Probabilistic Lossy Channel Systems *

Purush Iyer and Murali Narasimha

Dept of Computer Science
North Carolina State University
Raleigh, NC 27695-8206

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* (against linear-time PTL requirements without the next state operator) problem.

1 Introduction

Finite state machines which communicate over unbounded channels have been used as an abstract model of computation for reasoning about communication protocols [4, 14] and form the backbone of ISO protocol specification languages Estelle [8] and SDL [17]. Ever since the publication of the Alternating bit protocol [3] (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 [10] 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 [1, 2] 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 [1] and the model-checking problem [2] against specifications in the linear time temporal logic PTL and the branching time temporal logic CTL^* [9]. 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 [5]. In [6], Cécé, Finkel and Iyer consider other sources of errors such as deletion and duplication

* Research supported in part by NSF grant CCR-9404619.

of messages. The significance of these results is 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, and 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, a 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 (without the next state operator), does the system $\mathcal{L}$ have the property ϕ with probability at least p, and tolerance ν?

Both algorithms involve computing a sequence of approximations to the probability with which a property holds. In the case of the model-checking algorithm, we prove a monotonicity lemma, a consequence of the fact that we consider PTL specifications in *positive normal form*, which provides for the successive-approximation strategy to work.

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. In Section 6, we conclude.

2 Definitions

The model of computation we will use is a probabilistic version of lossy channel systems [1], which consists of a finite control and multiple FIFO channels capable of losing messages – a particular rendition of Communicating Finite State Machines [14] [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

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

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. 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. Finally, if Σ is a set of propositions define $\mathcal{B}(\Sigma)$ to be the set of boolean expressions over Σ and $\chi : 2^{\Sigma} \rightarrow \mathcal{B}(\Sigma)$ as the characteristic function with the definition $\chi(X) = \bigwedge_{p \in X} p \bigwedge_{p \notin X} \neg p$.

Definition 1 PLCS. Fix a set $(\sigma \in)\Sigma$ of atomic propositions. A probabilistic lossy channel system $\mathcal{L}$ is a tuple $(S, s_0, C, M, Act, \Delta, P, p_\ell, f)$ 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*,
- $Act = \{c!m, c?m \mid c \in C, m \in M\}$ is a finite set of actions, where $c!m$ ($c?m$) denotes an output (input) action of message m on channel c.
- $(\rho \in)$ $\Delta \subseteq S \times Act \times S$ 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,
- $f : S \rightarrow 2^{\Sigma}$ is a labeling function that indicates which atomic propositions hold at a given state.

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.

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 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 ρ.

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

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

$\gamma \longrightarrow_p \gamma'$ 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 $\gamma' = \langle s', w[c := w[c]m] \rangle$, 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 $p = 0$.

Fig. 1. Definition of $\longrightarrow_p$

among the transitions enabled in a global state γ (in accordance with the relative probability assigned by P to the transition on local state).

The definition of $\longrightarrow_p$ is shown in Figure 1. 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 of a buffer where the local state does not change; since the removal of the message could be either 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.

Definition 2. Given a PLCS $\mathcal{L} = (S, s_0, C, M, A, \Delta, P, p_\ell, f)$ define the Markov chain associated with it as $\mathcal{M} = (\Gamma_\mathcal{L}, \longrightarrow, \gamma_0, \mathcal{I})$ where $\mathcal{I}(\langle s, w \rangle) = \chi(f(s))$ is the *interpretation function*, and $\Gamma_\mathcal{L}$ and $\longrightarrow$ are as defined earlier.

A *computation* of a PLCS $\mathcal{L}$ (and its associated Markov chain $\mathcal{M}$) is an infinite sequence of global states of the form $\gamma_0\gamma_1\gamma_2\cdots$ 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 $\gamma_0, \gamma_1, \gamma_2 \cdots, \gamma_k$ such that there is a sequence of transitions $\gamma_0 \longrightarrow_{p_1} \gamma_1 \longrightarrow_{p_2} \cdots \gamma_k$ where $p_1, p_2, \cdots p_k > 0$. 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. We will also use $\mathcal{M}$ to refer to a PLCS $\mathcal{L}$, when it needs to be viewed as a Markov chain. Finally, extend the interpretation function on global states $\mathcal{I} : \Gamma \rightarrow \mathcal{B}(\Sigma)$ to sequences of global states as $\mathcal{I}(\pi) = \mathcal{I}(\pi(0))\mathcal{I}(\pi(1))\ldots$.

