A LOTOS Extension for the Performance Analysis of Distributed Systems

A LOTOS Extension for the Performance Analysis of Distributed Systems

Marco Ajmone Marsan, *Senior Member, IEEE*, Andrea Bianco, Luigi Ciminiera, *Member, IEEE*, Riccardo Sisto, and Adriano Valenzano

***Abstract*— Performance analysis and formal correctness verification of computer communication protocols and distributed systems have traditionally been considered as two separate fields. However, their integration can be achieved by using formal description techniques as paradigms for the development of performance models. This paper presents a novel extension of LOTOS, one of the two formal specification languages that were standardized by ISO. The extension is specifically conceived to integrate performance analysis and formal verification. The extended language syntax and semantics are formally defined, along with a mapping from extended specifications to performance models. The mapping preserves the specified observable behavior. Two simple examples, a stop-and-wait protocol and a time-sharing system, are used to concretely demonstrate the new approach and to validate it.**

I. INTRODUCTION

THE design and the correct implementation of communication protocols are difficult tasks, for which many formal models have been proposed and employed. The aim of such models, also called formal description techniques (FDT), is to give precise, unambiguous and complete specifications, which are mandatory in order to avoid incompatibility among different implementations of a given protocol and to prove correctness properties. Initially, FDT's were based on existing modeling paradigms, such as state machines, Petri nets, or process algebras. More recently, *ad hoc* specification languages were developed, and three of them, namely SDL [1], LOTOS [2], and Estelle [3], became international standards.

FDTs were originally conceived to describe the protocol behavior and functionalities in a time-independent fashion. This is adequate when dealing with nontime-critical applications, i.e., when timing does not affect correctness, but only performance. If this is not the case, formal models including quantitative temporal specifications are necessary in order to formally verify self-consistency, as well as the desired functional properties. However, timed FDT's are also relevant outside the scope of time-critical systems, since they can be used as paradigms for the construction of performance models of protocols.

Manuscript received August 1993; revised December 1993; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor K. Sabnani.

M. Ajmone Marsan and A. Bianco are with the Dipartimento di Elettronica, Politecnico di Torino, Italy.

L. Ciminiera and R. Sisto are with the Dipartimento di Automatica e Informatica, Politecnico di Torino, Italy.

A. Valenzano is with Centro Elaborazione Numerale dei Segnali, CNR, Torino, Italy.

IEEE Log Number 9403560.

Traditionally, formal correctness verification and performance analysis have been two separate fields. Whereas the validation and verification processes are based on formal techniques, the classical approach to performance analysis is based on human ingenuity and experience, and consists in devising specific abstract models which can be analyzed by simulation or by applying stochastic process theory. A key problem of the traditional performance evaluation approach lies in the credibility of the model: the functional equivalence of the performance model and the actual protocol is almost impossible to prove. This is not the case for the model used for verification, since this is the specification of the protocol itself. This consideration suggests that formal description techniques should be used for performance analysis, besides formal verification. Such an integration brings along many other advantages. Among them, it makes performance prediction possible in the early design phases, thus avoiding costly redesign, and it facilitates the automation of the performance analysis process.

As already noted in [4], the use of a formal description language as a paradigm for performance modeling requires the extension of the language with temporal and probabilistic specifications. The temporal specifications are necessary to describe the time lapse between consecutive events, and the probabilistic specifications are necessary to describe the selection among different possible behaviors.

Many researchers [5]–[14] have considered the two types of extension separately, the timing extension being interesting in itself for formal verification of time-critical systems, and the probabilistic extension being interesting for probabilistic verification or testing, when an exhaustive validation is impossible. Their work gives a reasonable insight into the related problems, but merging probabilistic and timing information for performance modeling involves new aspects.

Pioneering work in this respect was done in the area of Petri Nets, with the formulation of some well-known timed and probabilistic extensions such as stochastic Petri Nets (SPN) [15]–[17] and their offsprings [18]–[23], or Timed Petri Nets [24]–[26]. Other interesting proposals for integrating performance analysis within the framework of formal specification techniques were based on state machine models or formal grammar models [27]–[30].

In the area of algebraic specification techniques, despite the growing interest in such models, and in LOTOS in particular, little attention was paid to the integration of performance analysis with formal specification and verification. The majority of

works dealing with quantitative extensions consider as a target only the verification of time-critical systems.

In [31], Nounou and Yemini address the problem of integrating performance analysis aspects into a variant of CCS, while [32] proposes an extension of CCS which could be used for performance analysis. In [33], Rico and Von Bochmann propose a method for introducing deterministic times and mass probabilities in LOTOS, so as to obtain a corresponding semi-Markov model which can be used for performance analysis. Their timing model however is inspired by an old proposal [34] which is not able to describe global timing constraints properly.

In this paper we describe a more general extension of LOTOS that is suitable for the construction of performance models of communication protocols, and distributed systems. The proposed version of extended LOTOS is derived from a previous timed extension proposal by Bolognesi *et al.* [35]–[37], but it considers some new features, such as the introduction of random variables for the probabilistic specification of timing, and the use of priorities and weights for the resolution of conflicts.

The reader is assumed to be familiar with LOTOS. A comprehensive tutorial can be found in [38].

For the sake of simplicity in the presentation, like in [35]–[37], we only consider a subset of basic LOTOS.

II. The Earlier Timed LOTOS Extensions

The first timed LOTOS extension which appeared in the literature was due to Quemada and Fernandez [5], and was later improved in [34]. According to this definition, action denotations are extended with time intervals: the notation `a<t1,t2>` indicates that the event represented by action denotation `a` *may* take place only within the time interval `[t1,t2]`, relative to the instant when `a` becomes locally enabled according to the LOTOS semantics. In this context, the word "locally" means that the constraints deriving from synchronizations with other parallel processes are not taken into account. As a consequence, parallel processes have independent timing specifications, and synchronizations *may* take place only if the absolute time intervals associated with a given synchronization event in the various parallel processes involved in the synchronization overlap.

The extension by Bolognesi, Lucidi and Trigila [35]–[37] tries to overcome one of the main problems in the earlier proposals, i.e., the inability to express global timing constraints. In the earlier proposals, if two parallel processes that must synchronize at gate g are both ready for an interaction at g, the delay associated with the synchronization at g cannot be specified directly, but is a consequence of the two local timing specifications. This makes it difficult to understand how global timing is affected by local timing. In [35]–[37] temporal specifications are given in terms of time intervals, but they are associated with gates rather than with action denotations. The syntactic construct `timer g<t1,t2> in B` indicates that in expression `B` the time interval `<t1,t2>` is assigned to gate `g`. The process described by `B` may execute an action at gate `g` only when the age of the action at gate `g` lies in the interval `<t1,t2>`, the age of an action prefix being defined as the cumulative amount of time during which the action prefix event has been continuously enabled. Note that `g` may be a gate at which several subprocesses of `B` synchronize, in which case the temporal specification assumes a global significance over all the subprocesses in `B`, and the interval indicates how long it takes from the last subprocess getting ready at `g` until the interaction takes place.

The `timer` construct defined in [35] can be used to describe not only global but also local temporal constraints of the kind introduced in [5]. For example, in

```
(timer g < t1,t2 > in B) |[g]|
                          (timer g < t3,t4 > in B')
```

`B` and `B'` are characterized by separate local temporal constraints on `g`.

In the following we shall call T-LOTOS the timed LOTOS extension defined in [37], as suggested by the authors.

III. The Novel LOTOS Extension

The LOTOS extension proposed here is based on the T-LOTOS timed extension [37]. However, in contrast with [37], which permits the specification of both global and local timing, we only admit the use of timer operators (including the extensions that will be introduced) with a global scope.

One first reason for this requirement is that in [37] the cascade of several `timer` operators is equivalent to a `timer` operator whose time interval is the intersection of the time intervals of the single operators, but, as will be clear in the following, the introduction of our probabilistic characterizations makes it impossible to conveniently define a similar meaning for the same operator combination. A second reason is that with this requirement each timed event is controlled by a single timer operator, and the assignment of timings is clear and easy to understand. Obviously, such clarity and linearity is obtained at the expense of a reduced compositionality in the language. This means, for example, that if we have separately written two parts of a specification and we want to combine them, we have to revise the assignment of timing and probabilistic characterizations for what concerns the gates at which the two parts interact. This point, which may be considered as a drawback, is indeed also an advantage, because it allows the user to freely decide how to combine the timing and probabilistic characterizations, and avoids errors due to the difficulty of understanding how global timing is affected by local timing.

A. Presynchronization Timer

The definition of T-LOTOS enables the specifier to directly express situations in which the processes interacting at a gate, after reaching a consensus on the synchronization, must wait for a given time before actually being committed in the synchronization. During this time interval, it is possible for each one of the processes involved in the synchronization to execute another action, thus aborting the synchronization.

In some cases, instead, it may be useful to be able to express situations in which the commitment is instantaneous, but a

certain time is necessary to perform common operations. Only after this time has elapsed, the involved processes are free to execute other actions. For example, let us consider the expression

```
(R ||| R) |[get]| prepare; get; stop
```

where each R is an instance of a receiver process defined as `R := get; ELABORATE [] timeout; stop` and the right hand side of the expression represents a provider process, which prepares a message (event `prepare`) for the receiver processes. If R does not time out before (event `timeout`), it can get a message (event `get`) and then elaborate it (process `ELABORATE`). If we assume that the transfer of the message from the provider to one of the receivers takes time, we can assign such time to gate `get`, but we must require that it elapses *after* the commitment at `get`, otherwise a receiver could time out while receiving the message.

This different kind of timed synchronization can simply be expressed in terms of the basic synchronization introduced in [37], by splitting the synchronization into a pre-synchronization event, which is an interaction with null delay, followed by the timed synchronization. For the example presented above, the timed expression would be:

```
hide h in timer h<0,0>, get< t1,t2>
in (R ||| R) |[h,get]| prepare; h;
get; stop where R := h; get; ELABO-
RATE [] timeout; stop
```

where we have used the shorthand

```
timer X,Y in B = timer
X in timer Y in B
```

Since the modifications necessary to implement this new kind of synchronization may be extensive, and may complicate the specification significantly, resulting in reduced readability, it seems useful to introduce a new timer operator, whose semantics can be expressed in terms of the original timer operator semantics. We chose to define the new operator as a macro expansion operator, which will be indicated by the new keyword `p_timer`, and will be assigned the same syntax as the `timer` operator. The macro expansion rule therefore is

```
p_timer g<t1,t2> in B = hide g1 in
timer g1<0,0>, g<t1,t2> in B'
```

where `g1` is a new unique identifier and `B'` is the same as `B` with all occurrences of action prefix `g;` replaced by `g1; g;`, and all occurrences of `g` in synchronization operators replaced by `g1,g`, the same replacement being recursively applied to all the processes directly or indirectly instantiated in `B`.

