FACULDADE DE INFORMÁTICA
PUCRS - Brazil
http://www.inf.pucrs.br

Modeling Finite Capacity Queueing Networks with Stochastic Automata Networks

P. Fernandes and B. Plateau

TECHNICAL REPORT SERIES

Number 004
July, 2000

Contact:

paulof@inf.pucrs.br

http://www.inf.pucrs.br/~paulof

P. Fernandes is a senior lecturer at PUCRS/Brazil, where he started working in 1988. His main research topics are performance evaluation, numerical methods and Markovian models. He is the head of the Scientific Computing Group at PUCRS.

B. Plateau has been a professor at the *Institut National Polytechnique de Grenoble* since 1987. His main research topics are parallel processing, performance evaluation and numerical methods. She is the head of the *ID – Informatique et Distribution* lab at INRIA-Grenoble.

Copyright © Faculdade de Informática – PUCRS

Published by the Campus Global – FACIN – PUCRS

Av. Ipiranga, 6681

90619-900 Porto Alegre – RS – Brazil

Modeling Finite Capacity Queueing Networks with Stochastic Automata Networks*

Paulo Fernandes
CNPq-PUCRS, Brazil
paulof@inf.pucrs.br

Brigitte Plateau
CNRS-INPG-INRIA-UJF, France
Brigitte.Plateau@imag.fr

Abstract

This paper presents a method to represent Finite Capacity Queueing Networks - FCQN - using an alternative formalism called Stochastic Automata Networks - SAN. This method can be performed automatically and the resulting SAN model can be solved by traditional solution methods. The solution of a SAN model can provide stationary results like throughput, servers utilization, response time and population of each queue. The FCQN models that can be handled by this method include, but are not limited to the following features: different classes of clients with or without priority; open and closed queueing systems with blocking due to restricted capacity; open systems with loss of clients due to restricted capacity or priority among classes; and variable routing patterns according to queues local states. The benefits of the use of SAN are related to other similar approaches in the conclusion.

keywords: performance evaluation, numerical solutions, finite capacity queueing networks, stochastic automata networks.

1 Introduction

Queueing Networks are certainly the most known formalism used in performance evaluation. The product-form solutions [1, 15] and the simplicity of modeling guarantee the popularity of Queueing Networks. However, most of the real problems do not fit well into the restrictions imposed by product-form assumptions, *e.g.*, the unlimited capacity of queues.

The use of Markov chains is a natural alternative formalism, but in this case the state-space explosion and the absence of product-form solutions do not recommend it. Nevertheless, the use of Stochastic Automata Networks (SAN) [6, 13] instead of straight-forward Markov chains can reduce the impact of the state-space explosion and provide efficient numerical solutions [6, 14].

The SAN formalism models a system as a collection of subsystems which interact occasionally. This approach can be easily adopted to model queueing systems, where each queue is a subsystem that interacts with the other queues by exchanging customers. Other types of interaction like blocking, priority, rerouting and mostly every real-world behavior can also be modeled with SAN. The only restriction, ironically, is that only finite capacity queues can be modeled, since SAN formalism only handles finite state-space problems. Therefore, in this paper we will be interested in a subset of the Queueing Networks formalism, called Finite Capacity Queueing Networks (FCQN).

The SAN formalism has the same power of description of the Markov chains. In fact the SAN formalism specifies a modular way to describe a Markov chain and a sparse method to store its infinitesimal generator as a sum of tensor products, called descriptor [7]. Therefore, stationary and transient solutions of any queueing network modeled as a SAN can be performed by traditional methods used to solve Markov chains.

In this paper we present a method to represent several FCQN models in SAN models. Specifically, we present techniques to represent FCQN with:

- different classes of clients with or without priority;
- open and closed queueing systems with blocking due to restricted capacity;

*This work is partially supported by research projects funded by CNPq, FAPERGS and INRIA.

- open systems with loss of clients due to restricted capacity or priority among classes;
- variable routing patterns according to queues local states.

Additionally, we present integration functions to the SAN model that allow us to compute stationary indexes such as throughput, servers utilization, response time and population of each queue. According to the proposed method, the translation of a queueing system to the correspondent SAN model can be performed automatically.

Section 2 presents a general overview of the FCQN formalism, with emphasis to the notation adopted in this paper to describe a FCQN model. Section 3 presents a general overview of the SAN formalism, with emphasis to the semantics of the SAN modeling primitives. To illustrate the conversion of a FCQN model to a SAN model, Section 4 presents four practical cases conversions from FCQN to SAN. Section 5 presents numerical results for each of the practical examples computed using a SAN solver, PEPS [19], by showing the memory and CPU costs to solve the models. The conclusion draws a generic comparison between our method and other solutions [2, 4, 8, 18] in terms of modeling flexibility, precision and computational costs to obtain a stationary solution.

2 Queueing Networks Formalism

The Queueing Networks formalism was introduced by Jackson in the 50's [9] and the major breakthroughs in this subject have been made with the product-form solutions proposed in the late 70's [1, 15]. The popularity of this formalism is based on a very intuitive idea of *costumers* (or *requests*) passing by *queues* (or *service centers*). A myriad of extensions to the basic formalism gives approximations [2, 8] and even propose some product-form solutions [4, 18]. However, it is very hard to combine all techniques proposed to extend the limits of the traditional product-form queueing networks.

