On monotonicity of finite capacity queueing networks and their perfect simulation

Jean-Marc Vincent*
Laboratoire ID-IMAG,Projet INRIA APACHE,
51, avenue Jean Kuntzmann, F-38330 Montbonnot, France
Jean-Marc.Vincent@imag.fr

March 3, 2005

Abstract

Traditional product form property of Markovian queueing networks usually is vanished when capacity of queues are finite and clients are blocked or rejected. A new efficient simulation method, derived from Propp & Wilson (Propp 1996) perfect simulation, is applied to the finite capacity queues context. We present an algorithm that samples directly states of the network according to stationary distribution. This method has been applied to queueing networks with various monotonous routing policies.

It is shown that, according to an adequate uniformization techniques usual Markovian queuing networks are monotonous. Such monotonous networks include networks with overflow or blocking, join the shortest queue routing policy, fork of customers... Consequently, perfect simulation could be improved by drawing trajectories from minimum and maximum states which reduces computation time. Moreover, for the estimation of a monotonous reward function, the simulation time could be reduced drastically as in (Vincent & Marchand 2004).

Some examples are given : loss estimation on Erlang models, usage of the last queue in a line of queues with blocking, saturation estimation for a multi-stage interconnection switch...

1 Introduction

Queuing systems are of fundamental interest for modeling communication networks, production lines, operating systems,.... Servers represent the access of customers to resources and queue capacity allows modeling of resource contention and storage before service. Two kinds of dimensioning are needed for systems optimization. Time dimensioning have to fix servers speed and space dimensioning define memory capabilities of nodes. In all cases, the estimation of service quality are useful before the system deployment.

Under Markovian assumptions (Poisson arrivals, exponential service time, probabilistic routing etc.), it has been shown that the network of queues is modelled by a multidimensional Markov jump process. Then the system performances are computed from the steady-state distribution of the process. Fortunately, when queues have an infinite capacity the steady-state distribution is product-form and could easily be computed in a reasonable time (Bolch, Greiner, de Meer & Trivedi 1998). In some cases the hypothesis of infinite capacity could be released, preserving the product form (Perros 1994, Balsamo, De Nitto Person & Onvural 2001). Unfortunately, in most cases, the steady state distribution is not in a product form and adequate approximation techniques should be applied. Many works cover the domain of queuing networks with finite capacity, bibliographies of (Onvural 1990, Balsamo, De Nitto Person & Inverardi 2003) provide pointers to related works.

Simulation approaches are alternative methods to estimate quality of service of such networks. Based on discrete event simulation (Banks, Carson, Nelson & Nicol 2001) or on Markov properties (MCMC methods) (Brémaud 1999), simulations estimate the steady-state distribution on long run trajectories. Drawbacks of simulations are the control of the warm up period or burn-in time (Robinson 2002) and the influence of the initial state on stochastic behavior. Moreover, because statistics are made on long run trajectories, assumptions on asymptotic independence are difficult to justify.

In this paper a specific simulation technique, “perfect simulation”, is used to sample the steady state distribution. Based on Propp & Wilson ideas (Propp 1996), an algorithm is proposed. By proving the monotonicity of

*This work was partially supported by ACI SurePath project and DECORE-IMAG project

queuing networks with blocking and rejection, we improve drastically simulation time. This technique have been applied on classical models to validate statistically the approach and on huge models to demonstrate the efficiency of the method.

The second section of this article establish the construction of a Markovian queuing network as a composition of independent monotone events. Section 3 is devoted to the description of perfect sampling technique and improvements obtained by monotonicity properties. Section 4 presents typical examples of non-product form Markovian queuing networks and gives some experimental results.

2 Multi-dimensional Markov jump process

2.1 Events in queuing networks

Consider a queueing network with K queues. The state space of each queue Q_i is the set of integers $\mathcal{X}_i = \{0, \cdots, C_i\}$, where C_i is the capacity of queue Q_i. The state space $\mathcal{X}$ of the system is the Cartesian product of all $\mathcal{X}_i$;

$$\mathcal{X} = \mathcal{X}_1 \times \cdots \times \mathcal{X}_K.$$

The natural order on integer is extended to a partial order on $\mathcal{X}$ using component-wise ordering.