B. Probabilistic Extension

In T-LOTOS, the temporal properties of a system are described using the `timer` construct which assigns to a gate `g` a time interval within which any action at `g` must be executed. We introduce random variables to describe the time lapse according to the same temporal specification, but in a probabilistic way. For each random variable it is necessary to define:

- a time interval which is the domain of the random variable;
- a probability density function of the random variable.

The first item is the time interval associated with the corresponding gate as in T-LOTOS, while the second item is an additional information which should be assigned to each time interval. Whenever an event becomes enabled, a random variable instance is extracted from the probability density function, and its value represents the actual delay of the event; the instance value lies in the domain of the random variable, as required by T-LOTOS semantics.

This characterization describes probabilistic time lapse in an explicit way, and probabilistic selection in an implicit way, because, according to T-LOTOS semantics, the action whose associated delay is shortest, is actually selected for execution when several actions are competing. Nevertheless, the explicit definition of a selection criterion is sometimes mandatory, e.g., to solve a conflict between enabled events with the same deterministic delay, or a conflict between events associated with delays described as random variables when the instances of the random variables are identical. For this purpose, we introduce a priority (ranging from p_{min} to p_{max}) and a weight (ranging from w_{min} to w_{max}) associated with each event. When a choice between enabled events with the same delay must be made, the priority and the weight are orderly used to solve the conflict: first of all, the priorities are examined, and a deterministic decision is taken; if the conflict is not solved (i.e., two or more enabled events have the same priority), the enabled events are assigned probabilities proportional to their weights, and a random choice is made according to this probability assignment.

Priorities and weights are assigned to gates just as time intervals are assigned to gates in [37]. The resulting syntactic extension to the `timer` operator is the following:

```
timer a< t1, t2, pdf(), priority,
weight> in B
```

where

- `a` is a gate;
- `t1,t2` (`t2` $\geq$ `t1`) describe the domain of the random variable representing the delay associated with the gate;
- `pdf()` is a function $[\texttt{t1},\texttt{t2}] \rightarrow [0,\infty]$ that describes the characteristic of the pdf ($\int_{t1}^{t2} \texttt{pdf}(x)dx = 1$) of the same random variable;
- `priority` is the priority used to solve conflicts in a deterministic way;
- `weight` is the weight used to solve in a probabilistic way the conflicts not solved by priorities;
- `B` is a behavior expression.

When values are not explicitly defined, default values are assigned to the extremes of the time interval, to the pdf type, to the priority and weight for a gate. The default time interval is `<0,0>`, so that the default timing is a deterministic null delay; the default pdf is uniform over the defined time interval; the default priority is $\frac{p_{max}+p_{min}}{2}$, and the default weight is $\frac{w_{max}+w_{min}}{2}$.

With respect to the T-LOTOS proposal, the probabilistic extensions do not alter the possible behaviors of the speci-

fication, but only make it possible to specify how likely the various behaviors are to occur. The semantics expressed in [37] are therefore still valid for the purpose of verification and, in general, when the probabilistic aspect is not of interest.

The probabilistic extension defined here can be applied to the pre-synchronization timer operator as well. The extension can be reasonably defined in the following way ("holes" in the parameter list represent default values):

```
p_timer g<t1,t2, pdf(), prior-
ity,weight> in B = hide g1 in timer
g1<0,0,,priority,weight>, g<t1,t2,
pdf(),,> in B'
```

Indeed, the priority and the weight are to be used to solve conflicts, which may arise only during pre-synchronization, i.e., with gate `g1`, whereas the pdf is needed only to characterize the interaction at `g`.

The probabilistic extension that we are introducing is substantially different from the existing proposals of probabilistic process algebras [12]–[14]. The main difference is that in [12]–[14] only mass probabilities associated with choice operators are considered, because the extension is conceived mainly for probabilistic testing, while we needed a more powerful model, including a probabilistic description of continuous time, for performance evaluation. Another important remark is that, according to the classification of probabilistic extensions made in [14], our proposal can be categorized as a *generative* model, i.e. it characterizes in a probabilistic way not only the internal choices of the system, but also the interactions with its environment (e.g., the system inputs). This feature is required for obtaining complete performance evaluation models, but it is not needed for probabilistic testing. Our probabilistic extension is also more general than the generative model proposed in [33] for LOTOS because the latter is based on deterministic times and mass probabilities associated with choice operators.

C. Time-Independent Probabilistic Choices

Expressing the probabilistic characterization of a nondeterministic choice by assigning specific time distributions to the various alternatives is satisfactory in some situations, but it may lead to difficulties when the timing information is naturally independent of the selection criterion. For example, suppose that we must model a communication channel with a given failure rate and with a given distribution of the transmission time. Since the distribution of the transmission time is fixed, the failure rate must be obtained by properly specifying a fictitious time distribution for the channel failure event. This surely results in a poorly readable model, and may induce errors in the cases in which the derivation of the fictitious distribution is not straightforward.

The introduction of presynchronization (the `p_timer` operator) helps in solving this difficulty: if the initial events representing the various alternatives are subject to `p_timer` operators (or if they are characterized by identical deterministic times), their priorities and weights determine a choice which is independent of the delay assignments. For example, the expression

```
p_timer out <5,10, ,1,998> in timer
loss< 0,0, ,1,2> in CH where CH :=
inp; (out; CH [] loss CH)
```

describes a channel `CH` which is characterized by a transmission time uniformly distributed over the interval [5, 10] and an error rate of 0.2%. The `p_timer` operator assigned to gate `out` introduces a new hidden event with null delay, so that the weights 998 and 2 are actually used to probabilistically select between the two branches of the choice.

D. Memory Timers

When modeling communication protocols and distributed systems, it is important to be able to describe different typical scenarios. In general, a system in a given state can be considered as a set of parallel activities requiring some (residual) time to complete. In T-LOTOS, each activity is a gate interaction, and a timer operator is used to express the time it takes from the instant it is enabled until the instant it completes. The *age* associated with actions and interactions represents the amount of time the activity has already spent, i.e., the amount of work already performed by the activity.

The simplest scenario is a memoryless one, in which the occurrence of any event has the effect of restarting all the parallel activities in the system. In practice, if more than one event is enabled, the one which takes place first causes the work performed by the other activities up to that instant to be lost. This scenario corresponds to rather uncommon situations, since normally in distributed systems each process keeps its memory when events occur in other concurrent processes.

The `timer` operator introduced in [37] enables the user to model a more realistic scenario, i.e., one in which the occurrence of an event does not restart all the activities: concurrent activities not synchronized with the one(s) having produced the event remain enabled after the state change and are not restarted, which means that the work they have already performed (i.e., their age) is not lost. Only the work performed by the other activities, which are disabled by the event occurrence, is lost: they will be restarted when they become enabled again. For example, this scenario applies to a protocol machine which, at a given time, is composed of two alternative activities representing the reception of an acknowledgment and a timeout, plus a third activity, running in parallel without synchronization with respect to the others, representing the transmission of another message. If the acknowledgment reception activity completes, the transmission activity continues to be enabled and is not restarted, whereas the timeout activity is no more enabled, and will be restarted when it is enabled again.

Even if this kind of behavior is useful in modeling a great variety of real systems, another typical scenario exists in distributed systems, which is not included in the timed model proposed in [37], but it is equally important. This scenario is one in which the occurrence of an event does not restart any activity, except the one associated with the event which has just taken place. Moreover, even if an activity which was enabled is no longer enabled in the new system state, the work it has previously performed is not lost: when the activity is

TABLE I
THE SYNTAX OF ET-LOTOS BEHAVIOR EXPRESSIONS

Operator	Syntax
action prefix	`a; B`
choice	`B1 [] B2`
parallel composition	`B1 |[G]| B2` requirement: $istimed_g(\texttt{B1}) = istimed_g(\texttt{B2}) = false\ \forall g \in \texttt{G}$
timer	`timer a<t1,t2,pdf(),priority,weight> in B` requirement: $istimed_g(\texttt{B}) = false$
pre-synchronization timer	`p_timer a<t1,t2,pdf(),priority,weight> in B` requirement: $istimed_g(\texttt{B}) = false$
memory timer	`m_timer a<t1,t2,pdf(),priority,weight> in B` requirement: $istimed_g(\texttt{B}) = false$
process instantiation	`P[G]` where `P` is a process name
hiding	`hide G in B`
relabelling	`B[G/H]` requirement: `H` and `G` are gate lists of the same length

enabled again later, it will resume its work from the point at which it was interrupted. For example, let us consider a server which serves one request at a time, with a preemption mechanism to serve immediately urgent requests. A state of the server can be modelled by means of two alternative activities, one representing the completion of the current request, and the other representing the occurrence of an urgent request. If this second activity completes, the former is disabled, but the work it has already performed must not be lost, because it has to be resumed when the urgent request has been served. Only the activity representing the arrival of an urgent request is to be restarted in this case.

In order to model this different kind of situations within the framework of timed LOTOS, it is necessary for the timed model proposed in [37] to be enhanced, and this can be done by introducing a new `timer` operator, whose semantics is defined according to the new timed behavior.