Definition 3 [12, 11, 15]. Given a Markov chain $\mathcal{M} = (\Gamma, \longrightarrow, \gamma_0, \mathcal{I})$ define the sequence space as $\wp(\mathcal{M}) = (\Omega,\ \mathcal{F},\ \mu)$, 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 $\gamma_0 \longrightarrow_{p_1} \gamma_1 \longrightarrow_{p_2} \ldots \gamma_n$.

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* ($\sigma \in \Sigma$), boolean operations ($\neg$, $\wedge$ and $\vee$), the unary temporal connective *next* ($\circ$) and the binary temporal connectives *until* ($\mathcal{U}$) and *while* ($\mathcal{V}$). Let ϕ and ψ range over PTL formulae. The syntax of PTL formulae is given by the following grammar:

$$\phi := \sigma \mid \neg\sigma \mid \circ\phi \mid \phi \wedge \psi \mid \phi \vee \psi \mid \phi\mathcal{U}\psi \mid \phi\mathcal{V}\psi$$

Definition 4. For a Markov chain $\mathcal{M}$, with interpretation function $\mathcal{I}$ and a computation π of $\mathcal{M}$, the satisfaction relation $\models$ is defined as:

- $\mathcal{M}, \pi, i \models \sigma$ iff $\mathcal{I}(\pi(i)) \Rightarrow \sigma$.
- $\mathcal{M}, \pi, i \models \neg\sigma$ iff $\mathcal{I}(\pi(i)) \Rightarrow \neg\sigma$.
- $\mathcal{M}, \pi, i \models \circ\phi$ iff $\mathcal{M}, \pi, i+1 \models \phi$.
- $\mathcal{M}, \pi, i \models \phi \wedge \psi$ iff $\mathcal{M}, \pi, i \models \phi$ and $\mathcal{M}, \pi, i \models \psi$.
- $\mathcal{M}, \pi, i \models \phi \vee \psi$ iff $\mathcal{M}, \pi, i \models \phi$ or $\mathcal{M}, \pi, i \models \psi$.
- $\mathcal{M}, \pi, i \models \phi\mathcal{U}\psi$ 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 $\mathcal{M}, \pi, k \models \phi$.
- $\mathcal{M}, \pi, i \models \phi\mathcal{V}\psi$ iff for all $j \geq i$ we have that if $\forall k (i \leq k < j) : \mathcal{M}, \pi, k \not\models \phi$ then $\mathcal{M}, \pi, j \models \psi$.

Define XPTL to be the set of PTL formulae which do not contain the next state operator $\circ$. Furthermore, 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 $\Box\phi = \phi\mathcal{V}false$. We say that a computation π of $\mathcal{M}$ *satisfies* ϕ, denoted $\mathcal{M}, \pi \models \phi$, iff $\mathcal{M}, \pi, 0 \models \phi$.

Note that the syntax of PTL is such that only formulae that are in *positive normal form* are allowed, i.e., negation can be applied only to atomic propositions. The conventional definition of PTL syntax gives formulae in which negation is not restricted to be an innermost operation. A PTL formula in the conventional syntax can be converted to positive normal form by using boolean identities to drive the negations to the innermost nesting level (and hence the two representations of the formula are equivalent). Given a PTL formula ϕ written in the conventional syntax, we will use $pn(\phi)$ to denote its positive normal form.

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 measure at least 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 XPTL 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 measure at least 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 $p \in [0, 1]$.
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_f as the measure of the set of computations that visit γ_f. Effectively, we can solve probabilistic reachability problem by computing the collective measure of finite execution sequences in which γ appears exactly once as the last state of the sequence. To facilitate the computation of probability of reaching a state γ_f in a PLCS $\mathcal{L}$ we define an execution tree $\mathcal{T}(\mathcal{L}, \gamma_f)$ which captures all execution sequences of $\mathcal{L}$ that end in γ_f.

The algorithm to check if a state γ_f of $\mathcal{L}$ is reachable with probability p_f involves creating the execution tree of $\mathcal{L}$ in a breadth-first fashion. The nodes of this tree are pairs of the form (γ, p). The root node of the tree is a pair $(\gamma_0, 1)$. Consider a path in the tree from $(\gamma_0, 1)$ to (γ, p). This tree path corresponds to a path from γ_0 to γ in $\mathcal{L}$ and p denotes the probability of reaching γ on it. In every step of the tree construction, we look for longer paths leading to γ_f and stop when we accumulate a probability of at least p_f. To formally define these notions we need the notion of non-probabilistic reachability.