Definition 1 (Event)
An event e is an application defined on $\mathcal{X}$, that associates to each state $x \in \mathcal{X}$ a new state denoted by $\Phi(x, e)$. Φ is called the **transition function** *of the system.*

One should note that the transition function is defined on $\mathcal{E} \times \mathcal{X}$. It is convenient to include inside the transition function the fact that some events could not be applied to a state. For example, the event *end of service* could be executed only if the number of customers in the queue is greater than one. As an example, we consider that applying an *end of service* event to an empty queue remains the global state unchanged (skip operation).

In a queueing network, a customer arrival, the end of a service and the following routing, a customer departure, are typical events in networks. The transition corresponding to an arrival in queue Q_i is an increment of x_i provided that $x_i < C_i$. In that case one should precise the routing policy (rejection, overflow on another queue,...).

Denote by $\mathcal{E} = \{e^1, \cdots, e^p\}$ the set of events. Usually, this set is supposed to be finite.

Definition 2 (Execution)
An **execution** *of the system is defined by an initial state $x_0 \in \mathcal{X}$ and a sequence of events $e = \{e_n\}_{n\in\mathbb{N}}$. The sequence of states $\{x_n\}_{n\in\mathbb{N}}$ defined by the recurrence $x_{n+1} = \Phi(x_n, e_{n+1})$ for $n \geqslant 0$ is called a* **trajectory**.

Coupling has been used in many ways in Markov chain analysis (Brémaud 1999) or (Lindvall 1997). In this paper, coupling is related to trajectories having the same sequence of generating events. This is a stronger definition than (Lindvall 1997) when probability measures are given on the initial value and the sequence of events.

Definition 3 (Coupling)
A sequence of events $e = \{e_n\}_{n\in\mathbb{N}}$ is said to be globally coupling if there exist some N such that the state at time N (and so after) does not depend on the initial state. The minimum value of N is called the global forward coupling time and is noted τ.

Definition 4 (Monotonous events)
An event $e \in \mathcal{E}$ is said to be **monotonous** *if it preserves the partial ordering on $\mathcal{X}$. That is*

$$\forall (x, y) \in \mathcal{X} \;\; x \leqslant y \;\Rightarrow\; \Phi(x, e) \leqslant \Phi(y, e).$$

If all events are monotonous, the global system is said to be monotonous.

The monotonicity property is of fundamental for improvement of simulation. Denote by M, (respectively m), the set of all maximal, (respectively minimal), elements of the finite partially ordered state space $\mathcal{X}$. Then

Proposition 1
For a given infinite sequence of events $e = \{e_n\}_{n\in\mathbb{N}}$. If all trajectories issued from any initial state in $M \cup m$ couple then global coupling occurs.

In the case when the state space is the Cartesian product of $\{0, \cdots, C_i\}$ there is a unique minimum state $(0, \cdots, 0)$ and a unique maximum state $(C_1, \cdots, C_K)$. So that it is sufficient to build 2 trajectories to capture the behavior of the system and compute coupling time.

2.2 Uniformization

Now, to achieve the model construction of the Markov process, a Poisson process with intensity λ_j is associated to each event e_j. These Poisson processes are supposed to be independent.

Theorem 1 (Uniformized process)
The uniformized process driven by the Poisson process with rate $\Lambda = \sum_{j=1}^{p} \lambda_i$ and generating at each time of the process an event $e \in \mathcal{E}$ according to the probability distribution $(\frac{\lambda_1}{\Lambda}, \cdots, \frac{\lambda_p}{\Lambda})$ is equivalent to the queueing network Markov process.

The proof of this result is obtained by writing down the infinitesimal transition equations. One should notice that the idea is to introduce the independence between events, when an event could not be applied to one state x the state is not changed (skip operation), the method is analogous with the rejection method in stochastic simulation. The forward simulation algorithm is then derived :

Algorithm 1 Forward simulation of a Markov chain

$x \leftarrow x_0$; {choice of the initial value of the process}
repeat
  $e \leftarrow$ Generate-event() {generation of event e according to the distribution $(\frac{\lambda_1}{\Lambda}, \cdots, \frac{\lambda_p}{\Lambda})$}
  $x \leftarrow \Phi(x, e)$; {computation of the next state x_{n+1} with event e}
until stopping criteria
return x

Moreover, because generated events are independent a simple global coupling condition could be established.