We shall call this new operator a **memory timer operator**, and we shall assign it a syntax similar to the one used for the `timer` operator in [37], incorporating the extensions introduced in the previous sections. The expression

```
m_timer a<t1, t2, pdf(), priority,
weight> in B
```

means that, in expression `B`, an interaction at gate `a` can take place when the total amount of time during which event `a` has been enabled since the last event at `a` reaches the value extracted from the probability density function `pdf()` (in the range `[t1,t2]`). Note that the enabling time that is considered here is not necessarily continuous, but it may be the sum of several enabling intervals, since the age accumulated during an enabling period is not lost if the event is disabled by the occurrence of another alternative event. Note also that different interactions at the same gate contribute to the same enabling age. For example, the expression

```
m_timer tx<10,50, ,1,> in m_timer
main<1000,1000, ,2,> in T where T :=
get; (tx; T [] main; tx; T)
```

models a protocol entity which iteratively takes a message (event `get`), and then transmits it (event `tx`). Every 1000 time units of transmission time, the entity executes a maintainence operation (event `main`), after which it resumes transmission of the interrupted message. The `m_timer` at `main` is needed because the timer counting the 1000 time units must remember its value from one transmission to the other, while the `m_timer` at `tx` is needed because, after the maintainence, the transmission must be restarted from the point at which it was interrupted. Note that here we have exploited the fact that the enabling age for the two occurrences of `tx` is the same. Would this feature not be required, it is sufficient to split a gate into several distinct gates. Event `main` has been assigned a priority higher than that of `tx`because, if the maintainence time comes exactly when a transmission time has elapsed, we want the maintainence to be made before going to the next transmission.

With the introduction of the memory timer operator, the semantics of the LOTOS extension defined in [37] is no longer sufficient for describing the functional (non-probabilistic) properties of a specification, but it must be extended accordingly. The formal definition of both the syntax and the semantics of our LOTOS extension, that we call *Extended Timed LOTOS (ET-LOTOS)*, is given in the next section.

E. Formal Definition of Extended Timed LOTOS

1) Syntax and Macro Expansions: Table I defines the syntax of behavior expressions in ET-LOTOS. In the table, `a` denotes a gate, `B`, `B1`, and `B2` denote behavior expressions, `G` and `H` denote gate lists, whereas `t1`, `t2`, `pdf()`, `priority`, and `weight` denote the parameters of timer operators, as defined in the previous section. Any one of such parameters can be omitted, thus taking the default value.

The structure of a specification remains as it is in basic LOTOS.

The boolean function $istimed_g()$ is used to formally define the restrictions concerning the composition of timer operators. $istimed_g(\texttt{B})$ takes value *true* for behavior expressions `B` already containing some timing specifications referred to gate

TABLE II
THE DEFINITION OF THE FUNCTION $istimed_g()$

Beh	$istimed_g$(Beh)	
a; B	$istimed_g$(B)	
B1[]B2	$istimed_g$(B1) OR $istimed_g$(B2)	
B1|[G]|B2	$istimed_g$(B1) OR $istimed_g$(B2)	
timer g<X> in B	*true*	
timer a<X> in B	$istimed_g$(B)	if $a \neq g$
m_timer g<X> in B	*true*	
m_timer a<X> in B	$istimed_g$(B)	if $a \neq g$
P[G]	$istimed_g$(B[G/H])	given P[H] := B
hide G in B	*false*	if $g \in$ G
hide G in B	$istimed_g$(B)	if $g \notin$ G
B[g1..gn/h1..hn]	$istimed_g$(B)	if $g \notin$ {g1..gn}
B[g1..gn/h1..hn]	$istimed_{hi}$(B)	if $g =$ gi

X = t1,t2,pdf(),priority,weight
g1..gn and h1..hn are gate lists.

g. The timer operators referred to gate g and the parallel operators having g as a synchronization gate can be applied only to behavior expressions B for which $istimed_g$(B) is *false*. This function is defined formally in Table II.

The following macro expansion rules apply:

1) `timer a<X>, b<Y> in B =`
   `timer a<X> in timer b<Y> in B`
2) `p_timer g<t1,t2,pdf(),priority,weight> in B =`
   `hide g1 in timer g1<0,0, ,priority, weight>, g<t1,t2,pdf(), ,> in B'`
   where B' is the same as B with all occurrences of g; replaced by g1; g; and all occurrences of g in synchronization operators or process instantiations replaced by g1,g. The same replacement must be recursively applied to all the processes directly or indirectly instantiated in B.

2) Semantics: The semantics of ET-LOTOS can be divided into two parts: a functional part and a probabilistic part. The former defines the possible behaviors of ET-LOTOS specifications, and is analogous to T-LOTOS semantics, whereas the latter defines in a probabilistic way how the system selects among its different possible behaviors.

The functional semantics can be defined in operational style, using the same formalism introduced in [37] for T-LOTOS, by means of the axioms and inference rules reported in Table III. Two kinds of transitions are defined: action transitions, representing the occurrence of events, and aging transitions, representing the time lapse. Action and aging transitions can be interleaved in any way, but it is always possible to merge successive aging transitions into a single one, thus describing, in the general case, a trace as a sequence of actions, possibly with a delay between consecutive actions, or, equivalently, as a sequence of actions occurring at nondecreasing time instants.

As in [37], we assign to each action denotation an age τ, which is syntactically represented as a superscript, and represents the cumulative amount of time during which the corresponding event has been continuously enabled. Moreover, the notation $B - a^{\tau} \rightarrow B'$ means that an event a with age τ can take place in the process defined by expression B, thus making B transform into B'. We have introduced also a second age parameter, η, which is associated with every occurrence of memory timer operators. The expression $m\_timer^{\eta}a<X>$ in B specifies that gate a was enabled in B for a total of η time units since the last occurrence of an event at a. This parameter enables us to represent the total amount of work that was performed at a given gate. The action denotations and memory timer operators that appear in ET-LOTOS specifications have no superscript, according to ET-LOTOS syntax, but an implicit age parameter equal to 0 is assigned to each of them. Thus, for example, an occurrence of a;B in a specification is to be interpreted as a^0;B in the application of the rules of Table III.

Function $age_g()$ has the same meaning as in [37], and is defined in Table IV.

The formal definition of the functional semantics of ET-LOTOS opens the possibility of applying the formal verification techniques for assessing correctness properties and equivalence relations.

The probabilistic part of the semantics of ET-LOTOS was already defined in almost formal terms in Section III-B, where the operational interpretation of the timer parameters pdf(), priority, and weight is explained. Such a definition is sufficient for the purpose of building and interpreting performance models of the specified systems.

IV. A General Framework for Performance Modeling

In this section we describe a general environment for the construction of performance evaluation models. The mapping of ET-LOTOS specifications onto performance models of this type is described in the next section.

The model is based on a state machine. Focusing our attention on the case in which the state space is finite, we can identify the following entities in the model:

- a set S of states, $S = \{s_i | 1 \leq i \leq M\}$;
- a (normally finite) set T of transitions, $T = \{t_j | 1 \leq j \leq N\}$; transitions are divided into two classes: transitions without memory ($U \subseteq T$) and transitions with memory ($V \subseteq T$);
- a mapping $E : S \rightarrow \mathcal{P}(T)$ ($\mathcal{P}$ denoting the power set operation), which defines, for each state s_i, the set of transitions enabled in that state, $E(s_i)$;
- a new-state partial function $N : S \times T \rightarrow S$, such that $N(s_i, t_j)$ is the state reached by firing transition t_j when in state s_i. The function is defined only for the pairs (s_i, t_j) such that $t_j \in E(s_i)$;
- a mapping $Q : S \times T \rightarrow \mathcal{P}(T)$ which defines, for each pair s_i, t_k such that $t_k \in E(s_i)$, the set $Q(s_i, t_k)$ of transitions whose associated delay must be resampled as a consequence of firing t_k in state s_i;
- a set D of random variables, $D = \{d_j | 1 \leq j \leq N\}$, d_j representing the delay associated with transition t_j;
- a set R of variables, $R = \{r_{ij} | 1 \leq i \leq M, 1 \leq j \leq N\}$, r_{ij} representing the residual delay associated with transition t_j in state s_i;
- a set P of priorities $P = \{p_j | 1 \leq j \leq N\}$, p_j being the priority assigned to transition t_j;

TABLE III
EXTENDED TIMED LOTOS FUNCTIONAL SEMANTICS (X=t1,t2,pdf(),priority,weight)

