Lot 4.2

Technologie de modélisation

Probabilités

Application d'une méthode de preuve probabiliste pour prouver la terminaison en temps moyen fini du protocole CSMA/CA 802.11b

Description :	Nous présentons une méthode de preuve qui permet de montrer la terminaison en temps moyen fini d'un algorithme probabiliste et distribué utilisé par le protocole WI-FI 802.11b.
Auteur(s) :	Olivier BOURNEZ, Florent GARNIER, Claude KIRCHNER
Référence :	AVERROES / Lot 4.2 / Fourniture 4.2 / V1.0
Date :	2 mars 2006
Statut :	validé
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)

1 Introduction

Term rewriting has shown to be a very powerful tool in contexts where efficient methods for reasoning with equations are required [1, 12]. In the last decade, term rewriting has also shown to provide a very elegant framework for specifying concurrency models [14] and deduction systems [7, 3].

When specifying probabilistic systems, it is rather natural to consider that the firing of a rewrite rule can be subject to some probabilistic laws. For that purpose, we proposed in [6] to add basic probabilistic strategies to rule based languages. The idea of adding probabilities to rewrite rules has also been explored in [10] in the context of probabilistic constraint handling rules, or in [15]. The idea of adding probabilities to high level models of reactive systems has also been explored for models like Petri Nets [2, 17], automata based models [8, 18], or process algebra [11].

In a recent work [4], we developed techniques to prove termination with a finite mean of reductions (called positive almost sure termination) of a set of probabilistic rewrite rules.

The Ethernet and WI-FI network protocols use probabilistic primitives that require appropriate probabilistic models and tools for their verification [13, 9]. In this paper, we provide a probabilistic rule based model of the CSMA/CA 802.11b protocol and we prove that this model terminates in a time whose mean is finite, using the previously proposed frameworks. More precisely, we first show that for a given station, the probability to make a "good" transition—a transition which leads to the success state—is always positive, then we build a function mapping the terms to the set of positive real numbers such that all probabilistic rewrite rule decrease the value of this function by some quantity whose mean is positive. Thanks to the results of [4], we conclude.

Finding such a function in the general case, is not easy. However, in our case, the rules can be split in two several subsets:

1. Some rules coding administrative tasks (e.g. putting some subterms in normal form, sorting, etc. . .).
2. Other rules applicable on the normal forms of the previous set of rules. These include the rules coding transitions between CSMA/CA's main states.

Our approach is to present a simple combination construction that allow to build the wanted function by considering these subsets of rules somehow independently. Indeed, we show that if there exists a function whose mean value decreases at least by a fixed positive real number between these classes of states —i.e. each time a rule of the second set of rule is fired—, and if the first set of rules terminates, then the whole rewrite system *terminates in a finite mean time*.

This construction, exemplified here, helps the proof of positive almost sure termination in the general case by allowing modular constructions.

This paper is organized as follows. In Section 2, we recall probabilistic rewrite systems and some results of [4]. In Section 3, we discuss the combination of a probabilistic rewrite system and a term rewrite system. In Section 4, we describe our coding of the protocol using probabilistic rewrite rules. In the following section, we prove the protocol to be positively almost surely terminating using our tools.

2 Probabilistic Rewrite Rules

In [6], we introduced probabilistic abstract reduction systems (PARS). In the same way that abstract reduction systems are also called *transition systems* in other contexts, PARS corresponds to *Markov Decision Processes* [16]. The main points are that, compared to usual definitions of Markov decision processes, we explicitly allow states to be terminal and we do not label transitions by actions.

Definition 1 (PARS) *Given some denumerable set S, we note $Dist(S)$ for the set of probability distributions on S: $\mu \in Dist(S)$ is a function $S \to [0,1]$ that satisfies $\sum_{i\in S} \mu(i) = 1$.*

A probabilistic abstract reduction system (PARS) is a pair $\mathcal{A} = (A, \to)$ consisting of a countable set A and a relation $\to \subset A \times Dist(A)$.

A PARS is said deterministic if, for all a, there is at most one μ with $a \to \mu$.
A state $a \in A$ with no μ such that $a \to \mu$ is said terminal.

We now need to explain how such systems evolve: a *history* (of length $n+1$) is a finite sequence $a_0 a_1 \cdots a_n$ of elements of the state space A. It is non-terminal if a_n is. A *policy* ϕ, that can also be called a *strategy*, is a function that maps non-terminal histories to distributions in such a way that $\phi(a_0 a_1 \cdots a_n) = \mu$ is always one (of the possibly many) distribution μ with $a_n \to \mu$. A history is said *realizable*, if for all $i < n$, if μ_i denotes $\phi(a_0 a_1 \cdots a_i)$, one has $\mu_i(a_{i+1}) > 0$.

