FUNCTIONAL AND PERFORMANCE MODELING AND ANALYSIS OF TOKEN RING USING EMPA

MARCO BERNARDO, MARIO BRAVETTI
Università di Bologna, Dipartimento di Scienze dell'Informazione
Mura Anteo Zamboni 7, 40127 Bologna, Italy
E-mail: {bernardo, bravetti}@cs.unibo.it

We present an application of the integrated approach based on the stochastically timed process algebra EMPA to the analysis of token ring. The protocol is formally represented in EMPA. The deadlock freeness of the protocol and the utilization of the channel are then assessed by automatically analyzing the algebraic description with the EMPA based software tool TwoTowers. Finally, we show the potentialities of the integrated approach by investigating the fairness of the protocol, which depends on both functional and performance aspects.

1 A Formal Approach to Performance Evaluation

Assessing the performance of a system is an important activity which has to be carried out during the development of the system itself. The desirability of taking into account the performance aspects of a system in the early stages of its design has been widely recognized [12,6]. Nevertheless, it often happens that a concurrent system is tested for efficiency only after it has been fully designed and tested for functionality. This results in two problems. On the one hand, the detection of poor performance causes the system to be designed again, so the cost of the project increases. On the other hand, the performance-related analysis is done on a model of a system extracted from the implementation while the functionality-related analysis is usually performed on a model derived from a system design. As an example, functional verification could be conducted on a Petri net [19] or an algebraic term [17] describing the system, while performance could be evaluated on a Markov chain or a queueing network model [16] of the system. As a consequence, great care must be taken to ensure that these models are consistent with one another, i.e. do reflect (different aspects of) the same system.

In this setting, yet another problem is given by the fact that linguistic support for describing performance aspects is not provided. Performance is usually modeled by resorting to stochastic processes such as Markov chains, which are represented as (possibly huge) state transition graphs whose construction by hand is quite error prone, or queueing network models, which are a bit too informal and too oriented to particular systems. From the designer viewpoint, it would be extremely advantageous to profit from a general purpose description technique resulting in readable models, possibly specified in a compositional way, which are unambiguous and can be formally analyzed.

To solve the problems mentioned above, in the past five years several proposals have been made to extend process algebras with performance modeling features, resulting in *stochastically timed process algebras* (see e.g. [13,14,4,8,7,18]). Like classical process algebras, they are algebraic languages endowed with a small set of powerful operators; moreover, they comprise a family of actions which permit to express both the type and the duration of the activities executed by the systems being modeled. Stochastically timed process algebras thus provide a linguistic, compositional, general purpose means to formally describe and analyze functional and performance properties of systems. They also turn out to be advantageous compared to stochastically timed extensions of Petri nets [1] because of the linguistic support they provide and the fact that they enable the designer to conduct an integrated system analysis by means of suitable notions of equivalence which relate terms describing systems with the same functional and performance characteristics.

In this paper we shall consider the stochastically timed process algebra *EMPA*$_{\mathrm{r}}$ *(Extended Markovian Process Algebra with Rewards)* [2], an extension of EMPA [5] allowing for the high level specification of performance measures. The name of the algebra stems from the fact that action durations are mainly expressed by means of exponentially distributed random variables (hence Markovian), but it is also possible to express prioritized probabilistic actions having duration zero as well as actions whose duration is unspecified (hence Extended). The reason why we focus on EMPA is its notable expressive power (as recognized in [5]) which allows one to model both continuous time and discrete time systems where nondeterministic and prioritized aspects can be present as well.

When developing the theory for EMPA, we had in mind the two-phase, integrated approach to the modeling and analysis of systems proposed in [4] which we briefly recall below. The first phase requires the designer to specify the system as a term in the stochastically timed process algebra. Because of compositionality, the designer is allowed to develop the algebraic representation of the system in a modular way: every subsystem can be modeled separately, then these models can be put together through the operators of the algebra. The analysis of the algebraic term is then carried out on its underlying integrated interleaving semantic model, which is a transition system labeled with both action types and durations. The integrated interleaving semantic model can be analyzed as a whole by a notion of integrated equivalence or projected on a functional semantic model and a performance semantic model. The second phase consists of deriving from the algebraic representation of the system an equivalent representation in the form of a stochastically timed Petri net. The net representation turns out to be useful whenever a less abstract representation is required highlighting dependencies, conflicts, and synchronizations among system activities, and helpful in detecting some properties that can be easily checked only in a distributed setting. Additionally, the net representation is usually more compact than the integrated interleaving semantic model resulting from the algebraic representation, since concurrency is kept explicit instead of being simulated by alternative computations obtained by interleaving actions of concurrent components.