ACTION TRANSITIONS	AGING TRANSITIONS ($t, t' > 0$)
a1) $\mathtt{a}^{\tau};\mathtt{B} - \mathtt{a}^{\tau} \rightarrow \mathtt{B}$	t1) $\mathtt{a}^{\tau};\mathtt{B} = t \Rightarrow \mathtt{a}^{\tau+t};\mathtt{B}$
a2) $\dfrac{\mathtt{B1} - \mathtt{a}^{\tau} \rightarrow \mathtt{B1}'}{\mathtt{B1[]B2} - \mathtt{a}^{\tau} \rightarrow \mathtt{B1}'}$	t4) $\dfrac{\mathtt{B1} = t \Rightarrow \mathtt{B1}', \mathtt{B2} = t \Rightarrow \mathtt{B2}'}{\mathtt{B1[]B2} = t \Rightarrow \mathtt{B1}'\mathtt{[]B2}'}$
a3) symmetric of rule a2)	
a4) $\dfrac{\mathtt{B1} - \mathtt{a}^{\tau 1} \rightarrow \mathtt{B1}', \mathtt{B2} - \mathtt{a}^{\tau 2} \rightarrow \mathtt{B2}', \mathtt{a} \in \mathtt{G}}{\mathtt{B1}\,\vert\,\mathtt{[G]}\,\vert\,\mathtt{B2} - \mathtt{a}^{min(\tau 1, \tau 2)} \rightarrow \mathtt{B1}'\,\vert\,\mathtt{[G]}\,\vert\,\mathtt{B2}'}$	t5) $\dfrac{\mathtt{B1} = t \Rightarrow \mathtt{B1}', \mathtt{B2} = t \Rightarrow \mathtt{B2}'}{\mathtt{B1}\,\vert\,\mathtt{[G]}\,\vert\,\mathtt{B2} = t \Rightarrow \mathtt{B1}'\,\vert\,\mathtt{[G]}\,\vert\,\mathtt{B2}'}$
a5) $\dfrac{\mathtt{B1} - \mathtt{a}^{\tau} \rightarrow \mathtt{B1}', \mathtt{a} \notin \mathtt{G}}{\mathtt{B1}\,\vert\,\mathtt{[G]}\,\vert\,\mathtt{B2} - \mathtt{a}^{\tau} \rightarrow \mathtt{B1}'\,\vert\,\mathtt{[G]}\,\vert\,\mathtt{B2}}$	
a6) symmetric of rule a5)	
a7) $\dfrac{\mathtt{B} - \mathtt{a}^{\tau} \rightarrow \mathtt{B}', \mathtt{t1} \leq \tau \leq \mathtt{t2}}{\mathtt{timer\ a<X>\ in\ B} - \mathtt{a}^{\tau} \rightarrow \mathtt{timer\ a<X>\ in\ B}'}$	t6) $\dfrac{\mathtt{B} = t \Rightarrow \mathtt{B}', age_{\mathtt{a}}(\mathtt{B}) + t \leq \mathtt{t2}}{\mathtt{timer\ a<\ X\ >\ in\ B} = t \Rightarrow \mathtt{timer\ a<\ X\ >\ in\ B}'}$
a8) $\dfrac{\mathtt{B} - \mathtt{b}^{\tau} \rightarrow \mathtt{B}', \mathtt{b} \neq \mathtt{a}}{\mathtt{timer\ a<X>\ in\ B} - \mathtt{b}^{\tau} \rightarrow \mathtt{timer\ a<X>\ in\ B}'}$	
a9) $\dfrac{\mathtt{B} - \mathtt{a}^{\tau} \rightarrow \mathtt{B}', \mathtt{t1} \leq \eta \leq \mathtt{t2}}{\mathtt{m\_timer}^{\eta}\mathtt{a<X>\ in\ B} - \mathtt{a}^{\tau} \rightarrow \mathtt{m\_timer}^{0}\mathtt{a<X>\ in\ B}'}$	t7) $\dfrac{\mathtt{B} = t \Rightarrow \mathtt{B}', \eta + t \leq \mathtt{t2}, age_{\mathtt{a}}(\mathtt{B}) \geq 0}{\mathtt{m\_timer}^{\eta}\mathtt{a<X>\ in\ B} = t \Rightarrow \mathtt{m\_timer}^{\eta+t}\mathtt{a<X>\ in\ B}'}$
a10) $\dfrac{\mathtt{B} - \mathtt{b}^{\tau} \rightarrow \mathtt{B}', \mathtt{b} \neq \mathtt{a}}{\mathtt{m\_timer}^{\eta}\mathtt{a<X>\ in\ B} - \mathtt{b}^{\tau} \rightarrow \mathtt{m\_timer}^{\eta}\mathtt{a<X>\ in\ B}'}$	t8) $\dfrac{\mathtt{B} = t \Rightarrow \mathtt{B}', age_{\mathtt{a}}(\mathtt{B}) = -\infty}{\mathtt{m\_timer}^{\eta}\mathtt{a<X>\ in\ B} = t \Rightarrow \mathtt{m\_timer}^{\eta}\mathtt{a<X>\ in\ B}'}$
a11) $\dfrac{\mathtt{B} - \mathtt{hi}^{\tau} \rightarrow \mathtt{B}', \mathtt{P[h1..hn]:=B}\ \text{is a proc. def.}}{\mathtt{P[g1..gn]} - \mathtt{gi}^{\tau} \rightarrow \mathtt{B}'\mathtt{[g1..gn/h1..hn]}}$	t9) $\dfrac{\mathtt{B} = t \Rightarrow \mathtt{B}', \mathtt{P[H]:=B}\ \text{is a proc. def.}}{\mathtt{P[G]} = t \Rightarrow \mathtt{B}'\mathtt{[G/H]}}$
a12) $\dfrac{\mathtt{B} - \mathtt{a}^{\tau} \rightarrow \mathtt{B}', \mathtt{a} \notin \mathtt{G}}{\mathtt{hide\ G\ in\ B} - \mathtt{a}^{\tau} \rightarrow \mathtt{hide\ G\ in\ B}'}$	t10) $\dfrac{\mathtt{B} = t \Rightarrow \mathtt{B}'}{\mathtt{hide\ G\ in\ B} = t \Rightarrow \mathtt{hide\ G\ in\ B}'}$
a13) $\dfrac{\mathtt{B} - \mathtt{a}^{\tau} \rightarrow \mathtt{B}', \mathtt{a} \in \mathtt{G}}{\mathtt{hide\ G\ in\ B} - \mathtt{i}^{\tau} \rightarrow \mathtt{hide\ G\ in\ B}'}$	
a14) $\dfrac{\mathtt{B} - \mathtt{a}^{\tau} \rightarrow \mathtt{B}', \mathtt{a} \notin \mathtt{H}}{\mathtt{B[G/H]} - \mathtt{a}^{\tau} \rightarrow \mathtt{B}'\mathtt{[G/H]}}$	t11) $\dfrac{\mathtt{B} = t \Rightarrow \mathtt{B}'}{\mathtt{B[G/H]} = t \Rightarrow \mathtt{B}'\mathtt{[G/H]}}$
a15) $\dfrac{\mathtt{B} - \mathtt{hi}^{\tau} \rightarrow \mathtt{B}'}{\mathtt{B[g1..gn/h1..hn]} - \mathtt{gi}^{\tau} \rightarrow \mathtt{B}'\mathtt{[g1..gn/h1..hn]}}$	

- a set W of weights $W = \{w_j | 1 \leq j \leq N\}$, w_j being the weight assigned to transition t_j.

The evolution of the system consists in state changes and in the elapsing of time. Each state change corresponds to a transition and is instantaneous.

When the system is in state s_i, each transition t_j has an associated residual delay r_{ij}. If there exists a transition t_k enabled in state s_i (i.e., $t_k \in E(s_i)$) whose residual delay in state i, r_{ik}, is such that $r_{ik} < r_{ij} \forall j | t_j \in E(s_i), j \neq k$ then t_k occurs when its associated delay r_{ik} has elapsed, and its occurrence determines the next system state $l = N(i, t_j)$.

If two or more residual delays in the set of enabled transitions assume the same minimum value, the priorities and the weights are used to establish which transition occurs, exactly

TABLE IV
THE DEFINITION OF THE FUNCTION $age_a()$

Beh	age_a(Beh)	
a$^{\tau}$;B	τ	
b$^{\tau}$;B	$-\infty$,	$a \neq$ b
B1[]B2	$max\{age_a(B1), age_a(B2)\}$	
B1 | [a] | B2	$min\{age_a(B1), age_a(B2)\}$	
B1 | [b] | B2	$max\{age_a(B1), age_a(B2)\}$,	$a \neq$ b
timer b < X > in B	$age_a(B)$	
m_timer$^{\eta}$ b < X > in B	$age_a(B)$	
P [G]	age_a(B[G/H]),	*given*P[H]:=B
hide G in B	$age_a(B)$,	$a \notin$ G
hide G in B	$-\infty$,	$a \in$ G
B [G/H]	$age_a(B)$,	$a \notin$ H
B [g1..gn/h1..hn]	$age_{hi}(B)$,	$a =$ gi

as priorities and weights are used in ET-LOTOS to determine the next event when two or more events are ready to occur.

The residual delays r_{lj} of the new set of enabled transitions in state l are computed as follows:

- for all the transitions without memory
  - a) $r_{ij} - r_{ik}$, if the transition t_j $(j \neq k)$ was enabled in state s_i, is still enabled in state s_l, and the delay of t_j must not be restarted as a consequence of the firing of t_k. This is the computation of the residual delay before the occurrence of the transition;
  - b) d_j^*, where d_j^* represents a new instance of the random variable d_j, if the transition was disabled in state s_i and is enabled in state s_l, or if $j = k$ and transition t_k is enabled in state s_l, or if the transition was enabled in state s_i, is still enabled in state s_l, and the delay of t_j must be restarted as a consequence of the firing of t_k;
  - c) $-\infty$, if the transition is not enabled in state s_l.
- for all the transitions with memory
  - a) $r_{ij} - r_{ik}$, if the transition t_j $(j \neq k)$ was enabled in state s_i. Again this is the way the residual time before the occurrence of the transition is computed;
  - b) if the transition was disabled in state s_i, it keeps its associated residual delay value;
  - c) d_k^* for the transition t_k which has just occurred, where d_k^* represents a new instance of the random variable d_k.

We can express the semantics of the model in a formal way by using the operational style. The global state of the system is defined by the $(N+1)$-tuple $(s_i, r_{i1}, ..., r_{iN})$, where $s_i \in S$ is a state, and r_{ij} is the actual residual delay of transition t_j in state s_i.

We use the notation $(s_i, r_{i1}, ..., r_{iN}) - t_j \rightarrow (s_l, r_{l1}, ..., r_{lN})$ to express the fact that transition t_j may occur in state $(s_i, r_{i1}, ..., r_{iN})$, leading to the new state $(s_l, r_{l1}, ..., r_{lN})$, and similarly $(s_i, r_{i1}, ..., r_{iN}) = \tau => (s_i, r'_{i1}, ..., r'_{iN})$ to express the fact that a time τ may elapse, so determining the state change from $(s_i, r_{i1}, ..., r_{iN})$ to $(s_i, r'_{i1}, ..., r'_{iN})$.

Then, the semantics of the model can be expressed by the following rules.

1) If there exists a value t such that $t \leq r_{ij}$ $\forall j | t_j \in E(s_i)$, then $(s_i, r_{i1}, ..., r_{iN}) = t => (s_i, r'_{i1}, ..., r'_{iN})$, with

$$r'_{ij} = \begin{cases} r_{ij} - t \text{ if } t_j \in E(s_i) \\ r_{ij} \text{ otherwise} \end{cases} \quad (1)$$

2) If there exists a $t_k \in E(s_i)$ such that $r_{ik} = 0$, then $(s_i, r_{i1}, ..., r_{iN}) - t_k \rightarrow (s_l, r_{l1}, ..., r_{lN})$, with $s_l = N(s_i, t_k)$ and

$$r_{lj} = \begin{cases} r_{ij} & \text{if } t_j \in E(s_i) \cap E(s_l) - Q(s_i, t_k),\ j \neq k \\ -\infty & \text{if } t_j \notin E(s_l) \\ d_j^* & \text{otherwise} \end{cases} \quad (2)$$

for $j | t_j \in U$, and

$$r_{lj} = \begin{cases} d_j^* \text{ if } j = k \\ r_{ij} \text{ otherwise} \end{cases} \quad (3)$$

for $j | t_j \in V$.

