Discrete Time Stochastic Automata Networks: using structural properties and stochastic bounds to simplify the SAN

J.M. Fourneau
INRIA project MESCAL
Laboratoire Informatique de Grenoble
CNRS UMR 5147
Montobonnot, France
jmf@prism.uvsq.fr

ABSTRACT

The Stochastic Automata Network (SAN in the following) methodology is in general associated to exact numerical analysis taking into account the tensor decomposition of the chain to improve matrix vector product. Here, we advocate a completely different approach : use the automata, their properties and the definition of tensor operations to derive structural properties of the chain and stochastic bounds to simplify the SAN. We focus on lumpability and stochastic bounds. Rather than a complete theory which remains to establish, we present a real example: the computation of the loss rate for a multistage ATM switch.

Categories and Subject Descriptors

C.4 [**Performance of Systems**]: Modeling techniques; Performance attributes; G.3 [**Probability and Statistics**]: Markov processes

Keywords

Discrete Time Markov Chain, Stochastic Bounds, Stochastic Automata Networks, Aggregation

1. INTRODUCTION

The Stochastic Automata Networks (SAN for short) approach was originally designed by B. Plateau [19] to evaluate the performance of distributed algorithms. SAN methodology consists of a decomposition into modules (automata) which are connected by synchronized transitions and functional transitions whose rates are functions of the state of the automata. Theses transitions are denoted as functional transitions or functional rate transitions. Associated to numerical solvers, SAN have been used for the specification and the evaluation phases : see [12] to obtain the loss rates in ATM networks, [2] for the blocking probabilities in multistage interconnection networks or [26] for the performance evaluation of a bandwidth allocation mechanism in Wireless ATM.

The first method associated to Stochastic Automata Networks was a numerical resolution based on the power method to compute the steady-state distribution of the Markov chain associated to a SAN [20]. It improves the complexity of the product vector-matrix taking into account the tensor decomposition of the transition matrix of a SAN. Since then, several numerical methods have been investigated: Arnoldi and GMRES [24], Gauss-Seidel [22] and some new algorithms have been proposed to again improve the product of a vector by the tensor decomposition of the transition matrix [11]. New techniques for SANs associated to Near Completely Decomposable chains have also been considered [9] and several papers have addressed some sufficient conditions of strong aggregation (or ordinary lumpability) for SANs [4, 10].

Recently, some analytical results for SAN have been presented. First B. Plateau et al. [21] have considered SAN without synchronizations. They proved that a product form steady-state distribution exists as soon as some local balances are satisfied. Even without synchronizations, the transitions of the automata are still dependent because of functional rates. In [3], Boujdaine et al. have also proved product form for SAN with a special case of synchronization where only two automata are active. However, to the best of our knowledge, these analytical results have not been used to solve real problems.

In this paper, we propose a new direction to efficiently solve some performance evaluation problems modelled as a SAN: we take into account some structural properties to modify the chain and make it lumpable. Lumpability allows to reduce the state space but many models are not lumpable or the lumped chains are still very large. Thus we advocate a SAN transformation method based on stochastic comparison which provides an upper bounding lumpable SAN. For the sake of our knowledge it is the first time that an upper bound is directly computed to obtain a SAN. We have previously shown in [15] how we can build a lumped transition probability matrix which is a stochastic upper bound of a SAN. Here we build an upper bounding SAN directly from the SAN description of the model.

It must be clear that such a result is not limited to SAN in discrete time. It is based on the tensor algebra representation and can be generalized to any formalism which uses such a description of the transition probability matrix (for

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
Valuetools-SMCTools '07, October 23-25, 2007, Nantes, France
Copyright 2007 ICST 978-963-9799-00-4.

instance PEPA [16]). Note that we use here stochastic comparison of Discrete Time Markov Chains. Another approach based on the polyhedral theory has been recently presented [7]. Stochastic comparison allows to obtain bounds on the steady-state and transient distributions but the approaches based on polyhedral theory provides in general more accurate results.

Here, we show how to prove some structural properties of the graph using properties of the automata. Rather than a general theory, which remains to establish, we present a real example: the computation of the loss rate for a multistage ATM switch. Remark that all the techniques presented here are based on the reduction of the state space. Indeed, using modular decomposition techniques, we are able to specify and store in memory models that we do not know how to solve. State reduction for exact results or stochastic bounds are clearly one way to fill the gap between our specification ability and our evaluation techniques.

The rest of the paper is organized as follows: in section 2, we describe discrete-time SAN. Note that we restrict ourselves to SAN without functional rates. Section 3 is devoted to stochastic comparison and strong aggregation. In section 4 we present the evaluation of the loss rates for the second stage of an ATM switch. This example was developed in [13] using evolution equations and analytical methods. Here, we show how our results about structure may be applied very easily to obtain these bounds. It is even possible to check the property and derive some bounds algorithmically.

2. DISCRETE-TIME SAN

For the sake of simplicity, we restrict ourselves to discrete-time SAN. Indeed our example is a discrete time model of an ATM switch. An automaton consists of states and transitions which represent the effects of events. These events are classified into two categories: local events or synchronizing events. A local event affects a single automaton and is modeled by a local transition. On the opposite, a synchronizing event modifies the state of more than one module (but loops are considered as valid transitions).

In this paper, we consider that the transition rates are fixed. The SAN methodology allows functional rates to model dependence between automata. However it is possible to replace functional rates by synchronizations with loops. Each value of the function is replaced by a synchronization with a fixed transition rate (i.e. the value of the function). Functions have been added in the SAN methodology to make more compact the representation, using less synchronizations. Furthermore, in discrete-time, as local events are independent, their global behavior is represented as their product. This is equivalent to a synchronization.

