Teknillinen Korkeakoulu
Teknillisen fysiikan koulutusohjelma
Mat-2.108 Sovelletun matematiikan erikoistyöt

ERIKOISTYÖ
30.9.2001

Monte Carlo Simulation of a Banyan Net

Vesa Timonen, e-mail: vesa.timonen@hut.fi
47494R

Monte Carlo Simulation of a Banyan Net

Vesa Timonen

30th September 2001

Contents

Abstract

Blocking probabilities characterize the grade of service in networks. Often it is computationally ineffective to calculate these probabilities exactly and probabilities are estimated via simulation instead. In this paper we compare estimating of the blocking probabilities in a Banyan net via static Monte Carlo method and via inverse convolution method. The blocking probabilities are rather small in actual networks and thus to achieve a sufficient accuracy, a great number of samples is required with the static Monte Carlo method. Inverse convolution method generates samples directly into the blocking region and thus less samples are required to achieve the same accuracy. However, to operate effectively it necessitates that the load of the system is low and it has a higher memory requirement. The numerical results show that in terms of variance, the inverse convolution method can be even 6700 times more effective than the static Monte Carlo method.

1 Introduction

A circuit switched network contains a set of links that join e.g. nodes of the network to each other. When a user creates a connection to a destination, this connection allocates a certain amount of resource units in networks links that the connection uses. Designing of the capacities is among the most general problems in teletraffic engineering.

Network's capability to carry connections can be described with blocking probabilities. *Time blocking probability* can be described as the probability that the system is full at an arbitrary instant of time, i.e. the proportion of time that the system is in blocking states and new connections can not be created. *Call blocking probability* denotes the probability that incoming call is lost, i.e. the proportion of the connections that are lost because of insufficient capacity. Connections are created by users of the system. The call blocking probability characterizes the grade of service that a single user of the system experiences and hence, it is usually more interesting than the time blocking probability.

These two probabilities do not usually have the same value, but are closely related to each other. The difference between time and call blocking depends on the user population. If there are a finite number of users, the time and the call blocking probabilities have different values. In case of the call blocking a single user is in focus and the call blocking probability denotes the probability that the system is full exclusive of connections created by the user. That equates with the time blocking probability of the system with the user in focus removed. If there are infinite number of users and the intensity of the calls adheres to Poisson distribution, the probabilities are equal.

Cell switching fabrics (*CSF*) are used to transfer data packets, i.e. cells between two functional blocks in asynchronous transfer mode switches. The main purpose of a CSF is to route cells from the inputs to the adequate outputs. One design of CSFs used is the multistage interconnection networks (*MIN*). Further, *a Banyan net* is a MIN that has some feasible features. A Banyan net that has N inputs and N outputs has exactly one route from each input to each output. The net consists of single type of switch elements that are placed on $\log_2 N$ levels and each switch element is connected with two links to two switch elements on next level. The implementation of the Banyan net is easy and the

physical limitations of the net are almost only restrictions that constrain the size of the net. These features are concerned e.g. in [1]. In this paper we examine the Banyan net with assumptions that instead of cells there are connections created through the net.

A Banyan net is blocking, i.e. there are connection combinations that are not possible. The blocking probabilities are of interest, especially the call blocking probability. The difficulty in calculating the blocking probabilities in a Banyan net is that the size of the state space and the number of the states grows fast as a function of the number of the inputs, i.e. in $N \times N$ net the number of the possible states of the inputs is $(N+1)^N$ in case that there is one connection maximum for a single input. Thus computing the blocking probabilities by going through all states is not calculatorily effective and the blocking probabilities are estimated via simulation. In reality, the blocking probabilities tend to keep small to achieve high grade of service. Thus probability that with *static Monte Carlo method* a randomly generated state of the system is a blocking state is small and to achieve a sufficient confidence samples have to be generated massively. An effective method called *inverse convolution method* for this kind of rare events is derived in [4]. With inverse convolution method samples are generated directly to the blocking states region and thus the efficiency is remarkably greater than with the static Monte Carlo method and less samples are needed to achieve the same confidence. To achieve these benefits, the inverse convolution method necessitates that the blocking probability is small; in cases of higher blocking probabilities the static Monte Carlo method becomes more effective. The inverse convolution method has also a higher memory requirement.