If there exist several t_k satisfying the condition expressed in rule 2, first priorities are used, and transitions $t_k \in E_p(s_i)$ are selected, where $E_p(s_i) = \{t_k \in T | (p_k \geq p_j \text{ or } r_{ij} > 0) \forall j \neq k\} \cap E(s_i)$. Finally, if $E_p(s_i)$ still contains more than one transitions, transition t_k is chosen with probability

$$\frac{w_k}{\sum_{x \in E_p(s_i)} w_x} \quad (4)$$

V. MAPPING ET-LOTOS SPECIFICATIONS ONTO PERFORMANCE MODELS

The performance model described above is based on a finite-state transition system. The main requirement for being able to map an ET-LOTOS specification onto it is therefore the existence of a finite transition system equivalent to the specification.

Since LOTOS is characterized by an expressive power which is the same as that of Turing machines, only a restricted class of LOTOS specifications can be modeled by means of a finite state transition system. Unfortunately, membership in this class was proved undecidable in [39]. It is however possible to state a set of sufficient conditions for the existence of a finite transition system equivalent to a given specification. Such conditions were formulated in different ways in [39]–[41]. As we are considering only a subset of LOTOS, the two following conditions are sufficient:

1) Every recursion is guardedly well defined.
2) Every sub-expression of the form A | [G] | B is such that all the derivations of A and B do not contain A | [G] | B as a sub-expression.

In the following we define the mapping from ET-LOTOS specifications to performance models. We omit the proof that observational equivalence is preserved for lack of space.

A. Mapping LOTOS Events onto Performance Model Transitions

In the performance model we have defined, each transition represents a possible event which is distinguished from the

others mainly by the fact that it is aged independently of the others. In order to correctly map LOTOS events onto performance model transitions, it is necessary that two events at the same gate be mapped onto the same transition only if they share the same age, and be mapped onto distinguished transitions if they are aged independently of each other.

In order to apply these concepts, we need to uniquely identify any LOTOS entity which is assigned a distinguishable age. For this purpose, we introduce extended action denotations and extended `m_timer` operators.

First of all, since the various occurrences of action denotations in a LOTOS text are aged independently, we extend each one of them with a different identifier, thus ensuring that each extended action denotation is unique within the LOTOS text. For example, if we use integer indexes as extensions, the expression `(a; b; stop [] b; stop)` becomes

```
(a(1); b(2); stop [] b(3); stop)
```

and the two occurrences of `b` which were undistinguishable are now uniquely identified.

The same can be done for the different occurrences of the `m_timer` operators in the LOTOS text.

Since LOTOS offers the possibility of instantiating more copies of the same process, and the ages of action denotations in the various copies are distinguished, unicity of identifiers within the LOTOS text is not sufficient. We further extend action denotations with a second component which uniquely identifies the parallel control flow to which the action denotation belongs. For example, the expression

```
P[a,b] |[b]| P[a,b] where P[a,b] :=
(a; b; stop)
```

would be expanded as

```
(a(1,0);b(2,0);stop) |[b]|
(a(1,1);b(2,1);stop)
```

where the two control flows introduced by the parallel operator are identified by 0 and 1.

Obviously, a combined use of recursion and parallel operators may generate an infinite number of different extended action denotations, since it may generate an infinite number of parallel control flows. However, this kind of use of recursion is forbidden by the conditions assuring that the LOTOS specification has a finite transition system.

As regards interactions, each one is uniquely identified by the extensions associated with the action denotations offered by the interacting processes. For example, the interaction generated by the above expression can be uniquely identified as `b((2,0),(2,1))`.

From a formal point of view, let us indicate an extended action denotation by $g(i)$, where g is a gate and i is an identifier which uniquely identifies the related event. The semantic rules of the language can be extended accordingly, in order to express precisely how extensions are generated and combined. The new axioms and inference rules are exactly the same as before, with the addition of extensions. We indicate by $\mathtt{B} - \mathtt{a}(i)^{\tau} \rightarrow \mathtt{B}'$ the fact that `B` can generate event $\mathtt{a}^{\tau}$ with extension i and become `B'`. Table V reports the extended semantic rules (aging transitions remain unchanged). The notation $(i1, i2)$ is used to denote an extension which is the concatenation of the two identifiers $i1$, $i2$. Note that the parallel control flows which are active in a system state can be represented as the leaves of a flow tree, and each one is uniquely identified by the concatenation of the binary digits representing the path starting from the tree root and ending in it.

All the action (or interaction) denotations referring to the same `m_timer` operator occurrence have the same extension, as they are indistinguishable according to the ET-LOTOS semantics (they have a common age).

The extended action denotations formally defined in Table V are such that they can be mapped one-to-one onto performance model transitions.

B. Definition of a Labeled Transition System

In the definition of the mapping from ET-LOTOS onto the corresponding performance model we take advantage of the uncoupling of timing information with respect to untimed behavior information, which is enforced by the semantics formulation, composed of two independent sets of transitions.

As regards the untimed behavior part, we first define a labelled transition system (LTS) which encompasses all the information related to this part (action transitions). If we neglect in action transitions (Table V) all the age parameters and related requirements, we get a set of inference rules (that we shall call untimed action transitions) which is substantially the same as that of basic LOTOS. Indeed, in this new system, the `timer` and `m_timer` operators do not affect the set of possible derivations of a behavior expression. The LTS that has to be considered is then essentially the same as the standard one, the only difference being that arcs must be labelled by extended action denotations instead of being labelled by gate identifiers. This is needed in order to be able to properly assign ages when time is superimposed onto the untimed structure.

From a formal point of view, the LTS of a behavior expression B is defined as a 4-tuple $< S, G, T, s_0 >$, where

- S is the set of the LTS states. The states are in one to one correspondence with the behavior expressions derivable from B, according to untimed action transitions.
- G is the LTS alphabet. It is a set of extended action denotations.
- $T = \{(E, g, E') | E - g \rightarrow E', g \in G\}$.
- $s_0 \in S$ is the initial state, labelled B.

Different techniques were proposed to build an LTS from a LOTOS specification, and here we use the constructive technique proposed in [41], which offers some interesting advantages over direct inductive application of inference rules, such as lower computational complexity and applicability to a wider class of specifications. The algorithm for building the LTS of a behavior expression B works as follows:

1) Start with $S = \emptyset$, $G = \emptyset$, $T = \emptyset$, and add $B = s_0$ to S. Note that every state of S is identified here by its associated behavior expression.
2) For every new state B_i just added to S, find the direct derivations of B_i according to untimed action transitions, i.e., the set $H(B_i) = \{(B_i, g, B_i') | B_i - g \rightarrow \cdots\}$

TABLE V
ACTION TRANSITIONS EXTENDED WITH IDENTIFIERS

a1)
$a(i)^{\tau}; B - a(i)^{\tau} \rightarrow B$

a2)
$$\frac{B1 - a(i)^{\tau} \rightarrow B1'}{B1 [] B2 - a(i)^{\tau} \rightarrow B1'}$$

a4)
$$\frac{B1 - a(i1)^{\tau 1} \rightarrow B1', B2 - a(i2)^{\tau 2} \rightarrow B2', a \in G}{B1 |[G]| B2 - a(i1, i2)^{min(\tau 1, \tau 2)} \rightarrow B1' |[G]| B2'}$$

a5)
$$\frac{B1 - a(i)^{\tau} \rightarrow B1', a \notin G}{B1 |[G]| B2 - a(i, 0)^{\tau} \rightarrow B1' |[G]| B2}$$

a6)
$$\frac{B2 - a(i)^{\tau} \rightarrow B2', a \notin G}{B1 |[G]| B2 - a(i, 1)^{\tau} \rightarrow B1 |[G]| B2'}$$

a7)
$$\frac{B - a(i)^{\tau} \rightarrow B', t1 \leq \tau \leq t2}{\texttt{timer a<X> in } B - a(i)^{\tau} \rightarrow \texttt{timer a<X> in } B'}$$

a8)
$$\frac{B - b^{\tau} \rightarrow B', b \neq a}{\texttt{timer a<X> in } B - b^{\tau} \rightarrow \texttt{timer a<X> in } B'}$$

a9)
$$\frac{B - a(i)^{\tau} \rightarrow B', t1 \leq \eta \leq t2}{\texttt{m\_timer}(j)^{\eta}\texttt{a< X > in } B - a(j)^{\tau} \rightarrow \texttt{m\_timer}(j)^{0}\texttt{a<X> in } B'}$$

a10)
$$\frac{B - b(i)^{\tau} \rightarrow B', b \neq a}{\texttt{m\_timer}(j)^{\eta}\texttt{a<X> in } B - b(i)^{\tau} - > \texttt{m\_timer}(j)^{\eta}\texttt{a<X> in } B'}$$

a11)
$$\frac{B - hi(i)^{\tau} \rightarrow B', P[h1..hn] := B \text{ is a proc. def.}}{P[g1..gn] - gi(i)^{\tau} \rightarrow B'[g1..gn/h1..hn]}$$

a12)
$$\frac{B - a(i)^{\tau} \rightarrow B', a \notin G}{\texttt{hide G in } B - a(i)^{\tau} \rightarrow \texttt{hide G in } B'}$$

a13)
$$\frac{B - a(i)^{\tau} \rightarrow B', a \in G}{\texttt{hide G in } B - i^{\tau} \rightarrow \texttt{hide G in } B'}$$

a14)
$$\frac{B - a(i)^{\tau} \rightarrow B', a \notin H}{B[G/H] - a(i)^{\tau} \rightarrow B'[G/H]}$$

a15)
$$\frac{B - hi(i)^{\tau} \rightarrow B'}{B[g1..gn/h1..hn] - gi(i)^{\tau} \rightarrow B'[g1..gn/h1..hn]}$$

where the derivation relation refers to untimed action transitions.

3) For each element (B_i, g, B_i') of $H(B_i)$, add it to T and, if B_i' is not yet included in S, add B_i' to S, and, if g is not yet included in G, add g to G.
4) If in step 3 some new state was added to S, go to step 2, otherwise stop.

The direct derivations $H(B_i)$ of a behavior expression B_i can be computed by recursively applying the inference rules of the language (untimed action transitions).

