Two Exercises with EMPA: Computing the Utilization of the CSMA/CD Protocol and Assessing the Performability of a Queueing System

Marco Bernardo

Università di Bologna, Dipartimento di Scienze dell'Informazione
Mura Anteo Zamboni 7, 40127 Bologna, Italy
E-mail: bernardo@cs.unibo.it

Abstract

We present two applications of the stochastically timed process algebra EMPA. The first one is concerned with the compositional modeling of the CSMA/CD protocol and the determination of its utilization. The second one is concerned with the description of a queueing system representing a computing center where failures and repairs can occur, and the compositional assessment of its performability. In both cases, the technique of rewards is used to determine performance measures.

1 Compositional performance modeling

The need of integrating the performance modeling and analysis of a concurrent system into the design process of the system itself has been widely recognized (see, e.g., [27, 7, 12, 4]) and stimulated a considerable research effort. The main problem to be tackled is that it often happens that a concurrent system is first fully designed and tested for functionality, and afterwards tested for efficiency. As a consequence, if the performance is detected to be poor, the concurrent system has to be designed again, thereby negatively affecting both the design costs and the delivery at a fixed deadline. Another relevant drawback is that tests for functionality and performance are usually carried out on two different models of the system, so one has to make sure that the functional model and the performance model are consistent, i.e. they really describe (different aspects of) the same system.

An emerging field in the area of the integration of formal methods and performance evaluation is the field of *stochastically timed process algebras* (see e.g. [8, 13, 2, 6, 5, 22]). The main reason for the adoption of stochastically timed process algebras is compositionality. More accurately, they allow for (*i*) *compositional model construction* thanks to their operators whereby it is possible to build complex system descriptions from smaller ones, (*ii*) *compositional model simplification* by means of appropriate congruences defined over their sets of terms [9, 13, 3, 6, 10, 23], and (*iii*) *compositional model solution* whenever special product form conditions are met [11, 24]. Another strength of stochastically timed process algebras (in particular with respect to formalisms such as stochastic Petri nets) is that they provide system descriptions that can be analyzed *as a whole* without building the underlying *(consistent)* functional and performance models, by virtue of the definition of notions of equivalence which relate algebraic descriptions of systems having the same functional and performance properties.

In this paper we shall focus our attention on the stochastically timed process algebra *EMPA (Extended Markovian Process Algebra)* [2, 3]. EMPA was originally developed in order to implement the integrated approach for modeling and analyzing functional and performance properties of concurrent systems proposed in [2]. However, as it has been recognized in [3], the expressive power of EMPA is considerable because it allows nondeterminism, priorities, probabilities and time to be modeled. The purpose of this paper is to exhibit such an expressive power by means of two examples. In Section 2 we present a brief overview of

EMPA that is necessary in order to understand the algebraic description of the CSMA/CD protocol given in Section 3 as well as the performability model of a queueing system given in Section 4. In both examples, some performance measures are specified and computed by resorting to the technique of rewards.

2 EMPA: syntax, semantics, equivalence

2.1 Syntax of EMPA terms

The building blocks of EMPA are actions. Each *action* is a pair $<a, \tilde{\lambda}>$ consisting of the *type* of the action and the *rate* of the action. Actions are divided into *external* and *internal* (τ) according to types, while they are classified as exponentially timed, immediate or passive according to rates:

- *Exponentially timed actions* are actions whose rate λ is a positive real number that uniquely identifies the exponential probability distribution function $F_X(t) = 1 - e^{-\lambda \cdot t}$ specifying the random variable X (with mean $1/\lambda$) expressing the duration of the action.
- *Immediate actions* are actions whose rate $\infty_{l,w}$ is infinite. Such actions have duration zero, and each of them is given a priority level $l \in \mathbf{N}_+$ and a weight $w \in \mathbf{R}_+$.
- *Passive actions* are actions whose rate denoted by $*$ is undefined. The duration of a passive action is fixed only by synchronizing it with an active action of the same type.

The classification of actions based on their rates implies that: (i) exponentially timed actions model activities that are relevant from the performance point of view, (ii) immediate actions model logical events as well as activities that are either irrelevant from the performance point of view or unboundedly faster than the others, and are useful to express prioritized and probabilistic choices, (iii) passive actions model activities waiting for the synchronization with timed activities, and are useful to express nondeterministic choices.

We denote the set of actions by $Act = AType \times ARate$ where $AType$ is the set of types and $ARate = \mathbf{R}_+ \cup Inf \cup \{*\}$, with $Inf = \{\infty_{l,w} \mid l \in \mathbf{N}_+ \wedge w \in \mathbf{R}_+\}$, is the set of rates. We use $a, b, c, \ldots$ as metavariables for $AType$, $\tilde{\lambda}, \tilde{\mu}, \tilde{\gamma}, \ldots$ for $ARate$, and $\lambda, \mu, \gamma, \ldots$ for $\mathbf{R}_+$. Finally, we denote by $APLev = \{-1\} \cup \mathbf{N}$ the set of *action priority levels*, and we assume $* < \lambda < \infty_{l,w}$.

Let $Const$ be a set of *constants*, ranged over by $A, B, C, \ldots$, and let $ARFun = \{\varphi : AType \longrightarrow AType \mid \varphi(\tau) = \tau \wedge \varphi(AType - \{\tau\}) \subseteq AType - \{\tau\}\}$ be a set of *action relabeling functions*.

Definition 2.1 The set $\mathcal{L}$ of *process terms* of EMPA is generated by the following syntax

$$E ::= \underline{0} \mid <a, \tilde{\lambda}>.E \mid E/L \mid E[\varphi] \mid E + E \mid E \parallel_S E \mid A$$

where $L, S \subseteq AType - \{\tau\}$. The set $\mathcal{L}$ will be ranged over by $E, F, G, \ldots$. We denote by $\mathcal{G}$ the set of guarded and closed terms of $\mathcal{L}$. ■

The *null term* "$\underline{0}$" represents a termination or deadlocked state.

The *prefix operator* "$<a, \tilde{\lambda}>._$" represents the sequential composition of an action and a term; so, term $<a, \tilde{\lambda}>.E$ can execute action $<a, \tilde{\lambda}>$ and then behaves as term E.

The *functional abstraction operator* "$\_/L$" abstracts from the type of the actions (executed by the term to which it is applied) whenever it is in L, i.e. the action type is turned into τ. The meaning of this operator is the same as that of the hiding operator of CSP [14].

The *functional relabeling operator* "$\_[\varphi]$" changes the type of the actions (executed by the term to which it is applied) according to φ. The meaning of this operator is the same as that of the relabeling operator of CCS [19].