The state space of the system is the cartesian product of the states of the automata which are combined in the network. The effective state space is in general only a subset of this product. Because of synchronizations, an automaton by itself is not Markovian. To obtain a multidimensional Markov chain for the whole network, we assume independence and geometrically distributed transition durations. The fundamental results allows a compact representation of the transition matrix of the chain [19]. Assume a lexicographic ordering for the states, the matrix description of a SAN Markov chain (P) is :

$$P = \bigotimes_{i=1}^{n} Q_i + \sum_{j=1}^{c} (\bigotimes_{i=1}^{n} S_{i,j}) + N \quad (1)$$

where n is the total number of automata in the network, c is the number of synchronizations, Q_i is the description of the local events, $S_{i,j}$ is the contribution matrix of automaton i to synchronization j and N is the normalization to obtain a stochastic matrix. $\bigotimes$ denotes tensor product. Remember that if $S_{1,1} = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$ and $S_{2,1} = \begin{bmatrix} \alpha & \beta \\ \gamma & \delta \end{bmatrix}$ then:

$$S_{1,1} \bigotimes S_{2,1} = \left[\begin{array}{cc|cc} a\alpha & a\beta & b\alpha & b\beta \\ a\gamma & a\delta & b\gamma & b\delta \\ \hline c\alpha & c\beta & d\alpha & d\beta \\ c\gamma & c\delta & d\gamma & d\delta \end{array}\right]$$

Here we only consider the following matrix:

$$\sum_{j=1}^{c} (\bigotimes_{i=1}^{n} S_{i,j}) \quad (2)$$

This equation is a little bit simpler than the usual one because we do not consider normalization and we represent local transitions as a new synchronization. Remember that we restrict ourselves to discrete-time SAN without functions. Note that in discrete-time, the matrix contains the tensor product of local events while in continuous-time we use the tensor sum of these events. As we focus on structural properties, we do not need the normalization. The theory for Discrete Time SAN allows to consider global synchronizations which acts upon every automaton (but the action may be a loop) and semi-global synchronization to allow a more efficient representation of synchronizations which are only loops. However to avoid some difficult problems due to superposition of semi-global synchronizations, it is easier to model the system with global synchronizations only. Note that this is not a restrictive assumption as it is possible that $S_{i,j}$ is a positive diagonal matrix (i.e. synchronization j has no effect on automaton i). The only consequence of this restriction is that we must consider a larger set of synchronizations. Indeed if some synchronizations are independent we must build and use the cartesian product of these independent events.

The main idea is to find properties on $S_{i,j}$ which are kept unchanged during a tensor product or a sum of matrices. In the following, we present some structural or numerical properties and the way they imply a simpler resolution of the SAN. We use the following method. First check if the exact aggregation results hold. If they do not, change the automata and the synchronizations following the two constraints for the stochastic ordering (monotonicity and comparison). And design a new SAN which gives a stochastic bound and which can be analyzed by exact aggregation methods. Let us first introduce the stochastic comparison of Markov chain and necessary condition for SAN lumpability.

3. STOCHASTIC BOUNDS FOR DTMC

Most of the result already published deal with DTMC where the state space is endowed with a total order (see [14] for a survey on the algorithmic aspects of strong stochastic bounds on a totally ordered states space). Here we must consider a partial ordering on the state space. Indeed most

of the models we consider are associated to a natural partial order. For instance in a network of queues, we use the product order and the states of a queue are naturally ordered using their size. If we only have a partial order, it is very simple to build a consistent total order and we can easily obtain a bound. However the bound accuracy will be very bad. Thus, to obtain accurate bounds, we have to find new techniques which only need a partial order. Note that SAN are naturally associated to product orders.

Let ε be a discrete denumerable state space, and $\preceq$ be at least a preorder (reflexive, transitive but not necessarily anti-symmetric) on ε. We give in the following the definition of the strong stochastic ordering associated to the preorder $\preceq$, first as an integral ordering and then as a set stochastic ordering [25], [18].

DEFINITION 1. *Let λ and μ be probability measures on ε,*

$$\lambda \preceq_{st} \mu \Leftrightarrow \int_{\varepsilon} f d\lambda \leq \int_{\varepsilon} f d\mu$$

for all $f : \varepsilon \to R$ increasing according to the $\preceq$, i.e., $\forall (x, y) \in \varepsilon \times \varepsilon \quad x \preceq y \Rightarrow f(x) \leq f(y)$;

DEFINITION 2. *Any subset Γ of ε is called an increasing set if and only if $x \preceq y$ and $x \in \Gamma \Longrightarrow y \in \Gamma$.*

DEFINITION 3. *$X \preceq_{st} Y$ if and only if $P(X \in U) \leq P(Y \in U)$, for all increasing sets $U \subset S$.*

For example, let us consider the space $\varepsilon = \{1, 2, 3, 4\}$ with the partial order defined by $1 \preceq 2 \preceq 4$ and $1 \preceq 3 \preceq 4$. Then $\emptyset, \{4\}, \{2, 4\}, \{3, 4\}, \{2, 3, 4\}, \varepsilon$ are all increasing subsets of ε. If we consider the random variables X, Y and Z with the following distribution vectors $(0.3, 0.4, 0.1, 0.2)$, $(0.3, 0.1, 0.3, 0.3)$ and $(0.1, 0.2, 0.3, 0.4)$, then we have $X \preceq_{st} Z$ and $Y \preceq_{st} Z$, but X and Y are not comparable in the $\preceq_{st}$-sense since $P(X = 4) = 0.2 < P(Y = 4) = 0.3$ but $P(X \in \{2, 4\}) = 0.6 > P(Y \in \{2, 4\}) = 0.4$.

The strong stochastic ordering is a sample path ordering, so we may use Strassen's theorem and the coupling method (see [25] for further information). The stochastic ordering on Markov chains is then simply defined as the preservation of stochastic ordering which is satisfied on the initial distribution.

DEFINITION 4. *Let $X(i)$, $Y(i)$ two discrete-time Markov chains. $X(i) \preceq_{st} Y(i)$ iff*

$$X(0) \preceq_{st} Y(0) \Rightarrow X(i) \preceq_{st} Y(i) \ \ \forall i > 0$$

Now we introduce the monotonicity property for transition matrices of homogenous discrete time Markov chains (DTMC). The monotonicity property together with the transition matrix comparison are sufficient conditions for stochastic comparison of two DTMC (Theorem 1).

DEFINITION 5. *A transition matrix P of a DTMC $\{X_t\}_{t \geq 0}$ is monotone if for all probability vectors x and y, $x \preceq_{st} y$ implies $xP \leq yP$.*

We have the following characterization of $\preceq_{st}$-monotonicity. If the chains are homogeneous, the stochastic comparison and the structural properties may be expressed using the transition matrices.

LEMMA 1. *Let $\{X_t\}_{t \geq 0}$ be a DTMC with a partially ordered state space $(S, \preceq)$. A transition matrix P is $\preceq_{st}$-monotone if for all $i, j \in S$ such that $i \preceq j$, $P_{i,*} \preceq_{st} P_{j,*}$.*

DEFINITION 6. *For transition matrices P and Q we say that $P \preceq_{st} Q$ if $P_{i,*} \preceq_{st} Q_{i,*}$ for all $i \in S$.*

The definition can be clearly generalized to substochastic matrices.

THEOREM 1. *Let S be a Polish space with a closed partial order $\preceq$ and let $\{X_t\}$ and $\{X_t\}$ be two DTMC and P and Q be their respective stochastic matrices. If*

- *$X_0 \preceq_{st} Y_0$,*
- *at least one transition matrix P or Q is $\preceq_{st}$-monotone,*
- *$P \preceq_{st} Q$,*

then $X(t) \preceq_{st} Y(t)$, for all $t > 0$. If X and Y have steady-state distributions π_X and π_Y, then $\pi_X \preceq_{st} \pi_Y$.