Therefore, EMPA has been equipped with a collection of semantics as well as a notion of integrated equivalence as depicted in Fig. 1. Each term has an integrated interleaving semantics, represented by a labeled transition system (LTS for short) whose labels consist of both the type and the duration of the actions, and an integrated net semantics represented by a generalized stochastic Petri net (GSPN) [1]. From the integrated interleaving semantic model, two projected semantic models can be obtained: a functional model, given by a LTS labeled only on the type of the actions, and a performance model, given by a Markov chain (MC for short).

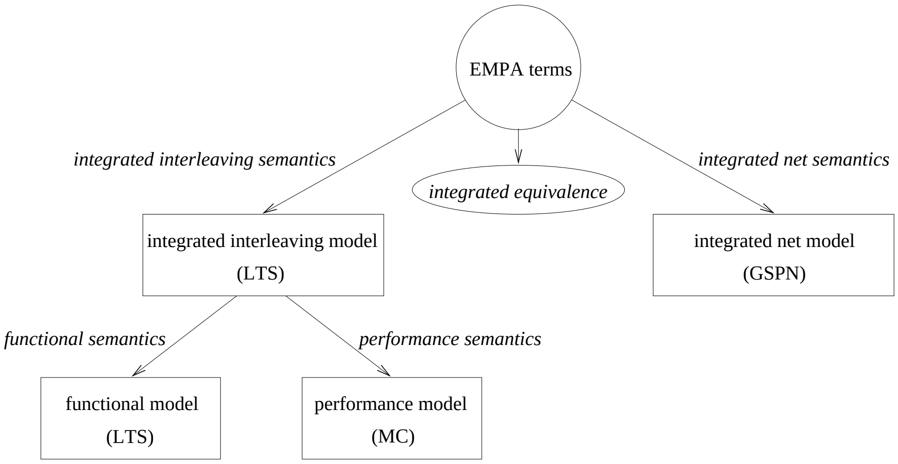

Figure 1: EMPA semantics and equivalence.

Now, since the semantics for EMPA have been formally defined in an operational style, it has been a relatively easy task to write a software tool, called TwoTowers [3], which compiles EMPA specifications to the various semantic models according to the semantic rules. Since the semantic models (i.e. LTSs, MCs, and GSPNs) have been chosen in such a way that tools already exist for their analysis, the implementation of the integrated approach proposed in [4] has just been a matter of interfacing our EMPA compiler with tools such as CWB-NC [11], MarCA [20], and GreatSPN [9]. This means that we needed not to write code for implementing the analysis routines, thereby making the development of TwoTowers itself easier and faster, and users are provided with a wide range of automated techniques helping them during the study of systems.

In a nutshell, starting from a real problem such as providing a suitable support for performance modeling and analysis in the early stages of system design, we have developed a theory which has served as both mathematical basis for the support above and specification basis for the automation of the support itself (Sect. 2). The purpose of this paper is to show an application of the machinery above to token ring [21], a widely used medium access control protocol for local area networks which avoids message collisions and guarantees a fair use of the medium. By means of this case study we shall highlight the power of the integrated approach and the related theory as well as the usefulness of having an automated support for the analysis.

The token ring protocol will be compositionally modeled in Sect. 3 as a family of terms in $\mathrm{EMPA_r}$ and its properties will be assessed by means of TwoTowers. More precisely, we shall verify a functional property (deadlock freeness) via equivalence checking and we shall evaluate a performance index (channel utilization) through the technique of rewards [15]. Moreover, we shall show a property (fair circulation of the token) that can be proven only by resorting to both functional and performance arguments, thereby stressing the importance of being able to model both functional and performance aspects. To the best of our knowledge, this is the first time that fairness issues are addressed in stochastically timed process algebras.

