Probabilistic Lossy Channel Systems[1]

Purush Iyer and Murali Narasimha
Dept of Computer Science
North Carolina State University
Raleigh, NC 27695-8206
Tel: +1 919 515 7291
Fax: +1 919 515 7896
e-mail: `purush@csc.ncsu.edu, mnarasi@eos.ncsu.edu`

Abstract

Consider a system of finite state machines communicating with each other over unbounded FIFO buffers. Such a model of computation is, clearly, turing powerful. This model has been used as the backbone of ISO protocol specification languages Estelle and SDL, as it allows one to abstract away from the details, such as errors in communication, that occur at lower levels of the protocol stack. It has recently been shown (in the literature) that realistic models which implicitly model errors in the communication buffers are more tractable than models which assume perfect communication. In this paper, we propose to make the model more realistic by modeling the probability of loss in the buffers. Given specifications in such a model we provide algorithms for the *probabilistic reachability* problem and the *probabilistic model-checking* (in linear-time PTL) problem.

[1]Research supported in part by NSF grant CCR-9404619.

1 Introduction

Finite state machines which communicate over unbounded channels have been used as an abstract model of computation for reasoning about communication protocols [5, 12] and form the backbone of ISO protocol specification languages Estelle [9] and SDL [20]. Ever since the publication of the Alternating bit protocol [4] (the first ever computer communication protocol) it has been customary to assume, while modeling a protocol, that the communication channels between the processes are free of errors. Possible errors in the communication channels are treated separately, or are completely ignored. In [11] Finkel considered a model of errors, called *completely specified protocols*, in which messages from the front of a queue can be lost. He showed that the *termination* problem is solvable for this class. In [2, 3] Abdulla and Jonsson consider a slightly more general notion of message lossiness: they assume that messages from anywhere in the queue can be lost. They considered the reachability problem [2] and the model-checking problem [3] against specifications in the linear time temporal logic PTL and the branching time temporal logic CTL^* [10]. They show that the reachability problem is decidable and that the model-checking problem for both logics is undecidable. This is in sharp contrast to finite state machines communicating over perfect channels, which are equivalent to turing machines [6]. In [7], Ćeće, Finkel and Iyer consider other sources of errors such as deletion and duplication of messages. The significance of these results lies in the fact that by modeling errors in a protocol, we would be modeling real situations more closely.

While errors are possible in a communication medium, it is generally the case that the manufacturer provided assurances, or guarantees, in the form of a measure of its reliability. Clearly, a system/component with out such guarantees, with a high degree of unreliability is completely useless. Consequently, we believe that it is more realistic to model the measure of guarantee in a protocol. Given a model where the probability of message losses is taken into account, the natural question to ask of a protocol is "Is the probability of something bad happening, in spite of the errors, low?" Alternatively, we could ask: "Does a property ϕ hold of a protocol/system with probability greater than p?" Answers to such questions can conceivably be used in the context of Formal methods and Performance evaluation of protocols (or systems with lossiness).

Technically, we address the *probabilistic reachability* and *probabilistic model-checking* questions in this paper. Given a description $\mathcal{L}$ of a *probabilistic lossy channel systems*, a probability $p \in (0, 1]$ and any arbitrarily small level of tolerance $\nu > 0$, the contributions of this paper are algorithmic solutions for the following problems:

Probabilistic Reachability problem: Is a state γ of the system $\mathcal{L}$ reachable with probability at least p and tolerance ν?

Probabilistic Model-checking problem: Given a Propositional Temporal Logic (PTL) formula ϕ, does the system $\mathcal{L}$ have the property ϕ with probability at least p, and tolerance ν?

The organization of the paper is as follows: in Section 2 we provide the necessary definitions, in Section 3 we state and summarize the results, in Sections 4 and 5 we provide algorithms for probabilistic reachability and probabilistic model-checking, respectively. We conclude, in Section 6, with a comparison of our work with that of others.

2 Definitions

In this section we present the necessary definitions. The model of computation we will use is a probabilistic version of lossy channel systems [2], which consists of a finite control and multiple FIFO channels capable of losing messages – a particular rendition of Communicating Finite State Machines [17] [2].

Let $(m \in)$ M be a finite set of messages and let $(c \in)$ C be a finite set of channels. Let $(w \in)$ $W(C, M)$ be the set of all string vectors over the index set C and strings $(x, y \in)$ M^*. Given a string vector w let $w[c := x]$ denote the new string vector w' such that $w'(c) = x$, and $w'(d) = w(d)$ for $d \neq c$. We will use ϵ to denote the string vector that maps all elements of C to the empty string ε. Finally, we will use $\mid x \mid$, and $\mid w \mid$, to denote the length of the string x, and the sum of the lengths of all of the strings in the vector w, respectively.

Definition 2.1 [PLCS] Fix a set $(\sigma \in)\Sigma$ of atomic propositions. A probabilistic lossy channel system $\mathcal{L}$ is a tuple $(S, s_0, C, M, A, \Delta, P, p_\ell, f)$ where

- $(s \in)$ S is a finite set of *control states*, and $s_0 \in S$ is the *initial control state*,
- C is a finite set of *channels*,
- M is a finite set of *messages*,
- $A = \{c!m, c?m | c \in C, m \in M\}$ is a finite set of actions, where $c!m$ $(c?m)$ denotes an output (input) action of message m on channel c.
- $(\rho \in)$ $\Delta \subseteq S \times A \times S$ is the transition relation.
- $P : \Delta \rightarrow [0, 1]$ is a probability function on transitions,
- p_ℓ, a constant, denotes the probability of losing a message from some channel at any given time, and
- $f : S \rightarrow 2^\Sigma$ is an interpretation function that indicates which atomic propositions hold at a given state. Without loss of generality, we will assume that f is injective.