Proposition 2
Suppose there exist some finite sequence (pattern) of event $\{e_1, \cdots, e_k\}$ such that the image of $\mathcal{X}$ by $\Phi(\Phi(\cdots, \Phi(\mathcal{X}, e_k), e_{k-1}), \cdots, e_1)$ is reduced to one point, then the random sequence generated by the algorithm is globally coupling almost surely.

The proof is obvious because of independence of events. The probability that the pattern never occurs is 0. So after a sufficiently large number of iterations, the state of the chain does not depend on the initial state.

Proposition 3 (Uniformized monotonous system)
Provided that the system is monotonous, the discrete time Markov chain embedded in the uniformized process is also monotonous.

Consequently, global coupling could be obtained by conditions like : there exist a sequence of events that flush the system, it will be the case for all examples presented in the last section. The representation of the chain and the coupling condition is given by the model construction.

2.3 Two queues example

For example consider the following system described in figure 1.

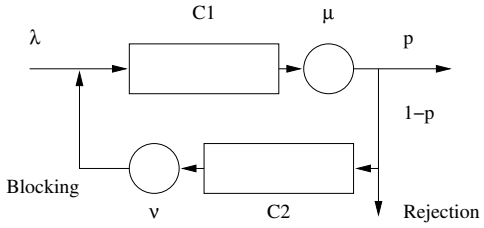

Figure 1: Queues with 2 types of contention

There are 4 events:

e_1 customer external arrival with rate $\lambda_1 = \lambda$.

e_2 end of service of the customer on the first queue and routing outside the network with rate $\lambda_2 = p\mu$.

e_3 end of service on the first queue and routing to the second queue, if the second queue is full the customer is lost. The rate associated to e_3 is $\lambda_3 = (1-p)\mu$.

e_4 end of service on the second queue, if the first queue is full the customer is blocked in the second queue and start again a service. The rate associated to e_4 is $\lambda_4 = \nu$.

End of service events, e_2, e_3 and e_4 make a transition if the corresponding queue is not empty. When the queue is empty the event is just a skip operation.

The uniformized process has a rate $\Lambda = \lambda + \mu + \nu$ and the sequence of C_1 events e_2 followed by a sequence of $C2$ events e_4 and e_2 flush the system. The system is globally coupling almost surely.

3 Monotone events in queuing networks

The difficulty for detecting monotonicity property in queueing networks is to define simultaneously a state-space structure and events on the structure. Some ideas developped in this section could be find in (Vincent 2005).

3.1 Finite capacity single server queue

If we consider a single server queue defined by an input Poisson process (rate λ) and exponential service time (rate μ). Denote by C the queue capacity The state space associated to the queue is $\mathcal{X}) = \{0, \cdots, C\}$. Two types of

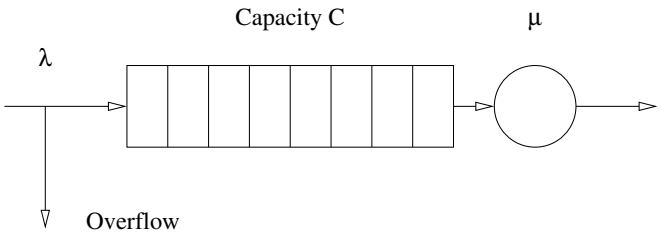

Figure 2: Single server queue $M/M/1/C$

events occur in the system :

Type	Rate	Action on $x \in \mathcal{X}$
arrival	λ	$x \longmapsto \min\{x+1, C\}$
departure	μ	$x \longmapsto \max\{x-1, 0\}$

It is clear that arrival and departure are monotone events as composition of $\max$, $\min$, $+$ functions.