C. Mapping Definition

Given an ET-LOTOS specification, an LTS can be built from it as indicated in the previous section. Moreover, due to the syntactic requirements imposed on the language, and to the default timing and probabilistic assumptions, each LTS alphabet element has a corresponding unique `timer` or `m_timer` operator. The performance model corresponding to an ET-LOTOS specification is then defined in the following way:

- The performance model states are in one-to-one correspondence with the LTS states. We shall use the notation $\Sigma(B)$ to denote the performance model state corresponding to the LTS state labelled B.
- The performance model transitions are in one-to-one correspondence with the LTS alphabet elements (i.e., extended action denotations). We shall use the notation $T(e)$ to indicate the performance model transition corresponding to the LTS alphabet element e. As each LTS alphabet element has an associated `timer` or `m_timer` operator, the same applies to the performance model transitions. Transitions with memory are those corresponding to `m_timer` operators, while transitions without memory are those corresponding to `timer` operators.
- The performance model random variable d_j is described by the pdf associated with the `timer` or `m_timer` operator which corresponds to transition t_j.
- The priorities and weights assigned to transitions are the values assigned to the corresponding `timer` or `m_timer` operators.
- The mappings E and N are defined by the LTS structure:

$$E(s_i) = \{t_k | \Sigma^{-1}(s_i) - T^{-1}(t_k) \rightarrow B'\} \quad (5)$$

$$N(s_i, t_k) = \Sigma(B') \quad (6)$$

where B' is such that $\Sigma^{-1}(s_i) - T^{-1}(t_k) \rightarrow B'$.
- The mapping $Q(s_i, t_k)$ can be determined considering that a transition t_i belongs to $Q(s_i, t_k)$ iff the following relations hold:

$$\Sigma^{-1}(s_i) - (T^{-1}(t_i))^{\tau} \rightarrow B' \quad (7)$$

$$\Sigma^{-1}(s_i) - (T^{-1}(t_k))^{\eta} \rightarrow B'' - (T^{-1}(t_i))^{0} \rightarrow B''' \quad (8)$$

The first one, states that t_i is enabled in state s_i with an age τ, while the second states that if t_k is executed, the new system state (B'') is such that t_i is again enabled but with null age.

VI. PERFORMANCE MODEL EVALUATION

The description of the dynamic behavior of the performance model into which an ET-LOTOS specification can be mapped provides sufficient details for the implementation of a simulator that can be instrumental for the computation of the performance indexes of interest. Of course, the extensions to the original timer constructs are such that the performance

analysis can be obtained independently of any restriction on the probabilistic characterization of the temporal specification. It may however be interesting to discuss under what conditions an analytical approach to performance evaluation is possible and convenient.

First of all, it is important to note that by an analytical approach we mean the study of the stochastic model generated from the ET-LOTOS specification by numerical methods, since the complexity of even the simplest toy examples immediately rules out the possibility of any closed-form solution.

Obviously, the use of exponential distributions for the characterization of the random variables associated with the action timers allows the mapping of the ET-LOTOS specification onto a continuous-time Markov chain. This opens the possibility of using the numerical tools developed in many years of lively research in the field of efficient numerical techniques for the steady-state solution of Markovian models. The present state of the art in this field is such that models comprising few hundred thousand states can be analyzed with acceptable time and space complexity.

The adoption of probability distributions formed by adequately combining exponential stages also leads to Markovian models, but in this case the number of states of the Markov chain is much larger than the number of states of the ET-LOTOS specification. Similarly, by introducing some (tight) restrictions on the use of general distributions in combination with a majority of exponentially distributed timers would lead to semi-Markov models, that in principle can be analyzed, but at very high cost.

In summary, this means that a numerical approach to the analysis of the performance model derived from an ET-LOTOS specification seems feasible only in the case of exponential timing and "reasonable" state space sizes. In all other instances, simulation is probably more convenient. This statement should not be interpreted as a claim that the simulation of extraordinarily large models is simple, regardless of the timing specifications; on the contrary, the model size makes obtaining *reliable* performance estimates extremely difficult, but no simple alternative is known.

VII. Examples

In this section we present two simple application examples: the first one is a version of the stop-and-wait protocol (the example that is always used in the literature), while the second one describes the behavior of a time-sharing system composed of a host and two terminals. The stop-and-wait protocol model provides an application example of the performance evaluation approach based on a continuous-time Markov chain and makes a model validation possible (by comparison with results found in the literature), while the time sharing system model is presented to describe a possible use of memory timers.

A. A Stop-and-Wait Protocol

We consider a stop-and-wait protocol with one bit frame numbering, and the timeout mechanism at the transmitter. The ET-LOTOS specification of the protocol is reported in Fig. 1.

```
Specification stopwait[tf0,tf1,ra0,ra1,rf0ta0,rf1ta1,
timeout]:noexit

behavior
timer tf0 < 0, infty, exp(13.47), , >,
 tf1 < 0, infty, exp(13.47), , > in
timer timeout < 0, infty, exp(1000), , > in
timer rf0ta0 < 0, infty, exp(120.14), , >,
 rf1ta1 < 0, infty, exp(120.14), , > in
p_timer ra0 < 0, infty, exp(106.7), , >,
 ra1 < 0, infty, exp(106.7), , > in

TX[tf0,tf1,ra0,ra1,timeout]
|[tf0,tf1,ra0,ra1,timeout]|
( TIMER[tf0,tf1,timeout] |[tf0,tf1]|
CH[tf0,tf1,rf0ta0,rf1ta1,ra0,ra1] )

where
process TX[tf0,tf1,ra0,ra1,timeout]:noexit :=
tf0; WA[tf0,tf1,ra0,ra1,timeout]

where
process WA[tf0,tf1,ra0,ra1,timeout]:noexit :=
 timeout; TX[tf0,tf1,ra0,ra1,timeout]
[] ra0; TX[tf1,tf0,ra1,ra0,timeout]
[] ra1; WA[tf0,tf1,ra0,ra1,timeout]
endproc
endproc

process TIMER[start0,start1,timeout]:noexit :=
 start0; SET_TIMER[start0,start1,timeout]
[] start1; SET_TIMER[start0,start1,timeout]

where
process SET_TIMER[start0,start1,timeout]:noexit :=
 start0; SET_TIMER[start0,start1,timeout]
[] start1; SET_TIMER[start0,start1,timeout]
[] timeout; TIMER[start0,start1,timeout]
endproc
endproc

process CH[tf0,tf1,rf0ta0,rf1ta1,ra0,ra1]:noexit:=
CH1[tf0,tf1,rf0ta0,rf1ta1] |[rf0ta0,rf1ta1]|
CH1[rf0ta0,rf1ta1,ra0,ra1]

where
process CH1[t0,t1,r0,r1]:noexit :=
hide ok, err in
timer ok < 0, 0, , , 95> in
timer err < 0, 0, , , 5> in
 t0; (ok; r0; CH1[t0,t1,r0,r1]
 [] err; CH1[t0,t1,r0,r1])
[] t1; (ok; r1; CH1[t0,t1,r0,r1]
 [] err; CH1[t0,t1,r0,r1])
endproc
endproc

endspec
```

Fig. 1. The ET-LOTOS specification of the stop-and-wait protocol.

The protocol behavior is modeled by three parallel processes: a transmitter `TX`, a communication channel `CH`, which acts also as the receiver, and a `TIMER`, which models the timeout mechanism of the protocol.

Gates `tf0` and `tf1` represent the transmission of a frame with sequence number 0 and 1 respectively; gates `rf0ta0` and `rf1ta1` represent the propagation and the reception of a frame, and the corresponding acknowledgment transmission at the receiver; gates `ra0` and `ra1` represent the acknowledgment propagation and reception; gate `timeout` represents the timeout expiration at the transmitter.

The notation `exp(x)` indicates an exponential pdf with mean value `x`, while the notation `infty` indicates the value ∞. Default values of timer parameters are simply indicated by "holes" in the parameter list. For example, gate `tf0` is extended with an exponential pdf with mean value 13.47 and default priority and weight.

Frame transmission is described by the interaction of process `TX` with processes `CH` and `TIMER` at gates `tf0` and `tf1`. After the frame transmission, `TX` instantiates the `WA` (Wait for Acknowledgment) process, which discards the acks referring to out of order frames (i.e. interactions offered at gate `ra0` while waiting for an acknowledgment at gate `ra1` or vice versa), until a correct ack is received or the timeout occurs. If a correct ack is received, process `TX` is restarted, but now gate `tf1` takes the place of gate `tf0` (and vice versa), and the same holds for gates `ra0` and `ra1`, so as to obtain the correct frame numbering. If a timeout occurs (i.e. an interaction at gate `timeout` takes place) before the correct acknowledgment is received, process `TX` is restarted without inverting the gates, since the same frame will be transmitted again.

Process `TIMER` models a timer which is set whenever a frame is transmitted (an interaction at gate `tf0` or gate `tf1` takes place), and expires after an exponentially distributed time, if not reset by a new frame transmission. The choice of an exponential distribution comes only from the need to make a markovian analysis possible. Should performance analysis be done by simulation, any distribution could be used.

Process `CH` is composed of two parallel processes `CH1(0)` and `CH1(1)`, each one representing a unidirectional channel: one conveys frames, while the other conveys acknowledgments. After a frame or ack transmission (`t0` or `t1`), the channel can lose a frame with probability 0.05 (event `err`) or successfully propagate it (event `ok`) and offer it at the corresponding gate (`r0` or `r1`). If an error occurs on the channel, the `timeout` action takes place at the transmitter because no other interaction can occur. The two unidirectional channels are connected in such a way that, if a frame reaches the receiver end, process `CH1(0)` interacts at gate `rf0ta0` with process `CH1(1)`, this event representing both frame reception at one unidirectional channel and ack transmission at the other one.

Note the use of a pre-synchronized timer for gates `ra0` and `ra1` to prevent action `timeout` from taking place before the incorrect pending acknowledgment is discarded. This behavior must be excluded in order to avoid a possible deadlock.

The use of our extension allows a direct specification of the 5% channel error probability, simply using appropriate weights for gates `err` and `ok`, because both gates are extended with deterministic 0 time.