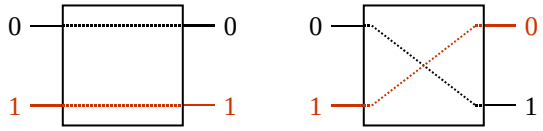
Organization of the paper is as follows: Section 2 presents structure, features and functioning of a Banyan net and defines procedures to constitute sets containing links and classes. Section 3 defines the state space, the set of valid states and blocking states. In Section 4 the user model is introduced, distributions are derived and exact definitions of time and call blocking probabilities are presented. Section 5 introduces static Monte Carlo method and derives inverse convolution method with assumptions made in previous sections. Section 6 presents numerical results from simulations made via both methods with varied parameter values and paper is concluded in Section 7.

2 Banyan net

A Banyan net is an interconnection network that has N input and outputs, $N = 2^K$, where $K \in \mathbb{Z}_+$. The net consists of $KN/2 = K2^{K-1}$ switch elements set in K levels, $N/2$ switches per level. Every switch element has two inputs and outputs and there are two possible couplings, direct and cross connection as described in Figure 1.

Excluding switch elements on the first and on the last level, every switch is connected with links to two switches on previous and next level alike. Links incoming to the switch elements on the first level are the inputs of the net and links outgoing from the switch elements on the last level are the outputs of the net, see Figure 2(a).

There are $J = N(K+1) = (K+1)2^K$ links in a $N \times N$ Banyan net and $\mathcal{J} = \{0, 1, \ldots, J-1\}$ denotes the set of all links. Let $\mathcal{K} = \{0, 1, \ldots, K\}$ denote the set of link levels and $\mathcal{M} = \{0, 1, \ldots, N-1 = 2^K - 1\}$ denote the set

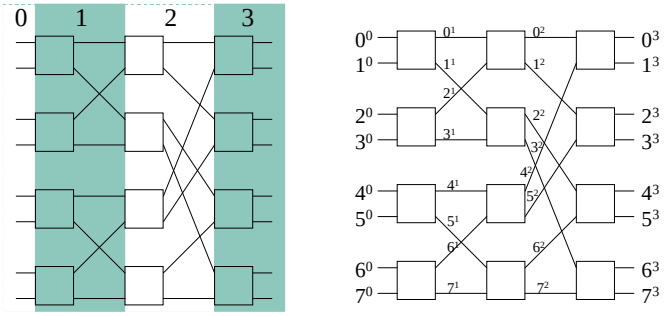


(a) Direct connection: input 0 is connected to output 0 and input 1 to output 1.

(b) Cross connection: input 0 is connected to output 1 and input 1 to output 0.

Figure 1: Connection possibilities of 2×2 switch element.

of consecutive indices of the outputs of switch elements on an arbitrary level. Further let ℓ^k denote link on level k outgoing from ℓ^{th} output of level k switch elements, where $k \in \mathcal{K}$ and $\ell \in \mathcal{M}$. There are no switch elements on level 0, thus parameter ℓ of links on level 0 is here defined to be the same as the consecutive number of the input of switch elements on level 1. The notation is displayed in Figure 2(b).



(a) Link and switch levels.

(b) Link notation.

Figure 2: Structure of an 8×8 Banyan net.

Each link ℓ^k is associated with a index j explicitly with mapping ι and respectively indices ℓ and k are restored with mappings κ and λ:

$$\begin{aligned} \iota : \mathcal{K} \times \mathcal{M} \to \mathcal{J} : &\quad \iota(\ell^k) = kN + \ell \\ \kappa : \mathcal{J} \to \mathcal{K} : &\quad \kappa(j) = \lfloor \tfrac{j}{N} \rfloor \\ \lambda : \mathcal{J} \to \mathcal{M} : &\quad \lambda(j) \equiv j \pmod N. \end{aligned}$$

Now let $\mathcal{J}^k$ denote the set of links on link level k:

$$\mathcal{J}^k = \{j \in \mathcal{J} \mid \kappa(j) = k\}, \; k \in \mathcal{K}.$$

There is exactly one route from every input to every output, total $R = N^2 = 2^{2K}$ routes. Every route is unique, i.e. it can identified by input $b \in \mathcal{J}^0$ and by output

$d \in \mathcal{J}^K$. Thus, we associate each route with an index r given by a mapping ρ:

$$\rho : \mathcal{J}^0 \times \mathcal{J}^K \to \{0, \ldots, R-1\} : \rho(b, d) = N\lambda(b) + \lambda(d),$$

where $\mathcal{J}^0 \times \mathcal{J}^K$ denotes the set of all possible input and output combinations. Let $\mathcal{R}(r)$ denote the set of all links that are on route $r = \rho(b, d)$, including links b and d.