2.1 FCQN definition

The method proposed in this paper can be applied to virtually any finite capacity queueing network, but in this paper we will be concerned with networks that can be described with the following parameters:

Let

M	The number of queues of the network;
R	The number of different classes of costumers;
K_i	The maximum number of costumers in the queue i (capacity);
C_i	The number of servers available in queue i;
S_i^r	The average time† needed to a server to serve one costumer of class r in queue i;
$P_{i,j}^r$	The routing probability† of a costumer of the class r which receives his service on queue i to leave this queue and to go to queue j (these parameters can be used to compute the average visit rate of costumers of class r in queue i, usually denoted by V_i^r);
$B_{i,j}^r$	The behavior of costumers of class r routed from queue i to queue j when queue j is full: *loss* (the costumer leaves the model) or *blocking*[1] (the queue i is blocked);
$\vec{D}_i$	The discipline of service stating the priority order among classes in the queue i, *e.g.*, $(2, 1, 3)$ indicates a higher priority for costumers of class 2, then for costumers of class 1 and finally costumers of class 3 (the absence of this parameter indicates no priority, *i.e.*, a *FCFS* discipline;
L_i^r	The average arrival rate† of costumers of class r at queue i from outside the model (workload for open models, usually denoted for all queues and all classes as $\vec{L}$);
N^r	The population of costumers of the class r in the network (workload for closed models, usually denoted for all classes as $\vec{N}$).

[1] Only open networks may have the *loss* option, since closed networks must be conservative and therefore can only have the *blocking* behavior.

The parameters with the sign † can be a discrete function of the current number of costumers of any queue over Real numbers.

The numerical performance indexes that can be computed with the method proposed in this paper are not limited to, but include:

Let

d_i^r The average throughput of costumers of class r through queue i;

u_i^r The average utilization index of queue i by costumers of class r;

n_i^r The average number of costumers of class r present in queue i (average population of costumers);

w_i^r The average response time of costumers of class r in queue i (wait and service time for costumers).

The literature of Queueing networks usually indicates these performance indexes as functions of the model workload, *e.g.*, $d_i^r(\vec{N})$ for closed networks or $d_i^r(\vec{L})$ for open networks.

3 Stochastic Automata Networks Formalism

In this section we introduce the basic concepts of the SAN formalism without a formal description. The reader interested in a formal description of the formalism can consult previous publications [6, 13, 5], or the original work on the subject [12].

The SAN formalism describes a complete system as a collection of subsystems that interact with each other. Each subsystem is described as a stochastic automaton, *i.e.*, an automaton in which the transitions are modeled by a continuous-time stochastic process[2]. The state of a SAN model, called *global state*, is defined by the combination of the states of all automata, each of them called *local state*. Figure 1 represents a SAN model with 2 automata completely independent and its equivalent Markov chain.

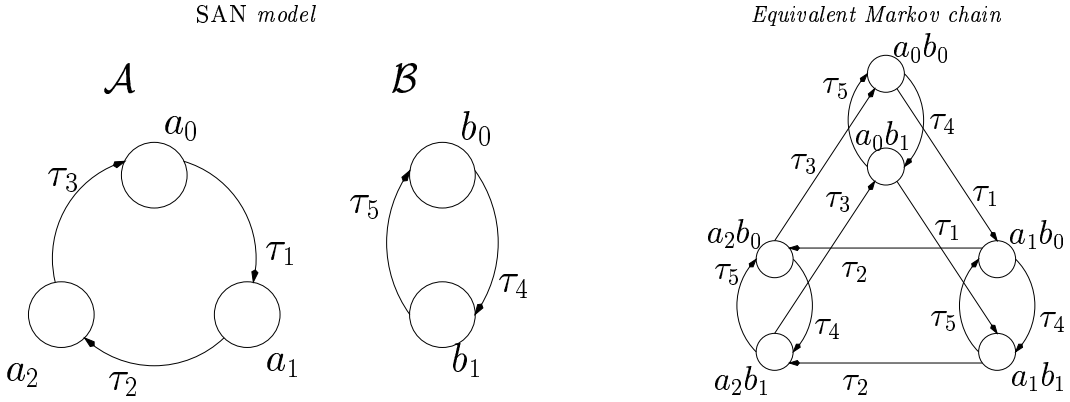


Figure 1: A SAN model with 2 independent automata

Note that in the model in Figure 1 there is no interaction between the two automata. In the following sections we will extend this model to illustrate the use of two SAN formalism primitives (the *synchronized events* and the *functional rates*) to represent the interactions among automata.

3.1 Synchronized Events

Two types of events can change the global state of a SAN model: *local events* and *synchronized events*. The local events change the global state by changing only one local state, *i.e.*, passing from a global state to another that differs only by one local state. The synchronized events can change simultaneously more than one local state, *i.e.*, two or more automata can change their local states

[2]Queueing Networks formalism usually describe the inter-arrival and service durations in a continuous-time scale. Therefore, in the context of this paper only continuous-time SAN will be considered, but discrete-time SAN can also be employed without any loss of generality.

simultaneously. The model in Figure 1 has only local events, Figure 2 represents a slightly different model where a synchronized event has been included.

The occurrence of a synchronized event *forces* all automata concerned to fire a transition corresponding to this event. Thus, the transitions representing a synchronized event will no longer be represented by a single rate (a non-negative real number), but those transitions will be represented by the *name* of the synchronized event (an identifier), its *firing rate* and its *probability of occurrence*:

- The *name* of the event is necessary to identity which transitions must fire simultaneously.
- The *firing rate* describes the rate in which the event occurs, but since it appears in all automata concerned by the event, one single automaton is chosen to indicate the event rate[3]. All other automata concerned by the event will indicate a symbolic rate equal to 1.
- The *probability of occurrence* establishes a relationship among all transitions corresponding to a same event that can be fired from the same local state, and therefore cannot be fired simultaneously.