A *derivation of $\mathcal{A}$* is then a stochastic sequence where the non-deterministic choices are given by some policy ϕ, and the probabilistic choices are governed by the corresponding distributions. Formally:

Definition 2 (Derivations) *Assuming $\perp$ to be a fresh constant symbol (in particular not in A), a derivation π of $\mathcal{A}$ over policy ϕ is a stochastic sequence $\pi = (\pi_i)_{i \in \mathbb{N}}$ on $A \cup \{\perp\}$ such that for all n, $P(\pi_{n+1} = \perp | \pi_n = \perp) = 1$, $P(\pi_{n+1} = \perp | \pi_n = s) = 1$ if $s \in A$ is terminal, $P(\pi_{n+1} = \perp | \pi_n = s) = 0$ if $s \in A$ is non-terminal and, for all $t \in A$, $P(\pi_{n+1} = t | \pi_n = a_n, \pi_{n-1} = a_{n-1}, \ldots, \pi_0 = a_0) = \mu(t)$ whenever $a_0 a_1 \cdots a_n$ is a realizable non-terminal history and $\mu = \phi(a_0 a_1 \cdots a_n)$.*

If a derivation is such that $\pi_n = \perp$ for some n, then $\pi_{n'} = \perp$ almost surely for all $n' \geq n$. Such a derivation is said to be *terminating*. In other words, a non-terminating derivation is such that $\pi_n \in A$ ($\pi_n \neq \perp$) for all n.

Definition 3 (Almost Sure Termination) *A PARS $\mathcal{A} = (A, \to)$ is* almost surely (a.s) terminating *when for any policy ϕ, the probability that a derivation $\pi = (\pi_i)_{i \in \mathbb{N}}$ under policy ϕ terminates is 1, i.e. for all ϕ, $P(\exists n \text{ s.t. } \pi_n = \perp) = 1$.*

Definition 4 (Positive Almost Sure Termination) *A PARS $\mathcal{A} = (A, \to)$ is* positively almost surely (+a.s.) terminating *when for all policies ϕ, for all states $a \in A$, the mean number of reductions required to reach a terminal state starting from a and following policy ϕ, is finite.*

A positively almost surely terminating PARS is almost surely terminating.

In [4], we proved:

Theorem 1 (Soundness) *A PARS $\mathcal{A} = (A, \to)$ is +a.s. terminating if there exist some function $V : A \to \mathbb{R}$, with $\inf_{i \in A} V(i) > -\infty$, and some $\epsilon > 0$, such that, for all states $a \in A$, for all μ with $a \to \mu$, the drift in a according to μ defined by:*

$$\Delta_\mu V(a) = \sum_i \mu(i) V(i) - V(a)$$

satisfies:

$$\Delta_\mu V(a) \leq -\epsilon.$$

We also showed that the technique is complete for finitely branching systems.

Definition 5 (Probabilistic Rewrite system) *Given a signature Σ and a set of variables X, the set of terms over Σ and X is denoted by $T(\Sigma, X)$.*

A probabilistic rewrite rule *is an element of $T(\Sigma, X) \times Dist(T(\Sigma, X))$. A* probabilistic rewrite system *is a finite set $\mathcal{R}$ of probabilistic rewrite rules.*

To manipulate terms, we use the standard term rewriting notations [1]: When $t \in T(\Sigma, X)$ is a term, let $Pos(t)$ be the set of its positions, where the position of a subterm denotes the path to reach the subterm from the root of the term. For $\rho \in Pos(t)$, let $t|_\rho$ be the subterm of t at position ρ, and let $t[s]_\rho$ denote the replacement of the subterm at position ρ in t by s. The set of all substitutions is denoted by Sub.

Definition 6 (Reduction relation) *To a probabilistic rewrite system $\mathcal{R}$ is associated the following PARS $(T(\Sigma, X), \to)$ over terms: $t \to_{\mathcal{R}} \mu$ if there is a rule $(g, M) \in \mathcal{R}$, some position $p \in Pos(t)$, some substitution $\sigma \in Sub$, such that $t|_p = \sigma(g)$, and, for all t', $\mu(t') = \sum_{\{d \in T(\Sigma, X) | t' = t[\sigma(d)]_p\}} M(d)$.*

To a probabilistic rewrite system is associated a probabilistic abstract reduction system $(T(\Sigma, X), \rightarrow_{\mathcal{R}})$ over the set of terms $T(\Sigma, X)$ where $\rightarrow_{\mathcal{R}}$ is defined as follows:

We proposed several results to prove the positive almost sure termination of a set of probabilistic rewrite rules.

Definition 7 *For a given rewrite system $\mathcal{R}$ and for all $t \in T(\Sigma, X)$, $NF_{\mathcal{R}}(t)$ denotes the set of the normal forms of t. In the same vein $NF_{\mathcal{R}}(T(\Sigma, X))$ denotes the set of the normal terms for $\mathcal{R}$. When $\mathcal{R}$ is clear from the context, we may omit the subscript.*

3 Combination of a Probabilistic Rewrite System and a Term Rewrite System.

This section deals with the proof technique that will be used to certify that the modelisation of the CSMA/CA algorithm terminates within a finite mean time. Althrough we provide here a direct proof, this can also be seen as a consequence of the theorem about termination under strategies described in [5].