Note that, due to the exponential pdf on gates `rfa0ta0` and `ra0`, a timeout can occur also if the frame and ack transmissions take place without channel error but the process is so slow that the timeout expires. This is a realistic scenario that we can describe using the proposed extension. Moreover, we claim that the use of global timers greatly improves the readability of the specification.

The performance model is composed of 30 states and 68 transitions. The model is not reported here because the diagram is slightly complex. All the probability density functions of the random times are exponential, so as to allow a markovian analysis.

In [17], a model of the stop-and-wait protocol with exponentially distributed times is analyzed. In order to compare the results obtained with our approach with the ones reported in [17], the same numerical values for the protocol parameters must be used. However, as the two models are slightly different, our model parameters do not map one-to-one onto corresponding parameters used in [17], and this difference must be taken into account. In [17] one transition is used to describe the frame transmission and propagation times, and a separate transition is used for the frame reception time (the same is true for the ack); in our model instead we describe as three separate actions

- the frame transmission;
- the frame propagation, reception and ack transmission,
- the ack propagation and reception.

A proper interpretation of the values used in [17] gives for our model the following corresponding values:

- a frame transmission time (gates `tf0` and `tf1`) with mean value equal to 13.47 ms;
- a frame propagation, reception, and ack transmission time (gates `rf0ta0` and `rf1ta1`) with mean value 120.14 ms;
- an ack propagation and reception time (gates `ra0` and `ra1`) with mean 106.7 ms;
- a timeout with mean 1000 ms.

The error probability on the channel is 0.05 in both models.

The Markovian analysis with numerical techniques gives a protocol throughput equal to 2.79 messages/s, quite close to the value 2.75 reported in [17]. The difference in the numerical values is due to the different ways in which the error probability on the channel is described: in [17], an exponential distribution is used to induce the error probability, while we describe it explicitly via a probabilistic choice. Moreover, in our model a more realistic timeout scenario is used: whenever a frame is transmitted, a timeout is set, given that even in the case of a correct transmission the timeout may expire, while in [17] a timeout is set only if an error occurs on the channel.

B. A Time-Sharing System Model

We consider a system consisting of a host that serves two terminals. In order to keep the example really simple, we describe a dumb host, serving a terminal for a fixed time, even if the terminal does not require any service. The ET-LOTOS specification of the model is reported in Fig. 2.

The system behavior is modeled by the two processes named `HOST` and `TERMS`. They are synchronized at gates `t1` and `t2`, each one representing a terminal receiving some service from the host. Gates `start1` and `start2` are used to model the time between successive requests of service of the two terminals (uniformly distributed), while gate `intop` represents internal actions executed by the host at each service cycle. Gate `switch` represents the host switching, and its timing

```
Specification host_term[ t1,t2,start1,start2,intop,switch]:
noexit

Behavior

m_timer t1 < 0, 100, , , >, t2 < 0, 2, , , > in
timer intop < 0, 200, , , > in
timer switch < 20, 20, , , ,> in
timer start1 < 0, infty, exp(10), , >,
 start2 < 0, infty, exp(10), , > in

HOST[t1,t2,intop,switch] |[t1,t2]| TERMS[t1,t2,start1,start2]

where

process TERMS[t1,t2,start1,start2]:noexit := TERM[t1,start1]
||| TERM[t2,start2]
endproc

process TERM[t,start]:noexit :=
start; BUSY[t,start]
endproc

process BUSY[t,start]:noexit :=
t; BUSY[t,start] [] t; TERM[t,start]
endproc

process HOST[t1,t2,intop,switch]:noexit :=
intop; PHASE1[t1,t2,intop,switch]
endproc

process PHASE1[t1,t2,intop,switch]:exit :=
 t1; PHASE2[t1,t2,intop,switch]
[] switch; PHASE2[t1,t2,intop,switch]
endproc

process PHASE2[t1,t2,intop,switch]:exit :=
 t2; HOST[t1,t2,intop,switch]
[] switch; HOST[t1,t2,intop,switch]
endproc

endspec
```

Fig. 2. The ET-LOTOS specification of the time-sharing system.

describes the fixed time slice of 20 time units, assigned to each terminal by the host.

The TERMS process consists of two interleaved TERM processes, each one describing one terminal. Each terminal executes internal operations before asking service (event start); then it receives service, remaining in the BUSY state until the service is provided by the HOST process via an interaction at gate t1 (t2). Finally, the terminal can either immediately issue a new service request, or go back to the initial state.

The HOST process first executes internal operations (gate intop), then it provides a service to the first terminal, activating process PHASE1. The interaction at gate t1 occurs if the first terminal is busy and if the required service is shorter than the host switching time. Otherwise, after 20 time units the host switches (event switch) and activates process PHASE2, to provide service to the second terminal. Finally, the service cycle terminates and process HOST is activated again.

We give in Fig. 3 the diagram of the performance model derived from the specification. Transitions are labeled with the corresponding gates, while priorities and weights are not specified, because default values are used. The states are numbered from 1 to 12 (1 is the starting state). Gate identifiers t1 and t2 had to be extended as described in Section V-A.

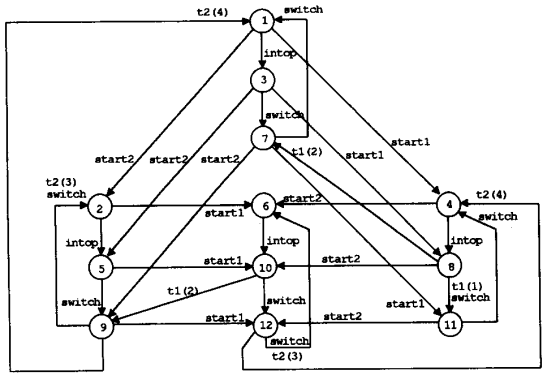


Fig. 3. The model for the time-sharing system example.

Four host "cycles" are easily identifiable in the diagram: we have four cycles because the two terminals can be in four different states while the host is working (no terminal requiring service, terminal 1 or 2 requiring service, both terminals requiring service). For example, cycle 1-3-7 describes the host working while no terminal is requiring service, whereas cycle 2-5-9 refers to a situation where terminal 2 is requiring service and so on. The transitions between these "groups" of states are caused by a new terminal requiring service (i.e. actions start1 and start2) or a terminal service completion (i.e. actions t1 and t2), when the terminal does not immediately require a new service.

The most interesting aspect of this example is the use of memory timers on gates t1 and t2. The memory timers ensure that, if the requested service cannot be completely satisfied within a service time, the work already done by the host is not lost; the next time the same terminal is being serviced, only the residual work is considered. This kind of behavior is very common and the memory timer extension helps in maintaining the LOTOS specification compact and readable.

VIII. Conclusions

"Extended Timed LOTOS" (ET-LOTOS) is a new extension of LOTOS which incorporates both timing and probabilistic specifications. ET-LOTOS maintains the same formal structure of LOTOS, and can therefore be similarly used to formally verify distributed systems, including those with time-critical features. The extension is downward compatible, in the sense that by neglecting extensions one gets a standard LOTOS specification describing the system features not related to time or probability. ET-LOTOS can be mapped onto a performance model which preserves the same observational behavior and is open to direct application of different performance evaluation techniques. The choice about which specific technique it is possible and convenient to use depends on the kind of timing and probabilistic characterizations used in the specification. The extensions introduced in the language enable the user to properly and easily specify the most common scenarios in distributed systems, including some which were not provided for in previous timed extensions of LOTOS.

REFERENCES

[1] CCITT/SGX/WP3-1, "Specification and Description Language SDL," *CCITT Recommendations Z.100-Z.104*, 1988.

[2] IS 8807: *Information Processing Systems, Open Systems Interconnection, LOTOS - A Formal Description Technique Based on the Temporal Ordering of Observational Behaviour*, ISO, 1989.

[3] ISO, *IS 9074, Information Processing Systems - Open Systems Interconnection: Estelle: a Formal Description technique Based on an Extended Finite State Transition Model*, ISO, 1989.

[4] G. Von Bochmann and J. Vaucher, "Adding Performance Aspects to Specification Languages," in *Protocol Specification, Testing and Verification VIII*, S. Aggarwal and K. Sabnani, Eds. New York: Elsevier Science, 1988.

[5] J. Quemada, and A. Fernandez, "Introduction of quantitative relative time into LOTOS," in *Protocol Specification, Testing, and Verification VII*, H. Rudin, and C. H. West, Eds. New York: Elsevier Science, 1987.

[6] G. M. Reed, and A. W. Roscoe, "A timed model for communicating sequential processes," *Proc. 13th ICALP*, LNCS 226, Springer-Verlag, 1986, pp. 314-321.

[7] R. Gerth, and A. Boucher, "A timed failures model for extended communicating processes," *Proc. 14th ICALP*, LNCS 267, Springer-Verlag, 1987, pp. 95-114.

[8] F. Moller, and C. Tofts, "A temporal calculus of communicating systems," *Lecture Notes in Computer Science 458*, Springer-Verlag, 1990, pp. 401-415.

[9] W. Yi, "Real-time behaviour of asynchronous agents," *Lecture Notes in Computer Science 458*, Springer-Verlag, 1990, pp. 502-520.

[10] N. F. Maxemchuk, and K. Sabnani, "Probabilistic verification of communication protocols," in *Protocol Specification, Testing, and Verification VII*, H. Rudin, and C.H. West, Eds. New York: Elsevier Science, 1987.

[11] D. D. Dimitrijevic, and M. S. Chen, "An integrated algorithm for probabilistic protocol verification and evaluation," in *Proc. IEEE INFOCOM'89*, Ottawa, Ont., Canada, Apr. 1989.

[12] A. Giacalone, C. Jou, and S. A. Smolka, "Algebraic reasoning for probabilistic concurrent systems," in *Proc. IFIP TC2 Working Conf. Programming Concepts and Methods*, 1989.

[13] K.J. Larsen and A. Skou, "Bisimulation through probabilistic testing," in *Proc. 16th ACM Symp. Principles Programming Languages*, 1989.

[14] R. Van Glabbeek, S. A. Smolka, B. Steffen, and C. M. N. Tofts, "Reactive, generative, and stratified models of probabilistic processes," in *Proc. 5th IEEE Int. Symp. Logic Comput. Sci.*, 1990.