Given a probabilistic lossy channel system (PLCS) $\mathcal{L}$ we formalize its semantics as a (possibly infinite state) Markov chain. The states of the Markov chain (referred to, henceforth, as global states) are tuples of the form $\langle s, w \rangle \in \Gamma_\mathcal{L} = S \times W(C, M)$, where s is a finite control state and w is the buffer contents. We will write $\gamma \in \Gamma_\mathcal{L}$ for a typical global state, and will drop the subscript $\mathcal{L}$ when the PLCS $\mathcal{L}$ is clear from the context. We will use $\gamma_0 = \langle s_0, \epsilon \rangle$ to denote the initial global state. Finally, we will use $control(\gamma)$ to refer to the control state of γ.

The transitions of the Markov chain, associated with a PLCS $\mathcal{L}$, is a function $\longrightarrow: \Gamma \times \Gamma \rightarrow [0, 1]$ capturing the probability $p = \longrightarrow (\gamma, \gamma')$ with which the system may move from the global state γ

[2] A CFSM consists of a set of finite state machines interacting, asynchronously, with each other over unbounded FIFO buffers.

to the global state γ'. In the following we will write $\gamma \longrightarrow_p \gamma'$ instead of $p = \longrightarrow (\gamma, \gamma')$. A natural condition that $\longrightarrow$ should satisfy is the Markovian condition: $\forall \gamma : (\sum_{\gamma' \in \Gamma_{\mathcal{L}}} \longrightarrow (\gamma, \gamma') = 1)$.

A transition $\rho \in \Delta$ is said to be *enabled* in a global state γ provided

- ρ is an output transition $(s, c!m, s')$ and $\gamma = (s, w)$, or
- ρ is an input transition $(s, c?m, s')$ and $\gamma = (s, w[c := mx])$, i.e., the first message in the channel c is the message, m, which will be removed by the transition Δ.

Let $enabled(\gamma) = \{\rho \in \Delta | \rho \textit{ is enabled in } \gamma\}$.

In assigning probability to a move of the system, from a state γ to a state γ', the probability of loss p_ℓ will be distributed among the (implicit) loss transitions (to be defined) and the probability of non-lossiness $(1 - p_\ell)$ will be distributed among the transitions enabled in a global state γ (in accordance with the relative probability assigned by P to the transition on local state).

We are now ready to define the probabilistic transitions of moves between the global states:

Definition 2.2 $\gamma \longrightarrow_p \gamma'$ provided

- Output out of empty buffers: If $\gamma = \langle s, \epsilon \rangle$ and there exists a transition $\rho = (s, c!m, s') \in \Delta$ then $\gamma' = \langle s', \epsilon[c := m] \rangle$ and probability $p = \frac{P(\rho)}{\sum_{\rho' \in enabled(\gamma)} P(\rho')}$.
- Output: If $\gamma = \langle s, w \rangle$, $w \neq \epsilon$ and there exists a transition $\rho = (s, c!m, s') \in \Delta$ then $\gamma' = \langle s', w[c := w[c]m]$, and the probability $p = \frac{(1-p_\ell) \times P(\rho)}{\sum_{\rho' \in enabled(\gamma)} P(\rho')}$.
- Input or Loss: If $\gamma = \langle s, w[c := mx] \rangle$ and there exists a transition $\rho = (s, c?m, s) \in \Delta$ then $\gamma' = \langle s, w[c := x] \rangle$. The probability p in this case should also include the fact that the first message in the queue could have been lost; consequently, $p = \frac{(1-p_\ell) \times P(\rho)}{\sum_{\rho' \in enabled(\gamma)} P(\rho')} + \frac{p_\ell}{|w|}$.
- Input: If $\gamma = \langle s, w[c := mx] \rangle$, $s \neq s'$ and there exists a transition $\rho = (s, c?m, s') \in \Delta$ then $\gamma' = \langle s', w[c := x] \rangle$ and the probability $p = \frac{(1-p_\ell) \times P(\rho)}{\sum_{\rho' \in enabled(\gamma)} P(\rho')}$.
- Loss: If $\gamma = \langle s, w[c := xmy] \rangle$, and either $x \neq \varepsilon$ or $x = \varepsilon$ and there is no input transition of the form $(s, c?m, s)$ then $\gamma' = \langle s, w[c := xy] \rangle$ and the probability p is $\frac{p_\ell}{|w|}$.
- If none of the above conditions hold then γ' can be any arbitrary global state and $p = 0$.

The first two clauses in the definition given above characterize when an output can take place, and the probability of an output action. Note that the first clause deals with a global state in which there are no messages in the buffer; in this case the probability of loss p_ℓ has no effect on the probability of the transition. The third and the fourth clause deal with input actions. The third clause deals with the removal of a message from the front a buffer where the local state does not change; since the removal of the message could either be due to a loss or due to an input action of the PLCS there are two terms in the calculation of the transition probability. Finally note that when a message is lost from the buffer the finite control remains in the same local state.

A *computation* of a PLCS $\mathcal{L}$ (and its associated Markov chain $\mathcal{M}$) is an infinite sequence of global states of the form $\gamma_0\gamma_1\gamma_2\ldots$ such that there is a sequence of transitions $\gamma_0 \longrightarrow_{p_1} \gamma_1 \longrightarrow_{p_2} \cdots$ where $p_1, p_2, \ldots > 0$. An *execution* of a PLCS (and its Markov chain) is a finite sequence of global states $\gamma_0, \gamma_1, \gamma_2 \ldots, \gamma_k$ such that there is a sequence of transitions $\gamma_0 \longrightarrow_{p_1} \gamma_1 \longrightarrow_{p_2} \ldots \gamma_k$ where $p_1, p_2, \ldots p_k > 0$. We will let π range over computations and α over executions. Furthermore, let $\pi(i)$ and $\alpha(i)$ refer to the i-th element of π and α, respectively.