The direct application of Theorem 1 yielding almost sure termination is not appropriate because, in the full description of the model, we have more rules than it appears, e.g. we need to code some priority queue, sorting operations etc, and building a function whose mean decreases for so many transitions is far from being obvious. Hopefully, thanks to the result described below, we will see that it is possible on one hand to only consider the probabilistic rewrite rules which must satisfy the decreasing property for a given lower bounded function if, on the other hand, we can prove that the derivations generated by the remaining rules have a length that can be explicitly bounded. Let us now see this in a formal way,

Let $T_p = (T(\Sigma, X), \rightarrow_{\mathcal{R}})$ be a probabilistic rewrite system and $T_s = (T(\Sigma, X), \rightarrow)$ a (classical) rewrite system coded as a probabilistic rewrite system (e.g. all the probabilities are fixed to 0 or 1). Suppose that there exists a set A of terms, so that if a term t is included in A, it is reduced using a probabilistic rule of T_p else if t is not in A, it is reduced using a rule of T_s. We suppose that the set of the normal forms of T_s equals A.

Definition 8 (The Sequential Combination of a TRS and a PRS) *The sequential combination of a PRS T_p and a TRS T_s is the PRS $(T(\Sigma, X), \rightsquigarrow)$ whose probabilistic rewrite relation is defined by:*

- *$t_1 \rightsquigarrow t_2$ when there exists t_1' such that $t_1 \rightarrow t_1'$ by PRS T_p in one step and $t_1' \xrightarrow{!} t_2$, where $t_1' \xrightarrow{!} t_2$ means that t_1 rewrites by a sequence of derivation of TRS T_s to $t_2 \in NF(t_1')$.*

We denote $T_s!$ the strategy consisting in normalizing using the rules of T_s.

We provide sufficient conditions to insure that a subset B of A will be reached in a finite mean time, independently from the choice of the first term to reduce and independently from the followed policy [4].

Definition 9 (Drift between normal forms) *Assume that we have a function V from A to $\mathbb{R}$. The* drift between normal forms *for V, for a given policy ϕ and the history $t_1 \dots t_n$ is the following value:*

$$\Delta_{\phi(t_1 \dots t_n)}^{NF(T(\Sigma,X),\rightarrow)} V(t_n) = \sum_{t \in T(\Sigma,X)} V(NF_{\phi}(t)) \times \phi(t_1 \dots t_n)(t) - V(t_n)$$

which represents the mean variation of V once the term is put in normal form by $(T(\Sigma, X), \rightarrow)$.

Let us recall that we are looking for sufficient conditions entailing that $B \subseteq A$ a set of terms considered as terminal, will be reached in a finite mean time, knowing that after that a term of A is reduced with a probabilistic rule of T_p the reduced term will be reduced to a normal form of T_s

Lemma 1 (Positive almost surely termination of the combination) *If there exits some ϵ, some $V : A \rightarrow \mathbb{R}$, some function $Size : T(\Sigma, X) \rightarrow \mathbb{N}$ and some non-decreasing function $L : \mathbb{N} \rightarrow \mathbb{N}$, and:*

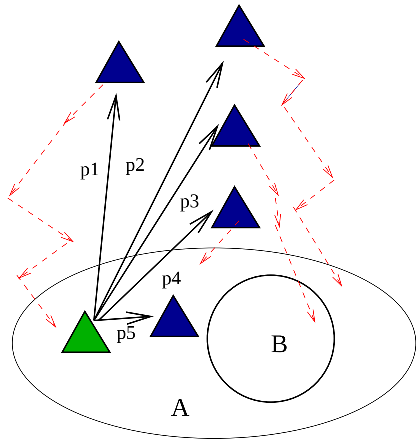


Figure 1: An example of reductions: green term is rewritten into the the five blue other terms by a probabilistic rewrite rule. Once a successor of the green term has been chosen, it is rewritten using the classical term rewrite system (in red) until a normal form is reached.

1. *The length of any derivation from a term t of $T(\Sigma, X)$ by T_s is less or equal to $L(Size(t))$.*
2. *If $t \rightsquigarrow t'$ (formally $\rightsquigarrow (t,t') > 0$) then $Size(t) \geq Size(t')$*
3. $\forall t \in A, \forall \phi, \forall t_1 \dots t_n \ \Delta^{NF(T(\Sigma,X),\rightarrow)}_{\phi(t_1 \dots t_n)} V(t_n) \leq -\epsilon$
4. *V is lower bounded and reaches its minimum in B and only in B.*

Then $T_p \cup T_s$ is positively almost surely terminating. Furthermore, the terminal terms of $T_p \cup T_s$ are precisely the terms of B.

Proof. Condition 3 and 4, combined with Theorem 1, provide positive almost sure termination of probabilistic relation $\rightsquigarrow$. Conditions 1 and 2, guarantee that each rewrite step of $\rightsquigarrow$ on a term t correspond to at most $1 + L(Size(t))$ rewrite steps of $T_p \cup T_s$. $\square$

4 Model

We now consider our model of the CSMA/CA protocol in infrastructure mode, that is to say in the case where there exists a central station called "hub", that regulates the transmission, centralises messages and broadcasts to every other stations. The CSMA/CA protocol is widely used today in wireless networks. It aims at handling appropriately collisions, i.e. the situation where two stations emit during the same time slot, by using a virtual locking mechanism.