[15] F. J. W. Symons, "Introduction to numerical petri nets, a general graphical model of concurrent processing systems," *Australian Telecommun. Res.*, vol. 14, no. 1, pp. 28-33, Jan. 1980.

[16] G. Florin and S. Natkin, "Les Reseaux de Petri Stochastiques," *Technique et Science Informatiques*, vol. 4, no. 1, Feb. 1985.

[17] M. K. Molloy, "Performance analysis using stochastic Petri Nets," *IEEE Trans. Comput.*, vol. 31, pp. 913-917, Sept. 1982.

[18] M. Ajmone Marsan, G. Balbo, and G. Conte, "A class of generalized stochastic Petri Nets for the performance analysis of multiprocessor systems," *ACM Trans. Comput. Syst.*, vol. 2, May 1984.

[19] M. Ajmone Marsan, G. Balbo, G. Chiola, and G. Conte, "Generalized stochastic Petri Nets revisited: Random switches and priorities," in *Proc. Int. Workshop Petri Nets Perform. Models*, IEEE-CS Press, Madison, WI, pp. 44-53, Aug. 1987.

[20] M. Ajmone Marsan, G. Balbo, A. Bobbio, G. Chiola, G. Conte, and A. Cumani, "The effect of execution policies on the semantics and analysis of stochastic Petri Nets," *IEEE Trans. Software Eng.*, vol. 15, pp. 832–846, July 1989.

[21] J.B. Dugan, K.S. Trivedi, R.M. Geist, and V.F. Nicola, "Extended Stochastic Petri Nets: Applications and analysis," in *Proc. PERFORMANCE '84*, Paris, France, Dec. 1984.

[22] M. Ajmone Marsan and G. Chiola, "On Petri Nets with deterministic and exponentially distributed firing times," in *Advances in Petri Nets '87* G. Rozenberg, Ed. New York: Springer-Verlag, LNCS, vol. 266, 1987, pp. 132-145.

[23] J. F. Meyer, A. Movaghar, and W. H. Sanders, "Stochastic activity networks: Structure, behavior, and application," in *Proc. Int. Workshop Timed Petri Nets*, IEEE-CS Press, Torino, Italy, July, 1985.

[24] R. R. Razouk, and C. V. Phelps, "Performance analysis using timed Petri Nets," in *Proc. Int. Conf. Parallel Processing*, Aug. 1984, pp. 126-129.

[25] M. A. Holliday, and M. K. Vernon, "A generalized timed Petri Net model for performance analysis," in *Proc. Int. Workshop Timed Petri Nets*, IEEE-CS Press, Torino, Italy, July, 1985.

[26] W. M. Zuberek, "M-timed Petri Nets, priorities, preemptions, and performance evaluation of systems," in *Advances in Petri Nets '85*, G.Rozenberg, Ed. New York: LNCS no. 222, Springer Verlag, 1986.

[27] H. Rudin, "An improved algorithm for estimating protocol performance," in *Protocol Specification, Testing and Verification IV*, Y. Yemini, R. Storm, and S. Yemini, Eds. New York: Elsevier Science, 1985.

[28] P. S. Kritzinger, "Analyzing the time efficiency of a communication protocol," in *Protocol Specification, Testing and Verification IV*, Y. Yemini, R. Storm, and S. Yemini, Eds. New York: Elsevier Science, 1985.

[29] P. S. Kritzinger, "A performance model of the OSI communication architecture," *IEEE Trans. Commun.*, vol. COM-34, June 1986.

[30] F. J. Lin, and M. T. Liu, "An integrated approach to verification and performance analysis of communication protocols," in *Protocol Specification, Testing and Verification VIII*, S. Aggarwal and K. Sabnani, Eds. New York: Elsevier Science, 1988.

[31] N. Nounou, and Y. Yemini, "Algebraic specification-based performance analysis of communication protocols," in *Protocol Specification, Testing and Verification IV*, Y. Yemini, R. Storm, and S. Yemini, Eds. New York: Elsevier Science, 1985.

[32] H. Hansson, and B. Jonsson, "A calculus for communicating systems with time and probabilities," in *Proc. IEEE 11th Real-Time Systems Symp.*, 1990.

[33] N. Rico, and G. Von Bochmann, "Performance description and analysis for distributed systems using a variant of LOTOS," in *Protocol Specification, Testing and Verification XI*, B. Jonsson, J. Parrow and B. Pehrson, Eds. New York: Elsevier Science, 1991.

[34] J. Quemada, A. Azcorra, and D. Frutos, "A timed calculus for LOTOS," in *Proc. FORTE'89 2nd Int. Conf. Formal Description Techniques Distrib. Syst. Commun. Protocols*, Vancouver, B.C., Canada, Dec. 1989.

[35] T. Bolognesi, F. Lucidi, and S. Trigila, "From timed Petri Nets to timed LOTOS," in *Protocol Specification, Testing and Verification X*, L. Logrippo, R.L. Probert, H. Ural, Eds. New York: Elsevier Science, 1990.

[36] ——, "New Proposals for Timed-Interaction LOTOS," Fondazione Ugo Bordoni, Italy, Rep. FUB: 5B5590, 1990.

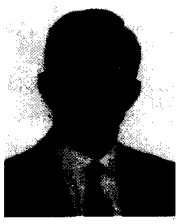
[37] T. Bolognesi and F. Lucidi, "LOTOS-like process algebras with urgent or timed interactions," in *Proc. of FORTE'91: 4th Int. Conf. Formal Dscription Techniques*, K. Parker and G. Rose, Eds. New York: Elsevier Science, 1992.

[38] T. Bolognesi and E. Brinksma, "Introduction to the ISO specification language LOTOS," *Comput. Networks ISDN Syst.*, vol.14, pp. 25-59, 1987.

[39] A. Fantechi, S. Gnesi, and G. Mazzarini, "How much expressive are LOTOS behaviour expressions?," in *Proc. of FORTE'90: 3rd Int. Conf. on Formal Dscription Techniques*, J. Quemada, J. Manas, and E. Vasquez, Eds. New York: Elsevier Science, 1991.

[40] H. Garavel and E. Najm, "TILT: From LOTOS to labelled transition systems," in *The formal description technique LOTOS*, P. H. J. Van Eijk, C. A. Vissers, and M. Diaz, Eds. New York: Elsevier Science, 1989.

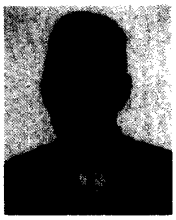
[41] A. Valenzano, R. Sisto, and L. Ciminiera, "An Abstract Execution Model for Basic LOTOS," *IEE Software Eng. J.*, vol. 5, no. 6, pp. 311-318, 1990.



Marco Ajmone Marsan (SM'86) was born in Torino, Italy, in 1951. He holds a Dr. Ing. degree in Electronic Engineering from Politecnico di Torino, and a Master of Science from the University of California, Los Angeles.

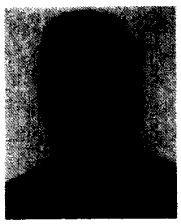
He is a Full Professor at the Electronics Department of Politecnico di Torino, in Italy. From November 1975 to October 1987 he was at the Electronics Department of Politecnico di Torino, first as a Researcher, then as an Associate Professor. From November 1987 to October 1990 he was a Full Professor at the Computer Science Department of the University of Milan, in Italy. During the summers of 1980 and 1981 he was with the Research in Distributed Processing Group, Computer Science Department, UCLA. His current interests are in the fields of performance evaluation of communication networks and computer systems, Petri nets and queueing theory.

Dr. Ajmone Marsan has coauthored about 150 journal and conference papers in the areas of Communications and Computer Science, as well as the book *Performance Models of Multiprocessor Systems* (Cambridge, MA, M.I.T. Press). He received the best paper award at the Third International Conference on Distributed Computing Systems in Miami, FL, in 1982. His e-mail address is ajmone@polito.it.



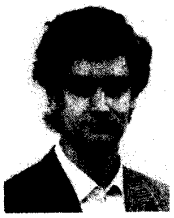
Andrea Bianco is a Ph.D. student in the Electronics Department at Politecnico di Torino, Italy. He received the B.S. degree in 1986 from Politecnico di Torino.

In 1993 he spent a one-year at the Hewlett-Packard Laboratories, Palo Alto, CA, researching ATM congestion control issues. His research interests are in the fields of all-optical networks, ATM congestion control, and formal description techniques. His e-mail address is bianco@polito.it.



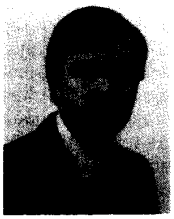
Luigi Ciminiera (M'80) received the Electronic Engineering degree from the Politecnico di Torino, Italy, in 1977.

He is currently Professor of Computer Engineering at Politecnico di Torino. Formerly, he has also been, in different positions, with the University of Bari and L'Aquila, Italy. He authored several technical papers and co-authored two books on microprocessors and industrial computer networks. His research interests include computer arithmetic, computer graphics and computer networks and protocols. His e-mail address is ciminiera@polito.it



Riccardo Sisto received the Electronic Engineering degree in 1987, and received the Ph.D degree in computer engineering in 1992, both from Politecnico di Torino, Torino, Italy.

Since 1991 he has been working at Politecnico di Torino, in the Computer Science Department. His current research interests are in the areas of computer networks, communication protocol engineering, and formal description techniques for parallel and distributed systems. His e-mail address is sisto@polito.it.



Adriano Valenzano was born in Turin, Italy, on August 11, 1955. He received in Electronic Engineering degree from the Polytechnic of Turin in 1980.

From 1980 to 1983 he joined the Department of Computer Engineering of the Polytechnic of Turin, where he was engaged in research on parallel processing, special purpose parallel processors and microprocessor-based local area networks under the Computer Science Program of the Italian National Research Council (CNR). Since 1983 he has been a Researcher at the Italian National Research Council and in 1991 be became Director of Research. He is currently with Centro Elaborazione Numerale dei Segnali (CENS) where he is responsible for researches concerning distributed systems, local area networks and communication protocols. His current research interests are in the areas of computer networks, formal description techniques, communication protocols and distributed systems.

Dr. Valenzano has authored more than 80 technical papers and co-authored two books on *Advanced Microprocessor Architectures* and *MAP and TOP Communications: Standards and Applications*. His e-mail address is valenzano@polito.it.