Theorem 5 Non-probabilistic Reachability [1]. *For a probabilistic lossy channel system $\mathcal{L}$ and a finite representation of a set of global states $\Gamma' \subseteq \Gamma$, we can compute a recognizable representation of the set $\{\gamma' |$ there is a path, of non-zero probability, from γ' to γ and $\gamma \in \Gamma'\}$.*

Note that by this previous result we can only determine whether a state is reachable, not what the probability of its reachability is. Let $Reach(\gamma, \gamma')$ be the subroutine that decides (non-probabilistically) whether γ' is reachable from γ. We are now ready for a formalization of the idea behind probabilistic reachability.

Definition 6 Execution tree. Given a PLCS $\mathcal{L}$ and a global state γ_f of $\mathcal{L}$ we define the execution tree $\mathcal{T}(\mathcal{L}, \gamma_f)$ as $(S_\mathcal{T}, \nabla, s^0_\mathcal{T})$ where:

- $S_\mathcal{T}$ is the set of nodes of the tree,
- $\nabla : S_\mathcal{T} \longrightarrow \Gamma \times (0, 1]$ is a labeling function,
- $s^0_\mathcal{T}$ is the root node with label $(\gamma_0, 1)$, and
- Let $s \in S_\mathcal{T}$ and $\nabla(s) = (\gamma, p)$. s is a leaf node provided $\gamma = \gamma_f$ or if $\neg Reach(\gamma, \gamma_f)$. Otherwise the children of s are $s_1, s_2 \ldots s_n$ with respective labels $(\gamma_1, p_1), (\gamma_2, p_2), \ldots (\gamma_n, p_n)$ such that $\forall i(1 \leq i \leq n)\gamma \longrightarrow_{p_i/p} \gamma_i$.

Let $P_k(\gamma_f)$ denote the probability of reaching a global state γ_f of the PLCS $\mathcal{L}$ through paths consisting of exactly k transitions which do not repeat γ_f. Let $depth(s)$ denote the depth of s in the tree $\mathcal{T}(\mathcal{L}, \gamma_f)$. Then

$$P_k(\gamma_f) = \sum_{\nabla(s)=(\gamma_f,p) \wedge depth(s)=k} p$$

. We now have the following obvious property of the execution tree.

Lemma 7. *The measure of all computations that visit γ_f is exactly $\sum_{i=0}^{\infty} P_i(\gamma_f)$.*

We will now show that the construction of the execution tree can be stopped at a finite depth. To that end, define

$$R_k(\gamma_f) = \{c \in S_\mathcal{T} | \nabla(c) = (\gamma, p) \wedge depth(c) = k \wedge Reach(\gamma, \gamma_f)\}$$

which characterizes those nodes at level k from which it is possible to reach γ_f. The following lemma provides us with the condition under which we can declare that γ_f is not reachable with probability greater than or equal to a given p.

Lemma 8. *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*

$$\sum_{i=0}^{k} P_i(\gamma_f) + \sum_{c \in R_k(\gamma_f) \wedge \nabla(c)=(\gamma,p_\gamma)} p_\gamma < p.$$

Proof. The first term of the sum is the probability of reaching γ_f in i steps where $i \leq k$. The second term covers the sum of probabilities of those sequences that need more than k steps to visit γ_f. If we have a k that satisfies the condition, we have that the sum of the measure of computations that visit γ_f in the first k steps and the measure of computations that visit γ_f after k steps is less than p. Then, by definition of reachability, γ_f is not reachable with probability p. For the other direction of the proof, we prove the contra-positive statement i.e., if for all k:

$$\sum_{i=0}^{k} P_i(\gamma_f) + \sum_{c \in R_{k+1}(\gamma_f) \wedge \nabla(c)=(\gamma,p_\gamma)} p_\gamma \geq p,$$

then γ_f is reachable with probability p. For any positive integer k, the left side of the above inequality is greater than or equal to the measure of all the computations that visit γ_f. As k approaches ∞, the second summand approaches 0, and the left side approaches the measure of all computations that visit γ_f (by Lemma 7) and we have the result.