The two following lemmas are quite easy. But they are the key properties to propagate bounds on the chain from the bounds on the automata.

LEMMA 2. *Let P and Q be two monotone substochastic matrices on the same state space endowed with a partial (or total) ordering such that $P+Q$ is stochastic or substochastic, then $P + Q$ is monotone.*

Proof : let $u \preceq_{st} v$, then according to definition $uP \preceq_{st} vP$ and $uQ \preceq_{st} vQ$. Then we have $uP + uQ \preceq_{st} vP + vQ$. After factorization we get: $u(P+Q) \preceq_{st} v(P+Q)$. As $P+Q$ is substochastic $P + Q$ is monotone.

LEMMA 3. *Let P and Q be two monotone substochastic matrices on state space $E1$ and $E2$ endowed respectively with order $\preceq_1$ and $\preceq_2$, then $P \bigotimes Q$ is monotone for the product order.*

Before proceeding with the proof, let us show on an example that this is not true for a total order. Consider the following two matrices A and B.

$$A = \begin{bmatrix} 1-a & a \\ 1-b & b \end{bmatrix} \quad B = \begin{bmatrix} 1-c & c \\ 1-d & d \end{bmatrix}$$

If $b \geq a$ then matrix A is monotone. Similarly if $d \geq c$ matrix B is monotone. Matrix $A \bigotimes B$ is equal to:

$$\left[\begin{array}{cc|cc} (1-a)(1-c) & (1-a)c & a(1-c) & ac \\ (1-a)(1-d) & (1-a)d & a(1-d) & ad \\ \hline (1-b)(1-c) & (1-b)c & b(1-c) & bc \\ (1-b)(1-d) & (1-b)d & b(1-d) & bd \end{array}\right]$$

Clearly if $ad > bc$ matrix $A \bigotimes B$ is not monotone due to the comparison of row 2 and 3. And the assumptions on a, b, c and d do not forbid to have $ad > bc$ (i.e. we only know that $b \geq a$ and $d \geq c$). To avoid to compare these two rows, one must consider a partial order instead of a total order. When we consider a product order, row 2 which is associated to state $(1, 2)$ cannot be compared to row 3 which is associated to state $(2, 1)$.

Let us know turn back to the proof of the lemma. Assume that $u \preceq_{st} v$, we want to prove that $uP \bigotimes Q \preceq_{st} vP \bigotimes Q$. Let $u = (u1, u2)$ and $v = (v1, v2)$ with the usual lexicographic representation of the product associated to tensor. As $\preceq_{st}$ is the product order it means that either

$$u1 = v1 \;\; and \;\; u2 \preceq_2 v2$$

or

$$u1 \preceq_1 v1 \;\; and \;\; u2 = v2.$$

Without loss of generality we assume that $u1 = v1$ and $u2 \preceq_2 v2$. Remember that $(u1, u2)\ P \bigotimes Q = (u1\ P, u2\ Q)$. As $u1 = v1$ we have $u1\ P = v1\ P$. As $u2 \preceq_2 v2$ and as Q is $\preceq_2$ monotone we have: $u2\ Q \preceq_2 v2\ Q$. Thus:

$$u\ P \bigotimes Q \preceq_{st} v\ P \bigotimes Q,$$

and the lemma is proved.

To handle the high level representation of a SAN, it is better to consider an event rather than the row of the matrices and we consider event monotonicity rather than monotonicity. This concept of event-monotone models will clearly help to reduce the computation cost to check that a SAN is monotone. Let us now define more formally the event based monotonicity.

Definition 7 (event). *An event e is an application defined on the state space that associates to each state x a new state denoted by $\Phi(x, e)$. Φ is called the **transition function** of the system.*

Definition 8 (execution). *An **execution** of the system is defined by an initial state x_0 and a sequence of events $e = \{e_n\}_{n\in\mathcal{N}}$. The sequence of states $\{x_n\}_{n\in\mathcal{N}}$ defined by the recurrence $x_{n+1} = \Phi(x_n, e_{n+1})$ for $n \geq 0$ is called a **trajectory**.*

Definition 9 (monotone events). *An event e is said to be monotone, if it preserves the partial ordering $\preceq$ on the state space :*

$$\forall (x, y) \;\; x \preceq y \rightarrow \Phi(x, e) \preceq \Phi(y, e)$$

If all events are monotone, the global system is said to be event-monotone.

Definition 10 (upper bounding events). *An event e is said to be an upper bound of event f if and only if*

$$\forall x \;\; \Phi(x, f) \preceq \Phi(x, e)$$

Event-monotonicity is much simpler to check algorithmically on the high level specifications than the stochastic monotonicity based on the matrix. And event monotone systems are also stochastically monotone (take care that the converse is false). As the proof is simple it is omitted here.

Lemma 4. *Event monotonicity implies strong stochastic monotonicity.*

In a Discrete Time SAN, the events are synchronizations. Thus one must provide a method or an algorithm to check if a SAN synchronization is event monotone. The following property and algorithm provide such a method.

Property 1. *For a Discrete Time SAN, endowed with a product order the verification of the event monotonicity is simple. Indeed for all states x and y, $x \preceq y$ implies that all the components are equal except one. Let i this component.*

Assume that the SAN has n automata and let $P1$,..Pn be the transition matrices which describe the synchronization we check. Due to the product ordering and the tensor representation we just have to study Pi when the states only differ in component i. Thus it is sufficient to check every matrix Pi in isolation and not to consider the cartesian product of the states and the tensor product of the matrices.

Again this is a new idea to illustrate the usefulness of tensor representation.

Finally if a SAN is not monotone, one may ask which transform we may apply to make it monotone. The main idea is to obtain upper and lower bound in the stochastic sense using the algorithms presented in [14]. As we deal with synchronization which are represented by matrix and event-monotonicity the simplest method consists in the modification of the local matrices of the synchronization. However the algorithm is highly dependent of the ordering of the local states of automata (i.e. $\preceq_1$ and $\preceq_2$ orders in previous lemmas). When these orders are total orders on the integers, Vincent's algorithm [14] provides a matrix representation of an upper bound events if the synchronization is lumpable. Indeed in this case, matrix $S_{i,j}$ has constant row sum and one can easily check that the bound also has constant row sum. For the sake of completeness we give here the text of the algorithm. P is the initial matrix representation of the synchronization and Q is the matrix representation of the upper bound monotone synchronization. We change the matrix notation to present a simpler algorithm.