Given an interpretation function $f : S \to 2^{\Sigma}$ we will extend it to global states $f : \Gamma \to 2^{\Sigma}$ as $f(\langle s, w\rangle) = f(s)$ and to sequences of global states as $f(\pi) = f(\pi(0))f(\pi(1))\ldots$.

Definition 2.3 Given a PLCS $\mathcal{L} = (S, s_0, C, M, A, \Delta, P, p_\ell, f)$ define the Markov chain associates with it as $\mathcal{M} = (\Gamma_{\mathcal{L}}, \longrightarrow, \gamma_0, f)$ where $f : \Gamma_{\mathcal{L}} \to 2^{\Sigma}$ is the interpretation function, and $\Gamma_{\mathcal{L}}$ and $\longrightarrow$ are as defined above.

Given a Markov chain $\mathcal{M} = (\Gamma, \longrightarrow, \gamma_0, f)$ we follow [16, 13, 18] in constructing a sequence space, $\wp(\mathcal{M}) = (\Omega, \mathcal{F}, \mu)$, for assigning probabilities, where

- $\Omega = \Gamma^{\omega}$ is the set of all infinite sequences of states of M starting at γ_0,
- $\mathcal{F}$ is a Borel field generated from the *basic cylindric sets*

$$\mathcal{F}(\gamma_0\gamma_1\ldots\gamma_n) = \{\pi \in \Omega | \pi = \gamma_0\gamma_1\ldots\gamma_n\ldots\}$$

- μ is a probability function defined by

$$\mu(\mathcal{F}(\gamma_0\gamma_1\ldots\gamma_n)) = p_1 \times p_2 \times \ldots p_n$$

  where $\gamma_0 \longrightarrow_{p_1} \gamma_1 \longrightarrow_{p_2} \cdots \gamma_n$.

Propositional Temporal Logic We will now define how *Propositional Temporal Logic (PTL)* formulae are to be interpreted over a Markov chain. We assume that PTL formulae are built from the set of *atomic propositions* (Σ), boolean connectives ($\wedge$ and $\vee$), the unary temporal connective *next* ($\circ$) and the binary temporal connective *until* ($\mathcal{U}$). Let ϕ and ψ range over PTL formulae. For a Markov chain $\mathcal{M}$, with a labeling function f and a computation π of $\mathcal{M}$, the satisfaction relation $\models$ is defined as:

- $\mathcal{M}, \pi, i \models b$, for $b \in \Sigma$ iff $b \in f(s)$ where s is the control state of $\pi(i)$
- $\mathcal{M}, \pi, i \models \phi \wedge \psi$ iff $\mathcal{M}, \pi, i \models \phi$ and $\mathcal{M}, \pi, i \models \psi$
- $\mathcal{M}, \pi, i \models \neg\phi$ iff $\neg(\mathcal{M}, \pi, i \models \phi)$
- $\mathcal{M}, \pi, i \models \circ\phi$ iff $\mathcal{M}, \pi, i+1 \models \phi$
- $\mathcal{M}, \pi, i \models \phi\mathcal{U}\psi$ iff for some $j \geq i$ we have $\mathcal{M}, \pi, j \models \psi$, and for all $i \leq k < j$ it is the case that $\mathcal{M}, \pi, k \models \phi$

We define the other typical operators in the usual way: $\Diamond$ (*eventually*) and $\Box$ (*always*) are defined as: $\Diamond\phi = true\mathcal{U}\phi$ and $\Box\phi = \neg\Diamond\neg\phi$. We say that a computation π of $\mathcal{M}$ *satisfies* ϕ, denoted $\mathcal{M}, \pi \models \phi$, iff $\mathcal{M}, \pi, 0 \models \phi$.

In our discussion on model-checking (later in the paper) we will make use of an automata-theoretic characterization of the computations which satisfy a formula ϕ. To that end, we now define *Büchi automata* on infinite words (in our case infinite strings of propositions) as follows:

Definition 2.4 A *Büchi automaton* is a tuple $A = (\tau, q_0, F)$ where:

- $\tau = (\Sigma, Q, \delta)$ is a table where $(q \in)Q$ is the set of states, Σ is the alphabet and $\delta : Q \times \Sigma \to 2^Q$ is the transition function,
- $F \subseteq Q$ is a set of accepting states.

A run of τ over an infinite word $\eta = \sigma_0\sigma_1 \ldots \in \Sigma^\omega$ is an infinite sequence of states $r = q_0 q_1 \ldots$ such that $q_{i+1} \in \delta(q_i, \sigma_i)$ for $i \geq 0$. Given a run r let $inf(r)$ be the set of states that appear infinitely often in r. A accepts an infinite word η if there exists a run, r, of τ on the word η such that $inf(r) \cap F \neq \emptyset$.

It turns out that for any given PTL formula ϕ it is possible to construct a Büchi automaton A_ϕ such that A_ϕ accepts a word $f(\pi)$ exactly when a computation π in some Markov chain $\mathcal{M}$ (under interpretation function f) satisfies ϕ[3]. [1, 19]

3 Problems of interest and Summary of Results

Given a PLCS $\mathcal{L}$ we say that a state $\gamma \in \Gamma_{\mathcal{L}}$ is reachable with probability p provided the set of computations containing γ has the measure p, i.e.,

$$\mu(\{\alpha\gamma\pi | \alpha \in \Gamma_{\mathcal{L}}^*,\ \pi \in \Gamma_{\mathcal{L}}^\omega\}) \geq p$$

Similarly, we will say that a PTL formula ϕ holds of a PLCS system $\mathcal{L}$ with probability p, written $\mathcal{L} \models_p \phi$, provided the set of computations which satisfy ϕ has the measure p, i.e.,

$$\mu(\{\pi \mid \mathcal{L}, \pi \models \phi\}) \geq p$$

Given these definitions we now summarize the results of this paper:

Probabilistic Reachability Problem:

Given: A PLCS $\mathcal{L}$, a global state γ, a $p \in (0, 1]$ and a tolerance $\nu > 0$.