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 tree could be an infinite tree. 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, i.e., $\sum_{i=0}^{k} P_i(\gamma_f) \geq p - \nu$ and $\sum_{c \in R_{k+1}(\gamma_f) \wedge \nabla(c)=(\gamma,p_\gamma)} p_\gamma < \nu$ and yet their sum is at least p. In this case we will report that the formula ϕ holds of $\mathcal{L}$ with in the required tolerance ν. As k increases the nodes of the tree (γ, p) will have p decreasing. Thus we are assured of termination. Formally, we have

Theorem 9. *There exists an algorithm that 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 XPTL 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. The main technique consists of computing successively 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 finite piece of the global Markov chain. But in doing so, recall the following important property of Markov chains:

Let S be a set of infinite sequences of a Markov chain. The measure of S is the same as the measure of $S' \subseteq S$ where S' does not contain any sequences with an infinite suffix of *transient* states.

Consequently, we are interested in computing the probability of those sequences which satisfy a property ϕ and which has an infinite tail in one of the closed connected components of the Markov chain $\mathcal{M}$ corresponding to $\mathcal{L}$. We will now characterize the closed connected components of $\mathcal{M}$.

Definition 10. A set of states $\Gamma' \subseteq \Gamma$ is a closed connected components provided (a) for every state $\gamma, \gamma' \in \Gamma'$ there is a path from γ to γ' and from γ' to γ, and (b) for every state $\gamma \in \Gamma'$ and $\gamma' \in \Gamma - \Gamma'$ there is no path from γ to γ'.

For a closed connected component Γ' define $proj(\Gamma') = \{(s, \epsilon) | (s, w) \in \Gamma'\}$. For every closed connected component Γ' of $\mathcal{M}$, arising from a lossy channel system $\mathcal{L}$, it is easy to show that $proj(\Gamma')$ uniquely represents Γ'. This is due to the fact that Γ' is a downward closed set[3] and that all states in Γ' are reachable from each other. We, thus, only have a finite number of closed SCCs for the Markov chain $\mathcal{M}$. Given a set of global states $\Gamma' \subseteq S \times \{\epsilon\}$, it is easy to check whether Γ' represents a closed SCC (i.e., is a projection of a closed SCC): check whether the states in Γ' are reachable from each other, that no state with a control component not represented in Γ' is reachable from some state in Γ', and that the states in Γ' are reachable from the start state γ_0 (note all of this can be carried out with the aid of the reachability algorithm for lossy channel systems [1]). Finally, it is also easy to check whether an arbitrary state γ belongs to some closed SCC of $\mathcal{M}$. Formally, we have

Lemma 11. *Given a lossy channel system $\mathcal{L}$ and its Markov chain $\mathcal{M}$ it is decidable whether*

- *A state γ is a member of a closed SCC of $\mathcal{M}$.*
- *A set of states $\Gamma' \subseteq S \times \{\epsilon\}$ represents a closed strongly connected component of $\mathcal{M}$.*

When building a representation of the Markov chain for model-checking we will use a representation of the closed connected component, defined as follows:

Definition 12. Let $\Gamma' \subseteq \Gamma$ be a set of states that form a closed SCC. Define $\hat{\Gamma}' = proj(\Gamma')$ and $rep(\hat{\Gamma}')$ as the graph defined as follows:

- The states of $rep(\hat{\Gamma}')$ are $\hat{\Gamma}'$,
- The edges of $rep(\hat{\Gamma}')$ are are follows: for every $\langle s, \epsilon \rangle$ and $\langle s', \epsilon \rangle$ in $\hat{\Gamma}'$ add an edge between them if (a) there is a send edge between s to s' in the finite control of $\mathcal{L}$, or (b) if there is a receive edge $\xrightarrow{c?m}$ between s and s' in $\mathcal{L}$ and $\langle s, \epsilon[c := m] \rangle$ is reachable and is in the closed SCC Γ' (note: it is enough to check whether $\langle s, \epsilon[c := m] \rangle$ is in some closed SCC, as closed SCCs have mutually disjoint sets of states).

[3] With respect to the sub-word ordering $\preceq \subseteq \Sigma^* \times \Sigma^*$ defined as $a_1 \ldots a_n \preceq x_0 a_1 x_1 a_2 \ldots x_{n-1} a_n x_n$, lifted to vectors of words.

- The probability adorning each of the edges of $rep(\hat{\Gamma}')$ is inversely proportional to the out-degree of the source of that edge, and
- The interpretation of each node in this graph is the interpretation of the same node in the Markov chain $\mathcal{M}$.