Algorithm 1 Construction of the matrix representation of an monotone upper bound synchronization:

$q_{1,n} = p_{1,n}$;
for $i = 2, 3, \ldots, n$ **do**
  $q_{i,n} = \max(q_{i-1,n}, p_{i,n})$;
end for
for l = n-1 downto 1 **do**
  $q_{1,l} = p_{1,l}$;
  for $i = 2, 3, \ldots, n$ **do**
    $q_{i,l} = \max(\sum_{j=l}^{n} q_{i-1,j}, \sum_{j=l}^{n} p_{i,j}) - \sum_{j=l+1}^{n} q_{i,j}$;
  end for
end for
return q

Thus if we use a full matrix implementation of the automata description, checking if a synchronization is monotone requires a number of instructions equal to $\sum_i n_i^2$ where n_i is the size of Automaton i. Of course we can reduce the complexity if we use a sparse matrix representation. But we just want to remark that if we build the global matrix and use it to check the monotonicity the complexity will be $\prod_i n_i^2$ if we assume that all the states are reachable. Again we have to compare $c\sum_i n_i^2$ with $\prod_i n_i^2$ and if c is not that large, the tensor based approach associated to the product ordering of the states is again more efficient.

One can also use a similar algorithm to obtain a monotone lower bound for the synchronization.

If $Q = P$ at the end of the algorithm it proves that P is already monotone. When the synchronization is not lumpable

and the row sum is not constant, we have two solutions:

- Make the event monotone first before checking the monotonicity. We follow this approach in the next section.
- Perform the monotonicity checking with an ordered list of synchronization and keep into account the probability we add to make the matrix monotone. This probability has to be removed when necessary from other automata. Indeed the final matrix must still be a stochastic matrix. This eventually leads to the deletion of synchronizing events which has a zero probability. This algorithm is much more complex and we do not get into the details. For a similar approach one can consider the various versions of LL algorithms presented recently by Busic in [1] where the events are modified independently but in an ordered manner and where some events may disappear.

PROPERTY 2. *Note that to obtain an upper bounding synchronization we just have to move the transitions to upper states. Similarly lower bounding synchronizations are obtained by moving transitions to lower states.*

3.1 Aggregation of SAN

In this section we just mention a sufficient condition for lumpability of SAN which has been published some years ago [10]. This first result is based on numerical conditions and it uses the Kemeny and Snell's theorem on lumpability.

It is well known that P is ordinarily lumpable if each block in the partitioning of P has equal row sums. Lumpability of SAN has been considered for a long time (see for instance [4, 5]). In these works Buchholz defined equivalence relations among states in a component of the model or among the components. For instance, in [4] where exact aggregation is applied to hierarchical Markovian models, aggregation is done with respect to identical classes of customers inside low level models, with respect to identical low level models, and with respect to identical states of the high-level model. In [9] Dayar and his coauthors have considered a SAN in its general form with functional transitions. They assumed that the descriptor corresponding to the SAN is a sum of generalized tensor products and they derived easy to check conditions on descriptions of automata and their ordering. These results allow to identify a class of ordinarily lumpable partitionings in which lumping happens automaton by automaton. The goal of Dayar's work was to identify ordinarily lumpable partitionings of P induced by the block structure of tensor product. Obviously, this kind of ordinarily lumpable partitionings is a special case of the lumpability and performance equivalence considered by Buchholz in [4, 5]. On the other hand, it is important here to have a simple condition for ordinary lumpability. Indeed we assume that the SAN is not lumpable and we modify it to satisfy the condition. Here the condition is even simpler because we do not have functional transitions in the SAN.

DEFINITION 11. *The list of transitions for a node of an automaton is the list of pairs (transition rate, synchronization number).*

THEOREM 2 (LUMBABILITY CONDITION). *Consider a discrete time SAN without functions associated to an irreducible chain. And assume that automaton A_1 has the same list of transitions for each state, then the Markov chain is lumpable. And the lumbability classes are the states of the chain which only differ by the state of A_1.*

Idea of the proof: verify the assumptions of the theorem on lumbability of Markov chains (the proof is in [10]).

Using the same event representation, we obtain a very simple sufficient condition for a SAN lumpability. The algorithm is quite simple and we do not present it here because of the limited size of the paper.

PROPERTY 3. *A synchronization is event-lumpable for automaton A*1 *if its probability is constant and if it occurs in every state of A*1*. If all the synchronizations of a SAN are event-lumpable, then the SAN is lumpable.*

These properties show the basic operations one can perform when studying a Discrete Time SAN. First check if all synchronizations are event-lumpable and if this property holds then perform the aggregation or use the IAD algorithm proposed in [10].

If some synchronizations are not lumpable, change them to be lumpable and make all of them be monotone and upper bounding to obtain a lumpable bound described as a SAN. One can perform this task algorithmically but one can also study the model to perform a transformation of a set of synchronizations. We illustrate this approach by an example where we group synchronizations and modify them to make the SAN lumpable.

4. BOUNDING THE LOSS RATE IN AN ATM SWITCH

We study the cell loss rates in a multistage ATM switch. Such a switch is decomposed into several queues with feed-forward routing; and external arrivals always take place at the first stage. We assume a discrete-time switching as the ATM cells have a constant size. All the queues are finite. Thus losses occur in all queues due to the variability of the input processes. the topology suggests to use a decomposition to find loss rates stage by stage.

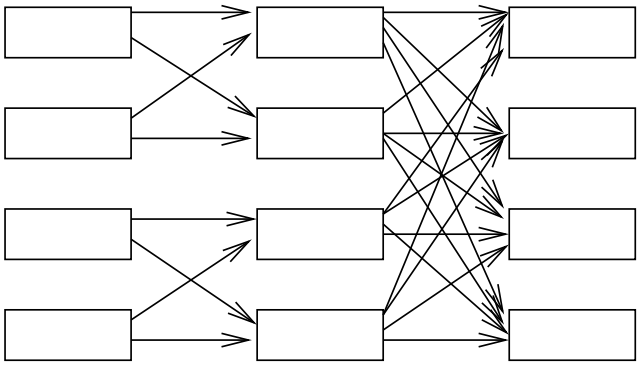

Figure 1: A multistage ATM switch

The analysis of the first stage is not very difficult. If we assume i.i.d batch arrivals or Markov modulated batch arrivals (MMBP), we can easily build a Markov chain of one buffer. the second stage is much more difficult to analyze.