The *alternative composition operator* "$\_ + \_$" expresses a choice between two terms. The choice is solved according to durations in the case of exponentially timed actions *(race policy)* and according to priorities and weights in the case of immediate actions *(preselection policy)*, while it is purely *nondeterministic* in the case of passive actions.

The *parallel composition operator* "$\_ \|_S \_$" is based on two synchronization disciplines. The synchronization discipline on action types is the same as that of CSP [14], hence two actions can synchronize only if they have the same type, and this coincides with the resulting type. The synchronization discipline on action rates states that action $<a, \tilde{\lambda}>$ can be synchronized with action $<a, \tilde{\mu}>$ only if $\min(\tilde{\lambda}, \tilde{\mu}) = *$, and the resulting rate is given by $\max(\tilde{\lambda}, \tilde{\mu})$ up to normalization. In other words, in a synchronization at most one active action can be involved and its rate determines the rate of the synchronization itself, up to normalization.

2.2 Semantics of EMPA terms

The main problem to tackle when defining the semantics for EMPA is that the actions executable by a given term may have different priority levels, and only those having the highest priority level are actually executable. Let us call *potential move* of a given term a pair composed of (i) an action executable by the term, and (ii) a derivative term obtained by executing that action. To solve the problem above, we compute inductively the multiset [1] of the potential moves of a given term regardless of priority levels, and then we select those having the highest priority level.

The formal definition of the integrated interleaving semantics for EMPA is based on the transition relation $\longrightarrow$, which is the least subset of $\mathcal{G} \times Act \times \mathcal{G}$ satisfying the inference rule reported in the first part of Table 1. This rule selects the potential moves having the highest priority level, and then merges together those having the same action type, the same priority level and the same derivative term. The first operation is carried out through functions $Select : \mathcal{M}u_{fin}(Act \times \mathcal{G}) \longrightarrow \mathcal{M}u_{fin}(Act \times \mathcal{G})$ and $PL : Act \longrightarrow APLev$, which are defined in the third part of Table 1. The second operation is carried out through function $Melt : \mathcal{M}u_{fin}(Act \times \mathcal{G}) \longrightarrow \mathcal{P}_{fin}(Act \times \mathcal{G})$ and partial function $Min : (ARate \times ARate) \multimap ARate$, which are defined in the fourth part of Table 1. The name *Min* should recall the adoption of the race policy: the minimum of a set of random variables has to be computed. We regard *Min* as an associative and commutative operation, thus we take the liberty to apply it to multisets of rates.

Example 2.2 If we consider term

$$E \equiv <a, \lambda>.F + <a, \lambda>.F$$

then we have two identical potential moves $(<a, \lambda>, F)$ which are merged into $(<a, 2 \cdot \lambda>, F)$ by means of *Melt* and *Min*. Note that rates of exponentially timed moves are summed up because of the adoption of the race policy: the minimum of several independent exponentially distributed random variables is an exponentially distributed random variable whose rate is the sum of the original rates. ■

The multiset $PM(E) \in \mathcal{M}u_{fin}(Act \times \mathcal{G})$ of potential moves of $E \in \mathcal{G}$ is defined by structural induction in the second part of Table 1. The normalization of the rates of potential moves resulting from the synchronization of the same active action with several independent or alternative passive actions is carried out through partial function $Norm : (AType \times ARate \times ARate \times \mathcal{M}u_{fin}(Act \times \mathcal{G}) \times \mathcal{M}u_{fin}(Act \times \mathcal{G})) \multimap ARate$ and function $Split : (ARate \times \mathbb{R}_{]0,1]}) \longrightarrow ARate$, which are defined in the fifth part of Table 1. Note that $Norm(a, \tilde{\lambda}_1, \tilde{\lambda}_2, PM_1, PM_2)$ is defined if and only if $\min(\tilde{\lambda}, \tilde{\mu}) = *$, which is the condition on action rates we have required in Section 2.1 in order for a synchronization to be permitted.

Example 2.3 Consider terms

$$\begin{aligned} E_1 &\equiv <a, \lambda>.\underline{0} \|_{\{a\}} (<a, *>.\underline{0} \|_{\emptyset} <a, *>.\underline{0}) \\ E_2 &\equiv <a, \lambda>.\underline{0} \|_{\{a\}} (<a, *>.\underline{0} + <a, *>.\underline{0}) \end{aligned}$$

In both cases, the left-hand operand of "$\|_{\{a\}}$" has one potential move $(<a, \lambda>, \underline{0})$ and the right-hand operand has two potential moves whose action is $<a, *>$, hence the whole term has two potential moves whose type is a. Since both terms consist of a single active action which is exponentially timed with rate λ, the rate of each of the two potential moves cannot be λ otherwise the mean sojourn time of the states corresponding to

[1] We use "$\{\!|$" and "$|\!\}$" as brackets for multisets, "$\_ \oplus \_$" to denote multiset union, $\mathcal{M}u_{fin}(S)$ ($\mathcal{P}_{fin}(S)$) to denote the collection of finite multisets (sets) over set S, $M(s)$ to denote the multiplicity of element s in multiset M, and $\pi_i(M)$ to denote the multiset obtained by projecting the tuples in multiset M on their i-th component. Thus, e.g., $(\pi_1(PM_2))(<a, *>)$ in the fifth part of Table 1 denotes the multiplicity of tuples of PM_2 whose first component is $<a, *>$.

$$\frac{(<a,\tilde{\lambda}>,E') \in Melt(Select(PM(E)))}{E \xrightarrow{a,\tilde{\lambda}} E'}$$

$$PM(\underline{0}) = \emptyset$$

$$PM(<a,\tilde{\lambda}>.E) = \{\!|\, (<a,\tilde{\lambda}>,E) \,|\!\}$$

$$PM(E/L) = \{\!|\, (<a,\tilde{\lambda}>,E'/L) \mid (<a,\tilde{\lambda}>,E') \in PM(E) \wedge a \notin L \,|\!\} \oplus \{\!|\, (<\tau,\tilde{\lambda}>,E'/L) \mid (<a,\tilde{\lambda}>,E') \in PM(E) \wedge a \in L \,|\!\}$$

$$PM(E[\varphi]) = \{\!|\, (<\varphi(a),\tilde{\lambda}>,E'[\varphi]) \mid (<a,\tilde{\lambda}>,E') \in PM(E) \,|\!\}$$

