taken depending on whether constraints on the clocks are satisfied. Furthermore, a set of clocks may be reset when an edge is executed. Probabilistic timed automata are timed automata for which discrete probability distributions range over the edges of the control graph.

Let $\mathcal{T} \in \{\mathcal{R}, \mathcal{N}\}$ 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 $\mathcal{T}$. A point $v \in \mathcal{T}^{|\mathcal{X}|}$ is refereed to as a *clock valuation*. Let $\mathbf{0} \in \mathcal{T}^{|\mathcal{X}|}$ be the clock valuation which assigns 0 to all clocks in $\mathcal{X}$. 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 $Zones(\mathcal{X})$ be the set of *zones* over $\mathcal{X}$, which are conjunctions of (closed) atomic constraints of the form $x \sim c$ for $x \in \mathcal{X}$, $\sim \in \{\leq, =, \geq\}$, and $c \in \mathcal{N}$. The clock valuation v *satisfies* the zone ζ, written $v \lhd \zeta$, if and only if ζ resolves to true after substituting each clock $x \in \mathcal{X}$ with the corresponding clock value from v. Let $Dist(Q)$ be the set of discrete probability distributions over subsets of Q.

Definition 5 *A* probabilistic timed automaton *is a tuple* $(L, \bar{l}, \mathcal{X}, \Sigma, inv, prob)$ *where :*
- L *is a finite set of* locations *including the* initial location $\bar{l} \in L$ *;*
- Σ *is a finite set of* events, *of which* $\Sigma_u \subseteq \Sigma$ *is declared urgent ;*
- *the function* $inv : L \to Zones(\mathcal{X})$ *is the invariant condition ;*
- *the finite set* $prob \subseteq L \times Zones(\mathcal{X}) \times \Sigma \times Dist(2^{\mathcal{X}} \times L)$ *is the* probabilistic edge relation.

A *state* of a probabilistic timed automaton is a pair (l, v) where $l \in L$ and $v \in \mathcal{T}^{|\mathcal{X}|}$ are such that $v \lhd inv(l)$. Informally, the behavior of a probabilistic timed automaton can be understood as follows. The model starts in the initial location $\bar{l}$ with all clocks set to $\mathbf{0}$, and hence the initial state is $(\bar{l}, \mathbf{0})$. 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 $(l, g, \sigma, p) \in prob$ with source location l which is *enabled*; that is g is satisfied by the current clock valuation v. Then the probability of moving to the location l' and resetting all the clocks in X to 0 is given by $p(X, l')$. In case (2), the option of letting the 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.

A notion of urgency can be associated with locations, in addition to events. Once an urgent location is entered, it must be left immediately, without time passing. Integer variables with bounded ranges, which can be tested withing enabling conditions and reset by edge distributions, can also be represented syntactically within the probabilistic timed automaton framework above by encoding the values of such variables within locations.

The *parallel composition* of timed automata extends naturally to the framework of probabilistic timed automata (See [21] for details.)

The semantics of probabilistic timed automata is defined in terms of transitions exhibiting both nondeterministic and probabilistic choice. Such models are Markov decision processes (see [10]).

Probabilistic Timed Automata As Markov Decision Processes We now give the semantics of probabilistic timed automa in terms of Markov decision processes. Strictly speaking, the definition should be parameterized both by the time domain $\mathcal{T}$ (either $\mathcal{R}$ or $\mathcal{N}$) and the time increment (addition on reals or integers). We focus here on the integer semantics. This is correct because we are interested here in stating the maximal (minimal) reachability probability of reaching a target location, which are equal in the continuous and integer semantics (see Theorem 1 of [21]). Hereafter, for any clock valuation $v \in \mathcal{N}^{|\mathcal{X}|}$ and time duration $t \in \mathcal{N}$, the expression $v \oplus t$ denotes the clock valuation of $\mathcal{X}$ which assigns the value $min\{v_x + t, k_x + 1\}$ to all clocks $x \in \mathcal{X}$, where k_x denote the greatest constant the clock x is compared to in the zones of PTA.

Definition 6 *Let $PTA = (L, \bar{l}, \mathcal{X}, \Sigma, inv, prob)$ be a probabilistic timed automaton. The* semantics *of PTA is the Markov decision process $(S, \overline{s}, Act, Steps)$ where :*

- $S \subseteq L \times \mathcal{T}^{|\mathcal{X}|}$ *and $(l, v) \in S$ if and only if $v \lhd inv(l)$;*
- $\overline{s} = (\bar{l}, \mathbf{0})$ *;*
- $Act = \mathcal{T} \cup \Sigma$ *;*
- $((l, v), a, \mu) \in Steps$ *if and only if one of the following condition holds :*
  - *Time transitions : $a \in \mathcal{T}$ and $\mu = \mu_{(l, v \oplus t)}$ such that*
    1. *$v \oplus t' \lhd inv(l)$ for all $0 \leq t' \leq t$, and,*
    2. *for all probabilistic edges of the form $(l, g, \sigma, -) \in prob$ if $v \lhd g$, then $\sigma \notin \Sigma_u$;*
  - *Discrete transitions : $a \in \Sigma$ and there exists $(l, g, \sigma, p) \in prob$ such that $v \lhd g$ and for any $(l', v') \in S$:*

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

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

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

The semantics of the parallel composition of two probabilistic timed automata corresponds to the semantics of the parallel composition of their individual semantic probabilistic systems.

B : Extra experiments