Using EMPA for the Performance Evaluation of an ATM Switch

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 an application of the stochastically timed process algebra EMPA_r to the performance modeling and analysis of an ATM switch. The switch is formally represented by means of a discrete time model based on the assumption of Bernoulli incoming traffic. Two algebraic descriptions are provided: the former concerns an ATM switch supporting UBR service only, the latter an ATM switch supporting UBR and ABR services. The two descriptions are then compared by automatically evaluating their performance by means of the EMPA_r based software tool TwoTowers.

1 Introduction

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. [10, 11, 4, 7, 6, 13]), algebraic languages endowed with a small set of powerful operators as well as a family of actions which permit to express both the type and the duration of the activities executed by the systems being modeled. The main reason for the adoption of stochastically timed process algebras is *compositionality*, both from the modeling and the analysis point of view.

In this paper we shall use the stochastically timed process algebra $\textit{EMPA}_\text{r}$ *(Extended Markovian Process Algebra with Rewards)* [1], an extension of EMPA [5] allowing for the algebraic specification of performance measures, to formally model ATM switches supporting different types of service and compare their performance based on the mean number of cells that are sent or lost by the switch per time unit, which will be computed by means of the EMPA_r based tool TwoTowers [3]. Unlike former EMPA_r models of concurrent systems which rely on exponential timing [4, 2, 3], the EMPA_r models of the ATM switches we shall develop here are discrete time, where the time unit is modeled

by means of a process term representing a clock, and the incoming traffic is assumed to follow a Bernoulli (instead of the usual Poisson) distribution. This should emphasize the expressive power of EMPA$_r$ as well as its versatility: both continuous time and discrete time models of probabilistic concurrent systems can be described using the same language. We also would like to point out that this work is concerned with performance evaluation, not probabilistic verification. More precisely, the technique of rewards [12] will be used.

The paper is organized as follows. In Sect. 2 we recall the basics of EMPA$_r$ and the related tool TwoTowers. In Sect. 3 we present the EMPA$_r$ models of an ATM switch supporting UBR service only and an ATM switch supporting UBR and ABR services, together with the related performance measures.

2 EMPA$_r$ and TwoTowers

2.1 The Language

The building blocks of EMPA$_r$ [1] are *actions*. Each action is a quadruple $<a, \tilde{\lambda}, \tilde{r}_1, \tilde{r}_2>$ where:

- a is the *type* of the action. According to types, actions are divided into *external* and *internal* (denoted by action type τ) depending on whether they can be seen by an external observer or not.

- $\tilde{\lambda}$ is the *rate* of the action, i.e. a concise way to represent the random variable specifying its duration. According to rates, we have the following classification:

  - *Exponentially timed actions* are actions whose rate is a positive real number. Such a number is interpreted as the parameter of the exponentially distributed random variable specifying the duration of the action.
  - *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.

  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.

- $\tilde{r}_1$ and $\tilde{r}_2$ are, respectively, the *yield reward* and the *bonus reward* of the action. Both of them are included within actions to allow for a high level specification of the performance measures of interest. The former expresses the speed at which gain is accumulated at the state executing the action, whereas the latter 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*.

Definition 2.1 The set $\mathcal{L}_{\rm r}$ of *process terms* of $\text{EMPA}_{\rm r}$ is generated by the following 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 $A \triangleq E$ allow for recursion.

Given a term $E \in \mathcal{G}_{\rm r}$, its *integrated interleaving semantics* $\mathcal{I}_{\rm r}[\![E]\!]$ 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 $\mathcal{I}_{\rm r}[\![E]\!]$ 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. $E \in \mathcal{G}_{\rm r}$ is said to be *performance closed* if $\mathcal{I}_{\rm r}[\![E]\!]$ does not contain passive transitions: we denote by $\mathcal{E}_{\rm r}$ the set of performance closed terms of $\mathcal{G}_{\rm r}$.

From the integrated interleaving semantic model, two projected semantic models can be obtained. The *functional semantics* of $E \in \mathcal{G}_{\rm r}$ is still represented as a transition system $\mathcal{F}_{\rm r}[\![E]\!]$ now labeled by action types only, derived from $\mathcal{I}_{\rm r}[\![E]\!]$ by dropping action rates and rewards. The *performance semantics* of $E \in \mathcal{E}_{\rm r}$ is represented as a homogeneous Markov reward chain basically derived from $\mathcal{I}_{\rm r}[\![E]\!]$ 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 $\mathcal{I}_{\rm r}[\![E]\!]$ 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 $\mathcal{I}_{\rm r}[\![E]\!]$ a suitable graph reduction rule, which eliminates vanishing states together with their outgoing immediate transitions and modifies rates of transitions entering such states accordingly.

For a more formal presentation of EMPA$_{\rm r}$ semantics, the interested reader is referred to [1].