$$PM(E_1 + E_2) = PM(E_1) \oplus PM(E_2)$$

$$PM(E_1 \parallel_S E_2) = \{\!|\, (<a,\tilde{\lambda}>,E_1' \parallel_S E_2) \mid a \notin S \wedge (<a,\tilde{\lambda}>,E_1') \in PM(E_1) \,|\!\} \oplus \{\!|\, (<a,\tilde{\lambda}>,E_1 \parallel_S E_2') \mid a \notin S \wedge (<a,\tilde{\lambda}>,E_2') \in PM(E_2) \,|\!\} \oplus \{\!|\, (<a,\tilde{\gamma}>,E_1' \parallel_S E_2') \mid a \in S \wedge (<a,\tilde{\lambda}_1>,E_1') \in PM(E_1) \wedge (<a,\tilde{\lambda}_2>,E_2') \in PM(E_2) \wedge \tilde{\gamma} = Norm(a,\tilde{\lambda}_1,\tilde{\lambda}_2,PM(E_1),PM(E_2)) \,|\!\}$$

$$PM(A) = PM(E) \quad \text{if } A \triangleq E$$

$$Select(PM) = \{\!|\, (<a,\tilde{\lambda}>,E) \in PM \mid PL(<a,\tilde{\lambda}>) = -1 \vee \forall (<b,\tilde{\mu}>,E') \in PM.\, PL(<a,\tilde{\lambda}>) \geq PL(<b,\tilde{\mu}>) \,|\!\}$$

$$PL(<a,*>) = -1 \qquad PL(<a,\lambda>) = 0 \qquad PL(<a,\infty_{l,w}>) = l$$

$$Melt(PM) = \{(<a,\tilde{\lambda}>,E) \mid (<a,\tilde{\mu}>,E) \in PM \wedge \tilde{\lambda} = Min\{\!|\, \tilde{\gamma} \mid (<a,\tilde{\gamma}>,E) \in PM \wedge PL(<a,\tilde{\gamma}>) = PL(<a,\tilde{\mu}>) \,|\!\}\}$$

$$* \; Min \; * = * \qquad \lambda_1 \; Min \; \lambda_2 = \lambda_1 + \lambda_2 \qquad \infty_{l,w_1} \; Min \; \infty_{l,w_2} = \infty_{l,w_1+w_2}$$

$$Norm(a,\tilde{\lambda}_1,\tilde{\lambda}_2,PM_1,PM_2) = \begin{cases} Split(\tilde{\lambda}_1, 1/(\pi_1(PM_2))(<a,*>)) & \text{if } \tilde{\lambda}_2 = * \\ Split(\tilde{\lambda}_2, 1/(\pi_1(PM_1))(<a,*>)) & \text{if } \tilde{\lambda}_1 = * \end{cases}$$

$$Split(*,\alpha) = * \qquad Split(\lambda,\alpha) = \lambda \cdot \alpha \qquad Split(\infty_{l,w},\alpha) = \infty_{l,w\cdot\alpha}$$

Table 1: Inductive rules for EMPA integrated interleaving semantics

E_1 and E_2 would be $1/(2 \cdot \lambda)$ instead of $1/\lambda$: a normalization must take place so that the sum of the rates of the two potential moves turns out to be λ. Assuming that independent or alternative passive actions have the same execution probability when they are involved in a synchronization, *Norm* computes the rate of each of the potential moves above by dividing λ by the number of independent or alternative passive actions with which the synchronization can take place. As a consequence, the rate of each of the two potential moves is $\lambda/2$. ■

Definition 2.4 The *integrated interleaving semantics* of $E \in \mathcal{G}$ is the labeled transition system (LTS) $\mathcal{I}[\![E]\!] = (\uparrow E, Act, \longrightarrow_E, E)$ where $\uparrow E$ is the set of states reachable from E, and $\longrightarrow_E$ is $\longrightarrow$ restricted to $\uparrow E \times Act \times \uparrow E$. ■

Definition 2.5 $E \in \mathcal{G}$ is *performance closed* if and only if $\mathcal{I}[\![E]\!]$ does not contain passive transitions. We denote by $\mathcal{E}$ the set of performance closed terms of $\mathcal{G}$. ■

Given a term $E \in \mathcal{G}$, its integrated interleaving semantics $\mathcal{I}[\![E]\!]$ fully represents the behavior of E because transitions are decorated by both the action type and the action rate. One can think of obtaining the *functional semantics* $\mathcal{F}[\![E]\!]$ and the *performance semantics* $\mathcal{P}[\![E]\!]$ of E from $\mathcal{I}[\![E]\!]$ by simply dropping action rates and action types, respectively. As a matter of fact, this is the case for the functional semantics.

Definition 2.6 The *functional semantics* of $E \in \mathcal{G}$ is the LTS $\mathcal{F}[\![E]\!] = (\uparrow E, AType, \longrightarrow_{E,\mathcal{F}}, E)$ where $\longrightarrow_{E,\mathcal{F}}$ is $\longrightarrow_E$ restricted to $\uparrow E \times AType \times \uparrow E$. ■

The definition of the performance semantics requires instead a more careful treatment due to the possible presence of immediate and passive transitions. In order to avoid the presence of passive transitions (which cause the performance model to be underspecified), we restrict ourselves to performance closed terms. In order to cope with the coexistence of exponentially timed transitions and weighted immediate transitions, i.e. the coexistence of states whose sojourn time is exponentially distributed *(tangible states)* and states whose sojourn time is zero *(vanishing states)*, we have devised an algorithm that eliminates immediate transitions together with the related vanishing states, and produces homogeneous continuous-time Markov chains (HCTMCs) [16]. These are formalized as probabilistically rooted labeled transition systems (PLTSs), which are LTSs where the initial state is replaced by a function that specifies for each state the probability that it is the initial one.

Given $E \in \mathcal{E}$, the algorithm comprises several steps. The first step consists of dropping action types, removing selfloops composed of an immediate transition (hereafter called immediate selfloops for short), changing the weight of each immediate transition into the corresponding execution probability, and determining the initial state probability function. Formally, from the LTS $\mathcal{I}[\![E]\!] = (\uparrow E, Act, \longrightarrow_E, E)$ we obtain the PLTS $\mathcal{P}_1[\![E]\!] = (S_{E,1}, \mathbb{R}_+ \cup Inf, \longrightarrow_{E,1}, P_{E,1})$ where: [2]

- $S_{E,1} = \uparrow E$.
- Let $PM_1(s) = Melt(\{\!|\, (\tilde{\lambda}, s') \mid s \xrightarrow{a,\tilde{\lambda}}_E s' \wedge a \in AType \,|\!\})$ for any $s \in S_{E,1}$. Then $\longrightarrow_{E,1}$ is the least subset of $S_{E,1} \times (\mathbb{R}_+ \cup Inf) \times S_{E,1}$ such that:
  - If s is tangible and $(\lambda, s') \in PM_1(s)$, then $s \xrightarrow{\lambda}_{E,1} s'$.
  - If s is vanishing and in $PM_1(s)$ there are exactly $m \geq 1$ potential moves (∞_{l,w_j}, s_j), $1 \leq j \leq m$, such that $s_j \not\equiv s$, then there are m transitions $s \xrightarrow{\infty_{l,w_j/w}}_{E,1} s_j$, $1 \leq j \leq m$, where $w = \sum_{j=1}^{m} w_j$.
- $P_{E,1} : S_{E,1} \longrightarrow \mathbb{R}_{[0,1]}$, $P_{E,1}(s) = \begin{cases} 1 & \text{if } s \equiv E \\ 0 & \text{if } s \not\equiv E \end{cases}$.