Question: Is γ reachable with probability at least p and tolerance ν, in $\mathcal{L}$ (Is the measure of the set of computations that visit γ at least $p - \nu$)?

We give an algorithm to decide reachability for $p \in [0, 1]$.

[3] Note that PTL formulae can only express "regular" properties.

Algorithm Prob-Reach
Input: A PLCS $\mathcal{L}$, a global state γ_f, $p \in (0,1]$ and a tolerance ν
Output: Is γ_f reachable from γ_0 with a probability greater than on equal to p and tolerance ν.
var $\gamma_f Prob$, $QProbSum$: real
var Q : queue
begin
    $\gamma_f Prob := 0$
    add $\langle \gamma_0, 1 \rangle$ to Q;
    while Q is not empty **do**
        remove $\langle \alpha, p_\alpha \rangle$ from the head of Q;
        if $lstate(\alpha) = \gamma_f$ **then**
            $\gamma_f Prob := \gamma_f Prob + p_\alpha$;
            if $\gamma_f Prob \geq p$ **then**
                exit(true)
        else
            foreach α' such that $(\alpha, \alpha', p_1) \in \hookrightarrow$ for some $p_1 \in (0,1]$ and
                    Reach($lstate(\alpha'), \gamma_f$) **do**
                add $\langle \alpha', p \times p_1 \rangle$ to Q
            end; (* for *)
        $QProbSum := \sum_{\langle \alpha, p_\alpha \rangle \in Q} p_\alpha$
        if $QProbSum + \gamma_f Prob < p$ **then**
            exit(false);
        if $QProbSum < \nu$ **then**
            exit(true, but with tolerance ν);
    end; (* while *)
end.

Figure 1: Algorithm for probabilistic reachability

Probabilistic Model-checking:
Given: A PLCS $\mathcal{L}$, a labeling function f, a PTL formula ϕ, a $p \in (0,1)$ and a tolerance $\nu > 0$.
Question: Does $\mathcal{L}$ satisfy ϕ with probability at least p and tolerance ν (Is the measure of the set of computations that satisfy ϕ at least $p - \nu$)?
We give an algorithm to show that probabilistic satisfaction is computable for $p \in (0,1]$.

4 Probabilistic Reachability

We have defined the probability of reaching a state γ as the measure of the set of computations that visit γ. But, fortunately, it turns out that it is enough to consider finite sequences in which no state is repeated. Consider two finite computation sequences $\alpha_1 = \gamma_0 \alpha \gamma$ and $\alpha_2 = \gamma_0 \alpha \gamma \alpha' \gamma$, where γ is visited more than once in α_2. Since the basic cylindric set associated with α_1 includes the basic cylindric set associated with α_2, the probability of reaching γ in α_2 (i.e., $\mu(\mathcal{F}(\alpha_2))$) would have been accounted for in the probability of reaching γ in α_1, (i.e., $\mu(\mathcal{F}(\alpha_2))$). So, effectively, we can solve probabilistic reachability problem by computing the probabilities on finite execution sequences that have no loops.

4.1 Trace Automata and Probabilistic Reachability

Let $\mathcal{L} = (S, s_0, C, M, A, \Delta, P, p_l, f)$ be a PLCS. Define the set of non-loop executions ($\aleph(\mathcal{L})$) of a PLCS $\mathcal{L}$ as

$$\aleph(\mathcal{L}) = \{\alpha | \alpha = \gamma_0\gamma_1 \ldots \gamma_n \textit{ is an execution of } \mathcal{L} \textit{ and } \forall\, i, j\, , 0 < i, j \leq n \wedge i \neq j\, \gamma_i \neq \gamma_j\}$$

We define an infinite state automaton, called the *trace automaton*, which reflects the behavior of the PLCS and has state space $\aleph(\mathcal{L})$. For an execution α, let $lstate(\alpha)$ and $laction(\alpha)$ denote the last state and the action of the last transition of α.

Definition 4.1 For a PLCS $\mathcal{L}$ the *trace automaton* of $\mathcal{L}$, denoted by $\mathcal{T}(\mathcal{L})$, is the tuple $(\aleph(\mathcal{L}), \alpha_0, \hookrightarrow)$ where

- $\aleph(\mathcal{L})$ is the set of non-loop executions of $\mathcal{L}$.
- $\alpha_0 = \gamma_0$ is the start state, and
- $\hookrightarrow : \aleph \times \aleph \rightarrow (0, 1]$ is the probabilistic transition function such that $\hookrightarrow (\alpha_1, \alpha_2) = p$ provided $\alpha_2 = \alpha_1\gamma$, $lstate(\alpha_1) \longrightarrow_p \gamma$ and $p > 0$.

We say that a state α of $\mathcal{T}(\mathcal{L})$ is reachable (from the start state) with a probability p if the product of the probabilities of the transitions (defined by $\hookrightarrow$) in the (unique) path from the start state α_0 to α is p; in such a case define $\mu(\alpha) = p$.

Given that the range of $\hookrightarrow$ is $(0, 1]$ it should be clear that the trace automaton of $\mathcal{L}$ has a finite branching tree structure. Reachability of a state α in the trace automaton implies reachability of $lstate(\alpha)$ in $\mathcal{L}$. This follows from the definition of the trace automaton. Different states of the trace automaton with the same *lstate* correspond to the different executions of $\mathcal{L}$ that end in $lstate(\alpha)$. Given the following lemma we can solve probabilistic reachability by constructing a trace automaton.

Lemma 4.2 *The probability of reaching a state γ is exactly the measure of the set of computations that have a prefix $\alpha \in \aleph(\mathcal{L})$ such that $lstate(\alpha) = \gamma$.*

The proof follows from our earlier discussion about basic cylindric sets.