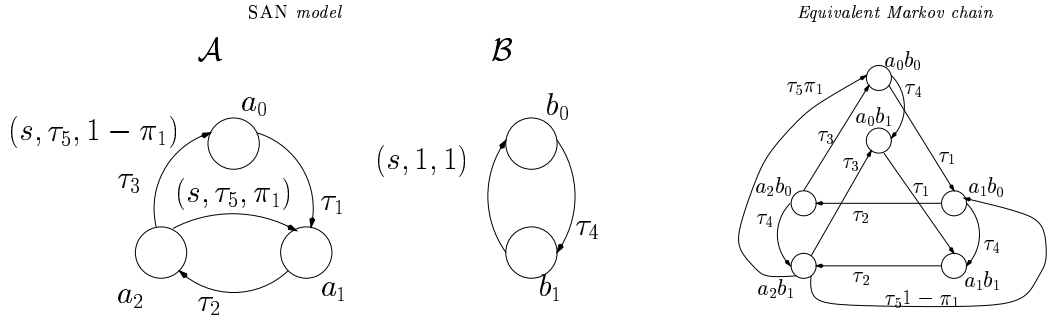


Figure 2: A SAN model with 2 automata and 1 synchronized event

In the model of Figure 2, transitions $a_0 \to a_1$, $a_1 \to a_2$, $a_2 \to a_0$, and $b_0 \to b_1$ represent local events. Transitions $a_2 \to a_0$, $a_2 \to a_1$, and $b_1 \to b_0$ represent the synchronized event s. Note that transition $a_2 \to a_0$ can be fired by the occurrence of a local event as well as by the occurrence of the synchronized event s. In this example, the occurrence of the synchronized event s (which will happen at rate τ_5) will cause one of the two situations:

- automata $\mathcal{A}$ will pass from state a_2 to state a_1 and at the same time automata $\mathcal{B}$ will pass from state b_0 to state b_1 (with probability π_1); or
- automata $\mathcal{A}$ will pass from state a_2 to state a_0 and at the same time automata $\mathcal{B}$ will pass from state b_0 to state b_1 (with probability $1 - \pi_1$).

3.2 Functional Rates

The use of functional rates is the second form of interaction among automata. A functional rate is no longer a non-negative Real number (like the ordinary rates), but a discrete function of the local state of some automata over the non-negative Real numbers. Figure 3 represents a variation of the example in Figure 2. In this new example, the transition of the state b_0 to the state b_1 is no longer independent of the automaton $\mathcal{A}$, but now it is a function called f defined as:

$$f = \begin{cases} \lambda_1 & \text{if automaton } \mathcal{A} \text{ is in the state } a_0; \\ 0 & \text{if automaton } \mathcal{A} \text{ is in the state } a_1; \\ \lambda_2 & \text{if automaton } \mathcal{A} \text{ is in the state } a_2; \end{cases}$$

In this model, the firing of the transition from state b_0 to b_1 will occur with rate λ_1 if automaton $\mathcal{A}$ is at state a_0, or with rate λ_2 if automaton $\mathcal{A}$ is at state a_2. If automaton $\mathcal{A}$ is at state a_1 the transition from state b_0 to b_1 will not occur. According to the SAN formalism, the expression of this function can be:

$$f = [\lambda_1 \, (st(\mathcal{A}) = a_0)] + [\lambda_2 \, (st(\mathcal{A}) = a_2)]$$

[3]This particular choice to indicate a rate of a synchronized event only in one automaton is arbitrary and it does not affect the semantics of the SAN model. More detailed information about synchronized events can be found in [5, 7].

The interpretation of a function can be viewed as the evaluation of an expression of non-typed programming languages, *e.g.*, C language. Each comparison is evaluated to 1 for true values and to 0 for false values.

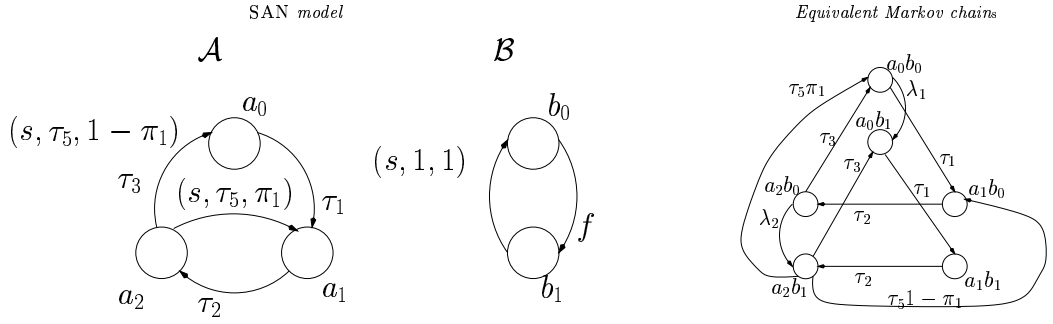


Figure 3: A SAN model with 2 automata, 1 synchronized event, and 1 functional rate

Note that the use of functional rates is not limited to local events. In fact, for synchronized events not only the event rate, but also the probability of occurrence, can be expressed as a function. The use of functional transitions is a powerful primitive of the SAN formalism, since it allows us to describe very complex structures in a very compact format. The computational costs to handle functional rates has decrease enormously with the recent developments of numerical solutions for SAN models, *e.g.*, the algorithms for generalized tensor products [6].