In the following we will assume that closed SCCs $\Gamma_1', \ldots, \Gamma_n'$ are represented by the graphs $rep(\hat{\Gamma_i}'), 1 \leq i \leq n$, as defined above.

There are two questions that need to be addressed: (a) how will the the (possibly infinite) graph corresponding to the Markov chain be explored? and (b) how will model-checking be carried out, and the probabilities computed? The answer to the second question follows from the following result, where the authors show that by using a Büchi automaton characterization of PTL[16] one can identify those closed SCCs of a finite state systems all of whose sequences satisfy ϕ.

Theorem 13 [7]. *There exists an algorithm* **PTL-sat** *to compute the probability of the set of sequences of a finite markov chain which satisfy a PTL formula ϕ.*

5.1 k-bounded graphs

In the k^{th} iteration of the algorithm we propose to construct a Markov chain which is restricted to contain states whose buffer size is limited to k messages, and which contains representatives of reachable closed SCCs. We will first show that by increasing k we will obtain a 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 $\mathcal{M} = (\Gamma, \longrightarrow, \gamma_0, \mathcal{I})$ where Γ is the set of global states of the PLCS, γ_0 is the start state, $\longrightarrow$ is the transition probability matrix and $\mathcal{I} : \Gamma \rightarrow \mathcal{B}(\Sigma)$ is the 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.

Definition 14. Given a PLCS $\mathcal{L} = (S, s_0, C, M, A, \Delta, P, p_\ell, f)$ define a family

$$\mathcal{M}_k = (\Gamma_k, \longrightarrow^k, \gamma_0, \mathcal{I}_k)$$

and their corresponding sequence space $\wp(\mathcal{M}_k) = (\Gamma_k^\omega, \mathcal{F}_k, \mu_k)$ where

- Let $\Gamma_k = \Gamma_k^1 \cup \Gamma_k^2 \cup \{D\}$, where $\Gamma_k^1 \subseteq (S \times W_k(C, M))$ contains states that are reachable from γ_0, through a path containing states in $S \times W_k(C, M)$, but does not contain any states that is in some closed SCC. Γ_2 is the union of the representatives of those closed components which have at least one state in $S \times W_k(C, M)$ reachable by a path of $S \times W_k(C, M)$ states.
- $\longrightarrow^k$: $\Gamma_k \times \Gamma_k \rightarrow [0, 1]$ is the transition probability matrix, defined as:
  - if γ_1, γ_2 are members of some closed SCC $\hat{\Gamma_i}'$ and there is edge with probability p between γ_1 and γ_2 in the graph $rep(\hat{\Gamma_i}')$ then $\gamma_1 \longrightarrow_p \gamma_2$.

  - if $\gamma_1, \gamma_2 \in \Gamma_k - \{D\}$, neither is a member of any closed SCC and there is an edge in $\mathcal{M}$ with probability p between γ_1 and γ_2 then $\gamma_1 \longrightarrow_p \gamma_2$.
  - if $\gamma_1 \in (S \times W_k(C, M))$ and $\gamma_2 = D$ then $\gamma_1 \longrightarrow_p^k \gamma_2$ where

$$p = \sum\{p' | \gamma_2 \notin (S \times W_k(C, M)) \wedge \gamma_1 \longrightarrow_{p'} \gamma_2 \wedge p' > 0\}$$

  - $D \longrightarrow_1^k D$.
  - $D \longrightarrow_0^k \gamma$, for all $\gamma \in (S \times W_k(C, M))$.
- $\mathcal{I}_k : \Gamma_k \rightarrow \mathcal{B}(\Sigma)$ agrees with $\mathcal{I} : \Gamma \rightarrow \mathcal{B}(\Sigma)$ on elements of $\Gamma_k - \{D\}$ and $\mathcal{I}_k(D) = \mathtt{true}$.