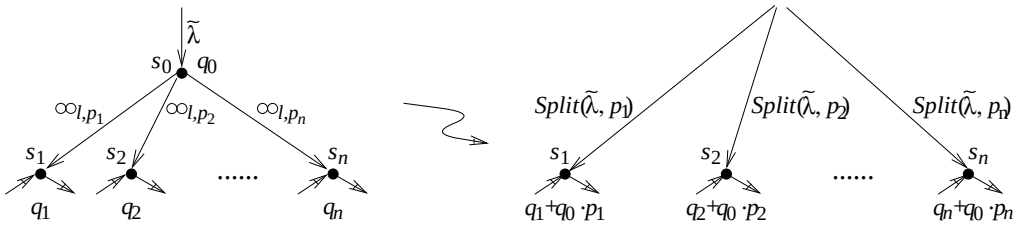

Figure 1: Graph reduction rule

The k-th step, $k \geq 2$, handles a vanishing state by eliminating the state itself as well as its outgoing immediate transitions, splitting the transitions entering the vanishing state, removing immediate selfloops created by splitting immediate transitions entering the vanishing state and exiting from states reached by the eliminated immediate transitions, and distributing the initial state probability associated with the vanishing state among the states reached by the eliminated immediate transitions. Formally, if we assume that the vanishing state considered at the k-th step is the one shown in Figure 1, we build PLTS $\mathcal{P}_k[\![E]\!] = (S_{E,k}, \mathbb{R}_+ \cup Inf, \longrightarrow_{E,k}, P_{E,k})$ where:

- $S_{E,k} = S_{E,k-1} - \{s_0\}$.
- Let $PM_k(s) = Melt(\{\!|\, (\tilde{\lambda}, s') \mid s \xrightarrow{\tilde{\lambda}}_{E,k-1} s' \wedge s' \not\equiv s_0 \,|\!\} \oplus \{\!|\, (Split(\tilde{\lambda}, p_i), s_i) \mid s \xrightarrow{\tilde{\lambda}}_{E,k-1} s_0 \wedge 1 \leq i \leq n \,|\!\})$ for any $s \in S_{E,k}$. Then $\longrightarrow_{E,k}$ is the least subset of $S_{E,k} \times (\mathbb{R}_+ \cup Inf) \times S_{E,k}$ such that:
  - If s is tangible, or vanishing but $s \notin \{s_i \mid 1 \leq i \leq n\}$, and $(\tilde{\lambda}, s') \in PM_k(s)$, then $s \xrightarrow{\tilde{\lambda}}_{E,k} s'$.
  - If s is vanishing, $s \equiv s_i$ and in $PM_k(s)$ there are exactly $m \geq 1$ potential moves (∞_{l,p_j}, s_j), $1 \leq j \leq m$, such that $s_j \not\equiv s$, then there are m transitions $s \xrightarrow{\infty_{l,p_j/p}}_{E,k} s_j$, $1 \leq j \leq m$, where $p = \sum_{j=1}^{m} p_j$.
- $P_{E,k} : S_{E,k} \longrightarrow \mathbb{R}_{[0,1]}$, $P_{E,k}(s) = \begin{cases} P_{E,k-1}(s) & \text{if } s \notin \{s_i \mid 1 \leq i \leq n\} \\ P_{E,k-1}(s) + P_{E,k-1}(s_0) \cdot p_i & \text{if } s \equiv s_i \end{cases}$.

Definition 2.7 The *Markovian semantics* of $E \in \mathcal{E}$ is the PLTS $\mathcal{M}[\![E]\!] = (S_{E,\mathcal{M}}, \mathbb{R}_+, \longrightarrow_{E,\mathcal{M}}, P_{E,\mathcal{M}})$ obtained by applying the algorithm above. ■

Theorem 2.8 Let $E \in \mathcal{E}$. If $\mathcal{I}[\![E]\!]$ has finitely many states, then the algorithm terminates and $\mathcal{M}[\![E]\!]$ has no immediate transitions, has finitely many states, and is unique. ■

For the proof of the theorem above, the reader is referred to [2].

2.3 Equivalence for EMPA terms

The notion of equivalence for EMPA terms [3] has been developed by adapting the idea of *probabilistic bisimulation* proposed in [17] according to the various kinds of actions. In order to come up with a congruence, we have introduced a *priority operator* "$\Theta(\_)$" such that priority levels are taken to be potential, and they become effective only within the scope of such an operator. We have thus considered the language $\mathcal{L}_\Theta$ generated by the following syntax

$$E ::= \underline{0} \mid <a, \tilde{\lambda}>.E \mid E/L \mid E[\varphi] \mid \Theta(E) \mid E + E \mid E \parallel_S E \mid A$$

whose semantic rules are those in Table 1 except that the rule in the first part is replaced by

[2] With abuse of notation, we apply function *Melt* to multisets of pairs whose first components are rates instead of actions.

$$\frac{(<a,\tilde{\lambda}>,E') \in Melt(PM(E))}{E \xrightarrow{a,\tilde{\lambda}} E'}$$

and the following rule for the priority operator is introduced in the second part

$$PM(\Theta(E)) = Select(PM(E))$$

It is easily seen that EMPA coincides with the set of terms $\{\Theta(E) \mid E \in \mathcal{L}\}$. The reason why the priority operator is not included in the syntax of EMPA stems from modeling issues explained in [3].