3.3 Performance Indexes

A SAN model can be used to compute stationary measures based on the computation of the eigenvector of the equivalent Markov chain. Therefore, integration functions over the resulting probability vector can be applied. Reward values can be associated to the probability of every and each global state. The format of such integration functions in SAN formalism is similar to the definition of functional rates. In the examples expressed in Figures 1, 2, and 3, integration functions can be defined to compute the probability of each local state. For example, the integration function "$st(\mathcal{A}) = a_0$" could be used to compute the probability of the local state a_0, *i.e.*, the sum of the probabilities of all global states where the automaton $\mathcal{A}$ is in state a_0.

4 Modeling Examples

The basic technique to model queueing networks with SAN is to describe each queue by one automaton, in which each local state represents the number of costumers in the queue. The exchange of costumers between two queues is represented by one synchronized event, which *forces* the two automata to change local states at the same time.

4.1 Open Queueing Networks with Loss and Blocking

In this section we will focus our attention in the difference between *loss* and *blocking* behaviors. Figure 4 describes a three queues open network with one single class of costumers. The main interest in this model is that it has one queue with a blocking behavior (for costumers from queue 1 to queue 2) and another queue with loss behavior (for costumers from queue 1 to queue 3).

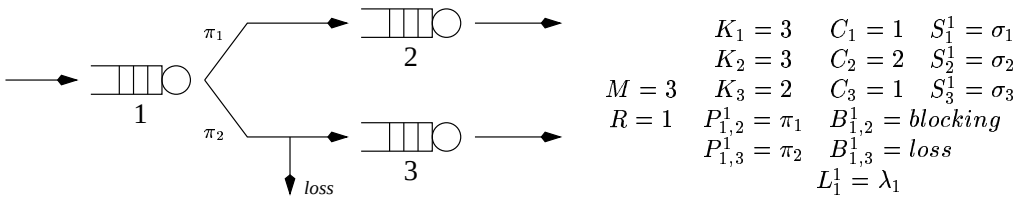


Figure 4: Open queueing network with loss and blocking

Figure 5 presents the SAN model equivalent to the FCQN model in Figure 4. For the example above, three automata ($\mathcal{A}^{(1)}$, $\mathcal{A}^{(2)}$ and $\mathcal{A}^{(3)}$) and two synchronized events were used. The synchronized events represent the passage of costumers from queue 1 towards queue 2 (event e_{12}) and the passage of costumers from queue 1 towards queue 3 (event e_{13}). The arrival of costumers in queue 1, and the departure of costumers from queues 2 and 3 are represented by local events. The occurrence of event e_{12} forces the simultaneous change of automata $\mathcal{A}^{(1)}$ and $\mathcal{A}^{(2)}$. When automata $\mathcal{A}^{(2)}$ is in local state 3, *i.e.*, queue 2 is full, event e_{12} cannot occur, which corresponds to the blocking of the departure of costumers from queue 1 to queue 2. For event e_{13}, the existence of an extra loop transition in the last state of automaton $\mathcal{A}^{(3)}$ allows not only the departure of costumers from queue 1 and simultaneously the arrival in queue 3 (when queue 3 is not full), but it also allows the loss of clients when queue 3 is full.

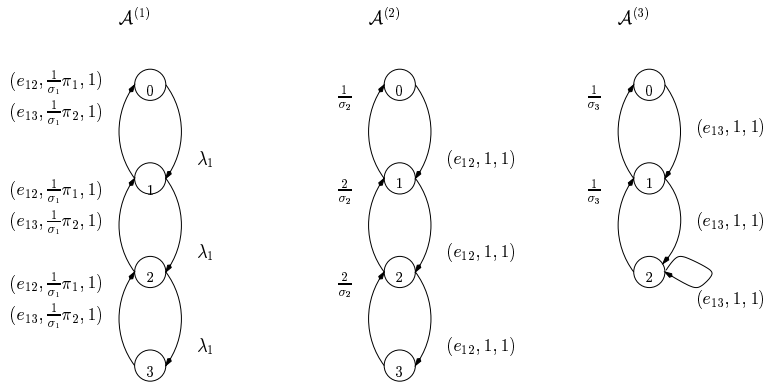


Figure 5: SAN model for an open queueing network with loss and blocking

The performance indexes for this model are simply extracted from weight over the probability of each local state. The average population of each queue is simply expressed by the weight of each global state by the number of costumers present in each queue[4]. The integration function to compute the average population of queue i is: "$st(\mathcal{A}^{(i)})$". The other performance indexes follow the same idea, *e.g.*, the average throughput integration function expresses the throughput of each global state.

4.2 Open Queueing Networks with Two Classes of Costumers and Priority

In this section the model in Figure 4 will be extended to include a second class of costumers. The costumers of class 2 enter in the model by queue 1 and go exclusively to queue 3 with a blocking behavior. Additionally, in queue 3, class 2 costumers are served with priority over class 1 costumers (see Figure 6).

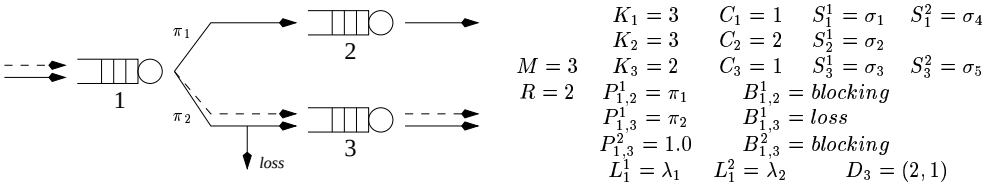


Figure 6: Open queueing network with two classes of costumers and priority

In this second example, queues visited by two classes of costumers are modeled by two automata. Thus, the SAN model for this example has five automata ($\mathcal{A}^{(1^1)}$, $\mathcal{A}^{(1^2)}$, $\mathcal{A}^{(2^1)}$, $\mathcal{A}^{(3^1)}$ and $\mathcal{A}^{(3^2)}$). There are three synchronized events to represent:

- the passage of a class 1 costumer from queue 1 to queue 2 (e_{12}^1);
- the passage of a class 1 costumer from queue 1 to queue 3 (e_{13}^1);

[4]It is important to remember that the integration functions are evaluated to each global state and the resulting value is multiplied by its computed probability.