Note that for every proposition σ, neither σ nor its negation is true in the dead state D. However, our definition of satisfaction (from Sec 2) carries over easily to these markov chains. The implication of using the notion of dead state is that there are sequences involving the dead state which neither satisfy ϕ, nor its negation $pn(\neg\phi)$. Note, however, that for any computation sequence which does not involve the dead state, we have the following (which is established by an easy induction on the structure of the formula):

Lemma 15. *For all $k \geq 0$, for all paths $\pi \in (\Gamma_k - \{D\})^\omega$, for all PTL formulae ϕ in positive normal form and for all $i \geq 0$: either $\mathcal{M}_k, \pi, i \models \phi$ or $\mathcal{M}_k, \pi, i \models pn(\neg\phi)$*

The paths in the family $\mathcal{M}_k, k \geq 0$, and the complete markov chain $\mathcal{M}$ have a relation to each other. To formalize it we define a notion similar to stuttering equivalence [13]. Define a relation $\preceq$ on sequences as follows:

Definition 16. Define *control* : $\cup_{k \geq 0}\Gamma_k \rightarrow S \cup \{D\}$ as $control(\langle s, w\rangle) = s$ and $control(D) = D$. Furthermore, define *blocks* : $(S \cup \{D\})^\omega \rightarrow (S \cup \{D\})^\omega \cup (S \cup \{D\})^*$ such that the result of $blocks(\pi)$ is the computation obtained by removing all repeating states in π. Note that $blocks(\pi)$ could be finite. Given $\pi, \pi' \in \Gamma^\omega \cup \Gamma^* D^\omega$, define $\pi \preceq \pi'$ provided

- If $blocks(control(\pi))$ is a finite sequence then there is a there is a prefix of $\pi'' = blocks(control(\pi'))$ such that the $blocks(control(\pi)) = blocks(control(\pi''))$, or $blocks(control(\pi)) = blocks(control(\pi''))D$
- If $blocks(control(\pi))$ and $blocks(control(\pi'))$ are both infinite sequences then $blocks((\pi)) = blocks(control(\pi'))$.

One of the nice properties of the relation $\preceq$ is that it preserves XPTL formulae, as has been shown in [13]. Formally,

Theorem 17 [13]. *Let $\pi \preceq \pi'$. For every XPTL formula ϕ if $\pi \models \phi$ then $\pi' \models \phi$.*

Given two markov chains $\mathcal{M}_1$ and $\mathcal{M}_2$ among the sequence of markov chains built define $\mathcal{M}_1 \trianglelefteq \mathcal{M}_2$ provided for every computation π in $\mathcal{M}_1$ there is a sequence in π' in $\mathcal{M}_2$ such that $\pi \preceq \pi'$. Furthermore, it is also easy to see that for all $i \geq 0 \mathcal{M}_i \trianglelefteq \mathcal{M}$. We are now ready for the monotonicity lemma, which forms the basis for our algorithm:

```
Algorithm PTL-sat
Input: A PLCS $\mathcal{L}$, PTL formula $\phi$, $p \in (0,1]$ and a tolerance $\nu$
Output: 'true' if $\mathcal{L}$ satisfies $\phi$ with probability $p$ and tolerance $\nu$ and 'false' otherwise.
var: p-sat, p-nosat : real;
var: $k$ : integer;
var: $states_k$ : set;
begin
    p-sat:=0; p-nosat:=0;
    $k$:=0;
    do
        if p-sat $\geq p$ then
            exit(true);
        if p-nosat $\geq 1 - p$ then
            exit(false);
        Construct $\mathcal{M}_k$;
        if $k > 0$ and $states_k = states_{k-1}$ then (* PLCS is finite state *)
            exit(false);
        p-sat := Sat-Prob($\mathcal{M}_k, \phi$);
        p-nosat := Sat-Prob($\mathcal{M}_k, pn(\neg\phi)$);
        $k := k + 1$;
    while p-sat + p-nosat $< 1 - \nu$;
    exit(true with in tolerance $\nu$);
end.
```

Fig. 2.

Lemma 18. *Let $\mathcal{M}_1$ and $\mathcal{M}_2$ be two markov chains such that $\mathcal{M}_1 \trianglelefteq \mathcal{M}_2$. If μ_1 and μ_2 are the probability functions defined by the sequence spaces of $\mathcal{M}_1$ and $\mathcal{M}_2$, respectively, then for every XPTL formula ϕ,*

$$\mu_1(\{\pi \mid \mathcal{M}_1, \pi \models \phi\}) \leq \mu_2(\{\pi \mid \mathcal{M}_2, \pi \models \phi\}).$$

Proof. Consider a basic cylindric set in $\{\pi|\mathcal{M}_1, \pi \models \phi\}$ with a prefix $\gamma_0\gamma_1 \dots \gamma_n$. By Theorem 17 there is a basic cylindric set of $\mathcal{M}_2$ with prefix $\gamma_0\gamma_1 \dots \gamma_n$. By Theorem 17 all computations in this basic cylindric set satisfy ϕ. Hence we have $\mu_2(\{\pi \mid \mathcal{M}_2, \pi \models \phi\}) \geq \mu_1(\{\pi \mid \mathcal{M}_1, \pi \models \phi\})$.