Indeed, it is quite impossible to know exactly the arrival process into a buffer in the second stage even if we assume a simple i.i.d batch arrival process at the first stage. The output process of the first stage is usually unknown due to the losses at the first stage and the superposition of such processes is unknown even if we assume independence. We have analyzed this problem using stochastic bounds in [13]. The bounds were derived using evolution equation and the Strassen's theorem to compare sample-paths. Here, we show how these bounds may be obtained quite easily using the theorems and the properties stated in the previous sections. For the sake of simplicity, we only present here the derivation of the upper bound (the most interesting part of the problem).

The loss rate R is the expected number of lost cells in a queue of the second stage per unit of time. It is defined as a reward function on the steady-state distribution.

$$R = \sum_{s \in S} \pi(s) \sum_{j=1}^{m} p[j,s]((n_0 - 1)^+ + j - B_0)^+ \qquad (3)$$

where n_0 is the cell number in this buffer, B_0 is the buffer size and $p[j,s]$ is the probability that j arrivals take place into this buffer when the chain is in state s. As usual x^+ denotes $max(0,x)$. This reward is an increasing function of the probability distribution π. Therefore it is compatible with the st-comparison of chains.

4.1 Model

For the sake of simplicity we have considered a system with two input buffers and one second stage buffer. We will show on this problem how to perform bounds and aggregation. The numerical results are obtained on larger problems (4 input buffers). This is not a limitation of the approach but we need to present a small model to illustrate the transforms on the SAN.

The model consists of one automaton per queue. The automaton models the queue size between 0 and B_i for automaton i. We have used 4 synchronizations denoted as $S0$, $S1$, $S2$ and $S12$. Synchronization Si means that queue i has released a customer which may join the second stage buffer if the routing is positive. $S0$ means that the two input buffers are empty. As we only consider one output buffer such a set of synchronizations is sufficient to represent the packet movements between queues. If we must also study the correlation between output queues connected to the same set of input queues (buddy nodes in the Multistage Interconnection Network terminology) one must consider a larger set of events. Indeed in such a model, an event must explicitly model the source and the destination of the packet. Here we do need such a precise representation as we only study a single output and we have a smaller set of events. However the representation of these events are more complex.

In Figure 3, we have represented automaton A_2 which models one input buffer. Automaton A_3, depicted in Figure 4, represents the behavior of the second stage buffer. Note that we do not represent all the transitions but only the ones exiting a state. And as the behavior is different at the boundary we have represented an arbitrary state i, and the two boundary states 0 and B_i. The description of the other input buffer is easily obtained from Automaton A_2.

The arrival process at the first stage is the superposition of two independent Bernoulli processes with rate p and q.

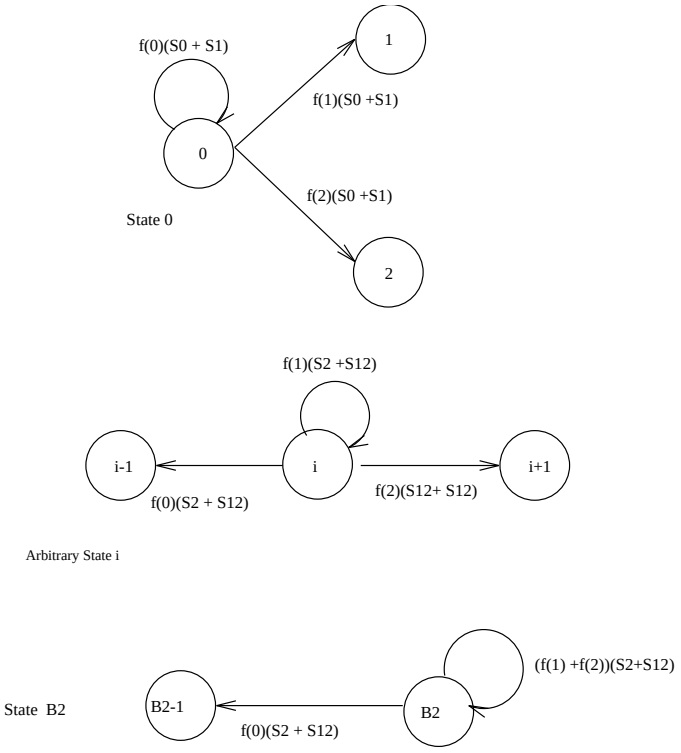

Figure 2: Automaton A_2

These arrivals probabilities are combined into the probabilities $f0$, $f1$ and $f2$ which are used in the figure for the sake of readability. Clearly we have: $f0 = (1-p)(1-q)$, $f1 = p(1-q) + q(1-p)$ and $f2 = pq$. Note that this assumption is not necessary to the analysis. The routing to enter the second stage queue (or join another queue not represented) is also independent and Bernoulli with rate $beta1$ and $beta2$.

4.2 Designing the bound

Clearly, Automaton A_2 have a tridiagonal structure with distinct behaviors for state 0 and $B2$. Also it must be clear that the model is not event-lumpable because $S0$ only exists when Automaton A_2 is in state 0. We show how we can modify the synchronizations to obtain a lumpable SAN.

Looking carefully at Automaton A_2, it is clear that if we change synchronization $S0$ into $S2$ and $S1$ into $S12$, we create a new version of Automaton A_2 such that the list of transitions is the same for all nodes of Automaton A_2. So Theorem 1 will apply and the SAN is now lumpable.

Let us now consider the monotonicity problem. We may prove that either the original SAN or the modified SAN is event-monotone. As the modified SAN has a smaller set of events (2 instead of 4) we consider it. However it is worthy to remark that the original SAN is also monotone even if the proof is more complex.

Consider the new synchronizing event $S2$ for instance for Automaton A_3. We can check very easily using Vincent's algorithm that the event is monotone. Indeed its matrix is simply (for buffer size equal to 4):

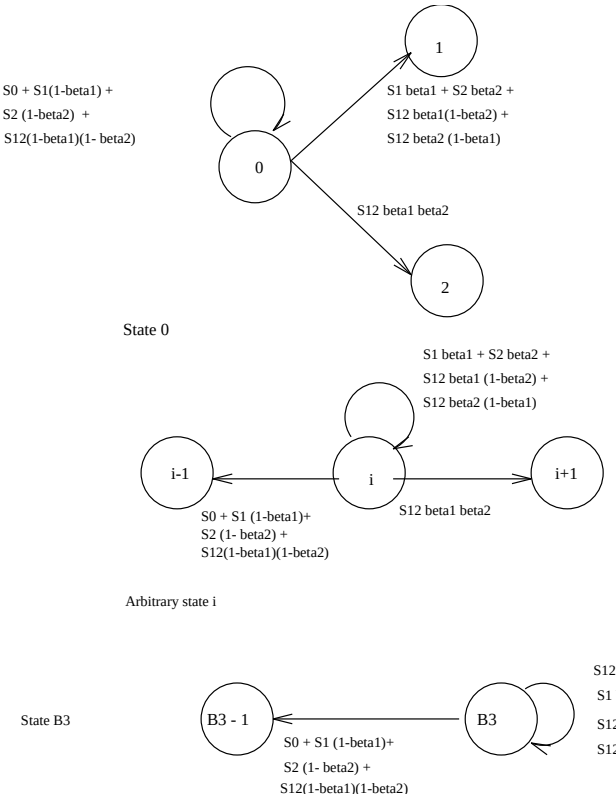

Figure 3: Automaton A_3

$$\begin{bmatrix} 1-beta2 & beta2 & & & & \\ 1-beta2 & beta2 & & & & \\ & 1-beta2 & beta2 & & & \\ & & 1-beta2 & beta2 & & \\ & & & 1-beta2 & beta2 \end{bmatrix}$$