Definition 2.9 We define partial function $Rate : (\mathcal{G}_\Theta \times AType \times APLev \times \mathcal{P}(\mathcal{G}_\Theta)) \nrightarrow ARate$ by

$$Rate(E,a,l,C) = Min\{\!|\, \tilde{\lambda} \mid E \xrightarrow{a,\tilde{\lambda}} E' \wedge PL(<a,\tilde{\lambda}>) = l \wedge E' \in C \,|\!\}$$ ■

Definition 2.10 An equivalence relation $\mathcal{B} \subseteq \mathcal{G}_\Theta \times \mathcal{G}_\Theta$ is a *strong extended Markovian bisimulation (strong EMB)* if and only if, whenever $(E_1,E_2) \in \mathcal{B}$, then for all $a \in AType$, $l \in APLev$ and $C \in \mathcal{G}_\Theta/\mathcal{B}$

$$Rate(E_1,a,l,C) = Rate(E_2,a,l,C)$$

In this case we say that E_1 and E_2 are *strongly extended-Markovian bisimilar (strongly EMB)*. ■

Proposition 2.11 Let $\sim_{EMB}$ be the union of all the strong EMBs. Then $\sim_{EMB}$ is the largest strong EMB. ■

Definition 2.12 We call $\sim_{EMB}$ the *strong extended Markovian bisimulation equivalence (strong EMBE)*, and we say that $E_1, E_2 \in \mathcal{G}_\Theta$ are *strongly extended-Markovian bisimulation equivalent (strongly EMBE)* if and only if $E_1 \sim_{EMB} E_2$. ■

Theorem 2.13 Let $E_1, E_2 \in \mathcal{G}_\Theta$. If $E_1 \sim_{EMB} E_2$ then:

(*i*) For every $<a,\tilde{\lambda}> \in Act$, $<a,\tilde{\lambda}>.E_1 \sim_{EMB} <a,\tilde{\lambda}>.E_2$.

(*ii*) For every $L \subseteq AType - \{\tau\}$, $E_1/L \sim_{EMB} E_2/L$.

(*iii*) For every $\varphi \in ARFun$, $E_1[\varphi] \sim_{EMB} E_2[\varphi]$.

(*iv*) $\Theta(E_1) \sim_{EMB} \Theta(E_2)$.

(*v*) For every $F \in \mathcal{G}_\Theta$, $E_1 + F \sim_{EMB} E_2 + F$ and $F + E_1 \sim_{EMB} F + E_2$.

(*vi*) For every $F \in \mathcal{G}_\Theta$ and $S \subseteq AType - \{\tau\}$, $E_1 \parallel_S F \sim_{EMB} E_2 \parallel_S F$ and $F \parallel_S E_1 \sim_{EMB} F \parallel_S E_2$.

Furthermore, $\sim_{EMB}$ is preserved by recursive definitions. ■

We conclude by showing in Table 2 a sound and complete axiomatization of nonrecursive EMPA terms with respect to $\sim_{EMB}$. All the proofs of the results above can be found in [3].

3 Modeling the CSMA/CD protocol

One of the most commonly used medium access control protocol for local area networks with bus/tree topology is CSMA/CD: as an example, its original baseband version is seen in Ethernet. In this section we describe informally such a protocol, then we formalize it by means of EMPA, and finally we derive its utilization.

Given a set of n stations connected through a local area network, CSMA/CD [25] works as follows. When a station wants to transmit a message, it first listens to the channel in order to determine whether another transmission is in progress ("listen before talk"). If the channel is sensed to be idle then the station transmits its message, otherwise the station backs off a random amount of time and then senses the channel again.

$$
\begin{array}{ll}
(\mathcal{A}_1) & (E_1 + E_2) + E_3 = E_1 + (E_2 + E_3) \\
(\mathcal{A}_2) & E_1 + E_2 = E_2 + E_1 \\
(\mathcal{A}_3) & E + \underline{0} = E \\
(\mathcal{A}_4) & <a, \tilde{\lambda}_1>.E + <a, \tilde{\lambda}_2>.E = <a, \tilde{\lambda}_1 \; Min \; \tilde{\lambda}_2>.E \quad \text{if } PL(<a, \tilde{\lambda}_1>) = PL(<a, \tilde{\lambda}_2>) \\
\\
(\mathcal{A}_5) & \underline{0}/L = \underline{0} \\
(\mathcal{A}_6) & (<a, \tilde{\lambda}>.E)/L = \begin{cases} <a, \tilde{\lambda}>.(E/L) & \text{if } a \notin L \\ <\tau, \tilde{\lambda}>.(E/L) & \text{if } a \in L \end{cases} \\
(\mathcal{A}_7) & (E_1 + E_2)/L = E_1/L + E_2/L \\
\\
(\mathcal{A}_8) & \underline{0}[\varphi] = \underline{0} \\
(\mathcal{A}_9) & (<a, \tilde{\lambda}>.E)[\varphi] = <\varphi(a), \tilde{\lambda}>.(E[\varphi]) \\
(\mathcal{A}_{10}) & (E_1 + E_2)[\varphi] = E_1[\varphi] + E_2[\varphi] \\
\\
(\mathcal{A}_{11}) & \Theta(\underline{0}) = \underline{0} \\
(\mathcal{A}_{12}) & \Theta(\sum_{i \in I} <a_i, \tilde{\lambda}_i>.E_i) = \sum_{j \in J} <a_j, \tilde{\lambda}_j>.\Theta(E_j) \\
& \qquad \text{where } J = \{i \in I \mid \tilde{\lambda}_i = * \vee \forall h \in I.\, PL(<a_i, \tilde{\lambda}_i>) \geq PL(<a_h, \tilde{\lambda}_h>)\} \\
\\
(\mathcal{A}_{13}) & (\sum_{i \in I_1} <a_i, \tilde{\lambda}_i>.E_i) \parallel_S (\sum_{i \in I_2} <a_i, \tilde{\lambda}_i>.E_i) = \sum_{j \in J_1} <a_j, \tilde{\lambda}_j>.(E_j \parallel_S \sum_{i \in I_2} <a_i, \tilde{\lambda}_i>.E_i) + \\
& \qquad \sum_{j \in J_2} <a_j, \tilde{\lambda}_j>.(\sum_{i \in I_1} <a_i, \tilde{\lambda}_i>.E_i \parallel_S E_j) + \\
& \qquad \sum_{k \in K_1} \sum_{h \in H_k} <a_k, Split(\tilde{\lambda}_k, 1/n_k)>.(E_k \parallel_S E_h) + \\
& \qquad \sum_{k \in K_2} \sum_{h \in H_k} <a_k, Split(\tilde{\lambda}_k, 1/n_k)>.(E_h \parallel_S E_k) \\
& \qquad \text{where } J_1 = \{i \in I_1 \mid a_i \notin S\} \\
& \qquad\quad J_2 = \{i \in I_2 \mid a_i \notin S\} \\
& \qquad\quad K_1 = \{k \in I_1 \mid \exists h \in I_2.\, a_h = a_k \in S \wedge \tilde{\lambda}_h = *\} \\
& \qquad\quad K_2 = \{k \in I_2 \mid \exists h \in I_1.\, a_h = a_k \in S \wedge \tilde{\lambda}_h = *\} \\
& \qquad\quad H_k = \begin{cases} \{h \in I_2 \mid a_h = a_k \wedge \tilde{\lambda}_h = *\} & \text{if } k \in K_1 \\ \{h \in I_1 \mid a_h = a_k \wedge \tilde{\lambda}_h = *\} & \text{if } k \in K_2 \end{cases} \\
& \qquad\quad n_k = |H_k|
\end{array}
$$

Table 2: Axioms for $\sim_{EMB}$

After starting transmission, the station continues to listen to the channel until it has finished ("listen while talk"): if a collision is detected, the station ceases transmitting its message, transmits a short signal in order to let all stations know there has been a collision, and backs off a random amount of time before attempting to transmit again. In case of collision, messages are lost.