2.2 The Tool

EMPA$_{\rm r}$ is the language used by TwoTowers [3], a software tool for the analysis of functional and performance properties of concurrent systems which implements the 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 is written in Tcl/Tk and allows the user to edit EMPA$_{\rm r}$ specifications, compile them, and run the various analysis routines.

The tool driver, which is written in C and uses Lex and YACC, includes routines for parsing EMPA$_{\rm r}$ specifications and performing lexical, syntactic and static semantic checks on the specifications.

The purpose of the integrated kernel is to perform those analyses that require both functional and performance information and therefore cannot be performed by factoring out information to be passed to either the functional or performance kernels. The integrated kernel contains the routines to generate the integrated interleaving semantic model of correct EMPA$_{\rm r}$ specifications, to check two EMPA$_{\rm r}$ specifications for strong extended Markovian reward bisimulation equivalence [1], and to carry out

the simulative analysis of the performance of an EMPA_r specification.

The functional kernel generates the functional semantic model of correct 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 [9] that was retargeted for EMPA_r using the PAC-NC [8]. 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 EMPA_r specifications. The analysis of the Markov reward chains is then carried out by means of MarCA [14]: transient and stationary probability distributions are calculated, then performance indexes are derived using the rewards expressed in the specifications themselves.

3 Evaluating the Performance of an ATM Switch

ATM [15] is the technology adopted to implement broadband integrated services digital networks, i.e. those broadband networks which provide end-to-end digital connectivity to support a wide range of services, including voice and nonvoice services, to which users have access by a limited set of standard multipurpose user network interfaces.

ATM is a packet switching transfer mode based on virtual connections which implements a slotted, asynchronous time division multiplexing. Virtual connections between sources and destinations are established before transmission takes place by suitably setting up the routing tables of the switches in order to allocate a given set of resources which should guarantee the quality of service, and are released once trasmission is over. Information to be transmitted along the network is grouped in 53 byte long packets called cells which flow through the corresponding virtual connections by sharing the available physical links in a slotted manner without any predefined schema, thereby reflecting the real traffic load of the network.

The aim of this section is to formally model an ATM switch with EMPA_r in order to compute some performance indices such as the mean number of cells that are sent or lost by the switch per time unit. More precisely, two algebraic descriptions will be provided: the former concerns an ATM switch supporting UBR service only, the latter an ATM switch supporting UBR and ABR services. The ABR (Available Bit Rate) service is designed for bursty traffic whose bandwidth range is known roughly (e.g. browsing the web). Using ABR service avoids having to make a long term commitment to a fixed bandwidth. ABR is the service in which the network provides rate feedback to the sender, asking it to slow down when congestion occurs. Assuming that the sender complies with such requests, cell loss for ABR traffic is expected to be low. Instead, the UBR (Unspecified Bit Rate) service makes no promises and gives no feedback about congestion. This category is well suited to sending IP packets, since IP also makes no

promises about delivery. All UBR cells are accepted, and if there is capacity left over, they will also be delivered. If congestion occurs, UBR cells will be discarded, with no feedback to the sender and no expectation that the sender slows down. The two descriptions we are going to provide are then compared by automatically evaluating their performance measures by means of TwoTowers.

3.1 ATM Switch Supporting UBR Service

An ATM switch is composed of n input ports as well as n output ports, and its task consists of forwarding incoming cells to the output ports according to the routing table that has been set up for the connections to which cells belong. The switch we are going to consider allows for unicast connections only, i.e. those requiring just one output port, and is nonblocking, because the presence of a buffer at each output port resolves possible internal link contentions among several incoming cells headed to the same output port. Because of the slotted nature of the transmission, and the fact that we are interested in the mean number of cells that are sent or lost by the switch per time unit, we resort to a discrete time EMPA_r description of the switch containing only immediate actions. In order to model the abstraction of the time unit, we introduce the action type *clock* on which all the switch components must periodically synchronize. The switch can be modeled as follows:

$$\begin{array}{rcl} \mathit{UBRATMSwitch}_n & \triangleq & \mathit{Clock} \parallel_T ((\mathit{InputPort}_1 \parallel_T \ldots \parallel_T \mathit{InputPort}_n) \parallel_{I \cup T} \\ & & \qquad (\mathit{OutputPort}_{1,0} \parallel_T \ldots \parallel_T \mathit{OutputPort}_{n,0})) \\ T & = & \{\mathit{clock}\} \\ I & = & \{\mathit{arrive\_cell}_{i,j} \mid 1 \leq i, j \leq n\} \end{array}$$

where $\mathit{arrive\_cell}_{i,j}$ is the action type describing the arrival of a cell at $\mathit{InputPort}_i$ and its subsequent forwarding to $\mathit{OutputPort}_j$.