By Lemma 18 we see that (a) the satisfaction probability of ϕ by $\mathcal{M}_k$, for any integer k, is a lower bound for the satisfaction probability of ϕ by $\mathcal{M}$ and (b) the lower bounds form a monotonic sequence. That this monotonic sequence converges to the probability with which $\mathcal{M}$ satisfies ϕ is proved in Lemma 21

5.2 Algorithm

To check the satisfaction of a PTL formula by a PLCS, the question to be answered is, for a given PLCS $\mathcal{M}$, a PTL formula ϕ, $p \in [0,1)$ and a tolerance (accuracy) $\nu > 0$, "Does $\mathcal{M} \models_p \phi$?". The algorithm (shown in Figure 2) inputs the representation of the PLCS and computes successive approximations to the satisfaction probability of ϕ in $\mathcal{M}_k$, and the satisfaction probability of $pn(\neg\phi)$ in $\mathcal{M}_k$, and records them in the variables *p-sat* and *p-nosat*, respectively. If in some iteration 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, and the algorithm can terminate. In the following, we will show that the algorithm always terminates and that it computes the correct solution. To that end, note that the computations of a markov chain $\mathcal{M}_k$ can be divided into the following five, mutually disjoint, sets:

1. $C^k_{sat,\hat{D}}$: computations that satisfy ϕ and do not end in D (characterized by its measure $p^k_{sat,\hat{D}}$)
2. $C^k_{sat,D}$: Computations that satisfy ϕ and end in D (with measure $p^k_{sat,D}$)
3. $C^k_{nosat,D}$: computations that satisfy $pn(\neg\phi)$ and do end in D (measure $p^k_{nosat,D}$)
4. $C^k_{nosat,\hat{D}}$: computations that satisfy $pn(\neg\phi)$ and do not end in D (measure $p^k_{nosat,\hat{D}}$), and
5. $C^k_{unknown}$: computations that satisfy neither ϕ nor $pn(\neg\phi)$ (and end in D) - measure $p^k_{unknown}$

Since these computations of $\mathcal{M}_k$ are mutually disjoint, it is the case that

$$p^k_{sat,D} + p^k_{sat,\hat{D}} + p^k_{nosat,D} + p^k_{nosat,\hat{D}} + p^k_{unknown} = 1$$

Call the values of the variables *p-sat* and *p-nosat* computed at the k^{th} iteration as *p-sat*^{k} and *p-nosat*^{k}. We then have the relations *p-sat*$^k = p^k_{sat,\hat{D}} + p^k_{sat,D}$, and *p-nosat*$^k = p^k_{nosat,D} + p^k_{nosat,\hat{D}}$. Note that by Lemma 18 the variables *p-sat*^{k} and *p-nosat*^{k} are non-decreasing, as k increases. Therefore, the quantity $p^k_{unknown}$ is non-increasing across the iterations of the algorithm **PTL-sat**.

We will now prove that $p^k_{unknown}$ decreases as k increases.

Lemma 19. *Suppose $\mathcal{M}_{k_1}$ is a Markov chain and $p^{k_1}_{unknown} > 0$. Then there exists a $k_2 > k_1$ such that $\mathcal{M}_{k_1}$ and $\mathcal{M}_{k_2}$ are distinct Markov chains and $p^{k_1}_{unknown} > p^{k_2}_{unknown}$.*

Proof. Consider a computation $\pi_1 = \gamma_0\gamma_1 \cdots \gamma_{k_1} D^\omega$ of $\mathcal{M}_{k_1}$. There must be a path from γ_{k_1} to a closed SCC of $\mathcal{M}$. Otherwise D captures a closed SCC of $\mathcal{M}$ (which is impossible from the definition of $\mathcal{M}_k$). This path appears in some $\mathcal{M}_{k_2}$ where $k_2 > k_1$. Thus $p^k_{unknown}$ eventually decreases.

Given that $p^k_{unknown}$ is decreasing, we have

Theorem 20. *The algorithm* **PTL-sat** *terminates on all inputs.*

We now discuss the correctness of the algorithm. Given that the probabilities are over rationals/reals the correctness criteria will involve both p and ν. But, we first prove that the sequence *p-sat*^{k} that we compute, converges to the right result.