To lock the medium, one station has to wait that the medium is free during $\mathtt{DIFS} + \mathtt{backoff}(c)$ time units, where:

- *`DIFS`* is a constant fixed in the 802.11 specification,
- *`backoff`*(c) is a random variable following an uniform law on the set $0, \dots, 2^{min(c,10)}$, and
- c is the number of collisions since last successful transmission.

If during this amount of time, the station has not eared anything then it sends a RTS message (Request To Send) and wait the central station authorization, given by a CTS message (Clear to Send) during a period of at most `timeout` time units. If the `CTS` message is received on time, then the station emits.

When the whole message has been received by the central station, an acknowledgment message (ACK) is sent to the sender, with a checksum of the message.

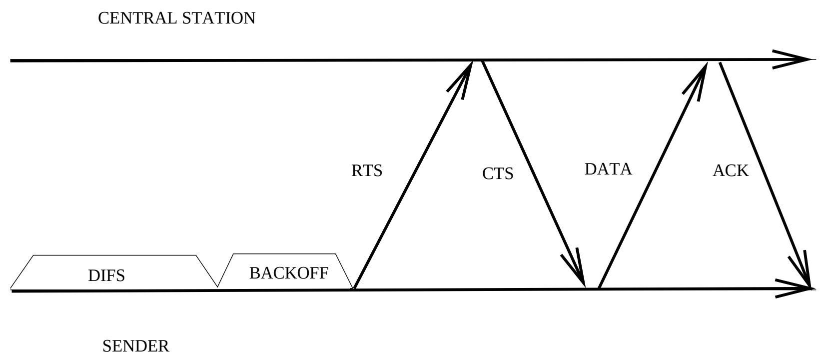
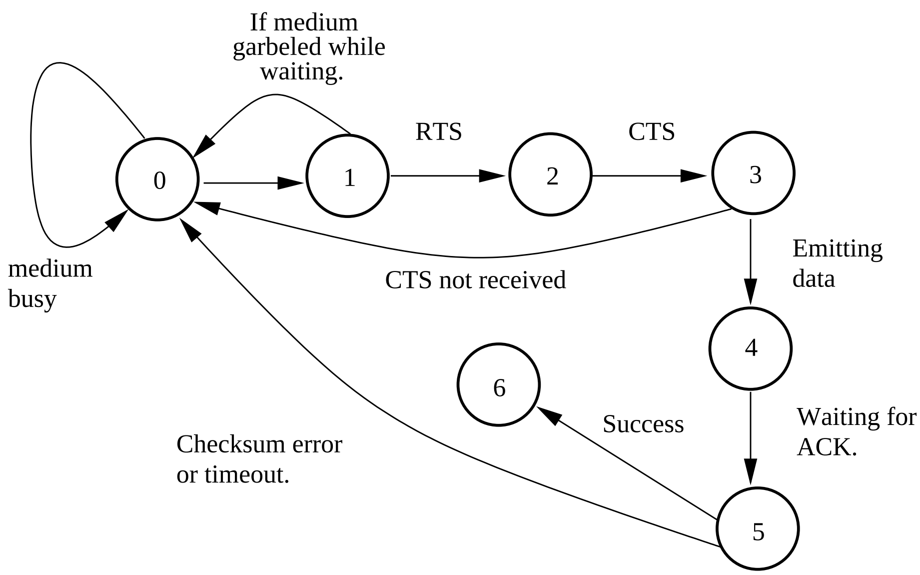


Figure 2: Successful CSMA/CA's dialogue between a station and the central station.

The algorithm in each sender can be represented by the following automata:



with the following meaning of the states:

State	description
0	Ear the medium, wait it become free, calculate backoff(c)
1	Has waited DIFS+backoff(c) time units.
2	Central station received RTS, send a CTS
3	Station received CTS.
4	Station send a data tram.
5	Station received an ACK tram.
6	Packet successfully transmitted.

This automata is implemented using (probabilistic) rewrite rules as follows:

- A sender is represented as a 5-tuples $(state, wait\_time, nb\_mess, c, write)$ where
  1. `state` is the current state of the sender ($state \in \{0, 1, \ldots, 6\}$),

2. `wait_time` is the remaining time before next transition,
3. `nb_mess` is the number of messages still to be sent,
4. `c` is the number of collisions since last successful transmission,
5. `write` switch monitoring other sender access to the medium.

- The whole system is represented by (t, l), where t is an integer coding the current date, and l is the list of 5-tuples coding senders.

The following hypotheses are made:

1. Station 1 can here the $n - 1$ stations, called $2, \ldots, n$,
2. There is k hidden stations, $n + 1, \ldots, n + k$,
3. Each messages sent by the base station can be heard by all the stations.
4. Trams CTS and ACK are never garbled.
5. c_i denotes the number of collisions for station i, since last successful transmission.