We exploit compositionality in order to concentrate on the various components of the switch. First of all, we model *Clock* as follows:

$$\mathit{Clock} \triangleq <\mathit{clock}, \infty_{1,1}, 0, 0>.\mathit{Clock}$$

This timer endlessly performs action $<\mathit{clock}, \infty_{1,1}, 0, 0>$ on which every switch component must synchronize whenever no other local action logically belonging to the same time unit can occur. To enforce this, every term modeling a switch component will enable action $<\mathit{clock}, *, *, *>$ at any time, and will consist of actions whose priority level is greater than 1.

Now let us focus our attention on $\mathit{InputPort}_i$. Initially, it can receive a request for establishing a connection ($\mathit{request\_ubr\_conn}_i$). For the sake of simplicity, we assume that every input port can support at most one connection at a time:

$$\mathit{InputPort}_i \triangleq <\mathit{request\_ubr\_conn}_i, \infty_{8,1}, 0, 0>.\mathit{UBRService}_i$$

If the UBR connection can be accepted ($\mathit{accept\_ubr\_conn}_i$), the connection is established with $\mathit{OutputPort}_j$ with probability $1/n$ ($\mathit{open\_ubr\_conn}_{i,j}$), otherwise the connection is refused ($\mathit{refuse\_ubr\_conn}_i$). Whenever $\mathit{InputPort}_i$ is enabled to accept a cell, it can receive either a request for closing the connection with probability q_i ($\mathit{close\_conn}_i$) or a cell with probability $1 - q_i$ ($\mathit{keep\_conn}_i$). If a cell is enabled to arrive in a given

time unit, it does arrive with probability p_i ($arrive\_cell_{i,j}$) otherwise $InputPort_i$ is idle ($idle_{i,j}$), hence we are assuming a Bernoulli distribution for the incoming traffic. Since the UBR service does not guarantee the quality of service as every portion of unused bandwith is freely exploited without observing any flow control mechanism, $UBRService_i$ can be modeled as follows:

$$\begin{aligned}
UBRService_i &\triangleq <accept_ubr_conn_i, \infty_{7,1}, 0, 0>. \sum_{j=1}^{n} <open_ubr_conn_{i,j}, \infty_{6,1/n}, 0, 0>. \\
&\qquad\qquad UBRConn_{i,j} + \\
&\quad <refuse_ubr_conn_i, \infty_{7,1}, 0, 0>.InputPort_i \\
UBRConn_{i,j} &\triangleq <clock, *, *, *>.(<keep_conn_i, \infty_{5,1-q_i}, 0, 0>.UBRCell_{i,j} + \\
&\qquad\qquad <close_conn_i, \infty_{5,q_i}, 0, 0>.InputPort_i) \\
UBRCell_{i,j} &\triangleq <arrive_cell_{i,j}, \infty_{3,p_i}, 0, 0>.UBRConn_{i,j} + \\
&\quad <idle_{i,j}, \infty_{3,1-p_i}, 0, 0>.UBRConn_{i,j}
\end{aligned}$$

Finally, let us model output ports. $OutputPort_j$ is essentially a FIFO buffer with capacity c_j:

$$OutputPort_{j,m} \triangleq <clock, *, *, *>.ArriveCheck_{j,m,\{1,\ldots,n\}}, \;\; 0 \leq m \leq c_j$$

At each time unit $OutputPort_j$ can do nothing ($wait_j$) if it is empty and there is no incoming cell, send the first cell in the buffer ($send\_cell_j$) if it is not empty and there is no incoming cell, or accept an incoming cell ($arrive\_cell_{k,j}$) and send the first cell in the buffer ($send\_cell_j$) if the buffer is not empty and there is an incoming cell. Since $OutputPort_j$ can be simultaneously connected to several input ports, it can receive several incoming cells during the same time unit, and those cells which cannot be accommodated in the buffer are lost ($lose\_cell_j$). For $|K| = n$ we have:

$$\begin{aligned}
ArriveCheck_{j,0,K} &\triangleq \sum_{k \in K} <arrive_cell_{k,j}, *, *, *>.<send_cell_j, \infty_{3,1}, 0, 0>. \\
&\qquad\qquad ArriveCheck_{j,0,K-\{k\}} + \\
&\quad <wait_j, \infty_{2,1}, 0, 0>.OutputPort_{j,0} \\
ArriveCheck_{j,m,K} &\triangleq \sum_{k \in K} <arrive_cell_{k,j}, *, *, *>.<send_cell_j, \infty_{3,1}, 0, 0>. \\
&\qquad\qquad ArriveCheck_{j,m,K-\{k\}} + \\
&\quad <send_cell_j, \infty_{2,1}, 0, 0>.OutputPort_{j,m-1}, \;\; 1 \leq m \leq c_j
\end{aligned}$$

while for $1 < |K| < n$ we have:

$$\begin{aligned}
ArriveCheck_{j,m,K} &\triangleq \sum_{k \in K} <arrive_cell_{k,j}, *, *, *>.ArriveCheck_{j,m+1,K-\{k\}} + \\
&\quad OutputPort_{j,m}, \;\; 0 \leq m < c_j \\
ArriveCheck_{j,c_j,K} &\triangleq \sum_{k \in K} <arrive_cell_{k,j}, *, *, *>.<lose_cell_j, \infty_{3,1}, 0, 0>. \\
&\qquad\qquad ArriveCheck_{j,c_j,K-\{k\}} + \\
&\quad OutputPort_{j,c_j}
\end{aligned}$$

and for $|K| = 1$ we have:

$$\begin{aligned}
ArriveCheck_{j,m,\{i\}} &\triangleq <arrive_cell_{i,j}, *, *, *>.OutputPort_{j,m+1} + \\
&\quad OutputPort_{j,m}, \;\; 0 \leq m < c_j \\
ArriveCheck_{j,c_j,\{i\}} &\triangleq <arrive_cell_{i,j}, *, *, *>.<lose_cell_j, \infty_{3,1}, 0, 0>.OutputPort_{j,c_j} + \\
&\quad OutputPort_{j,c_j}
\end{aligned}$$

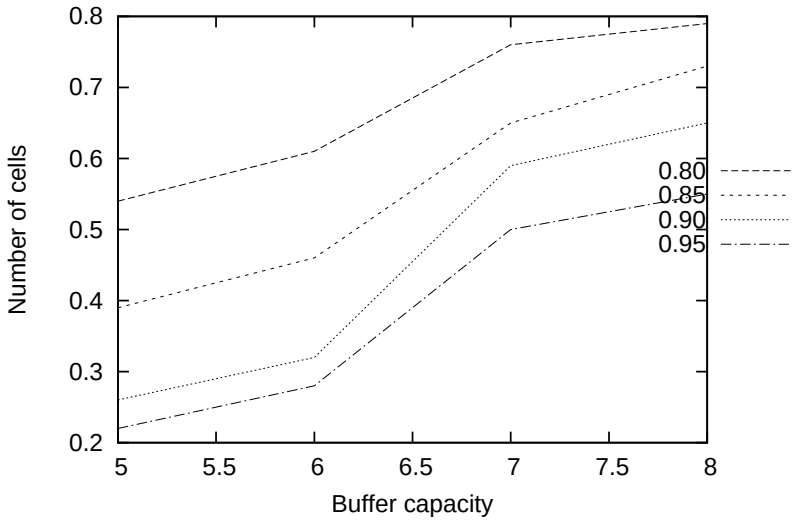

Figure 1: Mean number of cells sent per time unit

buffer capacity	states	transitions
5	2107	3672
6	2859	4982
7	3743	6522
8	4759	8292

Table 1: Size of $\mathcal{M}_{\mathrm{r}}[\![UBRATMSwitch_2]\!]$

We now compute by means of TwoTowers the mean number of cells that are sent or lost by the switch per time unit. We assume that $n = 2$, i.e. there are only two input ports and two output ports, that the input ports all have the same characteristics ($q = 10^{-6}$), and that the output ports all have the same characteristics as well. The two performance indices have been assessed by varying the buffer capacity c and the parameter p of the Bernoulli incoming traffic. We show in Table 1 the size of the performance semantics, which is a homogeneous discrete time Markov reward chain.

Since the two performance indices are related to the time unit, but the performance model used to calculate them is not atomic with respect to the time unit, in the sense that several transitions can occur during a time unit, a normalization is needed. First of all, we have to single out the actions carried out by the output ports during a time unit which are relevant from the point of view of the two performance indices. Such actions are those having type $send\_cell_j$, $lose\_cell_j$, and $wait_j$. By giving each of them yield reward 1 in turn, we determine the fractions of time π_s, π_l, and π_w during which

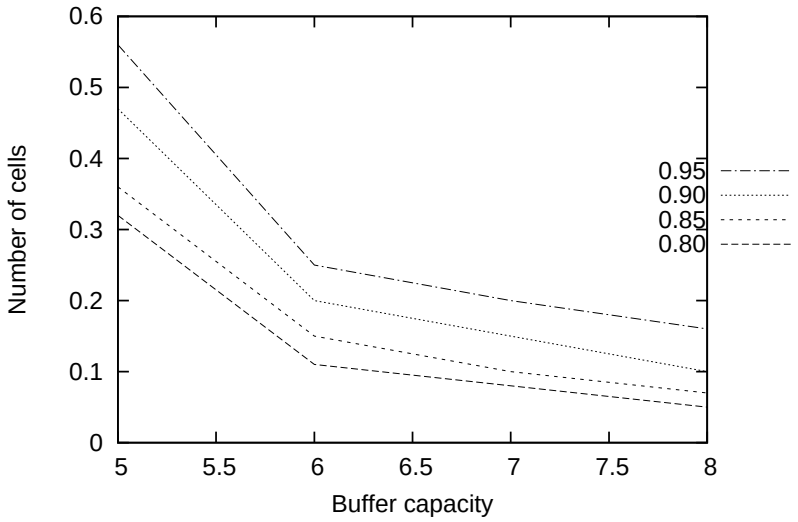

Figure 2: Mean number of cells lost per time unit

at steady state an output port is sending a cell, losing a cell, or waiting because the buffer is empty and there is no incoming cell. The fractions of time unit π'_s and π'_l during which an output port is sending or losing a cell are then given by:

$$\pi'_s = \pi_s/(\pi_s + \pi_l + \pi_w) \quad \text{and} \quad \pi'_l = \pi_l/(\pi_s + \pi_l + \pi_w)$$