Lemma 21. *As k approaches ∞, $p\text{-}sat^k$ approaches the probability of satisfaction of ϕ by $\mathcal{M}$.*

Proof. We know that $p\text{-}sat^k$ and $p\text{-}nosat^k$ are both monotonically increasing (Lemma 18) and that $p\text{-}sat^k + p\text{-}nosat^k$ approaches 1 as k approaches ∞. Suppose $\lim_{k\to\infty} p\text{-}sat^k$ is less than the probability of satisfaction of ϕ by $\mathcal{M}$. Then $lim_{k\to\infty} p\text{-}nosat^k$ is greater than the probability of satisfaction of $pn(\neg\phi)$ by $\mathcal{M}$ i.e., there is some k such that probability of satisfaction of $pn(\neg\phi)$ in $\mathcal{M}_k$ is greater than the probability of satisfaction of $pn(\neg\phi)$ by $\mathcal{M}$. By Lemma 18 this is not possible.

From these lemmata we conclude the following, final, correctness theorem:

Theorem 22.

Soundness *If* **PTL-sat**$(\mathcal{L}, \phi, p, \nu)$ *terminates with the result* `true` (`false`, `true` *with in* ν) *then* $\mathcal{L} \models_p \phi$ ($\mathcal{L} \not\models_p \phi$, $\mathcal{L} \models_{p'} \phi$ *where* $p - \nu < p' < p + \nu$, *respectively).*

Completeness *If* $\mathcal{L} \models_{p'} \phi$ *where* $p - \nu < p'$ *then* **PTL-sat**$(\mathcal{L}, \phi, p, \nu)$ *terminates with the result* `true` *or* `true` *with in* ν. *If* $\mathcal{L} \not\models_{p-\nu} \phi$ *then* **PTL-sat**$(\mathcal{L}, \phi, p, \nu)$ *terminates with the result* `false`.

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 modeling probability of errors in the specification, we believe, we have made protocol specifications more realistic. Two problems remain to be explored: that of complexity of the algorithm and efficient implementations, and the effect of numerical errors and stability during the calculation. Is the model-checking problem in the traditional sense: "$\mathcal{L} \models_1 \phi$" (with no notion of tolerance) decidable? Our conjecture is that it is probably not, given that non-probabilistic version of the problem is undecidable.

References

1. P. Abdulla and B. Jonsson. Verifying programs with unreliable channels. In *Proceedings of the IEEE Symposium on Logic in Computer Science*, 1993.
2. 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.
3. Bartlett, Scantlebury, and Wilkinson. A note on reliable full-duplex transmission over half duplex lines. *Comm. ACM*, 12(5):260–265, 1969.
4. G. Bochmann. Finite state description of communication protocols. *Comput. Networks*, 2:362–372, 1978.
5. D. Brand and P. Zafiropulo. On communicating finite state machines. *Journal of the Association of Computing machinery*, 30(2):323–342, 1983.

6. 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.
7. C. Courcoubetis and M. Yannakakis. Verifying temporal properties of finite-state probabilistic programs. In *Proc. 1988 IEEE Symp. on the Foundations of Comp. Sci.*, 1988.
8. M. Diaz, J.P. Ansart, P. Azema, and V. Chari. *The Formal Description Technique Estelle.* North-Holland, Amsterdam, 1989.
9. 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.
10. A. Finkel. Decidability of the termination problem form completely specified protocols. *Dist. Comput.*, 7:129–135, 1994.
11. 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.
12. J. G. Kemeny, J. L. Snell, and A. W. Knapp. *Denumerable Markov Chains.* Van Nostrand, New Jersey, 1966.
13. E. M. Clarke M. C. Browne and O. Grümberg. Characterizing finite kripke structures in propositional temporal logic. *Journal of TCS*, 59:115–131, 1988.
14. 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.
15. M. Y. Vardi. Automatic verification of probabilistic concurrent finite-state programs. In *IEEE Symposium on Foundations of Computer Science*, pages 327–338, 1985.
16. P. Wolper, M. Vardi, and A. Sistla. Reasoning about infinite computation paths. In *IEEE Symposium on Foundations of Computer Science (FOCS)*, 1983.
17. CCITT Recommendation Z.100. Specification and description language sdl. Blue Book Vols X.1-X.5, ITU General Secreterait, Geneva, 1989.