3.2 Erlang model

Consider K servers with rates $\mu_1, \cdots, \mu_K$. In such model, because servers are not identical, one should precise a priority order when the system is partially empty. It is supposed that the customer enters the first non-empty server in the classical different a single server queue defined by an input Poisson process (rate λ) and exponential service time (rate μ). Denote by C the queue capacity The state space associated to the queue is $\mathcal{X}) = \{0,1\}^K$. $(K+1)$

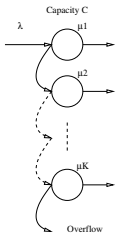

Figure 3: Single server queue $M/M/1/C$

events are defined as follows :

Type	Rate	Action on $x = (x_1, \cdots, x_K) \in \mathcal{X}$
arrival	λ	$x_i \longmapsto x_j + x_1 \dots x_{j-1}(1 - x_j)$
departure on server i	μ_i	$x_i \longmapsto \max\{x_i - 1, 0\}$

It is clear that departures are monotone events. For the arrival case, one should note that

$$x_1 \dots x_{j-1}(1 - x_j) = \begin{cases} 1 & \text{if } j \text{ is the first non-empty server,} \\ 0 & \text{in other cases.} \end{cases}$$

With this algebraic expression, each product added to the state vector is monotonous, so is the event.

3.3 Finite capacity multiple server queue

For the $M/M/K/C$ queue, events are just a combination of the two previous cases. The state space is $\{0, 1\} \times \cdots \times \{0, 1\} \times \{0, \cdots, C - K\}$. The K first coordinates indicate if the server is occupied and the last coordinate corresponds to customers in the waiting room.

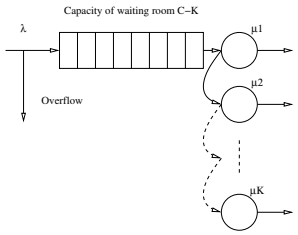

Figure 4: Multiple server queue $M/M/K/C$

For arrivals, the case is the same as in the Erlang model. For a departure on server i, we have to check if there are available customers in the waiting room. But the event is still monotonous.

3.4 Routing : rejecting or blocking

Consider now a typical routing event e, occurring in a queueing network. This event is defined by the origin queue Q_i and a list of destinations $Q_{j_1}, Q_{j_2}, \cdots, Q_{j_i}$ with the following semantic :
- if Q_i is empty do nothing;
- if Q_i is not empty find the first non empty queue Q_{j_k} in the list, decrements the number of customers in Q_i and increments the number of customers in Q_{j_k}
- if all queues are full then reject the customer out of the network.

Proposition 4 (Monotonicity of routing and rejection)
A routing event with rejection if all destination queues are full is a monotone event.

The proof is done by exploring all possibilities. Let $(x, y) \in \mathcal{X}^2$, such that $x \leqslant y$. Let e be a routing event defined by its origin Q_i and its list of destinations $Q_{j_1}, Q_{j_2}, \cdots, Q_{j_i}$.
- if $y_i = 0$ then $x_i = 0$ and the event does not modify state x and y; $\Phi(x, e) = x \leqslant \Phi(y, e) = y$;
- if $y_i > 0$ and $x_i \geqslant 0$, then $\Phi(y, e)|_i = y_i - 1 \geqslant x_i - 1 = \Phi(x, e)|_i$. Let Q_{j_k} the first non saturated queue in state y. Because $x \leqslant y$ the first non saturated queue for state x is either strictly before Q_{j_k} in the list and the corresponding queue is saturated in state y or Q_{j_k} is the first non saturated queue in state x and y. In both cases, the order is preserved.

The case of blocking queues is very similar to the previous case. The routing event is defined by the origin queue Q_i and a list of destinations $Q_{j_1}, Q_{j_2}, \cdots, Q_{j_i}$ with the following semantic :
- if Q_i is empty do nothing;
- if Q_i is not empty find the first non empty queue Q_{j_k} in the list, decrements the number of customers in Q_i and increments the number of customers in Q_{j_k}
- if all queues are full then block the customer in Q_i.

Proposition 5 (Monotonicity of routing and blocking)
A routing event with blocking in the original queue if all destination queues are full is a monotone event.

The proof is similar to the proof of the previous proposition and was established by (Mattson 2002).

Clearly arrivals from outside or routing directly outside (end of service) are also monotone events.

3.5 State dependent routing

In queuing networks routing could depend on queue state. That is the case for the "Join the shortest queue policy". We suppose that a priority is given for the queues, such that when shortest queues have the same number of customers the queue with higher priority is chosen.