2 $\mathbf{EMPA_r}$ and TwoTowers

2.1 The Language

The building blocks of $\mathrm{EMPA_r}$ [2] are *actions*. Each action is a quadruple $<a, \tilde{\lambda}, \tilde{r}_1, \tilde{r}_2>$ where a is the *type*, $\tilde{\lambda}$ is the *rate*, and $\tilde{r}_1$ and $\tilde{r}_2$ are the *yield reward* and the *bonus reward*, respectively. Types divide actions into *external* and *internal* (denoted by action type τ) depending on whether they can be seen by an external observer or not. Rates divide actions into *exponentially timed*, *immediate*, and *passive*. Exponentially timed actions are actions whose rate is a positive real number which is interpreted as the parameter of the exponentially distributed random variable specifying the duration of the action. The restriction to exponentially distributed durations implies that the semantics can be defined in the interleaving style, thanks to the memoryless property, and that the underlying performance models turn out to be Markov chains. Immediate actions are actions whose rate, denoted by $\infty_{l,w}$, is infinite. Such actions have duration zero and each of them is given a *priority level* $l \in \mathbb{N}_+$ and a *weight* $w \in \mathbb{R}_+$, useful for expressing prioritized and probabilistic choices respectively. Passive actions are actions whose rate, denoted by $*$, is undefined. The duration of a passive action is fixed only by synchronizing it with a nonpassive action of the same type. Finally, rewards are included within actions to allow for a high level specification of the performance measures of interest. The yield reward expresses the speed at which gain is accumulated at the state executing the action, whereas the bonus reward expresses the gain awarded upon executing the action. Performance measures are computed as weighted sums of state probabilities and transition frequencies, where rewards are used as weights. We denote by $Act_{\rm r} = \{<a, \tilde{\lambda}, \tilde{r}_1, \tilde{r}_2> \in AType \times ARate \times AYReward \times ABReward \mid \tilde{\lambda} = * \Longleftrightarrow \tilde{r}_1 = \tilde{r}_2 = *\}$ the set of actions, where $AType$ is the set of types, $ARate = \mathbb{R}_+ \cup Inf \cup \{*\}$, with $Inf = \{\infty_{l,w} \mid l \in \mathbb{N}_+ \wedge w \in \mathbb{R}_+\}$ is the set of rates, $AYReward = \mathbb{R} \cup \{*\}$ is the set of yield rewards, and $ABReward = \mathbb{R} \cup \{*\}$ is the set of bonus rewards.

Let $Const$ be a set of *constants* and $ARFun = \{\varphi : AType \longrightarrow AType \mid \varphi(\tau) = \tau \wedge \varphi(AType - \{\tau\}) \subseteq AType - \{\tau\}\}$ be a set of *action relabeling functions*. The set $\mathcal{L}_{\rm r}$ of *process terms* of EMPA$_{\rm r}$ is generated by the syntax

$$E ::= \underline{0} \mid <a, \tilde{\lambda}, \tilde{r}_1, \tilde{r}_2>.E \mid E/L \mid E[\varphi] \mid E + E \mid E \parallel_S E \mid A$$

where $L, S \subseteq AType - \{\tau\}$ and $A \in Const$. We denote by $\mathcal{G}_{\rm r}$ the set of guarded and closed terms of $\mathcal{L}_{\rm r}$.

The *null term* "$\underline{0}$" represents a termination or deadlocked state. The *prefix operator* "$<a, \tilde{\lambda}, \tilde{r}_1, \tilde{r}_2>.\_$" denotes the sequential composition of an action and a term. The *functional abstraction operator* "$\_/L$" hides the actions whose type belongs to L by changing their type to τ. The *functional relabeling operator* "$\_[\varphi]$" changes the types of actions according to φ. The *alternative composition operator* "$\_ + \_$" expresses a choice between two terms: the choice is resolved according to the *race policy* whenever exponentially timed actions are involved (the fastest one succeeds) or according to the *preselection policy* whenever immediate actions are involved (only the actions having the highest priority level are executable and each of these is given an execution probability proportional to its own weight), while the choice is purely *nondeterministic* whenever passive actions are involved. The *parallel composition operator* "$\_ \parallel_S \_$" expresses the concurrent execution of two terms according to the following synchronization discipline: two actions can synchronize if and only if they have the same type belonging to S and at most one of them is not passive. Finally, constants together with their corresponding defining equations of the form $A \stackrel{\Delta}{=} E$ allow for recursion.