Similarly for the same automaton, synchronization $S12$ is associated to matrix representation (again with a buffer size equal to 4):

$$\begin{bmatrix} a2a1 & c & b & & \\ a2a1 & c & b & & \\ & a2a1 & c & b & \\ & & a2a1 & c & b \\ & & & a1a2 & b+c \end{bmatrix}$$

with $a2 = 1 - beta2$, $a1 = 1 - beta1$, $c = a1beta2 + a2beta1$, and $b = beta1\ beta2$ to simplify the matrix representation.

It is very simple to check numerically that these matrices are monotone using Vincent's algorithm. And they are also upper bounding events of the former events. Furthermore, this modification is performed on lumpable automata. And this modification implies a st-comparison of the global matrix.

Note that the structure of the chain explains intuitively why the system is monotone. Indeed, the automata are associated to Batch/D/1/B queues which are known to be monotone. Here we obtain a simple verification procedure for an upper bound lumpable description based on events.

Again an intuitive explanation of the upper bound is simple: synchronization $S0$ is associated to subdiagonal transitions for Automaton A_3 while $S2$ represent subdiagonal and diagonal transitions. Similarly, changing $S1$ into $S12$ modifies the contribution in the global transition matrix to obtain something greater in the st-comparison.

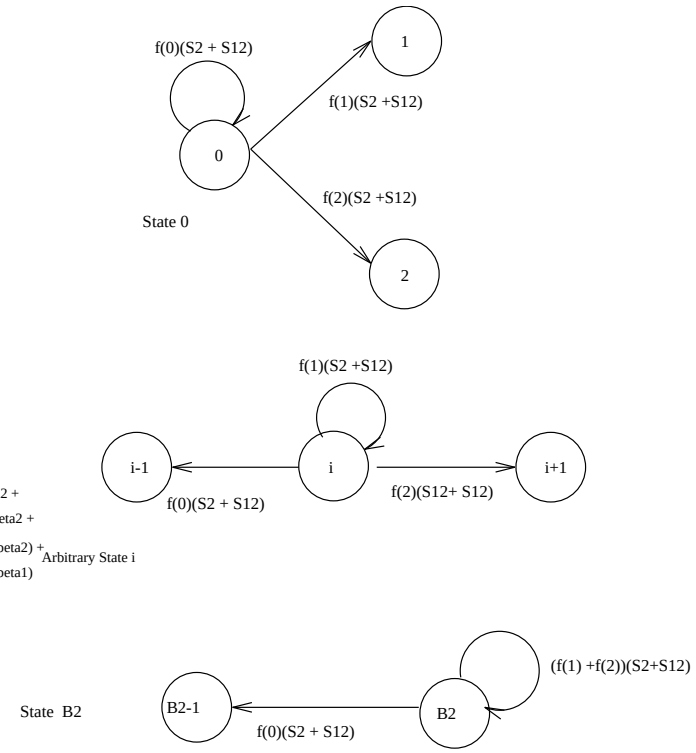

Figure 4: Modified Automaton 2

This system has a physical interpretation : we have simplified the original system by deleting some of the input buffers and we replace them by greedy sources (the black dot in figure 4.2). A greedy source always has a customer to send. This is equivalent to a queue which is always backlogged (i.e. these buffers behave as they are never empty). It is clear that this system provides a sample-path upper bound of the initial system. If the input buffer is never empty, it is not necessary to represent it to study the output queue. It is basically the modification that we have obtain when we have merged synchronization $S0$ with $S2$ and synchronization $S1$ with $S12$.

4.3 Using the bound

We apply this method to several topologies, several batch distributions of arrivals and several routing probabilities. We present here some typical results. We consider a system with 4 input buffers with the same size. Two cases are presented: buffers of size 10 and 20. The exact model is associated to a Markov chain of size $(B+1)^5$. The upper bound is obtained with a model of two input buffers and two sources. Thus the chain size is only $(B+1)^3$. The lower bounds are depicted here to show the quality of the results. But the bounds are obtained by another method that we cannot detail here (see [13]). The numerical computations have been conducted using the Gauss-Seidel's algorithm or

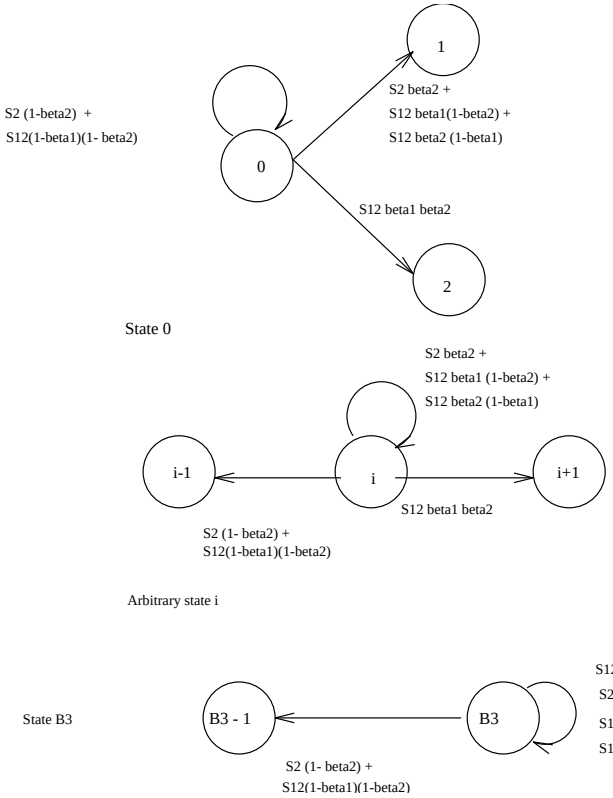

Figure 5: Modified Automaton 3

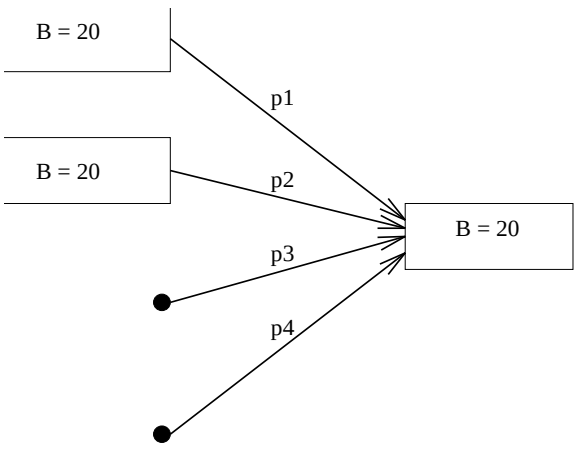

Figure 6: Physical Model for Upper Bound; the black node represents a greedy source of packets