Every link j has a capacity, say C_j, in resource units. A connection that uses an arbitrary route r has a capacity requirement b_r. We define *traffic class* as a tuple $e_r = (r, b_r)$. Connections using route r belong to the traffic class e_r. Index r associated with the route r determines also the class e_r explicitly. Hence, the class e_r is associated with the index r instead of e_r without loss of generality or doubt of causing confusion.

At this stage we set $C_j = 1$ for all j and $b_r = 1$ for all r. In an arbitrary link j a single class r connection in progress requires b_r^j resource units in such a way that

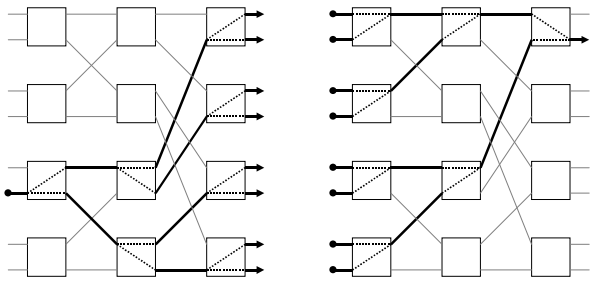
$$b_r^j = 1_{j \in \mathcal{R}(r)}.$$

Capacity requirements of classes on link j are denoted by $\mathbf{b}^j = (b_0^j \; b_1^j \; \cdots \; b_{R-1}^j)$.

2.1 Binary presentations

In this section we present functions to define a set that contains links that belong to a specific class and a set that contains classes that use a specific link. These functions are based on the binary presentations of the link and the class indices and were used in the implementation of the simulation methods.

For an arbitrary link $j \in \mathcal{J}$ on level $\kappa(j)$, the link on level $\kappa(j) + 1$ can be chosen between two links because the switch element to which the link j goes into has two possible couplings. Thereby routes starting from an arbitrary input establish a binary tree. Correspondingly, routes ending to an arbitrary output establish a binary tree, see Figure 3.



(a) Routes from input 5 to outputs $24, \ldots, 31$.

(b) Routes from inputs $0, \ldots, 7$ to output 25.

Figure 3: Binary tree structure in 8×8 Banyan net.

Binary presentation are used to present a function to define the link on a given level that belong to an arbitrary route $r = \rho(b, d)$ as follows: the binary

forms of $\lambda(b) = b'$ and $\lambda(d) = d'$ are

$$\begin{cases} b' = (b_K\ b_{K-1}\ \cdots\ b_2\ b_1)_2, & b_i \in \{0,1\} \\ d' = (d_K\ d_{K-1}\ \cdots\ d_2\ d_1)_2, & d_i \in \{0,1\} \end{cases} \forall i = 1, \ldots, K.$$

Now let $\chi(r,k)$ denote a function that gives $\lambda(j)$ of link $j \in \mathcal{R}(r)$ on level k:

$$\chi(r,k) = \begin{cases} b', & k = 0 \\ d', & k = K \\ (b_K\ b_{K-1}\ \cdots\ b_{k+1}\ d_K\ d_{K-1}\ \cdots\ d_{K-k+1})_2, & \text{otherwise,} \end{cases}$$

and thus

$$\mathcal{R}(r) = \{j \in \mathcal{J}^k \mid \lambda(j) = \chi(r,k),\ \forall k \in \mathcal{K}\}.$$

Resulting from the binary tree structure of the routes, an arbitrary link j on level $k = \kappa(j)$ can be reached from 2^k inputs $m \in \mathcal{J}^0$. Further, 2^{K-k} outputs $n \in \mathcal{J}^K$ can be reached from link j. Thus link j is passed through by $2^K = N$ routes $s = \rho(m,n)$ that establish set $\mathcal{L}(j)$ of classes using link j. To constitute links m and n, we first constitute the binary form of $\ell = \lambda(j)$:

$$\ell = (\ell_K\ \ell_{K-1}\ \cdots\ \ell_1)_2, \quad \ell_i \in \{0,1\}, \quad i = 1, \ldots, K,$$

and all possible inputs m and outputs n are constituted with the binary forms of $\lambda(m) = m'$ and $\lambda(n) = n'$:

$$\begin{cases} m' = (\ell_K\ \ell_{K-1}\ \cdots\ \ell_{k+1}\ m_k\ m_{k-1}\ \cdots\ m_1)_2, & m_i \in \{0,1\}, \quad i = 1, \ldots, k \\ n' = (\ell_k\ \ell_{k-1}\ \cdots\ \ell_1\ n_{K-k}\ n_{K-k-1}\ \cdots\ n_1)_2, & n_i \in \{0,1\}, \quad i = 1, \ldots, K-k. \end{cases}$$