Given a term $E \in \mathcal{G}_{\mathrm{r}}$, its *integrated interleaving semantics* $\mathcal{I}_{\mathrm{r}}[\![E]\!]$ is represented as a transition system with action-labeled edges. Because of the presence of actions with different priorities, as well as the fact that immediate actions take precedence over exponentially timed ones due to the race policy, the integrated interleaving model is generated by a two-layer semantics: first all the potential moves are derived, then those with lower priority are discarded. The states of $\mathcal{I}_{\mathrm{r}}[\![E]\!]$ are divided into *tangible*, *vanishing*, *open*, and *absorbing* depending on whether they have outgoing exponentially timed transitions, outgoing immediate transitions, outgoing passive transitions, or no outgoing transitions. $E \in \mathcal{G}_{\mathrm{r}}$ is said to be *performance closed* if $\mathcal{I}_{\mathrm{r}}[\![E]\!]$ does not contain passive transitions: we denote by $\mathcal{E}_{\mathrm{r}}$ the set of performance closed terms of $\mathcal{G}_{\mathrm{r}}$.

From the integrated interleaving semantic model, two projected semantic models can be obtained. The *functional semantics* of $E \in \mathcal{G}_{\mathrm{r}}$ is still represented as a transition system $\mathcal{F}_{\mathrm{r}}[\![E]\!]$ now labeled by action types only, derived from $\mathcal{I}_{\mathrm{r}}[\![E]\!]$ by dropping action rates and rewards. The *performance semantics* of $E \in \mathcal{E}_{\mathrm{r}}$ is represented as a homogeneous Markov reward chain basically derived from $\mathcal{I}_{\mathrm{r}}[\![E]\!]$ by dropping action types and by lifting yield rewards from transitions to states: the yield reward earned by a state is the sum of the yield rewards earned by its outgoing transitions. More precisely, the performance model is given by a homogeneous continuous time Markov reward chain or a homogeneous discrete time Markov reward chain depending on whether $\mathcal{I}_{\mathrm{r}}[\![E]\!]$ contains only exponentially timed or immediate transitions. If both kinds of transition coexist, the performance model is a homogeneous continuous time Markov reward chain built by applying to $\mathcal{I}_{\mathrm{r}}[\![E]\!]$ a suitable graph reduction rule, which eliminates vanishing states together with their outgoing immediate transitions and modifies rates of transitions entering such states accordingly.

2.2 The Tool

EMPA$_{\mathrm{r}}$ is the language used by TwoTowers[3], a software tool for the analysis of functional and performance properties of concurrent systems which implements the first phase of the integrated methodology proposed in [4]. TwoTowers is composed of a graphical user interface, a tool driver, an integrated kernel, a functional kernel, and a performance kernel.

The graphical user interface allows the user to edit, compile, and analyze EMPA$_{\mathrm{r}}$ specifications, while the tool driver includes routines for parsing EMPA$_{\mathrm{r}}$ specifications.

The integrated kernel contains the routines to generate the integrated interleaving semantic model of correct $\mathrm{EMPA_r}$ specifications, to check two $\mathrm{EMPA_r}$ specifications for Markovian bisimulation equivalence, and to carry out the simulative analysis of the performance of an $\mathrm{EMPA_r}$ specification.

The functional kernel generates the functional semantic model of correct $\mathrm{EMPA_r}$ specifications. The analysis of the functional semantic models as well as the terms themselves is then executed by a version of the CWB-NC [11] that was retargeted for $\mathrm{EMPA_r}$ using the PAC-NC [10]. The functional kernel therefore comprises all the analysis capabilities of the CWB-NC including interactive simulation of systems, reachability analysis to check safety properties, model checking in the μ-calculus or CTL, equivalence checking, and preorder checking.