The main point of probabilistic rewrite rules is to let the global state (t, l) of the system evolve according to the protocol. This is done in ELAN4 by rules of type:

```
[] simul((x,cons(snd,l)) => simul((x+snd.wait_time, runfirst_sortlist(cons(snd,l))
                            if jobleft(cons(snd.l))
[] simul((x,cons(snd,l)) => (x,cons(snd,l))
                            if !jobleft(cons(snd,l))

[] runfirst_sortlist((x,cons(snd,l)) => sortlist(nextstep(cons(snd,l)))
```

Where:

1. `x` is the current date,
2. `cons` is the list constructor,
3. `nextstep(l)`, where `l` is a list, aims at computing the result of the transition of the first sender of list `l` according to the automata.
4. `sortlist(l)`, where `l` is a list, aims at sorting senders according to waiting time: it returns list `l` in an order such that the first sender in the obtained list is the next one to evolve. In other words, it sorts the senders by value of `wait_time`.

For example, the transitions of the automata are encoded as follows:

```
[] nextstep ( cons(snd , l)) => cons((4,Tsendpacket,snd.nb_mess,snd.c,0),markwritten(l))
                                 if snd.write==1 and snd.state==3

[] nextstep ( cons(snd , l)) => cons((0,timeout,snd.nb_mess,snd.c,0),l)
                                 if snd.write==0 and snd.eta==3

[] nextstep ( cons(snd , l)) => cons((0,0,snd.nb_mess,snd.c+1,0),l)
                                 if snd.write==2 and snd.eta==3
```

These three rules code the outgoing transitions from the state number 3. The first applies if the `write` fields equals to 1 and the field `state` equals 3, and we modify the field `state` to 4 which means that the first sender of the list will transit to the state number 4 and the transition's duration will be `Tsendpacket`. The two other rules code a return to the zero state for the first sender because of errors notification.

Let now precise how the behavior of the different stations are synchronized in function of their access to the medium. Previously we mentioned that the field `write` of each station was used to testify if another station did a write access to the medium during the current transition.

The first rule of the example says that the list `l` is rewritten to `markwritten(l)` in the simulation term. The `markwritten` term triggers rewrite rules which change the value of the field `write` of each element of the list `l` to 2. Thanks to this, each term coding a sender, contains the information of what happened to the medium during their last transition. When we look at three previous rules, we can see that each time we take care of the write field's value, and we transit to the next state in function of this value.

5 Positive almost sure termination

We now prove termination that the protocol is correct, by proving the following fact: any term (t, l), where l encodes n senders with `wait_time`=0, is ultimately reduced in a number of rewrite rules whose mean is finite to a term (t', l'), where l' encoding a list of n senders with `state` = 6, `nb_mess`=0. I.e. all messages are ultimately sent in a finite mean time.

To do so, we use Lemma 1. We take B as the subset of the terms (t', l'), where l' encodes a list of senders with `state` = 6, `nb_mess`=0. These terms are irreducible.

We are going to call T_p all the probabilistic[1] rewrite rules acting on operator `nextstep`: i.e. all the rules encoding the transitions of the automaton.

Let T_s denote all the other rewrite rules. These correspond to classical rewrite rules, since no probabilities is involved.

We take A to be the set of terms $t \in T(\Sigma, X)$ such that there is some unique position $p \in Pos(t)$, such that the subterm at position p is of the form `nextstep(l)`, where `l` is a list sorted with respect to `wait_time`.

To apply Lemma 1, we need to build a function $V : A \to \mathbb{R}$ that satisfies the hypotheses of this lemma.

We prove that if there exist three positive real numbers α, β and γ such that:

- Any term coding a state where the first sender is in state 1 rewrites by $T_p \cup T_s!$ to a term where this sender is in state 2 with probability greater than α.
- Any term coding a state where the first sender is in state 3 rewrites by $T_p \cup T_s!$ to a term where this sender is in state 4 with probability greater than β.
- Any term coding a state where the first sender is in state 4 rewrites by $T_p \cup T_s!$ to a term where this sender is in state 5 with probability greater than γ.

Then there exist V certifying the positively almost sure termination of the simulation algorithm, and we compute those three real numbers latter.

In the same way, we suppose that the probability to go to 5 from 6 is positive, because of the virtual channel locking.