4.2 The Algorithm

The algorithm, given in Figure 1, proceeds by performing a breadth-first search of the space of non-loop executions $\aleph(\mathcal{L})$ starting with the execution sequence $\alpha_0 = \gamma_0$. In each iteration of the loop a sequence α, not considered thus far, is examined. If it turns out that the last state of the sequence is the required state γ_f then $\mu(\alpha)$ is added to a running total of the probability of reaching γ_f. If the last state of α is not the required state γ_f, its children (from $\aleph(\mathcal{L})$) are generated. A sequence α' generated is placed in the queue, to be considered later, provided it is possible to start from $lstate(\alpha')$ and reach the required state γ_f, on a path with non-zero probability. Note that this question is the same as checking whether γ_f is reachable, from a specified state, in the non-probabilistic model; a problem that is known to be decidable by the following theorem:

Theorem 4.3 (Non-probabilistic Reachability [2]) *For a lossy channel system $\mathcal{L}$ and a finite representation of a set of global states $\Gamma' \subseteq \Gamma$, we can compute a finite representation of the set $\{\gamma' |$ there is a path from γ' to γ and $\gamma \in \Gamma'\}$.*

Let $Reach(\gamma, \gamma')$ be the subroutine that decides whether γ' is non-probabilistically reachable from γ.

Clearly, our algorithm constructs longer and longer states from $\aleph(\mathcal{L})$. Let $P_k(\gamma)$ denote the probability of reaching a global state γ of the PLCS $\mathcal{L}$ through non-looping paths consisting of at most k transitions. Then $P_k(\gamma) = \sum_{lstate(\alpha)=\gamma \wedge |\alpha|=k} \mu(\alpha)$. Also let $R_k(\gamma_f)$ denote the set $\{\alpha \in \aleph(\mathcal{L}) \mid |\alpha| = k \wedge Reach(lstate(\alpha), \gamma_f)\}$, the set of sequences which can be extended into execution sequences which reach γ_f. The following lemma provides us the condition under which we can declare that γ_f is not reachable with probability greater than a given p.

Lemma 4.4 *For a global state γ_f of a PLCS and a $p \in (0, 1]$, γ_f cannot be reached with a probability greater than or equal p if and only if there is an integer k such that*

$$\sum_{i=0}^{k} P_i(\gamma_f) + \sum_{\alpha \in R_{k+1}(\gamma_f)} \mu(\alpha) < p$$

.

Proof: The first term of the sum is the probability of reaching γ_f in i steps where $i \leq k$. The second term is the sum of probabilities to depth k of those sequences that need more than k steps to visit γ_f. Clearly γ_f can be reached with probability p iff for all k the sum is greater than or equal to p. ■

If γ_f and p, in the above lemma, are such that $\lim_{k \to \infty} \sum_{i=0}^{k} P_i(\gamma_f) = p$ then the condition in Lemma 4.4 will never be true. To stop the algorithm, in such cases, we use the specified tolerance ν and halt when the probability of reachability is within ν of p, and the probability of the unexplored paths is less than ν. In this case we will report that the formula ϕ holds of $\mathcal{L}$ with in the required tolerance ν. As k increases the set of tuples $\langle \alpha, p_\alpha \rangle$ in the queue will be such that the length of α increases and p_α will decrease. Thus we are assured of termination. Formally, we have

Theorem 4.5 Prob-Reach *always terminates and decides whether the probability of reaching γ_f is greater than or equal to a given p, with tolerance ν.*

5 Model-checking against PTL formulae

In this section we will consider the model-checking problem: given a PLCS $\mathcal{L}$ and a PTL formula ϕ we show that it is possible to compute the probability with which $\mathcal{L}$ satisfies ϕ, with in a given limit of tolerance. Since the tolerance can be made as small as we want our algorithm can be used to compute the probability of satisfiability with arbitrary precision. Formally, we provide an algorithm which can answer the question: "Does $\mathcal{L}$ satisfy ϕ with probability greater than or equal to $p \in (0, 1]$, with in a given tolerance ν?". The technique consists of computing successive better lower bounds to the probability of satisfaction of ϕ by $\mathcal{L}$; this effect is achieved by constructing larger and larger portions of the state space of a lossy channel system and carrying out model-checking, at every step, on a small, and finite, piece of the global Markov chain. There are two questions that need to addressed: (a) which portion of the (possibly infinite) graph will be explored in each step? and (b) how will model-checking be carried out, and the probabilities computed? We address these two questions in the next two subsections.

5.1 k-bounded graphs

In each iteration of the algorithm we propose to carry out model-checking on a portion of the Markov chain where the states are restricted to have no more than k messages in each of the buffers. We will first show that by increasing k we will obtain better and better approximation to the probability p with which property ϕ holds of a system $\mathcal{L}$. To that end, recall, from Section 2, that a PLCS $\mathcal{L} = (S, s_0, C, M, A, \Delta, P, p_\ell, f)$ engenders a Markov chain $\mathcal{M} = (\Gamma, \longrightarrow, \gamma_0, f)$ where Γ is the set of global states of the PLCS, γ_0 is the start state, $\longrightarrow$ is the transition probability matrix and $f : \Gamma \to 2^\Sigma$ is an interpretation function. Furthermore, the markov chain $\mathcal{M}$ allows us to define a sequence space $\wp(\mathcal{M}) = (\Gamma^\omega,\ \mathcal{F},\ \mu)$.

We will now define a family of markov chains $\mathcal{M}_k, k \geq 0$, where $\mathcal{M}_k$ captures the markov chain constructed at the k^{th} iteration of our algorithm. As explained earlier, the state space of $\mathcal{M}_k$ contains only those global states whose buffers are all of size less than or equal to k.