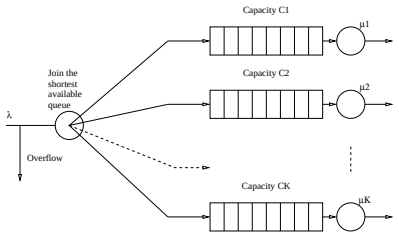

Figure 5: Join the shortest queue with K destination queues

The state space of the system is clearly $\{0, \cdots, C_1\} \times \cdots, \{0, \cdots, C_K\}$. Departure events are equivalent to departures in a $M/M/1$ queue and so are monotonous. For arrivals, consider states x and y such that $x \leqslant y$. Let i be the the index of the shortest queue for x and j for state y. If $i = j$ then $\Phi(x, e)|_i = x_i{+}1 \leqslant y_i{-}1 = \Phi(y, e)|_i$. If $i \neq j$ then $x_i \leqslant x_j \leqslant y_j < y_i$ and so $\Phi(x, e)|_i = x_i{+}1 \leqslant y_i = \Phi(y, e)|_i$ and $\Phi(x, e)|_j = x_j \leqslant y_i{+}1 = \Phi(y, e)|_i$. When queues are full, the proof is similar.

3.6 Monotone networks

It could be shown also that monotonous events appear in many situations. For example, fork events are monotonous but, join events are not. Negative customers (Chao, Miyazawa & Pinedo 1999), a generalization of join operation is monotonous only the case when external arrival of negative customers delete some customers.

For priority queues, one should note that the preemptive policy ensure the monotonicity.

Theorem 2 (Monotonous Markovian networks)
A Markovian network of multi-servers queues and finite capacity with rejection routing or blocking, state dependant monotonous routing is monotonous.

This result is clear according to the previous propositions. This result is an extension of a work from Mattson (2002) on blocking.

4 Perfect sampling method

In this section, the perfect sampling method is detailed for the general case then improved for monotonous systems and finally applied to queuing networks.

4.1 Global state iteration

Formally, when all the knowledge of the dynamics is included in the state description, the system is described by the transition function Φ, typically

$$X_{n+1} = \Phi(X_n, U_{n+1}); \tag{1}$$

where X_n is n^{th} observed state of the system, and $\{U_n\}_{n\in\mathbb{Z}}$ the sequence of inputs of the system, typically a sequence of calls to a `Random` function. This type of stochastic recursive sequence has been widely studied in a general framework (Borovkov & Foss 1994) or (Diaconis & Freedman 1999) and some results related with perfect simulation may be found in (Stenflo 1998, Stenflo 2001).

It is clear that, if the $\{U_n\}$ are independent and identically distributed, the process $\{X_n\}_{n\in\mathbb{Z}}$ defined by an initial value X_0 and the recursive equations (1) is a Markov chain. Conversely, given a transition matrix P, it is possible to find a transition function Φ such that a Markov chain defined by (1) has transition matrix P (Vincent & Marchand 2004).

Algorithm 2 Backward-coupling simulation (general version)

for all $x \in \mathcal{X}$ **do**
  $y(x) \leftarrow x$ {choice of the initial value of the vector y, $n = 0$}
end for
repeat
  u $\leftarrow$ Random; {generation of u_{-n}}
  for all $x \in \mathcal{X}$ **do**
    $y(x) \leftarrow y(\Phi(x, u))$; {computation of the state at time 0 of the trajectory issued from x at time $-n$}
  end for
until All $y(x)$ are equal
return $y(x)$

Based on a stochastic recurrent sequence formulation, the following algorithm provides directly a sample of the steady state distribution.

Provided that the coupling time is almost surely finite, it is shown (Propp 1996, Vincent & Marchand 2004) that the algorithm 2 generates a state distributed to steady state distribution.

4.2 Monotone perfect sampling

When the operator Φ is monotonous, as shown in section 2, the algorithm could be simplified by making iteration only on maximum and minimum values of the state space. In the open queuing networks situation, there is a unique minimum (all queues are empty) and a unique maximum (all queues are full). Then we only iterate simultaneously 2 trajectories and the time reduction is in the order of the size of the state space. Moreover, it has been shown (Propp 1996) that the mean coupling time is optimal when steps in the past are multiplied by 2 when trajectories issued from maximum and minimum states have not coupled at time 0. In the algorithm M (resp. m) denotes the set of maximal (resp. minimal) elements in the state space.