Consider the function $V : A \to \mathbb{R}$ defined by:

$$V(...(\texttt{nextstep}(l_1.l_2.\ldots.l_n)...) = K \times (\sum_{i=1}^{n} \texttt{nb\_mess}(l_i) + W(\texttt{state}(l_i)))$$

[1]Only the rule `[] nextstep ...` describing the transition from the state 0 is a probabilistic rewrite rule that depends on the `backoff` result.

where

$$
\begin{array}{rcl}
K & = & \frac{1-2(1-\alpha)(1-\beta)+3(1-\gamma)(1-\beta)(1-\alpha)}{(1-\alpha)(1-\beta)(1-\gamma)} \\
W(0) & = & K \\
W(1) & = & K-1 \\
W(2) & = & K-\frac{\alpha-2}{1-\alpha} \\
W(3) & = & K-\frac{2\alpha-3}{1-\alpha} \\
W(4) & = & K-\frac{\alpha-(1-\alpha)(1-\beta)}{(1-\alpha)(1-\beta)} \\
W(5) & = & K-\frac{-2(1-\alpha)(1-\beta)+\alpha}{(1-\alpha)(1-\beta)} \\
W(6) & = & 1 \\
nbmess((\texttt{state,...,nb\_mess,...,write})) & = & \texttt{nb\_mess} \\
state((\texttt{state,...,write})) & = & \texttt{state}
\end{array}
$$

Remark 1 *One can see that we choose $\epsilon = 1$ to simplify the reading of the function W and the computation of the function V.*

To shorten the notations, one denotes here by $V(i.l)$ the valuation of a list `cons(snd,l)` where `state(snd)=`i. This function satisfies for all l:

$$
\left\{
\begin{array}{lll}
\forall\ p_1 \geq \alpha > 0 & p_1 \times V(2.l) + (1-p_1) \times V(0.l) < V(1) - \epsilon & (1) \\
\forall\ 1-\beta \geq p_2 > \beta > 0 & p_3 \times V(3.l) + (1-p_3) \times V(0.l) < V(3) - \epsilon & (2) \\
\forall\ p_5 \geq \gamma > 0 & p_5 \times V(6.l) + (1-p_5) \times V(0.l) < V(5) - \epsilon & (3)
\end{array}
\right. \qquad (1)
$$

for $\epsilon = 1$.

It is quite easy to see that all permutations of the senders in the list do not change the system valuation by V, the same is true when the fields `write, c, waiting_time` are modified. In other words, when a nextstep rule is applied on a term of A, the valuation of term by V do not change by the rewrite rules of T_s.

$\forall t_1 \dots t_n,\ \forall \phi,$

$$
\begin{array}{rcl}
\Delta_{\phi(t_1 \dots t_n)}^{NF(T(\Sigma,X),\rightarrow)} V(t_n) & = & \sum_{t \in T(\Sigma,X)} V(NF_\phi(t)) \times \phi(t_1 \dots t_n)(t) - V(t_n) \\
 & = & \sum_{t \in T(\Sigma,X)} V(t) \times \phi(t_1 \dots t_n)(t) - V(t_n) \\
 & \leq & -\epsilon,
\end{array}
$$

for $\epsilon = 1$, where last inequality follows from inequations 1.

Hence, we have hypothesis 3 of lemma 1. The rewrite system T_s mostly codes the insertion of an element in a list of size n - where n is the size of the sender list- with respect to a simple order, plus a scan on the same list to update the `wait_time` field of each sender. In other word, the length of any derivation of T_s is always bounded by a polynomial function of the first degree in the number n of senders: hence we can take as $Size$ function the function that returns for any list coding a valid list of senders the number n of senders in the list, and as function L this polynomial function, to get hypothesis 1 of lemma 1.

Since all rules preserve the size n of lists, the rewrite derivation of T_s are finites, and hence hypothesis 2 of lemma 1 is satisfied.

The function V is a sum of a finite number of positive number, thus is lower bounded, and reaches its minimum when each station has zero message to send and is in the state number 6, which describe the set B of terminal terms. This shows that hypothesis 4 of lemma 1 is satisfied.

Lemma 1 then implies the correctness of the protocol: whatever is the initial state of the protocol, ultimately all messages will be sent in a finite mean time.

The only remaining point is to prove that α, β and γ are non equal to 0.

5.0.1 Computing α

The sole phenomenon which may cause the automaton to return 0 once in 1 is that one station among the $n-1$ other has emitted a signal. This signal will necessary be a RTS tram, because all the station are synchronized by the the base station for starting the waiting phase of the algorithm. The probability that the automaton 1 emits RTS in the state 1 equals the probability that:

$\texttt{backoff}(c_1) \leq \texttt{backoff}(c_i) \forall i \in \{1, \ldots, n\}$

$$P(Succes1)^2 = P(\bigcap_{i=2}^{n} \texttt{backoff}(c_1) \leq \texttt{backoff}(c_i))$$

$$P(Succes1) = \prod_{i=2}^{n} P(\texttt{backoff}(c_1) \leq \texttt{backoff}(c_i))$$

The worst case happens when 1 has more than 10 collisions since the last successfully sent message and that all the other station had no collisions.

$$P(Succes1) \geq \prod_{i=2}^{n} P(\texttt{backoff}(10) \leq \texttt{backoff}(1))$$

$$P(Succes1) \geq \frac{1}{2^{10\times n}}$$

We can then fix $\alpha = \frac{1}{2^{10\times n}}$

5.0.2 Computing β

Two phenomenon may cause a failure,

1. A RTS packet is sent during the same time slot by one station among the $n-1$ other visible stations.
2. There is a jamming caused by one of the k hidden stations.