- the passage of a class 2 costumer from queue 1 to queue 3 (e^2_{13}).

Using two automata to represent the same queue oblige us to define functions of automata local states to denote when a queue has reach its capacity. In the example above, it is necessary to define:

- $f_1 = \left(st(\mathcal{A}^{(1^1)}) + st(\mathcal{A}^{(1^2)}) \right) < K_1$, *i.e.*, queue 1 is not full;
- $f_3 = \left(st(\mathcal{A}^{(3^1)}) + st(\mathcal{A}^{(3^2)}) \right) < K_3$, *i.e.*, queue 3 is not full;
- $\bar{f}_3 = \left(st(\mathcal{A}^{(3^1)}) + st(\mathcal{A}^{(3^2)}) \right) \geq K_3$, *i.e.*, queue 3 is full.

These functions can be used to define a functional transition, *e.g.*, the local event of arrival at queue 1. For class 1 costumers the arrival at the first queue is expressed by: $\lambda_1\ f_1$, which is evaluated to 0.0 when the queue 1 is full, and as λ_1 otherwise. For class 2 costumers the arrival rate is similarly expressed by the function $\lambda_2\ f_1$, however functions can also be employed in a different way. Let us observe the description of the loss behavior of class 1 costumers leaving queue 1 to (possibly) enter queue 3, *i.e.*, the synchronized event e^1_{13}. Since, in all local states of automaton $\mathcal{A}^{(3^1)}$ queue 3 can be full (it depends on the local state of automaton $\mathcal{A}^{(3^2)}$), in all states of automaton $\mathcal{A}^{(3^1)}$ must exist a loop transition to be fired when queue 3 is full. For all local states of automaton $\mathcal{A}^{(3^1)}$ two transitions corresponding to the synchronized event e^1_{13} may occur:

- a transition corresponding to the arrival of the costumer in queue 3, if queue 3 is not full;
- a transition corresponding to the loss (automaton $\mathcal{A}^{(3^1)}$ remains at the same local state), if queue 3 is full.

This behavior is modeled by two alternative transitions that differ from each other by a probability expressed by a function, which in this case is the function f_3 to express that the queue is not full and $\bar{f}_3$ to express that the queue is full.

The class 2 costumers passage from queue 1 to queue 3 is also concerned by the fact that queue 3 is modeled by two automata. Because this passage has a blocking behavior, the synchronized event e^2_{13} must not occur if queue 3 is full. The SAN model describes this restriction defining the rate of the event e^2_{13} not only the inverse of the service time σ_4, but multiplying this rate by function f_3, *i.e.*, this event only will have a non zero rate is queue 3 is not full.

Finally, the priority of class 2 over class 1 costumers at queue 3 is expressed by another function. Class 1 costumers must be served in queue 3 only if there are no class 2 costumers in it. Function g_3 allows the service of class 1 costumers (local event in automaton $\mathcal{A}^{(3^1)}$) only when automaton $\mathcal{A}^{(3^2)}$ is in the local state 0:

$$g_3 \ = \ st(\mathcal{A}^{(3^2)}) \ = \ 0$$

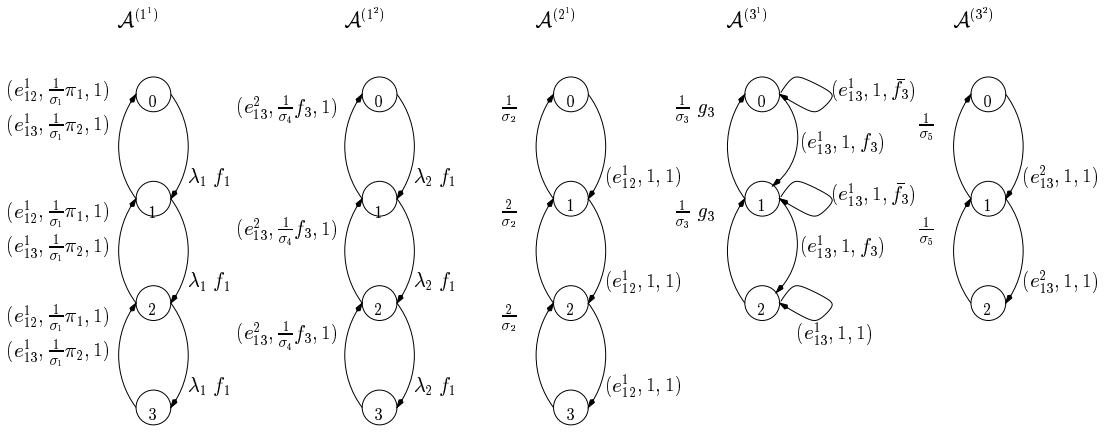


Figure 7: SAN model for an open queueing network with two classes of costumers and priority

An important remark to the proposed model is that not all the product state space of the SAN model is reachable. When two or more automata are used to describe the same queue, the SAN model includes non reachable global states. In the model above, some global states will not correspond to

valid situations of the FCQN model. The total number of costumers in queue 1 is limited to $K_1 = 3$, so all global states in which the sum of the states of automata $\mathcal{A}^{(1^1)}$ and $\mathcal{A}^{(1^2)}$ is greater than 3 are not reachable. There are 16 possible combinations of the local states for this two automata (4 states in each), but only 10 combinations are valid:

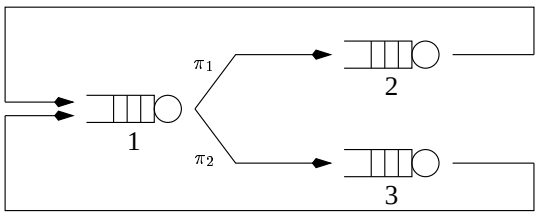
$(0,0)$	$(0,1)$	$(0,2)$	$(0,3)$	$(1,0)$	$(1,1)$	$(1,2)$	$(1,3)$	$(2,0)$	$(2,1)$	$(2,2)$	$(2,3)$	$(3,0)$	$(3,1)$	$(3,2)$	$(3,3)$
							×			×	×		×	×	×

The same phenomenon is observed in the representation of queue 3 (from the 9 possible combinations of automata $\mathcal{A}^{(3^1)}$ and $\mathcal{A}^{(3^2)}$, only 6 are valid). In this example, the product state space is equal to 576 states (the product of the size of each automata: $4 \times 4 \times 4 \times 3 \times 3$), but only 240 are reachable. Therefore to complete the SAN model is necessary to establish a reachability function that allows us to determine which states are reachable. The reachability functions have the same format as ordinary functions in stochastic automata, *i.e.*, a expression of the local states the automata. In this example, the reachability function depends on the states of the automata $\mathcal{A}^{(1^1)}$, $\mathcal{A}^{(1^2)}$, $\mathcal{A}^{(3^1)}$ and $\mathcal{A}^{(3^2)}$:

$$reachability \;=\; \left[\left(st(\mathcal{A}^{(1^1)}) + st(\mathcal{A}^{(1^2)})\right) \leq 3\right] \text{ and } \left[\left(st(\mathcal{A}^{(3^1)}) + st(\mathcal{A}^{(3^2)})\right) \leq 2\right]$$

4.3 Closed Queueing Networks with Dependable Service Rate

We proceed considering now the closed FCQN model of Figure 3 with three queues in which the service time of the first queue is a function of the population of the other two queues. We will assume the first queue serving slower when the other two queues have more costumers. In this example, we define numerically the average service time of queue 1 varying from σ_1 when the other queues have all costumers but one, to $\sigma_1/5$ when the others queues are empty[5].



$$\begin{array}{llll} & K_1 = 3 & C_1 = 1 & S_1^1 = f_1 \\ M = 3 & K_2 = 3 & C_2 = 2 & S_2^1 = \sigma_2 \\ R = 1 & K_3 = 2 & C_3 = 1 & S_3^1 = \sigma_3 \\ N = 5 & P_{1,2}^1 = \pi_1 & & B_{1,2}^1 = blocking \\ & P_{1,3}^1 = \pi_2 & & B_{1,3}^1 = blocking \end{array}$$

$$f_1 = \sigma_1 \tfrac{n_2+n_3+1}{5}$$

Figure 8: Closed queueing network with dependable service rate

Figure 9 represents the SAN model of the example in Figure 8. In this model the service rate of queue 1 is described with the help of a function defined as:

$$f_1 = \frac{st(\mathcal{A}^{(2)}) + st(\mathcal{A}^{(3)}) + 1}{5}$$

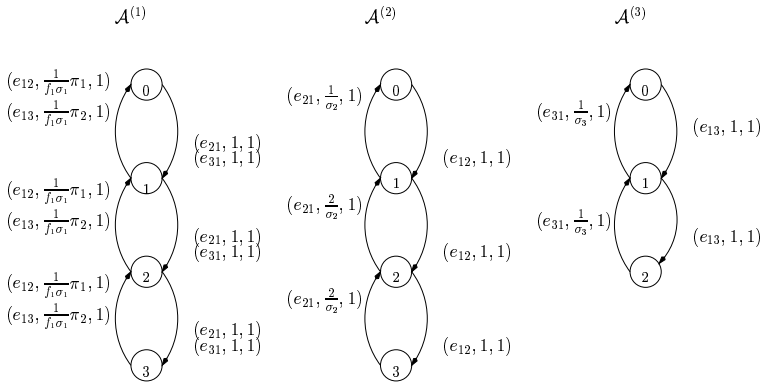


Figure 9: SAN model for a closed queueing network with dependable service rate

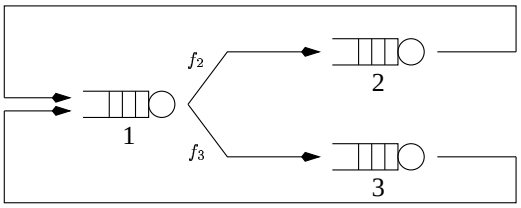
[5]Note that when all costumers are in the other queues, the first queue is empty. Therefore, its service time is irrelevant.