Algorithm 3 Backward-coupling simulation (monotonous version)

n=1;
E[1]=Generate-event()
repeat
  n=2n;
  for all $x \in M \cup m$ **do**
    $y(x) \leftarrow x$ {choice of the initial value of the vector y, $n = 0$}
  end for
  for i=n downto n/2+1 **do**
    E[i]=Generate-event() {generate event $-i$ according to distribution $(\frac{\lambda_1}{\Lambda}, \cdots, \frac{\lambda_p}{\Lambda})$}
    for all $x \in M \cup m$ **do**
      $y(x) \leftarrow \Phi(y(x), E[i])$ {apply the transition given by event $E[i]$ }
    end for
  end for
  for i=n/2 downto 1 **do**
    {event $-i$ has already been generated in a previous step}
    for all $x \in M \cup m$ **do**
      $y(x) \leftarrow \Phi(y(x), E[i])$
    end for
  end for
until All $y(x)$ are equal
return $y(x)$

An illustration of the algorithm is given in figure 6.

This algorithm have the same convergence properties as algorithm 2. The doubling period each scheme ensures that the coupling time for monotonous system is less than the coupling time for algorithm 2 multiplied by a factor of 2 which is small enough to improve the simulation time.

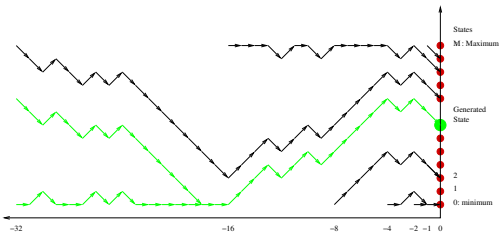

Figure 6: Convergence of monotone backward process

4.3 Functional coupling

In fact, in many case steady state is needed to compute rewards. When rewards are monotonous, the algorithm is improved by initiating the initial values of the process by reward values on maximum and minimum sets.

Algorithm 4 Backward-coupling simulation (monotonous reward version)

```
n=1;
E[1]=Generate-event()
repeat
  n=2n;
  for all x ∈ M ∪ m do
    y(x) ← x {choice of the initial value of the vector y, n = 0}
    R(x) = Reward(x) {initial value of the rewards}
  end for
  for i=n downto n/2+1 do
    E[i]=Generate-event() {generate event −i according to distribution (λ_1/Λ, ··· , λ_p/Λ)}
    for all x ∈ M ∪ m do
      y(x) ← Φ(y(x), E[i]) {apply the transition given by event −i e }
      R(x) = Reward(y(x)) {actualization of rewards}
    end for
  end for
  for i=n/2 downto 1 do
    for all x ∈ M ∪ m do
      y(x) ← Φ(y(x), E[i]) {apply the transition given by event −i }
      R(x) = Reward(y(x)) {actualization of rewards}
    end for
  end for
until All Reward(x) are equal
return Reward(x)
```

An illustration of the algorithm is given in figure 7. The simulation time is reduce from $n = 32$ to 8 iterations to get a sample of the steady state reward.

5 Examples

All simulations have been done on a 1Ghz Pentium 4 with 512Mo RAM, which is sufficient for our purpose.

5.1 Two queues

Consider the system in figure 1 with capacity 300, rates $\lambda = 1.2$, $\mu = 2$, $\nu = 0.8$ and $p = 0.6$ the routing probability. The estimation of saturation of queue 2 that generates losses is done on sample with size 10^6. Saturation probability is estimated to $2.27\,10^{-3} \pm 0.08\,10^{-3}$ with a 95% confidence interval.

The size of state space of this network is 90601. The mean number of iterations before coupling is about 10^7 simple transitions. The computation time of the generation of one state distributed according steady state is $2ms$.

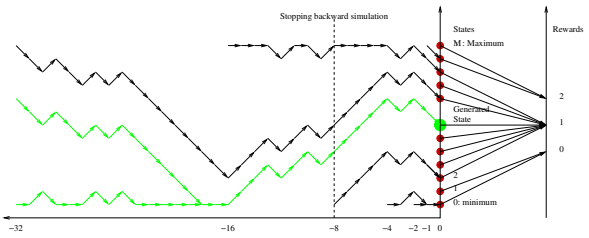

Figure 7: Convergence of monotone backward process

5.2 Priority servers

The basic model in performance evaluation of network is the Erlang model. It consists in parallel servers, arriving customers are served by the first non empty server. If all servers are busy the customer is rejected. If all servers

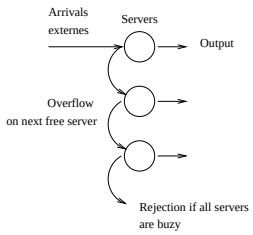

Figure 8: Erlang queueing system (overflow)

have the same service rate, it is well known that the system is reversible and the stationary distribution could be analytically computed. This property has been used to test the simulation software. By goodness fitting tests, it has been shown that the simulated sample is accepted by χ^2 tests as representative of the theoretical distribution.