Definition 5.1 Given a PLCS $\mathcal{L} = (S, s_0, C, M, A, \Delta, P, p_\ell, f)$ define a family $\mathcal{M}_k = (\Gamma_k, \longrightarrow^k, \gamma_0, f_k)$ and their corresponding sequence spaces $\wp(\mathcal{M}_k) = ((\Gamma_k)^\omega, \mathcal{F}_k, \mu_k)$ where

- $\Gamma_k = (S \times W_k(C, M)) \cup \{D\}$ where $W_k(C, M)$ denotes the set of all string vectors of $\mathcal{M}$ which have at most k messages in each channel. The dead state, D, captures the rest of the states, and computation, in the global Markov chain $\mathcal{M}$.
- $\longrightarrow^k : \Gamma_k \times \Gamma_k \to [0, 1]$ is the transition probability matrix, defined as:
  - if $\gamma_1 \longrightarrow_p \gamma_2$ and $\gamma_1, \gamma_2 \in (S \times W_k(C, M))$ then $\gamma_1 \longrightarrow_p^k \gamma_2$.
  - if $\gamma_1 \in (S \times W_k(C, M))$ and $\gamma_2 = D$ then $\gamma_1 \longrightarrow_p^k \gamma_2$ where $p = \sum_{\gamma_2 \notin (S \times W_k(C,M)) \wedge \gamma_1 \longrightarrow_{p'} \gamma_2 \wedge p' > 0} p'$
  - $D \longrightarrow_1^k D$.
  - $D \longrightarrow_0^k \gamma$, for all $\gamma \in (S \times W_k(C, M))$.
- $f_k : \Gamma_k \to 2^\Sigma$ is a restriction of $f : \Gamma \to 2^\Sigma$ to the domain $\Gamma_k - \{D\}$ and $f_k(D) = \emptyset$.

Given that for each computation (or execution) η of $\mathcal{M}_k$ there is a computation η' of $\mathcal{M}$ such that all but the D states of η form a prefix of η', we have:

Lemma 5.2 *If $\mathcal{M}$ satisfies a PTL formula ϕ with a probability p, and if $\mathcal{M}_k$ satisfies ϕ with probability p_k then $p \geq p_k$.*

By Lemma 5.2 the satisfaction probability of ϕ by $\mathcal{M}_k$ for any integer k, is a lower bound for the satisfaction probability of ϕ by $\mathcal{M}$. Next we show that the lower bounds form a monotonic sequence and converge to the probability of satisfaction of ϕ by $\mathcal{M}_1$.

Lemma 5.3 *Let $k_1 < k_2$. Let $\mathcal{M}_{k_i}$, $i \in \{1, 2\}$, satisfy ϕ with probability p_i, $i \in \{1, 2\}$, respectively. We then have $p_1 \geq p_2$.*

5.2 Satisfaction probability of an PTL formula by a finite Markov chain

The probability with which an PTL formula holds of a Markov chain $\mathcal{M} = (\Gamma_{\mathcal{L}}, \longrightarrow, \gamma_0, f)$ is the measure of all computations of $\mathcal{M}$ that satisfy the formula.

Definition 5.4 A strongly connected component (SCC) of a Markov chain is *closed* provided every state that is reachable from a state of the SCC is in the SCC.

Definition 5.5 [Product of Markov chain and Büchi automaton] Let $\mathcal{M} = (\Gamma, \longrightarrow, \gamma_0, f)$ be a Markov chain and $A_\phi = (\tau, q_0, F)$ (where $\tau = (\Sigma, Q, \delta)$ is a table and F is the set of accepting states) be a Büchi automaton for a PTL formula ϕ. We define a product automaton $A_{\mathcal{M},\phi} = (\tau_{\mathcal{M},\phi}, (\gamma_0, q_0), G)$ over a one letter alphabet as follows:

- $\tau_{\mathcal{M},\phi} = (\{a\}, \Gamma \times Q, \delta_{\mathcal{M},\phi})$
- $\delta_{\mathcal{M},\phi}((\gamma, q), a) = \{\gamma', q') : \gamma \longrightarrow_p \gamma' \wedge p > 0 \wedge q' \in \delta_{\mathcal{M},\phi}(q, f(\gamma))\}$, and
- $G = \Gamma \times F$ is the set of accepting states.

Theorem 5.6 *The measure of the set of computations of a Markov chain $\mathcal{M}$ that satisfy a PTL formula ϕ is equal to the measure of the projections onto $\mathcal{M}$ of computations of $A_{\mathcal{M},\phi}$ which loop in a closed SCC containing an accepting state.*

Proof: See [15]. ■

5.3 Algorithm

By Theorem 5.6 to measure the probability of satisfaction of a PTL formula by a finite state Markov chain, we only need to identify the accepting computations of $A_{\mathcal{M},\phi}$ that loop in a closed SCC and compute the measure of their projections onto $\mathcal{M}$. Figure 5.3 shows a scheme to do this. We replace the closed SCCs C_i of $A_{\mathcal{M},\phi}$ which have an accepting state, by single nodes m_i, and use a finite prefix that leads to an m_i to represent the set of all computations that have that prefix and loop in C_i. We then take the projections of these prefixes onto $\mathcal{M}$ and compute the measure of their Borel field.

```
Algorithm PTL-sat
Input: A PLCS 𝓛, PTL formula φ, p ∈ (0,1] and a tolerance ν
Output: 'true' if 𝓛 satisfies φ with probability p and tolerance ν and 'false' otherwise.
var: p-sat, p-dead : real;
var: k : integer;
var: states_k : set;
begin
        p-sat:=0; p-dead:=0;
        k:=0;
        do
           if p-sat ≥ p then
              exit(true);
           states_k := {γ | γ is a global state of 𝓜_k};
           if states_k = states_{k-1} then (* PLCS is finite state *)
              exit(false);
           p-sat := Sat-Prob(φ, 𝓜_k);
           p-dead := Dead-State-Prob(φ, 𝓜_k);
           k := k + 1;
        while p-sat + p-dead ≥ p and p-dead ≥ ν;
        if p-sat+p-dead ≥ p then
           exit(true with tolerance ν);
        else
           exit(false);
end.
```

Figure 2:

Subroutine Sat-Prob

Input: A PTL formula ϕ and a finite state Markov chain $\mathcal{M}_k = (\Gamma_k, \longrightarrow^k, \gamma_0, f_k)$.