To obtain the mean number of cells that are sent or lost by the switch per time unit, π'_s and π'_l must be multiplied by the incoming traffic at the output ports. Since one cell per time unit can arrive at a given input port, the probability that it does arrive is p, and the probability that it is directed to a given output port is $1/n$, the traffic is given by:

$$\sum_{j=1}^{n}\sum_{i=1}^{n} 1 \cdot p \cdot 1/n = n \cdot p \ \text{ cells/time unit}$$

The two performance indices are plotted in Fig. 1 and Fig. 2, respectively.

3.2 ATM Switch Supporting UBR and ABR Services

In order to cope with bursty traffic, we now assume that the switch is able to provide the ABR service as well. Congestion control is based on the idea that each sender has a current rate that falls between the minimum cell rate and the peak cell rate agreed upon. When congestion occurs, the rate is reduced. When congestion is absent, the rate is increased. The switch can be modeled as follows:

$$\begin{aligned} ABRUBRATMSwitch_n &\triangleq Clock \parallel_T ((InputPort_1 \parallel_T \ldots \parallel_T InputPort_n) \parallel_{I \cup T} \\ &\qquad (OutputPort_{1,0} \parallel_T \ldots \parallel_T OutputPort_{n,0})) \\ T &= \{clock\} \\ I &= \{arrive\_cell_{i,j}, increase\_speed_{i,j}, reduce\_speed_{i,j} \mid 1 \leq i,j \leq n\} \end{aligned}$$

where $increase\_speed_{i,j}$ and $reduce\_speed_{i,j}$ are action types describing flow control messages from $OutputPort_j$ to $InputPort_i$.

In the following, we only present the terms whose structure has changed with respect to the previous section. $InputPort_i$ is modified as follows:

$$\begin{aligned} InputPort_i \triangleq\ & <request\_abr\_conn_i, \infty_{8,1}, 0, 0>.ABRService_i + \\ & <request\_ubr\_conn_i, \infty_{8,1}, 0, 0>.UBRService_i \end{aligned}$$

Let us now model the ABR service. The ABR service guarantees a minimal bandwidth provided that a suitable flow control algorithm is implemented. If the ABR connection request is accepted ($accept\_abr\_conn_i$), then the connection is established with $OutputPort_j$ with probability $1/n$ ($open\_abr\_conn_{i,j}$), and the transmission proceeds at high speed, which means that $InputPort_i$ can accept an incoming cell every time unit. If $InputPort_i$ receives from $OutputPort_j$ a reduce speed message ($reduce\_speed_{i,j}$), then the transmission proceeds at middle speed, which means that $InputPort_i$ can accept an incoming cell every two time units. If $InputPort_i$ receives from $OutputPort_j$ an additional reduce speed message, then the transmission proceeds at low speed, which means that $InputPort_i$ can accept an incoming cell every three time units. The transmission can subsequently proceed faster if $InputPort_i$ receives from $OutputPort_j$ an increase speed message ($increase\_speed_{i,j}$). $ABRService_i$ can then be modeled as follows:

$$\begin{aligned} ABRService_i \triangleq\ & <accept\_abr\_conn_i, \infty_{7,1}, 0, 0>. \sum_{j=1}^{n} <open\_abr\_conn_{i,j}, \infty_{6,1/n}, 0, 0>. \\ & \qquad ABRConn_{high,i,j} + \\ & <refuse\_abr\_conn_i, \infty_{7,1}, 0, 0>.InputPort_i \end{aligned}$$

where high speed traffic is given by:

$$\begin{aligned} ABRConn_{high,i,j} \triangleq\ & <clock, *, *, *>.(<keep\_conn_i, \infty_{5,1-q_i}, 0, 0>.ABRCell_{high,i,j} + \\ & \qquad <close\_conn_i, \infty_{5,q_i}, 0, 0>.InputPort_i) \\ ABRCell_{high,i,j} \triangleq\ & <arrive\_cell_{i,j}, \infty_{3,p_i}, 0, 0>.ReduceCheck_{high,i,j} + \\ & <idle_{i,j}, \infty_{3,1-p_i}, 0, 0>.IncreaseCheck_{high,i,j} \end{aligned}$$

middle speed traffic is given by:

$$\begin{aligned} ABRConn_{mid,i,j} \triangleq\ & <clock, *, *, *>.IncreaseCheck_{mid,i,j} \\ ABRConn'_{mid,i,j} \triangleq\ & <clock, *, *, *>.(<keep\_conn_i, \infty_{5,1-q_i}, 0, 0>.ABRCell_{mid,i,j} + \\ & \qquad <close\_conn_i, \infty_{5,q_i}, 0, 0>.InputPort_i) \\ ABRCell_{mid,i,j} \triangleq\ & <arrive\_cell_{i,j}, \infty_{3,p_i}, 0, 0>.ReduceCheck_{mid,i,j} + \\ & <idle_{i,j}, \infty_{3,1-p_i}, 0, 0>.IncreaseCheck'_{mid,i,j} \end{aligned}$$

and low speed traffic is given by:

$$\begin{aligned} ABRConn_{low,i,j} \triangleq\ & <clock, *, *, *>.IncreaseCheck_{low,i,j} \\ ABRConn'_{low,i,j} \triangleq\ & <clock, *, *, *>.IncreaseCheck'_{low,i,j} \\ ABRConn''_{low,i,j} \triangleq\ & <clock, *, *, *>.(<keep\_conn_i, \infty_{5,1-q_i}, 0, 0>.ABRCell_{low,i,j} + \\ & \qquad <close\_conn_i, \infty_{5,q_i}, 0, 0>.InputPort_i) \\ ABRCell_{low,i,j} \triangleq\ & <arrive\_cell_{i,j}, \infty_{3,p_i}, 0, 0>.ReduceCheck_{low,i,j} + \\ & <idle_{i,j}, \infty_{3,1-p_i}, 0, 0>.IncreaseCheck''_{low,i,j} \end{aligned}$$

where:

$$
\begin{aligned}
IncreaseCheck_{high,i,j} &\triangleq <increase\_speed_{i,j},*,*,*>.ABRConn_{high,i,j} + \\
&\quad ABRConn_{high,i,j} \\
IncreaseCheck_{mid,i,j} &\triangleq <increase\_speed_{i,j},*,*,*>.ABRConn_{high,i,j} + \\
&\quad ABRConn'_{mid,i,j} \\
IncreaseCheck'_{mid,i,j} &\triangleq <increase\_speed_{i,j},*,*,*>.ABRConn_{high,i,j} + \\
&\quad ABRConn_{mid,i,j} \\
IncreaseCheck_{low,i,j} &\triangleq <increase\_speed_{i,j},*,*,*>.ABRConn'_{mid,i,j} + \\
&\quad ABRConn'_{low,i,j} \\
IncreaseCheck'_{low,i,j} &\triangleq <increase\_speed_{i,j},*,*,*>.ABRConn'_{mid,i,j} + \\
&\quad ABRConn''_{low,i,j} \\
IncreaseCheck''_{low,i,j} &\triangleq <increase\_speed_{i,j},*,*,*>.ABRConn_{mid,i,j} + \\
&\quad ABRConn_{low,i,j}
\end{aligned}
$$

and:

$$
\begin{aligned}
ReduceCheck_{high,i,j} &\triangleq <reduce\_speed_{i,j},*,*,*>.ABRConn_{mid,i,j} + \\
&\quad ABRConn_{high,i,j} \\
ReduceCheck_{mid,i,j} &\triangleq <reduce\_speed_{i,j},*,*,*>.ABRConn_{low,i,j} + \\
&\quad ABRConn_{mid,i,j} \\
ReduceCheck_{low,i,j} &\triangleq <reduce\_speed_{i,j},*,*,*>.ABRConn_{low,i,j} + \\
&\quad ABRConn_{low,i,j}
\end{aligned}
$$

As far as output ports are concerned, the flow control algorithm required by the ABR service introduces a lower threshold l_j and an upper threshold u_j: whenever u_j is exceeded reduce speed messages are issued, while increase speed messages are sent back to input ports if the number of cells in the buffer falls below l_j. For $|K| = n$ we have:

$$
\begin{aligned}
ArriveCheck_{j,0,K} &\triangleq \sum_{k\in K} <arrive\_cell_{k,j},*,*,*>.<send\_cell_j,\infty_{3,1},0,0>. \\
&\qquad\qquad ArriveCheck_{j,0,K-\{k\}} + \\
&\quad <wait_j,\infty_{2,1},0,0>.OutputPort_{j,0} \\
ArriveCheck_{j,l_j,K} &\triangleq \sum_{k\in K} <arrive\_cell_{k,j},*,*,*>.<send\_cell_j,\infty_{3,1},0,0>. \\
&\qquad\qquad ArriveCheck_{j,l_j,K-\{k\}} + \\
&\quad <send\_cell_j,\infty_{2,1},0,0>.IncreaseSpeedSignal_{j,K} \\
ArriveCheck_{j,m,K} &\triangleq \sum_{k\in K} <arrive\_cell_{k,j},*,*,*>.<send\_cell_j,\infty_{3,1},0,0>. \\
&\qquad\qquad ArriveCheck_{j,m,K-\{k\}} + \\
&\quad <send\_cell_j,\infty_{2,1},0,0>.OutputPort_{j,m-1},\ \ 1 \le m < u_j \wedge m \neq l_j \\
ArriveCheck_{j,m,K} &\triangleq \sum_{k\in K} <arrive\_cell_{k,j},*,*,*>.<send\_cell_j,\infty_{3,1},0,0>. \\
&\qquad\qquad ReduceSpeedSignal_{k,j,m,K} + \\
&\quad <send\_cell_j,\infty_{2,1},0,0>.OutputPort_{j,m-1},\ \ u_j \le m \le c_j
\end{aligned}
$$