Finally, the performance kernel generates the performance semantic model of correct performance closed $\mathrm{EMPA_r}$ specifications. The study of the Markov reward chains is then carried out by means of MarCA [20]: transient and stationary probability distributions are calculated, then performance indexes are derived using the rewards expressed in the specifications themselves.

3 Modeling and Analyzing Token Ring

Although the IEEE 802.3 standard for local area networks (also known as CSMA/CD) [21] is widely used in offices, during the development of the 802 standard people from several companies interested in factory automation had serious reservations about it. Due to the probabilistic medium access control protocol, with a little bad luck a station might have to wait arbitrarily long to send a frame. Moreover, IEEE 802.3 frames do not have priorities, making them unsuited for real time systems in which important frames should not be held up waiting for unimportant frames.

A simple system with a known worst case for the successful transmission of a frame is a ring in which the stations take turns sending frames. If there are n stations and it takes T sec to send a frame, no frame will ever have to wait more than $n \cdot T$ sec to be sent. Another attractive feature is the fact that a ring is not really a broadcast medium, but a collection of individual point-to-point links that happen to form a circle. These links involve a well understood and field proven technology and can run on twisted pair, coaxial cable, or fiber optics. Ring engineering is also almost entirely digital, whereas CSMA/CD has a substantial analog component for collision detection. A ring is also fair and has a known upper bound on channel access. For these reasons, IBM chose the ring as its local area network and IEEE has included the token ring standard as 802.5 [21].

A ring really consists of a collection of ring interfaces connected by point-to-point lines. Each bit arriving at an interface is copied into a 1-bit buffer and then copied out onto the ring again. While in the buffer, the bit can be inspected and possibly modified before being written out. A major issue in the design and analysis of any ring network is the physical length of a bit. If the data rate of the ring is R Mbps, a bit is emitted every $1/R$ μsec. With a typical signal propagation speed of about 200 m/μsec, each bit occupies $200/R$ m on the ring. This means e.g. that a 1 Mbps ring whose circumference is 1 Km can contain only 5 bits on it at once.

In a token ring a special 24 bit pattern, called the token, circulates around the ring whenever all the stations are idle. When a station wants to transmit a frame, it is required to seize the token and remove it from the ring before transmitting. Because there is only one token, only one station can transmit at a given instant, thus solving the channel access problem. An implication of the token ring design is that the ring itself must have a sufficient delay to contain a complete token to circulate when all the stations are idle.

Ring interfaces have two operating modes: listen and transmit. In listen mode, the input bits are simply copied to output. In transmit mode, which is entered only after the token has been seized, the interface breaks the connection between input and output, entering its own data onto the ring. As bits that have propagated around the ring come back, they are removed from the ring by the sender. The sending station can either save them, to compare with the original data to monitor ring reliability, or discard them. Because the entire frame never appears on the ring at one instant, this ring architecture puts no limit on the size of the frames. After a station has finished transmitting the last bit of its last frame or the token holding time (usually 10 msec) has elapsed, the station must regenerate the token. When the last bit of the last frame has gone around and come back, it must be removed, and the interface must switch back into listen mode immediately, to avoid removing the token that might follow if no other station has removed it.

When traffic is light, the token will spend most of its time idly circulating around the ring. Occasionally a station will seize it, transmit a frame, and then output a new token. However, when the traffic is heavy, so that there is a queue at each station, as soon as a station finishes its transmission and regenerates the token, the next station downstream will see and remove the token. In this manner the permission to send rotates smoothly around the ring, in round robin fashion. The network efficiency can begin to approach 1 under conditions of heavy load.

In the remainder of this section we present a formal algebraic model of token ring based on EMPA_r. The algebraic model is then automatically analyzed with TwoTowers in order to assess functional, performance, and combined properties of the protocol.