Output: Probability of satisfaction of ϕ in $\mathcal{M}_k$ (measure of all computations of $\mathcal{M}_k$ that satisfy ϕ).

Step 1: For PTL formula ϕ construct a Büchi automaton $A = (\tau, q_0, F)$ where $\tau = (\Sigma, Q, \delta)$ is a table and F is the set of accepting states.

Step 2: Construct product Büchi automaton
$A_{\mathcal{M}_k,\phi} = (\tau_{\mathcal{M}_k}, (\gamma_0, q_0), G)$ where
$\tau_{\mathcal{M}_k,\phi} = (\{a\}, \Gamma_k \times Q, \delta_{\mathcal{M}_k})$,
$\delta_{\mathcal{M}_k,\phi}((\gamma, q), a) = \{(\gamma', q') : \gamma \longrightarrow_p \gamma' \wedge p > 0 \wedge q' \in \delta\mathcal{M}_k, \phi(q, f_k(\gamma))\}$, and
$G = \Gamma_k \times F$.

Step 3: Compute strongly connected components of $A_{\mathcal{M}_k,\phi}$.

Step 4: Collect all strongly-connected-components $C_1, C_2, \ldots, C_n$ of $A_{\mathcal{M}_k,\phi}$ such that

1. each C_i passes contains at least one state of $\Gamma_k \times F$ and
2. the projection of C_i onto $\mathcal{M}_k$ is a closed SCC of $\mathcal{M}_k$.

Step 5: Replace each closed strongly connected component C_i from Step 4 by a new node m_i to get a new graph $A'_{\mathcal{M}_k,\phi}$. Mark each m_i as "closed".

Step 6:

prob := 0;
if $\exists i, 1 \leq i \leq n : (\gamma_0, q_0) \in C_i$ **then**
  prob := 1
else
  foreach $i \in \{1 \ldots n\}$ **do**
    satexecs := $\{\alpha | \alpha$ is the projection of β onto $\mathcal{M}$, β is a non-loop execution of $A_{\mathcal{M}_k,\phi} \wedge last(\beta) = m_i\}$;
    prob := *prob* + $\sum_{\alpha \in satexecs} \mu(\alpha)$
  end; (* foreach *)
end; (* if *)
return *prob*

Figure 3:

Subroutine Dead-State-Prob
Input: A finite PLCS $\mathcal{M}_k$ with channel lengths at most k and a PTL formula ϕ.
Output: Measure of computations that do not satisfy ϕ and go to D.
begin
    $nosatexecs = \{\alpha | \alpha \in \aleph(\mathcal{M}_k),\ last(\alpha) = D \wedge \alpha(D)^\omega$ does not satisfy $\phi\}$;
    return $\sum_{\alpha \in nosatexecs} \mu(\alpha)$
end.

Figure 4:

To check the satisfaction of a PTL formula by a PLCS, the question to be answered is, for a given PLCS $\mathcal{L}$, a PTL formula ϕ, $p \in [0,1)$ and a tolerance (accuracy) $\nu > 0$, "Does $\mathcal{L} \models_p \phi$?". The algorithm (shown in Figure 2) inputs the representation of the PLCS and computes successive approximations by computing the satisfaction probability of ϕ in $\mathcal{M}_k$. It uses the variable *p-sat* to record the measure of the computations that satisfy ϕ, and it uses the variable *p-dead* to record the measure of computations that do not satisfy ϕ, in each iteration of the loop. If for some k, the set of global states of $\mathcal{M}_k$ is the same as the set of global states of $\mathcal{M}_{k-1}$ then the PLCS has no global states that have more than k messages in a channel.

Computations of a markov chain $\mathcal{M}_k$ can be divided into the following four sets:

1. $C_{sat,\not{D}}$: Computations that satisfy ϕ and do not end in D (measure $p_{sat,\not{D}}$)
2. $C_{sat,D}$: Computations that satisfy ϕ and end in D (measure $p_{sat,D}$)
3. C_{nosat}: Computations that do not satisfy ϕ and do not end in D (measure p_{nosat})
4. C_{dead}: Computations that do not satisfy ϕ and end in D (measure *p-dead*)

Clearly *p-sat* $= p_{sat,\not{D}} + p_{sat,D}$. The invariant for the algorithm is *p-sat* + *p-dead* + $p_{nosat} = 1$. Note that by Lemma 5.2 *p-sat* is non-decreasing. We show below that *p-dead* is decreasing and therefore the stopping condition for the algorithm *p-sat* + *p-dead* $<\ p$ is reached, if ϕ does not hold for $\mathcal{L}$. When the loop index k is incremented to $k+1$ *p-sat* increases when computations in C_{dead} move to $C_{sat,\not{D}}$ or $C_{sat,D}$. When k is incremented p-dead necessarily decreases. This is because, at least one of the following has to happen:

1. Computations in C_{dead} move to C_{nosat} or $C_{sat,\not{D}}$. Here *p-dead* decreases.
2. Computations in C_{dead} remain in C_{dead}. Assuming the number of global states increases from k to $k+1$ (otherwise the algorithm halts), there is at least one such computation of $\mathcal{L}_{k+1}$ in which the state before D in $\mathcal{L}_{k+1}$ is a state that is not in $\mathcal{L}_k$. Let this state be κ and the computation be ρ. κ can have loss transitions and therefore, the probability on its transition to D is less than 1. Since ρ is the computation in C_{dead} of $\mathcal{L}_{k+1}$ that replaces a computation in C_{dead} of $\mathcal{L}_k$, *p-dead* decreases.