while for $1 < |K| < n$ we have:

$$ArriveCheck_{j,m,K} \triangleq \sum_{k \in K} <arrive\_cell_{k,j},*,*,*>.ArriveCheck_{j,m+1,K-\{k\}} + OutputPort_{j,m},\ 0 \leq m < u_j$$

$$ArriveCheck_{j,m,K} \triangleq \sum_{k \in K} <arrive\_cell_{k,j},*,*,*>.ReduceSpeedSignal_{k,j,m,K} + OutputPort_{j,m},\ u_j \leq m < c_j$$

$$ArriveCheck_{j,c_j,K} \triangleq \sum_{k \in K} <arrive\_cell_{k,j},*,*,*>.<lose\_cell_j,\infty_{3,1},0,0>.ReduceSpeedSignal_{k,j,m,K} + OutputPort_{j,c_j}$$

and for $|K| = 1$ we have:

$$ArriveCheck_{j,m,\{i\}} \triangleq <arrive\_cell_{i,j},*,*,*>.OutputPort_{j,m+1} + OutputPort_{j,m},\ 0 \leq m < u_j$$

$$ArriveCheck_{j,m,\{i\}} \triangleq <arrive\_cell_{i,j},*,*,*>.ReduceSpeedSignal_{i,j,m,\{i\}} + OutputPort_{j,m},\ u_j \leq m < c_j$$

$$ArriveCheck_{j,c_j,\{i\}} \triangleq <arrive\_cell_{i,j},*,*,*>.<lose\_cell_j,\infty_{3,1},0,0>.ReduceSpeedSignal_{i,j,c_j,\{i\}} + OutputPort_{j,c_j}$$

where:

$$IncreaseSpeedSignal_{j,K} \triangleq \sum_{k \in K} <increase\_speed_{k,j},\infty_{4,1},0,0>.IncreaseSpeedSignal_{j,K-\{k\}} + OutputPort_{j,l_j-1},\ |K| > 1$$

$$IncreaseSpeedSignal_{j,\{i\}} \triangleq <increase\_speed_{i,j},\infty_{4,1},0,0>.OutputPort_{j,l_j-1} + OutputPort_{j,l_j-1}$$

and:

$$ReduceSpeedSignal_{i,j,m,K} \triangleq <reduce\_speed_{i,j},\infty_{4,1},0,0>.ArriveCheck_{j,m,K-\{i\}} + ArriveCheck_{j,m,K-\{i\}},\ |K| = n \wedge u_j \leq m \leq c_j$$

$$ReduceSpeedSignal_{i,j,m,K} \triangleq <reduce\_speed_{i,j},\infty_{4,1},0,0>.ArriveCheck_{j,m+1,K-\{i\}} + ArriveCheck_{j,m+1,K-\{i\}},\ 1 < |K| < n \wedge u_j \leq m < c_j$$

$$ReduceSpeedSignal_{i,j,c_j,K} \triangleq <reduce\_speed_{i,j},\infty_{4,1},0,0>.ArriveCheck_{j,c_j,K-\{i\}} + ArriveCheck_{j,c_j,K-\{i\}},\ 1 < |K| < n$$

$$ReduceSpeedSignal_{i,j,m,\{i\}} \triangleq <reduce\_speed_{i,j},\infty_{4,1},0,0>.OutputPort_{j,m+1} + OutputPort_{j,m+1},\ u_j \leq m < c_j$$

$$ReduceSpeedSignal_{i,c_j,m,\{i\}} \triangleq <reduce\_speed_{i,j},\infty_{4,1},0,0>.OutputPort_{j,c_j} + OutputPort_{j,c_j}$$

Observe that modeling speed message related operations is made complicated by the fact that, for the sake of simplicity, $OutputPort_j$ does not keep track of the currently established connections nor the related kind of service. As a consequence, increase speed messages must be sent only to those input ports currently connected because of an ABR service request, and reduce speed messages must be sent to the input port causing the upper threshold overflow only in case of ABR service.

buffer capacity	states	transitions
5	19235	31146
6	24961	40568
7	31347	51102
8	38393	62748