3.1 Algebraic Model of Token Ring

The entities involved in the protocol are n stations, each equipped with a timer for the token holding time, and the token:

$$\begin{aligned} \mathit{TokenRing}_n &\stackrel{\Delta}{=} ((\mathit{Station}_1 \|_R \mathit{Timer}_1) \|_\emptyset \ldots \|_\emptyset (\mathit{Station}_n \|_R \mathit{Timer}_n)) \|_T \mathit{Token}_1 \\ T &= \{\mathit{get\_token}_i, \mathit{rel\_token}_i \mid 1 \le i \le n\} \\ R &= \{\mathit{start\_timer}, \mathit{int\_timer}, \mathit{signal\_tout}\} \end{aligned}$$

Now let us compositionally model the entities above. Each station is composed of a message generator, a queue, and a message trasmitter:

$$\mathit{Station}_i \stackrel{\Delta}{=} \mathit{MsgGen}_i \|_{\{\mathit{gen\_msg}_i\}} \mathit{Queue}_{i,0} \|_{\{\mathit{fetch\_msg}\}} \mathit{MsgTrans}_i$$

The message generator creates ($\mathit{gen\_msg}_i$) a Poisson stream of messages to be sent with rate λ_i:

$$\mathit{MsgGen}_i \stackrel{\Delta}{=} <\mathit{gen\_msg}_i, \lambda_i, 0, 0>.\mathit{MsgGen}_i$$

The queue has capacity c_i and is initially empty. The queue can either receive a new message to be sent ($\mathit{gen\_msg}_i$) or give the next message to be sent ($\mathit{fetch\_msg}$) to the message transmitter:

$$\begin{aligned} \mathit{Queue}_{i,0} &\stackrel{\Delta}{=} <\mathit{gen\_msg}_i, *, *, *>.\mathit{Queue}_{i,1} \\ \mathit{Queue}_{i,h} &\stackrel{\Delta}{=} <\mathit{gen\_msg}_i, *, *, *>.\mathit{Queue}_{i,h+1} + \\ &\quad\; <\mathit{fetch\_msg}, *, *, *>.\mathit{Queue}_{i,h-1}, \quad 0 < h < c_i \\ \mathit{Queue}_{i,c_i} &\stackrel{\Delta}{=} <\mathit{fetch\_msg}, *, *, *>.\mathit{Queue}_{i,c_i-1} \end{aligned}$$

The message transmitter comes into play upon seizing the token ($\mathit{get\_token}_i$). After starting the timer accounting for the token holding time ($\mathit{start\_timer}$), either the first message in the queue is fetched ($\mathit{fetch\_msg}$) or the token is released ($\mathit{rel\_token}_i$) and the timer interrupted ($\mathit{int\_timer}$) if the queue is empty. If both cases are possible, the trasmitter does fetch the message because action with type $\mathit{fetch\_msg}$ has been given higher priority than action with type $\mathit{rel\_token}_i$. After fetching a message, the message is transmitted ($\mathit{trans\_msg}_i$): should a timeout be signaled in the meanwhile ($\mathit{signal\_tout}$), the message is trasmitted until completion, then the token is released ($\mathit{rel\_token}_i$). Assuming that the length of a message is exponentially distributed, the message transmission time turns out to be exponentially distributed with rate γ_i:

$$\begin{aligned} \mathit{MsgTrans}_i &\stackrel{\Delta}{=} <\mathit{get\_token}_i, *, *, *>.<\mathit{start\_timer}, \infty_{1,1}, 0, 0>.\mathit{MsgTrans}'_i \\ \mathit{MsgTrans}'_i &\stackrel{\Delta}{=} <\mathit{fetch\_msg}, \infty_{2,1}, 0, 0>.(<\mathit{trans\_msg}_i, \gamma_i, 0, 0>.\mathit{MsgTrans}'_i + \\ &\qquad <\mathit{signal\_tout}, *, *, *>.<\mathit{trans\_msg}_i, \gamma_i, 0, 0>. \\ &\qquad\qquad <\mathit{rel\_token}_i, \infty_{1,1}, 0, 0>.\mathit{MsgTrans}_i) + \\ &\quad <\mathit{rel\_token}_i, \infty_{1,1}, 0, 0>.<\mathit{int\_timer}, \infty_{1,1}, 0, 0>.\mathit{MsgTrans}_i \end{aligned}$$