Let us compute a high bound of the probability that the cause number one occurs. Such a phenomenon happens when another station has the same `backoff` as the current one, that means,

$$P(Succes2)^3 = P(\forall i \ \texttt{backoff}(c_1) \neq \texttt{backoff}(c_i))$$

If, $\texttt{backoff}(c_1) \neq \texttt{backoff}(c_i)$ then $\texttt{backoff}(c_1) < \texttt{backoff}(c_i))$

$$P(Succes2) = P(\forall i \ \texttt{backoff}(c_1) < \texttt{backoff}(c_i))$$

$$P(Succes2) = \sum_{j=1}^{2^{10}} P(\texttt{backoff}(c_i) > i | \texttt{backoff}(c_1) = j) \times P(\texttt{backoff}(c_1) = j)$$

$$P(Succes2) = \sum_{j=1}^{2^{10}} \prod_{i=2}^{n} \frac{2^{c_i} - j}{2^{c_i}} \times 1_{j \leq 2^{c_i}} \times \frac{1}{2^{c_1}} \times 1_{\{j \leq 2^{c_1}\}}$$

where $1_{\{C\}}$ equals 1 when condition C is satisfied and 0 otherwise. Let us note $p = min_{i\in\{1,\ldots,n\}} c_i$

$$P(Succes2) = \sum_{j=1}^{2^{10}} \prod_{i=2}^{n} \frac{2^{c_i} - j}{2^{(n-1)c_1 + \sum_{i=2}^{n} c_i}} = \beta_1$$

Let us now compute the probability that the second phenomenon occurs,

[2] The set of events $Succes1$ denotes the successful firing of the rewrite rule coding the transition from the state 1 to 2
[3] The set of events $Succes2$ denotes the successful firing of the rewrite rule coding the transition from 3 to 4

$$P(Succes2) = P(\forall i \in \{n+1, \dots, n+k\}\ \texttt{backoff}(c_1) < \texttt{backoff}(c_i))$$

$$P(Succes2) = \sum_{j=1}^{2^{10}} \prod_{i=n+1}^{n+k} P(\texttt{backoff}(c_1) < \texttt{backoff}(c_i) = j) \times P(\texttt{backoff}(c_1) = j)$$

$$P(Sucess2) = \sum_{j=1}^{2^{10}} \prod_{i=n+1}^{n+k} \frac{j}{2^{c_i} \times 2^{c_1}} \times 1_{\{j \le 2^{c_i} \cap j \le 2^{c_1}\}}$$

Let us note $p' = min_{i \in \{n+1,\dots,n+k\}} c_i$

$$P(Succes2) = \sum_{j=1}^{2^{p'}} \prod_{i=n+1}^{n+k} \frac{j}{2^{c_1+c_i}}$$

$$P(Sucess2) = \sum_{j=1}^{2^{p'}} \frac{j^k}{2^{kc_1 + \sum_{i=n+1}^{n+k} c_i}}$$

$$1 > \beta_2 = \frac{j^k}{2^{kc_1 + \sum_{i=n+1}^{n+k} c_i}} > 0$$

To successfully transit from 3 to 4, the whole system must satisfy $Succes1$ and $Succes2$. Because of $Success1$ and $Success2$ are independent events, because of the two set of station are disjoints and each station's `backoff`is independent from all the other.

$$P(Success) = P(Sucess1 \cap Sucess2) = P(Sucess1) \times P(Sucess2)$$

$$P(Success) \ge \beta_1 \times \beta_2$$

$$\beta = \beta_1 \times \beta_2 > 0$$

Remark that, when transiting from state 3 to reach state 5 in two steps, all the other stations are aware that the current transmission is started. Therefore, they will wait at least `DIFS` time units with `DIFS` bigger than the time required to send the data packets and receiving the `ACK` data packet. The constant γ allows to model events such data corruption or transmission errors.

By computing the two constants α and β, we finished to prove the termination of the simulation algorithm in a finite mean number of rewrite steps. We can notice that the rules matching on terms containing the `nextstep` subterms are applied a finite mean number of time and increase the global date by at most 2^{10} time units. This is enough to say that the mean value of the global date will be finite, for all simulations.

6 Upperbounding the time before termination

We have shown that the protocol simulation algorithm terminates in a finite mean time. Moreover, the proofs in [4] show that any function V that satisfies the hypothesis of Theorem 1 with $\epsilon = 1$ actually provides an upper bound on the mean time before termination: any term t will rewrite to some terminal one in a number of rewrite steps whose mean is upper bounded by $V(t)$.

Combined with the constructions used here, we get that the function V built in previous section also provides an upper bound on the mean time before termination of the rules simulating the protocol.

More precisely, any term $\texttt{simul}(t, l)$ rewrites to a term of A in a number of rewrite steps whose mean is less than $V(\texttt{simul}(t, l))$.