Table 2: Size of $\mathcal{M}_{\mathrm{r}}[\![ABRUBRATMSwitch_2]\!]$

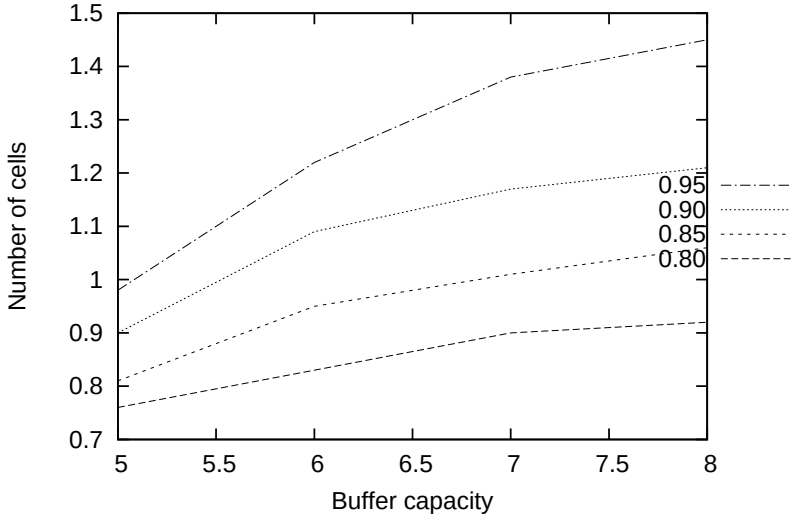

Figure 3: Mean number of cells sent per time unit

Finally, we show in Table 2 the size of the homogeneous discrete time Markov reward chain representing the performance semantics, whereas the two performance indices are plotted in Fig. 3 and Fig. 4, respectively.

3.3 Comparison and Conclusion

As far as the switch supporting the UBR service only is concerned, we observe that the mean number of cells that are sent (lost) by the switch per time unit increases (decreases) as the buffer capacity increases. This is the case for the switch supporting both services as well. As expected, the latter switch outperforms the former. Additionally, the performance of the latter switch improves as the traffic increases, whereas the reverse is true for the former switch because of the absence of a congestion control mechanism.

We conclude by noting the considerable number of states and transitions of the performance model underlying $ABRUBRATMSwitch_2$. In order to analyze the performance of switches with a more realistic number of ports, we are planning to use the

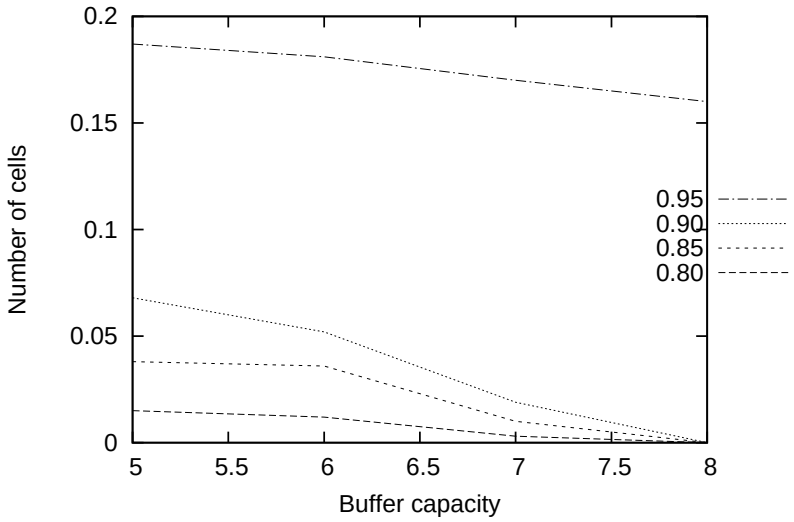

Figure 4: Mean number of cells lost per time unit

routines for the simulative analysis of performance implemented in TwoTowers so that the construction of the entire state space can be avoided.

Acknowledgements

We are grateful to Claudio Premici for actually carrying out the performance evaluation of the algebraic models with TwoTowers. This research has been partially funded by MURST and CNR Progetto Strategico "Modelli e Metodi per la Matematica e l'Ingegneria".

References

[1] 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

[2] M. Bernardo, *"Two Exercises with EMPA: Computing the Utilization of the CSMA/CD Protocol and Assessing the Performability of a Queueing System"*, in Proc. of *2nd ERCIM Int. Workshop on Formal Methods for Industrial Critical Systems (FMICS '97)*, pp. 5-17, Cesena (Italy), 1997

[3] M. Bernardo, R. Cleaveland, S. Sims, W. Stewart, *"TwoTowers: A Tool Integrating Functional and Performance Analysis of Concurrent Systems"*, submitted for publication, 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"*, to appear in Theoretical Computer Science, 1998

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

[7] 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

[8] 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

[9] 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

[10] 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

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

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

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

[14] W.J. Stewart, *"Introduction to the Numerical Solution of Markov Chains"*, Princeton University Press, 1994

[15] A.S. Tanenbaum, *"Computer Networks"*, Prentice Hall, 1996