This SAN model will present a clear disadvantage because of the number of unreachable global states. Only the global states in which the sum of costumers in all queues is equal to the number of clients (N) will be reachable. For this example, the product state space is equal to 48 states, but only 9 are reachable. The reachability function of the model in Figure 9 is:

$$reachability = \left(\sum_{i=1}^{M} st(\mathcal{A}^{(i)}) \right) = N$$

4.4 Closed Queueing Networks with Dependable Routing Pattern

This last example (Figure 10) illustrates the inclusion of dependable routing pattern in the previous example (Figure 8). For this model we will consider that the costumers exiting queue 1 will be sent to the shortest queue. In fact, the routing probability to the queue with less costumers will be 1.0 and if both queues have the same number of costumers, the routing probability will be of 0.5 to each queue.



$$\begin{array}{lllll} & K_1 = 3 & C_1 = 1 & S_1^1 = f_1 \\ & K_2 = 3 & C_2 = 2 & S_2^1 = \sigma_2 \\ M = 3 & K_3 = 2 & C_3 = 1 & S_3^1 = \sigma_3 \\ R = 1 & P_{1,2}^1 = f_2 & & B_{1,2}^1 = blocking \\ N = 5 & P_{1,3}^1 = f_3 & & B_{1,3}^1 = blocking \end{array}$$

$$f_1 = \sigma_1 \tfrac{n_2+n_3+1}{5}$$

$$f_2 = (n_2 < n_3) \text{ or } 0.5 * (n_2 = n_3)$$

$$f_3 = (n_2 > n_3) \text{ or } 0.5 * (n_2 = n_3)$$

Figure 10: Closed queueing network with dependable service rate and routing pattern

The example above can be described by the SAN model in Figure 11. Analogously to the previous model (Figure 9) this model has the same characteristics concerning the state space.

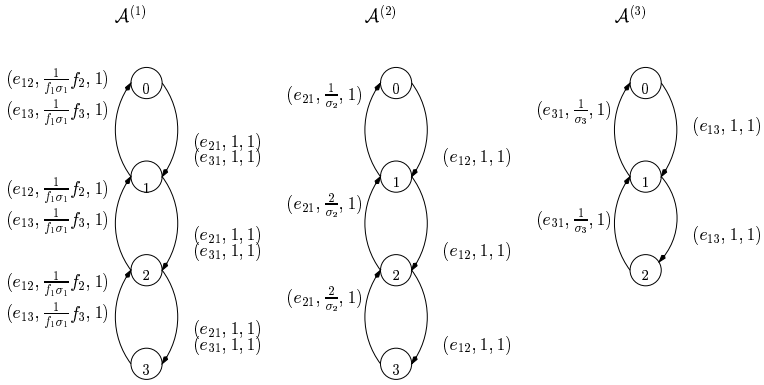


Figure 11: SAN model for a closed queueing network with dependable service rate and routing pattern

5 Numerical Results

The numerical experiments in this section were performed with the software tool PEPS, version 2000 [19]. PEPS - Performance Evaluation of Parallel Systems - is an academic tool to model and solve SAN models by iterative methods, *e.g.*, power method, Arnoldi or GMRES [16, 17]. The experiments were performed on an IBM-PC workstation (Pentium II, 366 MHz) with 64 Mbytes of memory (265 Mbytes of virtual memory) running under Linux operating system (Red Hat distribution version 6.1). The PEPS 2000 software is programmed in C++ language and its code was generated with gcc compiler full optimization option (-O3). All examples in this paper, as well as the PEPS code, can be retrieved free of charge for academic purposes in PEPS web page [19].

Table 1 summarizes the memory space used to store and the time spent to solve the examples of the previous section. It also includes the same examples considering larger queue capacities. The columns of Table 1 represent the following information about the SAN model:

Models				*pss*	*rss*	*mem*	*time*
SAN in Section 4.1	$K_1 = 3$	$K_2 = 3$	$K_3 = 2$	48	48	2	0.02
	$K_1 = 30$	$K_2 = 24$	$K_3 = 15$	12 400	12 400	101	2.53
SAN in Section 4.2	$K_1 = 3$	$K_2 = 3$	$K_3 = 2$	576	240	8	0.09
	$K_1 = 30$	$K_2 = 24$	$K_3 = 15$	6 150 400	1 686 400	48 059	864.30
SAN in Section 4.3	$K_1 = 3$	$K_2 = 3$	$K_3 = 2$	48	9	2	0.03
	$K_1 = 30$	$K_2 = 24$	$K_3 = 15$	12 400	200	102	11.27
SAN in Section 4.4	$K_1 = 3$	$K_2 = 3$	$K_3 = 2$	48	9	2	0.04
	$K_1 = 30$	$K_2 = 24$	$K_3 = 15$	12 400	200	102	16.10

Table 1: Memory use to store and CPU time to solve the SAN models in PEPS

- *pss*: product state space size:

  The product of the size of each automata is the main restriction to the SAN efficiency. Therefore, the product state space size expresses the problem size due to the SAN modeling approach.

- *rss*: reachable state space size:

  The reachable state space size expresses the real problem size in the user point of view, *i.e.*, if other methods, like straight-forward Markov chain, were adopted the problem could be reduced to that size.

- *mem*: memory requirements to store the model in Kbytes:

  The main advantage of tensor algebra approaches is the relatively small memory needs to store the infinitesimal generator of the equivalent Markov chain. This information gives the memory needs of the SAN model internal format adopted by PEPS tool.

- *time*: CPU time to perform 100 iterations steps in seconds:

  According to the numeric values adopted in each of the examples, the number of iterations necessary to found a solution may change, but the CPU time to perform one iteration depends only on the SAN structure, *e.g.*, number of synchronized events, number of functions, size of the automata. Therefore, this information allows us to establish a relationship among the computational cost of each example.

Observing Table 1 is easy to conclude that the proposed method is more suitable for open single class networks than for models with queues visited by multiple classes or closed networks. In both cases the SAN model suffers from the very low number of reachable global states into the product state space. This disadvantage do not appears clearly on the memory needs, which are relatively small, but the computational costs may easily become critical. It can also be observed that the impact of functional information, *e.g.*, dependable service rate, has an impact in comparison with the "constant cases" like the first example. However, once functional information already exists in the model, the addition of other dependable information is not very relevant. This can be observed by comparing the CPU time on the first model (with no functional information) with the results for the third model (with dependable service time) and the last model (with dependable service time and dependable routing).