Now we compositionally model the protocol CSMA/CD by means of EMPA. First of all, we recognize that we are dealing with $n+1$ entities we denote by $Station_i$, $1 \leq i \leq n$, and $Channel$, respectively. As a consequence, the protocol can be described as follows:

$$\begin{aligned} CSMA/CD &\triangleq (Station_1 \parallel_R \ldots \parallel_R Station_n) \parallel_S Channel \\ S &= \{sense\text{-}idle_i, sense\text{-}busy_i, trans\text{-}msg_i, prop\text{-}msg_i \mid 1 \leq i \leq n\} \cup R \\ R &= \{signal\text{-}coll\} \end{aligned}$$

where $sense\text{-}idle_i$ ($sense\text{-}busy_i$) is the action type describing the fact that $Channel$ is sensed idle (busy) by $Station_i$, $trans\text{-}msg_i$ ($prop\text{-}msg_i$) is the action type describing transmission (propagation) of a message sent by $Station_i$, and $signal\text{-}coll$ is the action type describing the propagation of a message indicating that there has been a collision.

By exploiting compositionality, we can focus our attention on each of the involved entities separately. Let us consider $Station_i$, $1 \leq i \leq n$. We assume that upper levels generate messages ($gen\text{-}msg_i$) that $Station_i$ has to send in such a way that they form a Poisson stream with rate λ_i. After receiving a message to be sent, $Station_i$ listens to the channel. If the channel is sensed idle ($sense\text{-}idle_i$) then the message is transmitted ($trans\text{-}msg_i$), otherwise ($sense\text{-}busy_i$) $Station_i$ backs off a random amount of time ($back\text{-}off_i$) and then listens to the channel again. After transmitting the message, it propagates along the channel ($prop\text{-}msg_i$) possibly colliding with other messages ($signal\text{-}coll$). $Station_i$ can be modeled as follows:

$$\begin{aligned} Station_i &\triangleq <gen\text{-}msg_i, \lambda_i>.Station_{i,sense} + <signal\text{-}coll, *>.Station_i \\ Station_{i,sense} &\triangleq <sense\text{-}idle_i, \infty_{1,1}>.<trans\text{-}msg_i, \infty_{1,1}>.Station_{i,prop} + \\ &\quad <sense\text{-}busy_i, \infty_{1,1}>.Station_{i,backoff} + <signal\text{-}coll, *>.Station_{i,sense} \\ Station_{i,prop} &\triangleq <prop\text{-}msg_i, \mu_i>.Station_i + <signal\text{-}coll, *>.Station_{i,backoff} \\ Station_{i,backoff} &\triangleq <back\text{-}off_i, \gamma_i>.Station_{i,sense} + <signal\text{-}coll, *>.Station_{i,backoff} \end{aligned}$$

Note that $sense\text{-}idle_i$ and $sense\text{-}busy_i$ have been modeled as immediate actions because they are irrelevant from the performance point of view. Also $trans\text{-}msg_i$ has been modeled as an immediate action: actually, it only describes the beginning of the transmission, as the duration of the whole operation has been attached to $prop\text{-}msg_i$ (we assume that the length of messages sent by $Station_i$ is exponentially distributed with rate μ_i). Furthermore, the random amount of time during which $Station_i$ backs off has been described by means of an exponentially distributed random variable with rate γ_i. Finally, we observe that action $<signal\text{-}coll, *>$ is enabled also in any state other than $Station_{i,prop}$: this is due to the fact that collisions are signaled also to stations that are not transmitting.

Let us consider now $Channel$. It can be viewed as being composed of a bidirectional error-free communication line and n sensors, one for each station, reporting the status of the communication line. As a consequence, $Channel$ can be modeled as follows:

$$\begin{aligned} Channel &\triangleq (Sensor_1 \parallel_\emptyset \ldots \parallel_\emptyset Sensor_n) \parallel_T Line \\ T &= \{set\text{-}busy_i, set\text{-}idle_i \mid 1 \leq i \leq n\} \end{aligned}$$

where $set\text{-}busy_i$ ($set\text{-}idle_i$) is the action type describing the fact that $Sensor_i$ must be set in such a way that $Station_i$ senses $Channel$ to be busy (idle). The rationale behind the introduction of $Sensor_i$, $1 \leq i \leq n$, is that they should simulate the presence or the absence of a message propagating along the medium. Every component $Sensor_i$, $1 \leq i \leq n$, can be described as follows:

$$\begin{aligned} Sensor_i &\triangleq <sense\text{-}idle_i, *>.Sensor_i + <set\text{-}busy_i, *>.Sensor_{i,busy} \\ Sensor_{i,busy} &\triangleq <sense\text{-}busy_i, *>.Sensor_{i,busy} + <set\text{-}idle_i, *>.Sensor_i \end{aligned}$$

Concerning $Line$, it waits for the beginning of the transmission of a message ($trans\text{-}msg_i$). If we denote by $1/\sigma$ the propagation delay between the two farthest stations in the network, then a collision can occur only within $1/\sigma$ time units from the beginning of the transmission (after $1/\sigma$ time units all the stations will sense the channel busy). Thus, if no collisions occur within $1/\sigma$ time units ($elapse\text{-}pd$) then the message

is successfully propagated ($prop\text{-}msg_i$), otherwise ($trans\text{-}msg_j$ for $j \neq i$) a collision is detected ($signal\text{-}coll$). $Line$ can then be represented as follows:

$$\begin{aligned}
Line &\triangleq \sum_{i=1}^{n} <trans\text{-}msg_i, *>.Line_{i,trans} \\
Line_{i,trans} &\triangleq <elapse\text{-}pd, \sigma>.Line_{i,success} + \sum_{j=1,j\neq i}^{n} <trans\text{-}msg_j, *>.Line_{i,collision} \\
Line_{i,success} &\triangleq <set\text{-}busy_1, \infty_{2,1}>.\ldots.<set\text{-}busy_n, \infty_{2,1}>.<prop\text{-}msg_i, *>. \\
&\quad <set\text{-}idle_1, \infty_{2,1}>.\ldots.<set\text{-}idle_n, \infty_{2,1}>.Line \\
Line_{i,collision} &\triangleq <set\text{-}busy_1, \infty_{2,1}>.\ldots.<set\text{-}busy_n, \infty_{2,1}>.<elapse\text{-}pd_i, \sigma>.<signal\text{-}coll, \sigma>. \\
&\quad <set\text{-}idle_1, \infty_{2,1}>.\ldots.<set\text{-}idle_n, \infty_{2,1}>.Line
\end{aligned}$$