The timer accounting for the token holding time is started upon seizing the token (*start_timer*). Then either the token holding time elapses ($elapse\_tht_i$) and a timeout is signaled (*signal_tout*) or the timer is interrupted by the message transmitter (*int_timer*). The token holding time has been described by means of an exponentially distributed random variable with rate μ_i:

$$\begin{aligned} Timer_i \triangleq\ & <start\_timer, *, *, *>.(<elapse\_tht_i, \mu_i, 0, 0>. \\ & \qquad <signal\_tout, \infty_{1,1}, 0, 0>.Timer_i + \\ & \qquad <int\_timer, *, *, *>.Timer_i) \end{aligned}$$

Finally, the token circulating around the ring is modeled as follows, where the time it takes to get from one station to the subsequent one is represented by means of an exponentially distributed random variable with rate σ_i:

$$Token_k \triangleq <get\_token_k, \sigma_k, 0, 0>.<rel\_token_k, *, *, *>.Token_{k+1},\ 1 \leq k \leq n-1$$
$$Token_n \triangleq <get\_token_n, \sigma_n, 0, 0>.<rel\_token_n, *, *, *>.Token_1$$

3.2 Functional, Performance, and Combined Analysis

We now analyze with TwoTowers the algebraic model of token ring given in the previous section. We consider a 10 Mbps local area network and we assume that all the stations have the same characteristics. We take $\lambda = 1$ msg / μsec, $c = 2$, $\gamma = 0.0195313$ (corresponding to an average message length of 512 bits), $\mu = 0.0001$ (corresponding to a 10 msec token holding time), and $\sigma = 0.416666$ (corresponding to a 24 bit token). The number n of stations is assumed to vary between 2 and 6.

As a functional property, we have proven that the protocol is deadlock free by verifying with the CWB-NC that $\mathcal{F}_{\rm r}[\![TokenRing_n / L]\!]$, where $L = AType - T$, is weakly equivalent to $\mathcal{F}_{\rm r}[\![Token_1]\!]$.

As a performance measure, we have computed the channel utilization with MarCA. To achieve this, every action having type $trans\_msg_i$ has been given yield reward 1 in order to single out those states in which the channel is used. We show in Fig. 2 the channel utilization. As expected, the utilization approaches 1 as the number of stations increases.

Finally, we prove the fairness of the protocol by taking into account both functional and performance aspects. Showing that token ring is fair amounts to demonstrate that the token does circulate and that the stations are cyclically given the chance to get the token. In order to highlight the relevant actions, the first property is proven by reasoning on $TokenRing_n/L$, where $L = AType - T$. The property holds because, after executing an observable

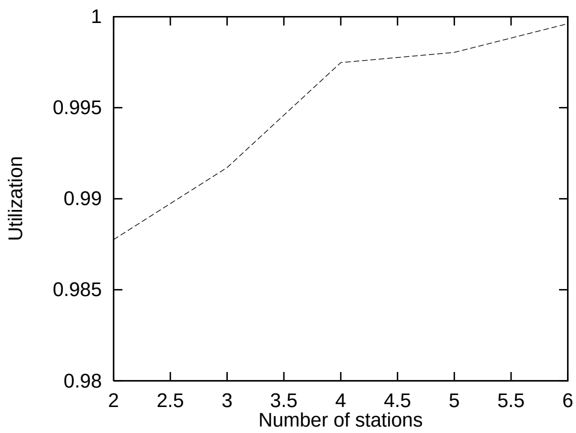

Figure 2: Channel utilization.

action, $TokenRing_n/L$ eventually performs another observable action without engaging forever in a τ loop. To verify this liveness property, it is sufficient to note that possible τ loops present in $\mathcal{I}_{\mathrm{r}}[\![TokenRing_n/L]\!]$ are always escaped with probability 1, as proven in [4] Theorem 4.11(iii). The second property is a straightforward consequence of the fact that $\mathcal{F}_{\mathrm{r}}[\![TokenRing_n/L]\!]$ is weakly equivalent to $\mathcal{F}_{\mathrm{r}}[\![Token_1]\!]$, as already mentioned, and the structure of $Token_1$.