Moreover, with non homogeneous servers, the system could be simulated easily. An experiment has been done with 30 servers, $\lambda = 20$ and $\mu_i = 1$ for $i \in \{1, \cdots, 10\}$, $\mu_i = 0.8$ for $i \in \{11, \cdots, 20\}$ and $\mu_i = 0.5$ for $i \in \{20, \cdots, 30\}$. Saturation probability is estimated to 0.0579 ± 4.710^{-4}, loss rate is estimated to 1.1 customers per second (the system is loaded).

In that case, the size of the system is $2^30 \simeq 10^9$ states, the computation time of the generation of one state distributed according to steady state is less than $0.4ms$ and the mean number of iterations is 577 per generated state.

To study more precisely the coupling time, the mean number of iterations is computed for 30 homogeneous servers with rate 1 and values of λ from 1 to 50. Sample size was 10000 which gives roughly a few percent precision.

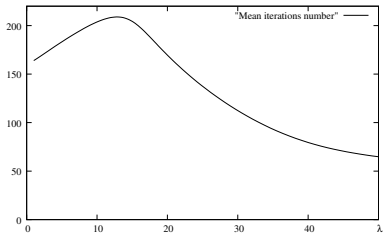

Figure 9: Erlang model : influence of input rate on coupling time

One should note that the curve is maximum when the rate is about 15 that corresponds to the maximum spread of the stationary distribution. We notice that the curve is not symmetric and heavy loaded systems converge more rapidly to the steady state.

5.3 Blocking line

To observe effect of blocking a line of queues is considered. Queue capacities have been fixed to 100, arrival rate

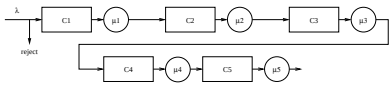

Figure 10: Line of queues with blocking

is 1.2 and service rates 0.8. Blocking probabilities for each queue have been estimated : $b_1 = 0.34$, $b_2 = 0.02$ $b_3 = 0.02$, $b_4, = 0.02$. In that case, the size of the system is $100^6 = 10^{12}$ states, the computation time of the generation of one state distributed according to steady state is less than $1ms$.

5.4 Delta networks

To deal with larger networks, a delta network with 4 stages have been implemented. All queues have a capacity of 100 customers. With a load of 0.9 on each input queue and a uniform routing at each stage (probability $1/2$), the

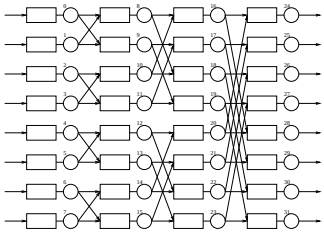

Figure 11: A regular network

mean number of iterations to get a state is 400000, the corresponding computation time is $135ms$ per generated state. It is important to notice that the state space is huge 10^{128} and the presented method remains efficient.

5.5 Call centers

A typical application of this approach concerns modeling of call centers. Gans, Koole & Mandelbaum (2003) provide a huge bibliography on this topic. We focus our example on a simple model with 3 types of servers (figure 12). The call center receive two types of requests. For the first type of requests, rate λ_1, servers of type I or type III could answer the request but requests are initially routed to type I servers. For the second type of requests, rate λ_2 servers of type II and III could answer in this order. The problem is to estimate the mean number of occupied servers of each type and estimate the probability for a call to be rejected.

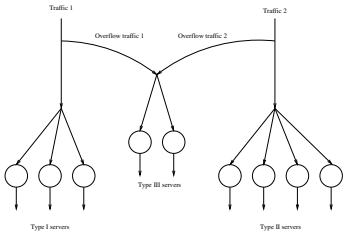

Figure 12: A typical call center with overflow

The configuration consist in $k_1 = 16$ $\mu_1 = 1$ servers of type I, $k_2 = 16$ of type II $\mu_2 = 1.5$ and $k_3 = 8$ of type III $\mu_3 = 2$. The size of the state space is 2^{40}. The input rate λ_1 vary from 0 to 60, the input rate λ_2 is fixed to 20. The second set of servers is lightly loaded.