6 Conclusion

Any Finite Capacity Queueing Networks with a Markovian (memoryless) behavior can be described by the method presented in this paper. The use of Stochastic Automata Networks allows the description of complex costumers exchange behaviors among queues. In fact, even other characteristics not exemplified in this paper could be incorporated to the proposed method. For example, complex fork and join behaviors (splitting and assembling costumers) could be easily modeled by the synchronized events. In all cases described in Section 4, the synchronized events were used to associate the departure of a costumer in one queue to the arrival into another one, establishing an one-to-one relationship. Fork and join procedures could correspond to the association of the departure from n queues with the arrival in m queues.

The proposed method can also extend the queue behavior from a simple *loss* or *blocking* choice to more complex behaviors. Other types of blocking behavior could be considered according to the

enormous possibilities of SAN modeling. In the case presented in Section 4.2, the blocking behavior of queue 3 stops the departure of class 2 costumers from queue 1, but queue 1 continues to serve class 1 costumers. A different behavior could be considered, *e.g.*, when queue 1 stops the departure of class 2 costumers it could also stops the departure of class 1 costumers. To summarize, the modeling possibilities of the SAN allow us to redefine the FCQN to include a myriad of new modeling primitives.

The main disadvantage of the proposed technique remains the numerical solution costs, which are not as difficult to handle as the straight-forward Markov chains techniques. The memory needs are reasonable, but the computational cost is still too high for large models. Nevertheless, the recent works in the subject have reduced the impact of the state explosion problem. Functional transitions have efficient algorithms to compute stationary solutions and on going works about product state space reduction [10], symmetries [11] and product-form solutions [14] have been proposed. The points carried out by these and other related works allow us to think that in the next decade the computational costs to solve a SAN model will reduce considerably.

Acknowledgments

The authors would like to express their gratitude to Professor Avelino Francisco Zorzo by his very pertinent and helpful remarks on the text.

References

[1] F.Baskett, K.M.Chandy, R.R.Muntz, F.G.Palacios. *Open, closed and mixed networks of queues with different classes of customers.* **Journal of the ACM**, vol. 22, no. 2, 1975, pp. 248-260.

[2] Y.Dallery, Z.Liu, D.Towsley. *Equivalence, reversibility, symmetry and concavity properties in fork-join queuing networks with blocking.* **Journal of the ACM**, vol. 41, no. 5, 1994, pp. 903-942.

[3] M.Davio. *Kronecker products and shuffle algebra.* **IEEE Transactions on Computers**, vol. C-30, no. 2, 1981, pp. 116-125.

[4] A.Economou, D.Fakinos. *Product form stationary distributions for queueing networks with blocking and rerouting.* **Queueing Systems**, vol. 30, 1998, pp. 251-260.

[5] P.Fernandes. **Méthodes numériques pour la solution de systèmes Markoviens à grand espace d'états**. INPG, Grenoble, 1998. (Ph.D. thesis)

[6] P.Fernandes, B.Plateau, W.J.Stewart. *Efficient descriptor-vector multiplication in stochastic automata networks.* **Journal of the ACM**, vol. 45, no. 3, 1998, pp. 381-414.

[7] P.Fernandes, B.Plateau, W.J.Stewart. *Optimizing tensor product computations in stochastic automata networks.* **Operations Research - RAIRO**, vol. 32, no. 3, 1998, pp. 325-351.

[8] A.Harel, S.Namn, J.Sturm. *Simple bounds for closed queueing networks.* **Queueing Systems**, vol. 31, 1999, pp. 125-135.

[9] J.R.Jackson. *Networks of waiting lines.* **Operations Research**, vol. 5, 1957, pp. 518-521.

[10] P.Kemper. *Numerical Analysys of superposed GSPNs.* **IEEE Transactions on computers**, vol. C-22, no. 9, 1996, 615-628.

[11] A.S.Miner, G.Ciardo, S.Donatelli. *Using the excat state space of a Markov model to compute approximate stationary mesures.* **Proceedings of the ACM SIGMETRICS**, Santa Clara, CA, USA, June, 2000, pp. 207-216.

[12] B.Plateau. **De l'Evaluation du Paralllisme et de la Synchronisation**. Paris-Sud, Orsay, 1984. (Ph.D. thesis)

[13] B.Plateau, K.Atif. *Stochastic automata networks for modelling parallel systems.* **IEEE transactions on software engineering**, vol. 17, no. 10, 1991, pp. 1093-1108.

[14] B.Plateau, W.J.Stewart. **Stochastic automata networks: product forms and iterative solutions**. INRIA, Rapport de Recherche no. 2939, 1996. Anonymous ftp *ftp://ftp.inria.fr/INRIA/Publication/RR/RR-2939.ps.gz*

[15] M.Reiser, S.S.Lavenberg. *Mean-value analysis of closed multichain networks.* **Journal of the ACM**, vol. 27, no. 2, 1980, pp. 313-322.

[16] Y.Saad. **Iterative methods for sparse linear systems**. Boston: PWS Publishing Company, 1995.

[17] W.J.Stewart. **Introduction to the numerical solution of Markov chains**. Princeton University Press, 1994.

[18] D.Towsley. *Queueing network models with state-dependent routing.* **Journal of the ACM**, vol. 27, no. 2, 1980, pp. 323-337.

[19] **PEPS Project**: Performance Evaluation of Parallel Systems. *http://www-apache.imag.fr/software/peps/*