We formally have:

Lemma 5.7 *p-dead necessarily decreases in each loop of the algorithm, and tends to zero as the number of iterations k tends to ∞.*

This leads us to the correctness of stopping conditions:

Lemma 5.8 *For a given PLCS $\mathcal{L}$, $p \in [0,1]$, a tolerance $\nu > 0$ and a PTL formula ϕ*

(a) *if $\mathcal{L} \models_p \phi$, then there exists a k such that the value of p-sat in the k-th loop of the algorithm is greater than or equal to $p - \nu$*

(b) *if $\mathcal{L} \not\models_p \phi$ then there exists a k such that the value of p-sat+p-dead in the k-th loop of the algorithm is less than p.*

Proof: (a) follows from Lemma 5.3.
(b) $\mathcal{L} \not\models_p \phi$ implies $p > p_{\mathcal{L}}$, where $p_{\mathcal{L}}$ is the measure of the computations of $\mathcal{L}$ that satisfy ϕ. As $k \longrightarrow \infty$, *p-sat*$\longrightarrow p_{\mathcal{L}}$ and *p-dead*$\longrightarrow 0$ (*p-sat* approaches $p_{\mathcal{L}}$ asymptotically and *p-dead* approaches 0 asymptotically). If $p_{\mathcal{L}} + \nu' = p$ then for some k *p-dead*$< \nu'$ so that *p-sat*+*p-dead*$< p$ (as *p-sat*$\leq p_{\mathcal{L}}$).

We finish up with a statement about the correctness of the algorithm.

Theorem 5.9 PTL-sat *terminates for all input, and checks whether ϕ holds for a PLCS $\mathcal{L}$ with probability greater than or equal to p with tolerance ν.*

6 Discussion

We have shown that probabilistic reachability and probabilistic model-checking problems are decidable for lossy channel systems, by providing algorithms for them. By modelling probability of errors in the specification, we believe, we have made protocol specifications more realistic. An open problem is to come up with more efficient algorithms/implementations (in the sense of algorithms used in SPIN [14, 8]).

The notion of probabilistic model-checking we use differs from that of [18]. Vardi, and all of the relevant papers on probabilistic model-checking, ask the question: "Does the set of computations that satisfy a formula ϕ have a measure exactly equal to 1?" Furthermore, they typically use zero-one laws of probability to reduce the probabilistic model-checking problems to model-checking problems on the traditional non-probabilistic models. Given that the model-checking problem for Lossy Channel Systems (without probabilities) is undecidable, it is highly unlikely that the probabilistic model-checking, in the traditional sense, would be decidable for PLCS.

References

[1] A. Sistla, M. Vardi, and P. Wolper. The complementation problem for Büchi automata with applications to temporal logic. *Theoretical Computer Science*, 49:217–237, 1987.

[2] P. Abdulla and B. Jonsson. Verifying programs with unreliable channels. In *Proceedings of the IEEE Symposium on Logic in Computer Science*, 1993.

[3] P. Abdulla and B. Jonsson. Undecidable verification problems for programs with unreliable channels. In *Proceedings of the Annual International Colloquium on Automata, Languages and Programming*, 1994.

[4] Bartlett, Scantlebury, and Wilkinson. A note on reliable full-duplex transmission over half duplex lines. *Comm. ACM*, 12(5):260–265, 1969.

[5] G. Bochmann. Finite state description of communication protocols. *Comput. Networks*, 2:362–372, 1978.

[6] D. Brand and P. Zafiropulo. On communicating finite state machines. *Journal of the Association of Computing machinery*, 30(2):323–342, 1983.

[7] G. Ćeće, A. Finkel, and S. P. Iyer. Unreliable channels are easier to verify than perfect channels. *Information and Computation*, 124(1):20–31, 1996.

[8] C. Courcoubetis, M. Vardi, P. Wolper, and M. Yannakakis. Memory efficient algorithms for verification of temporal properties. In *Computer Aided Verification*, 1990.

[9] M. Diaz, J.P. Ansart, P. Azema, and V. Chari. *The Formal Description Technique Estelle.* North-Holland, Amsterdam, 1989.

[10] E. A. Emerson and J. Y. Halpern. "sometimes" and "not never" revisited: On branching time versus linear time temporal logic. *JACM*, 33(1):151–178, 1986.

[11] A. Finkel. Decidability of the termination problem form completely specified protocols. *Dist. Comput.*, 7:129–135, 1994.

[12] M. G. Gouda, E. M. Gurari, T.H. Lai, and L.E. Rosier. On deadlock detection in systems of communicating finite state machines. *Comput. Artif. Intell.*, 6(3):209–228, 1987.

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

[14] G. Holzmann. *Design and Validation of Computer Protocols.* Prentice Hall, Englewood Cliffs, New Jersey, 1991.

[15] P. Iyer and M. Narasimha. "Almost always" and "definitely sometime" are not enough: Probabilistic quantifiers and probabilistic model checking. Technical Report TR-96-16, NC State University, July 1996. Submitted for publication.

[16] J. G. Kemeny, J. L. Snell, and A. W. Knapp. *Denumerable Markov Chains.* Van Nostrand, New Jersey, 1966.

[17] W. Peng and S. Purushothaman. Data flow analysis of communicating finite state machines. *ACM Transactions on Programming Languages and Systems*, 13(3):399–442, July 1991.

[18] M. Y. Vardi. Automatic verification of probabilistic concurrent finite-state programs. In *IEEE Symposium on Foundations of Computer Science*, pages 327–338, 1985.

[19] P. Wolper, M. Vardi, and A. Sistla. Reasoning about infinite computation paths. In *IEEE Symposium on Foundations of Computer Science (FOCS)*, 1983.

[20] CCITT Recommendation Z.100. Specification and description language sdl. Blue Book Vols X.1-X.5, ITU General Secreterait, Geneva, 1989.