the GTH algorithm when the convergence of the former algorithm is slow.

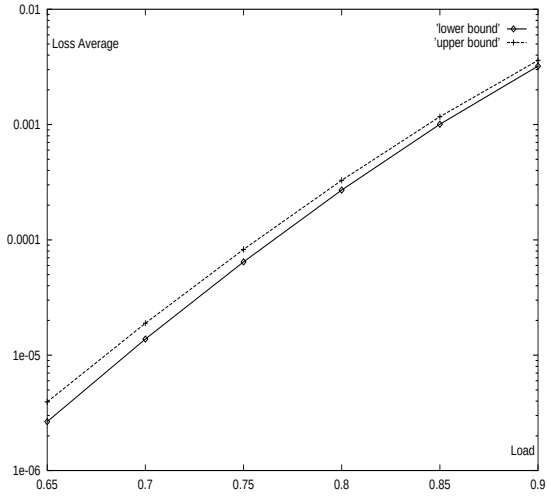

Figure 7: Buffer of size 10, q=0.01

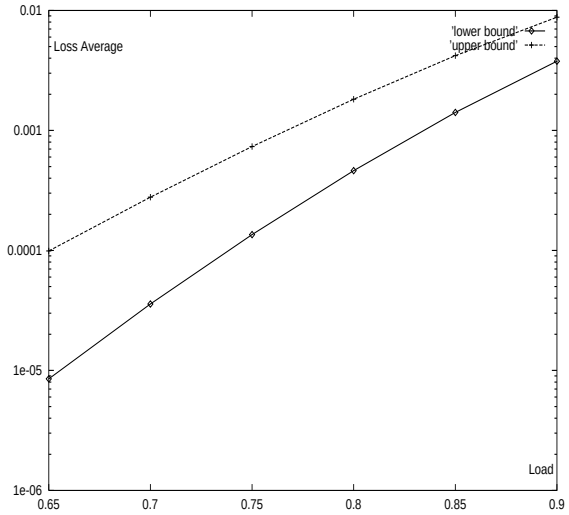

Figure 8: Buffer of size 10, q=0.1

The best results are obtained when the flows of arrivals from the input buffers are unbalanced. For instance, in figures 4.3 and 4.3, we present the bounds for buffer of size 10. We assume that the external arrivals batch is the superposition of 2 independent Bernoulli processes with probability p. So, the load in queues of the first stage is $2p$. The probabilities $beta1$ and $beta2$ are respectively $0.4 - q$ and $0.6 - q$.

The second example is a system with buffers of size 20. For the upper bounds the number of buffers replaced by sources is arbitrary. Clearly, this gives a hierarchy of bounds with a tradeoff between accuracy and computation times.

4.4 About the lower bound

We have reported in the previous drawings the upper bounds and the lower bounds. The lower bounds are actually computed using the sample path approach used in

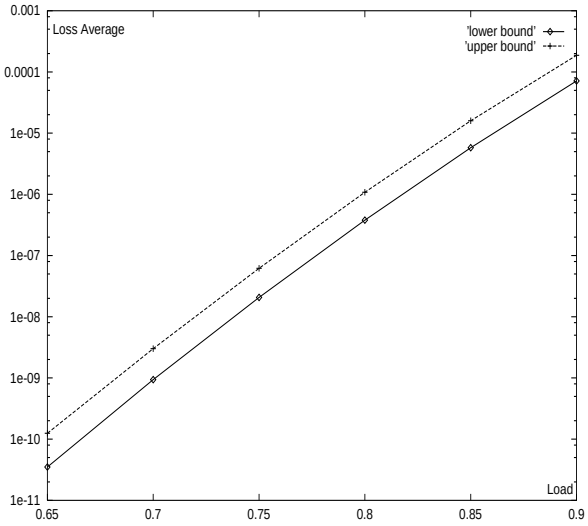

Figure 9: Buffer of size 20, q=0.01

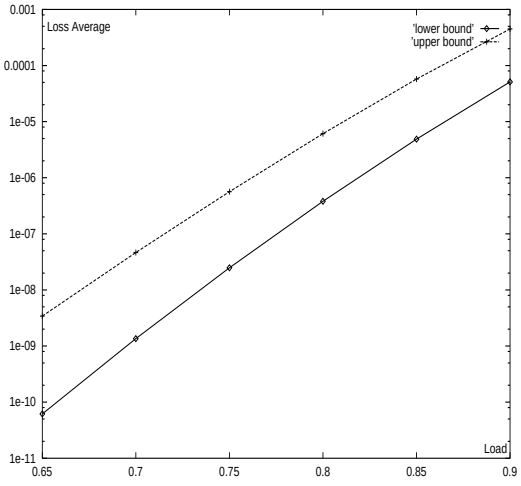

Figure 10: Buffer of size 20, q=0.05

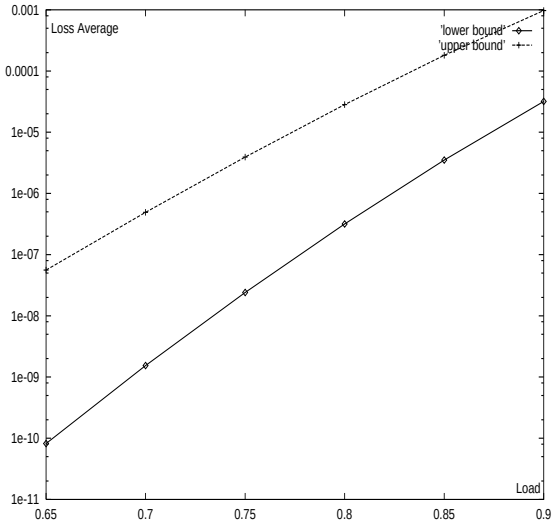

Figure 11: Buffer of size 20, q=0.1

[13]. We can also obtain lower bounds for that system using a similar modification of the SAN but we must first study SAN where an automaton is lumpable but cannot be removed from the model. We have just allowed to simplify the automaton and consider a SAN with the same number of automata but one of them is now smaller. We must first obtain some sufficient conditions on Discrete Time SAN before building the theory of SAN comparison. Such a theory is now under investigation.