We therefore obtain our main result:

Proposition 1 *Starting with n senders requesting to send some f packets,*

- *the whole simulation will end in a terminal state (all packets sent) in a number of rewrite steps less than* $V(\texttt{simul}(...\texttt{cons}(t,l)...) \times L(n)$.
- *the whole simulation will end in a terminal state (all packets sent) in a number of rewrite steps of* T_p *less than* $V(\texttt{simul}(...\texttt{cons}(t,l)...)$.
- *the whole protocol will end in a terminal state (all packets sent) in a time less than than* $V(\texttt{simul}(...\texttt{cons}(t,l)...) \times M$.

where $M = 2^{10}$ *and* $L : \mathbb{N} \to \mathbb{N}$ *is a first degree polynomial upper bounding the number of* T_s*-rewrite steps occurring in the reduction of a given simulation term to a simulation term of* A.

These results give us a qualitative information on the system and a quantitative one about upper-bounding the mean execution time of the algorithm. Such a work is a particular example of current investigations about proving quantitative and qualitative properties of probabilistic systems using rule based model.

References

[1] Franz Baader and Tobias Nipkow. *Term Rewriting and all That*. Cambridge University Press, 1998.

[2] Gianfranco Balbo. Introduction to stochastic Petri nets. *Lecture Notes in Computer Science*, 2090:84, 2001.

[3] Gilles Barthe, Horatiu Cirstea, Claude Kirchner, and Luigi Liquori. Pure Patterns Type Systems. In *Principles of Programming Languages - POPL2003, New Orleans, USA*. ACM, January 2003.

[4] Olivier Bournez and Florent Garnier. Proving positive almost sure termination. volume 3467 of *Lecture Notes in Computer Science*, pages 323–337, Nara, Japan, 2005. Springer.

[5] Olivier Bournez and Florent Garnier. Proving Positive Almost Sure Termination Under Strategies. In *To appear in RTA 2006*. Springer Verlag, 2006.

[6] Olivier Bournez and Claude Kirchner. Probabilistic rewrite strategies. Applications to ELAN. In S. Tison, editor, *13th International Conference on Rewriting Techniques and Applications - RTA'2002, Copenhagen, Denmark*, volume 2378 of *Lecture Notes in Computer Science*, pages 252–266. Springer, July 2002.

[7] Horatiu Cirstea and Claude Kirchner. The rewriting calculus — Part I *and* II. *Logic Journal of the Interest Group in Pure and Applied Logics*, 9(3):427–498, May 2001.

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

[9] Marie Duflot, Laurent Fribourg, Thomas Hérault, Richard Lassaigne, Frédéric Magniette, Stéphane Messika, Sylvain Peyronnet, and Claudine Picaronny. Probabilistic model checking of the CSMA/CD protocol using PRISM and APCM. In Michael R. A. Huth, editor, *Proceedings of the 4th International Workshop on Automated Verification of Critical Systems (AVoCS'04)*, volume 128 of *Electronic Notes in Theoretical Computer Science*, pages 195–214, London, UK, May 2005. Elsevier Science Publishers.

[10] Thom Frühwirth, Alexandra Di Pierro, and Herbert Wiklicky. Toward probabilistic constraint handling rules. In Slim Abdennadher and Thom Frühwirth, editors, *Proceedings of the third Workshop on Rule-Based Constraint Reasoning and Programming (RCoRP'01)*, Paphos, Cyprus, December 2001. Under the hospice of the International Conferences in Constraint Programming and Logic Programming.

[11] H. Hansson. *Time and Probability in Formal Design of Distributed Systems*. Series in Real-Time Safety Critical Systems. Elsevier, 1994.

[12] Jan Willem Klop. Term rewriting systems. In S. Abramsky, D. M. Gabbay, and T. S. E. Maibaum, editors, *Handbook of Logic in Computer Science*, volume 2, chapter 1, pages 1–117. Oxford University Press, Oxford, 1992.

[13] Marta Z. Kwiatkowska, Gethin Norman, and Jeremy Sproston. Probabilistic model checking of the IEEE 802.11 wireless local area network protocol. In *PAPM-PROBMIV*, pages 169–187, 2002.

[14] J. Meseguer. Conditional rewriting logic as a unified model of concurrency. *Theoretical Computer Science*, 96(1):73–155, 1992.

[15] Kumar Nirman, Koushik Sen, Jose Meseguer, and Gul Agha. A rewriting based model for probabilistic distributed object systems. In *Proceedings of 6th IFIP International Conference on Formal Methods for Open Object-based Distributed Systems (FMOODS'03)*, volume 2884 of *Lecture Notes in Computer Science*, pages 32–46. Springer, Paris, France, November 2003.

[16] M.L. Puternam. *Markov Decision Processes - Discrete Stochastic Dynamic Programming*. Wiley series in probability and mathematical statistics. John Wiley & Sons, 1994.

[17] William H. Sanders and John F. Meyer. Stochastic activity networks: Formal definitions and concepts. *Lecture Notes in Computer Science*, 2090:315, 2001.

[18] M. Y. Vardi. Automatic verification of probabilistic concurrent finite-state programs. In *focs85*, pages 327–338, 1985.