Acknowledgements

This research has been partially funded by MURST and CNR Progetto Strategico "Modelli e Metodi per la Matematica e l'Ingegneria".

References

1. M. Ajmone Marsan, *"Stochastic Petri Nets: An Elementary Introduction"*, in *Advances in Petri Nets '89*, LNCS 424:1-29, 1990
2. M. Bernardo, *"An Algebra-Based Method to Associate Rewards with EMPA Terms"*, in Proc. of the *24th Int. Coll. on Automata, Languages and Programming (ICALP '97)*, LNCS 1256:358-368, Bologna (Italy), 1997
3. M. Bernardo, R. Cleaveland, S. Sims, W. Stewart, *"TwoTowers: A Tool Integrating Functional and Performance Analysis of Concurrent Systems"*, to appear in Proc. of the *IFIP Joint Int. Conf. on Formal Description Techniques and Protocol Specification, Testing, and Verification (FORTE/PSTV '98)*, Paris (France), 1998

4. M. Bernardo, L. Donatiello, R. Gorrieri, *"A Formal Approach to the Integration of Performance Aspects in the Modeling and Analysis of Concurrent Systems"*, to appear in Information and Computation, 1998
5. M. Bernardo, R. Gorrieri, *"A Tutorial on EMPA: A Theory of Concurrent Processes with Nondeterminism, Priorities, Probabilities and Time"*, in Theoretical Computer Science 202:1-54, 1998
6. G.V. Bochmann, J. Vaucher, *"Adding Performance Aspects to Specification Languages"*, in Proc. of the *8th Int. Conf. on Protocol Specification, Testing and Verification (PSTV VIII)*, North Holland, pp. 19-31, Atlantic City (NJ), 1988
7. E. Brinksma, J.-P. Katoen, R. Langerak, D. Latella, *"A Stochastic Causality-Based Process Algebra"*, in Computer Journal 38:553-565, 1995
8. P. Buchholz, *"Markovian Process Algebra: Composition and Equivalence"*, in Proc. of the *2nd Workshop on Process Algebras and Performance Modelling (PAPM '94)*, pp. 11-30, Erlangen (Germany), 1994
9. G. Chiola, *"GreatSPN 1.5 Software Architecture"*, in Proc. of the *5th Int. Conf. on Modeling Techniques and Tools for Computer Performance Evaluation*, Elsevier, pp. 121-136, Torino (Italy), 1991
10. W.R. Cleaveland, E. Madelaine, S. Sims, *"A Front-End Generator for Verification Tools"*, in Proc. of the *1st Int. Conf. on Tools and Algorithms for the Construction and Analysis of Systems (TACAS '95)*, LNCS 1019:153-173, Aarhus (Denmark), 1995
11. W.R. Cleaveland, S. Sims, *"The NCSU Concurrency Workbench"*, in Proc. of the *8th Int. Conf. on Computer Aided Verification (CAV '96)*, LNCS 1102:394-397, New Brunswick (NJ), 1996
12. D. Ferrari, *"Considerations on the Insularity of Performance Evaluation"*, in IEEE Trans. on Software Engineering 12:678-683, 1986
13. 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 the *16th Int. Symp. on Computer Performance Modelling, Measurement and Evaluation (PERFORMANCE '93)*, LNCS 729:121-146, Rome (Italy), 1993
14. J. Hillston, *"A Compositional Approach to Performance Modelling"*, Cambridge University Press, 1996
15. R.A. Howard, *"Dynamic Probabilistic Systems"*, John Wiley & Sons, 1971
16. L. Kleinrock, *"Queueing Systems"*, John Wiley & Sons, 1975
17. R. Milner, *"Communication and Concurrency"*, Prentice Hall, 1989
18. C. Priami, *"Stochastic π-Calculus"*, in Computer Journal 38:578-589, 1995
19. W. Reisig, *"Petri Nets: An Introduction"*, Springer-Verlag, 1985
20. W.J. Stewart, *"Introduction to the Numerical Solution of Markov Chains"*, Princeton University Press, 1994
21. A.S. Tanenbaum, *"Computer Networks"*, Prentice Hall, 1996