A typical result is given in figure 13:

The computation time for one iteration is about $450ms$.

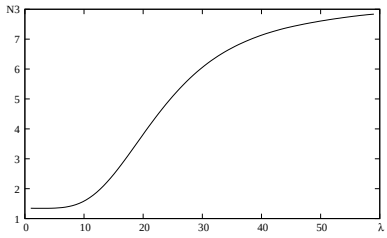

Figure 13: Influence of the input rate λ_1 on mean number of type III occupied servers

6 Future works

In this article have been established monotonicity properties of Markovian queueing networks with finite capacity queues. These properties are fundamental to improve stochastic simulation of such networks. It appears that the simulation time is short enough to estimate probability of rare events with a sufficient confidence interval. Moreover, perfect simulation of performances rewards could also be done by the method developed by (Vincent & Marchand 2004).

A free software is under development. Current version is at *http://www-id.imag.fr/Software/PSI2*. Based on an event description of the queueing network, it offers a simulation kernel based on "perfect simulation" algorithms.
Acknowledgement: Author would like to thank Bernard Tanzi for his very efficient development of PSI2 code.

References

Balsamo, S., De Nitto Person, V. & Inverardi, P. (2003), 'A review of queueing network models with finite capacity queues for software architectures performance prediction', *Performance Evaluation*.

Balsamo, S., De Nitto Person, V. & Onvural, R. (2001), *Analysis of Queueing Networks with Blocking*, Kluwer Academic Publishers.

Banks, J., Carson, J., Nelson, B. & Nicol, D. (2001), *Discrete-Event System Simulation*, Prentice-Hall.

Bolch, G., Greiner, S., de Meer, H. & Trivedi, K. (1998), *Queueing Networks and Markov Chains*, John Wiley & Sons.

Borovkov, A. & Foss, S. (1994), 'Two ergodicity criteria for stochastically recursive sequences', *Acta Appl. Math.*

Brémaud, P. (1999), *Markov Chains: Gibbs fields, Monte Carlo Simulation and Queues*, Springer-Verlag.

Chao, X., Miyazawa, M. & Pinedo, M. (1999), *Queueing Networks Customers, Signals and Product Form solutions*, John Wiley & Sons.

Diaconis, P. & Freedman, D. (1999), 'Iterated random functions', *SIAM Review* **41**(1), 45–76.

Gans, N., Koole, G. & Mandelbaum, A. (2003), 'Telephone call centers : Tutorial, review, and research prospects', *Manufacturing and Service Operation Management* **5**, 79–141.

Lindvall, T. (1997), 'Stochastic monotonicities in jackson queueing networks', *Probability in the Engineering and Informational Sciences* **11**, 1–9.

Mattson, D. (2002), On Perfect Simulation of Markovian Queueing Networks with Blocking, PhD thesis, Chalmers Göteborg University.

Onvural, R. (1990), 'Survey of closed queueing networks with blocking', *ACM Computing Surveys* **22**(2), 83–121.

Perros, H. (1994), *Queuing Networks with Blocking Exact and Approximate Solutions*, Oxford University Press.

Propp, J.and Wilson, D. (1996), 'Exact sampling with coupled Markov chains and applications to statistical mechanics', *Random Structures and Algorithms* **9**(1&2), 223–252.

Robinson, S. (2002), A statistical process control approach for estimating the warm-up period, *in* 'Winter Simulation Conference', Vol. 1, pp. 439– 446.

Stenflo, O. (1998), Ergodic theorems fory Iterated Function Systems controlled by stochastic sequences, Doctoral thesis n. 14, Umea university.

Stenflo, O. (2001), 'Ergodic theorems for markov chains represented by iterated function systems', *Bull. Polish Acad. Sci. Math*.

Vincent, J.-M. (2005), Perfect simulation of queueing networks with blocking and rejection, *in* 'Saint', pp. 268–271.

Vincent, J.-M. & Marchand, C. (2004), 'On the exact simulation of functionals of stationary markov chains', *Linear Algebra and its Applications* **386**, 285–310.