Observe that the maximum propagation delay $1/\sigma$ of a signal has been modeled by means of an exponentially distributed random variable with rate σ though the value of such a delay is fixed: the point is that this approximation can be made as accurate as we desire by means of a sequence of exponentially timed actions with the appropriate rates (the price to pay, as we can expect, is a state space growth). In case of success, the message is completely transmitted and propagates along the medium: due to the memoryless property of exponential distributions, the distribution of the time to completion of the operation above (described by $<prop\text{-}msg_i, *>$ in $Line_{i,success}$) is not affected by the fact that the maximum propagation delay has already elapsed. In case of collision, the worst case happens when one of the two farthest stations in the network is transmitting and the other starts transmitting just before the maximum propagation delay $1/\sigma$ has elapsed: this is the reason for action $<elapse\text{-}pd_i, \sigma>$ in $Line_{i,collision}$ (the remark about the memoryless property of exponential distributions applies to this case as well). Finally it is worth noting the use of priorities for actions $set\text{-}busy_i$ and $set\text{-}idle_i$, $1 \leq i \leq n$: since their priority level is two, they cannot be preempted by any other action, thereby making $Sensor_i$, $1 \leq i \leq n$, work as expected. Actually, the introduction of n components $Sensor_i$ might seem cumbersome. However, they constitute the means whereby to obtain a more accurate description of the protocol. As it turns out, $Sensor_i$ should be set immediately after the passage of the leading part of the message ($1/\sigma$ is only an upper bound): to do this, we should know the topology of the network as well as the maximum propagation delays between any pair of consecutive stations.

Other algebraic specifications of CSMA/CD have appeared in the literature. For example, in [21] an untimed description based on CCS is given, while in [20] a real time description based on ATP is presented. Our specification is similar to the one in [20] in the sense that they both take into account time. However, in [20] time is described in a deterministic way and special operators such as timeouts and watchdogs are used, the focus being on timing constraints. Instead, in our framework time is modeled in a stochastic way by means of actions which integrate functional and performance aspects, and there is no need for auxiliary operators due to the adoption of the race policy. From the analysis standpoint, we are more interested in measuring performance indices rather than verifying timing constraints. For example, we can compute the utilization of the medium, which is very sensitive to the duration of the backoff period and the maximum propagation delay: the values of these two parameters are essential in order to reduce the amount of wasted bandwidth due to idle periods and collisions. To compute the utilization of the medium, we can resort to the technique of rewards [15] according to the algebraic method proposed in [1]: every action becomes a triple where the third element is the reward gained by any state that can execute the action, every action is given reward zero except action $<prop\text{-}msg_i, \mu_i>$ which is given reward one, and finally the channel utilization is computed as the weighted sum (rewards being weights) of the steady state probabilities of the states of $\mathcal{M}[\![CSMA/CD]\!]$.

4 Assessing the performability of a queueing system

The performance of computing and communicating systems is often degradable because internal or external faults can reduce the quality of the delivered service even though that service remains proper according to its specification. It is therefore important to manage to measure their ability to perform, or *performability*, at different accomplishment levels specifying the extent to which a system is faulty, i.e. which resources are

faulty and, among them, which ones have failed, which ones are being recovered, and which ones contain latent faults [18].

From the modeling point of view, we would like to be able to describe both performance and dependability within a single model. On the other hand, this results in problems from the analysis standpoint such as largeness, caused by the presence of several resources working in parallel possibly at different operational levels, and stiffness, originated from the large difference of performance related event rates and rare failure related event rates implying numerical instability. As recognized in [26], this leads to a natural hierarchy of models: a higher level dependability model and a set of as many lower level performance models as there are states in the higher level model. This stems from the fact that the rate of occurrence of failure and repair events is smaller than the rate of occurrence of performance related events, hence the system achieves a quasi steady state with respect to the performance related events between successive occurrences of failure or repair events. This means that the system can be characterized by weighting these quasi steady state performance measures by the probabilities of the corresponding states of the higher level model.

In this section we show that these two seemingly conflicting requirements can be met by means of EMPA through suitable algebraic manipulations. The system we consider is a queueing system $M/M/n/q$ [16] with arrival rate λ, service rates μ_i $(1 \leq i \leq n)$, failure rates ϕ_i $(1 \leq i \leq n)$, and repair rates ρ_i $(1 \leq i \leq n)$. This system is composed of an arrival process, an initially empty queue with $q - n$ seats, and a set of n independent servers:

$$\begin{array}{rcl} System_{M/M/n/q} & \triangleq & Arrivals \parallel_a (Queue_0 \parallel_D Servers_n) \\ D & = & \{d_i \mid 1 \leq i \leq n\} \end{array}$$

where a stands for arrival of a customer whereas d_i stands for the delivery of a customer from the queue to the i-th server. The arrival stream constitutes a Poisson process with rate λ:

$$Arrivals \triangleq <a, \lambda>.Arrivals$$

The queue can be easily modeled as follows:

$$\begin{array}{rcl} Queue_0 & \triangleq & <a, *>.Queue_1 \\ Queue_h & \triangleq & <a, *>.Queue_{h+1} + \sum_{i=1}^{n} <d_i, *>.Queue_{h-1}, \quad 1 \leq h \leq q-n-1 \\ Queue_{q-n} & \triangleq & \sum_{i=1}^{n} <d_i, *>.Queue_{q-n-1} \end{array}$$

Finally we describe the set of servers as follows:

$$\begin{array}{rcl} Servers_n & \triangleq & S_1 \parallel_\emptyset S_2 \parallel_\emptyset \ldots \parallel_\emptyset S_n \\ S_i & \triangleq & <d_i, \infty_{1,1}>.S'_i, \quad 1 \leq i \leq n \\ S'_i & \triangleq & <s_i, \mu_i>.S_i + <f_i, \phi_i>.<r_i, \rho_i>.S'_i, \quad 1 \leq i \leq n \end{array}$$

where s_i stands for service, f_i stands for failure and r_i stands for repair.