5. CONCLUSION

Clearly, a general theory of exact aggregation and stochastic comparison technique for SAN has to be developed. We hope that this work will prove that such a theory may be useful to develop new algorithms for SAN. Note that our approach is quite different from the algorithms proposed in [14] even if we use the same properties (monotonicity and comparison of matrices). Indeed we modify the SAN to be lumpable and obtain an upper bound and we use a partial ordering. To the best of our knowledge it is the first approach where the output of the bounding method is a high level specification instead of a stochastic matrix. Thus bounding algorithms in this context may be considered as preprocessing before using a numerical technique based on tensor representation. Recently Daly, Buchholz and Sanders have also presented how to preserve bounds when we combine SANs [8]. We expect that it will be possible to combine all these results in a more general comparison method for SAN and other high level specification frameworks.

Acknowledgement : We thank Jean-Marc Vincent, Ana Busic and Nihal Pekergin for fruitful discussions and interesting research papers since a long time. We gracefully acknowledge their help for such an integration of SAN and stochastic bounds. Also many thanks to Lynda Mokdad for the numerical computations reported in this paper. This research is partially supported by project ANR (ANR-05-BLAN-009-0, Simulation and Stochastic Monotonicity) and ANR SETIN 2006 Checkbound.

6. REFERENCES

[1] A. Busic, J.M. Fourneau. Bounds based on lumpable matrices for partially ordered state space, Structured Markov Chains Tools Workshop in ValueTools 2006, Pisa, Italy, 2006.

[2] A. Badrah, J.M. Fourneau, and F. Quessette. Performance Evaluation of Multistage Interconnection Networks with blocking, Proc. of the 11th European Simulation MultiConference, Istanbul, 1997.

[3] F. Boujdaine, J.M. Fourneau, and N. Mikou. Product Form Solution for Stochastic Automata Networks with synchronization, 5th Process Algebra and Performance Modeling Workshop, Twente, Netherlands, 1997.

[4] P. Buchholz. Hierarchical Markovian models: symmetries and reduction, Performance Evaluation, V22, pp 93-110, 1995.

[5] P. Buchholz. Exact performance equivalence: an equivalence relation for stochastic automata, Theoretical Computer Science, V215, pp 263-287, 1999.

[6] P. Buchholz. Multilevel solution for structured Markov chains. SIAM journal on Matrix Analysis and Applications, V22, pp 342-357, 2000.

[7] P. Buchholz., An iterative bounding method for stochastic automata networks Performance Evaluation, V49, N1/4, pp 211-226, 2002.

[8] D. Daly, P. Buchholz and W. H. Sanders. Bound-Preserving Composition for Markov Reward Models, Third IEEE International Conference on the Quantitative Evaluation of Systems (QEST 2006), Riverside, California, USA, pp 243-252, September 2006,

[9] T. Dayar, O. Gusak and J.M. Fourneau. Stochastic Automata Networks and Near Complete Decomposability, SIAM Journal on Matrix Analysis and Applications, Vol 23, pp 581-599.

[10] T. Dayar, O. Gusak and J.M. Fourneau. Iterative disaggregation for a class of lumpable discrete-time SAN, Performance Evaluation, V53, N1, pp 436-451, 2003.

[11] P. Fernandes, B. Plateau, and W.J. Stewart. Efficient Descriptor-Vector Multiplications in Stochastic Automata Networks, RAIRO, V 32, N3, pp 325-351, 1998.

[12] J.M. Fourneau, L. Kloul, F. Quessette, N. Pekergin, and V. Vèque. Modeling Buffer Admission Mechanisms using Stochastic Automata Networks, Annales des Télécoms, Vol. 49, N 5-6, pp 337-349, 1994.

[13] J.M. Fourneau, L. Mokdad, N. Pekergin, Bounding the loss rate in an ATM switch, Modeling Techniques and Tools 97, St Malo, France, 1997, LNCS 1245.

[14] J.M. Fourneau, N. Pekergin, An algorithmic approach to stochastic bounds, LNCS 2459, Performance evaluation of complex systems: Techniques and Tools, 2002, pp 64-88.

[15] J.M. Fourneau, B. Plateau, I. Sbeity and W.J. Stewart, SANs and Lumpable Stochastic Bounds: Bounding Availability, in Computer System, Network Performance and Quality of Service, Edited by Imperial College Press, 2006.

[16] J. Hillston. A compositional approach to Performance Modeling, PHD Thesis, University of Edinburgh, 1994.

[17] J.G. Kemeny, J.L. Snell. Finite Markov Chains, Van Nostrand Ed, New York, 1960.

[18] W. Massey. Stochastic Ordering for Markov Processes on Partially ordered Spaces, Mathematics of operations Research, Vol.12, No. 2, 1987.

[19] B. Plateau. On the Stochastic Structure of Parallelism and Synchronization Models for Distributed Algorithms, Proc. ACM Sigmetrics Conference on Measurement and Modeling of Computer Systems, Austin, Texas, August 1985.

[20] B. Plateau, J.M. Fourneau, and K.H. Lee. PEPS: A Package for Solving Complex Markov Models of Parallel Systems, Proceedings of the 4th Int. Conf. on Modeling Techniques and Tools for Computer Performance Evaluation, Majorca, Spain, pp 291-305, 1988.

[21] B. Plateau, W.J. Stewart. Stochastic Automata Networks: Product Forms and Iterative Solutions, Inria Report 2939.

[22] F. Quessette, J. Tomasik. Iterative block methods for solving stochastic automata network models of a computer network switch, 5th Process Algebra and Performance Modeling Workshop, 1997, Twente, netherlands.

[23] W.J. Stewart. An Introduction to the Numerical Solution of Markov Chains, Princeton, 1993.

[24] W.J. Stewart, K. Atif, and B. Plateau. The numerical solution of Stochastic Automata Networks, European Journal of Operation Research, V86, N3, pp. 503-525, 1995.

[25] D. Stoyan. Comparison methods for queues and other stochastic models, J. Wiley and son, 1976.

[26] V. Vèque, J. Benothman. MRAP : Multiservice Resource Allocation Policy for Wireless ATM Networks, Int. Journal in Computer networks and ISDN systems, V29, pp 187-200, 1998.