Now, since the monolithic model above causes largeness and stiffness problems during its analysis, we algebraically manipulate it so as to build the hierarchy of models which should facilitate the analysis. Firstly we recognize that the higher level dependability model, i.e the failure-repair model, can be represented as follows:

$$\begin{array}{rcl} FR & \triangleq & FR_1 \parallel_\emptyset FR_2 \parallel_\emptyset \ldots \parallel_\emptyset FR_n \\ FR_i & \triangleq & <f_i, \phi_i>.<r_i, \rho_i>.FR_i, \quad 1 \leq i \leq n \end{array}$$

and can be efficiently studied since it trivially admits a product form solution. Each state of FR determines the set I of operational servers and the set J of failed servers ($I \cup J = \{1, \ldots, n\}$, $I \cap J = \emptyset$), so that the corresponding lower level performance model is given by

$$\begin{array}{rcl} System_{M/M/n/q,I,J} & \triangleq & System'_{M/M/n/q} \parallel_{D_J \cup F_I} \underline{0} \\ D_J & = & \{d_i \mid i \in J\} \\ F_I & = & \{f_i \mid i \in I\} \end{array}$$

where $System'_{M/M/n/q}$ is obtained from $System_{M/M/n/q}$ by substituting $*$ for the rates of actions whose type is f_i or r_i $(1 \leq i \leq n)$. The effect of the synchronization with $\underline{0}$ is that only operational servers can receive customers (D_J) and these servers cannot fail (F_I). It is easily seen that $System_{M/M/n/q,I,J}$ is equivalent

via $\sim_{EMB}$ to a queueing system $M/M/|I|/q$ with neither failures nor repairs, hence the manipulation above preserves the properties of the system under study.

We conclude by oberving that again we can resort to the algebraic reward based method proposed in [1] to specify and derive performance measures. For example, for every state of the higher level model we can compute the throughput of the corresponding lower level performance model by assigning reward μ_i to every action $<s_i, \mu_i>$. The overall throughput is obtained as a weighted sum of the previously computed values where the (product form) steady state probabilities of the states of the higher level model are used as weights.

Acknowledgements

We are grateful to Lorenzo Donatiello for the valuable discussions about the second example. This research has been partially funded by MURST and CNR.

References

[1] M. Bernardo, *"An Algebra-Based Method to Associate Rewards with EMPA Terms"*, to appear in Proc. of ICALP '97, Bologna (Italy), 1997

[2] M. Bernardo, L. Donatiello, R. Gorrieri, *"Integrating Performance and Functional Analysis of Concurrent Systems with EMPA"*, Technical Report UBLCS-95-14, University of Bologna (Italy), 1995

[3] M. Bernardo, R. Gorrieri, *"A Tutorial on EMPA: A Theory of Concurrent Processes with Nondeterminism, Priorities, Probabilities and Time"*, to appear in Theoretical Computer Science, 1997

[4] G. V. Bochmann, J. Vaucher, *"Adding Performance Aspects to Specification Languages"*, in Proc. of PSTV VIII, North Holland, pp. 19-31, Atlantic City (NJ), 1988

[5] E. Brinksma, J.-P. Katoen, R. Langerak, D. Latella, *"A Stochastic Causality-Based Process Algebra"*, in Computer Journal 38:553-565, 1995

[6] P. Buchholz, *"Markovian Process Algebra: Composition and Equivalence"*, in Proc. of PAPM '94, pp. 11-30, Erlangen (Germany), 1994

[7] D. Ferrari, *"Considerations on the Insularity of Performance Evaluation"*, in IEEE Trans. on Software Engineering 12:678-683, 1986

[8] N. Götz, U. Herzog, M. Rettelbach, *"Multiprocessor and Distributed System Design: the Integration of Functional Specification and Performance Analysis Using Stochastic Process Algebras"*, in Proc. of PERFORMANCE '93, LNCS 729:121-146, Rome (Italy), 1993

[9] H. Hermanns, M. Rettelbach, *"Syntax, Semantics, Equivalences, and Axioms for MTIPP"*, in Proc. of PAPM '94, pp. 71-87, Erlangen (Germany), 1994

[10] H. Hermanns, M. Rettelbach, T. Weiß, *"Formal Characterisation of Immediate Actions in SPA with Nondeterministic Branching"*, in Computer Journal 38:530-541, 1995

[11] P. Harrison, J. Hillston, *"Exploiting Quasi-Reversible Structures in Markovian Process Algebra Models"*, in Computer Journal 38:510-520, 1995

[12] C. Harvey, *"Performance Engineering as an Integral Part of System Design"*, in BT Technology Journal 4:143-147, 1986

[13] J. Hillston, *"A Compositional Approach to Performance Modelling"*, Cambridge University Press, 1996

[14] C.A.R. Hoare, *"Communicating Sequential Processes"*, Prentice Hall, 1985

[15] R.A. Howard, *"Dynamic Probabilistic Systems"*, John Wiley & Sons, 1971

[16] L. Kleinrock, *"Queueing Systems"*, John Wiley & Sons, 1975

[17] K.G. Larsen, A. Skou, *"Bisimulation through Probabilistic Testing"*, in Information and Computation 94:1-28, 1991

[18] J.F. Meyer, *"Performability: A Retrospective and some Pointers to the Future"*, in Performance Evaluation 14:139-156, 1992

[19] R. Milner, *"Communication and Concurrency"*, Prentice Hall, 1989

[20] X. Nicollin, J. Sifakis, S. Yovine, *"Compiling Real-Time Specifications into Extended Automata"*, in IEEE Trans. on Software Engineering 18:794-804, 1992

[21] J. Parrow, *"Verifying a CSMA/CD Protocol with CCS"*, in Proc. of PSTV VIII, North Holland, pp. 373-384, Atlantic City (NJ), 1988

[22] C. Priami, *"Stochastic π-Calculus"*, in Computer Journal 38:578-589, 1995

[23] M. Rettelbach, *"Probabilistic Branching in Markovian Process Algebras"*, in Computer Journal 38:590-599, 1995

[24] M. Sereno, *"Towards a Product Form Solution for Stochastic Process Algebras"*, in Computer Journal 38:622-632, 1995

[25] W. Stallings, *"Local Networks: An Introduction"*, Macmillan, 1984

[26] K.S. Trivedi, J.K. Muppala, S.P. Woolet, B.R. Haverkort, *"Composite Performance and Dependability Analysis"*, in Performance Evaluation 14:197-215, 1992

[27] Y. Yemini, J. Kurose, *"Towards the Unification of the Functional and Performance Analysis of Protocols, or Is the Alternating-Bit Protocol Really Correct?"*, in Proc. of PSTV II, 1982