Now the set $\mathcal{L}(j)$ is established by looking through all possible pairs $\{m,n\}$, i.e. all possible routes $s = \rho(m,n)$ and the set $\mathcal{L}(j)$ is

$$\mathcal{L}(j) = \{s \mid j \in \mathcal{R}(s)\}.$$

3 State space and blocking states

Let $x_r \in \{0,1\}$ denote the number of class r connections in progress and $\mathbf{x} = (x_0\ x_1\ \cdots\ x_{R-1})$ the state of the system. Thus the set of all states, i.e. the state space Ω is

$$\Omega = \{0,1\}^R.$$

Each class r is responsible for one dimension in the state space. Now taking into account the link capacities $C_j = 1$ for all j and $b_r = 1$ for all r, the set of allowed states Ω_S, i.e. the truncated state space is constituted as follows:

$$\Omega_S = \{\mathbf{x} \in \Omega \mid \mathbf{b}^j \cdot \mathbf{x} \leq 1,\ \forall j\} \subset \Omega. \tag{1}$$

Because each input can be used by connection of one class maximum, there can be connections of at most N different classes in progress in an $N \times N$ Banyan net.

Let $\mathcal{B}_r^j$ denote the set of blocking states of class r caused by the finite capacity of link j, i.e. the set of states in which establishing a new connection of class r causes exceeding the capacity $C_j = 1$ and $\mathcal{B}_r$ the set of all blocking states of class r connections:

$$\mathcal{B}_r^j = \{\mathbf{x} \in \Omega \mid \mathbf{b}^j \cdot (\mathbf{x} + \mathbf{e}_r) > 1\}, \tag{2}$$

and

$$\mathcal{B}_r = \Omega_S \cap \bigcup_{j \in \mathcal{R}(r)} \mathcal{B}_r^j, \tag{3}$$

where $\mathbf{e}_r = (1_{r=0}\ 1_{r=1}\ \cdots\ 1_{r=R-1})$ is a unit vector of the state space Ω. These sets are illustrated in Figure 4. Notice that also states that do not belong to set Ω_S are included in the definition of $\mathcal{B}_r^j$.

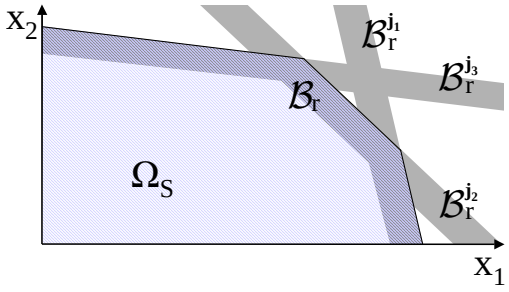


Figure 4: The set of allowed states Ω_S and blocking states in case of two classes and three restricting links.

4 Probabilities and density functions

Each input has one user that generates the connection. Let us first assume that the capacity of the system is infinite and the switch elements do not restrict the connections, thus all users are independent. We assume also that all the intensities of connection arrivals and the connection holding times are exponentially distributed. Thus, the users can be described with Markov-models. Figure 5 shows the user model of a single input b.

Let the random variable Z_b denote the state of input $b \in \mathcal{J}^0$ and the random variable $\mathbf{Z} = (Z_0\ Z_1\ \cdots\ Z_{N-1})$ the state of all inputs. Z_b adheres to distribution g_b, i.e.

$$g_b(z_b) = \mathrm{P}\{Z_b = z_b\} = \begin{cases} p_b^{\phi}, & z_b = -1 \mathrel{\widehat{=}} \text{"no connection on"} \\ p_{b,d_i}, & z_b = i \mathrel{\widehat{=}} \text{"connection of class } \rho(b,d_i) \text{ on"}, \end{cases}$$

where $z_b \in \{-1, 0, \ldots, N-1\}$, $p_b^{\phi}, p_{b,d_i} \in [0,1]$, $d_i = \iota(i^K)$ and $i \in \mathcal{M}$. Let ω denote the set of all states of inputs, $\omega = \{-1, 0, \ldots, N-1\}^N$. Each $\mathbf{z} = (z_0\ z_1\ \cdots\ z_{N-1}) \in \omega$ is associated with a state $\mathbf{x} = (x_0\ x_1\ \cdots\ x_{R-1}) \in \Omega$ as follows:

$$x_{\rho(b,d_i)} = 1_{z_b=i},\ \forall b, d_i,$$

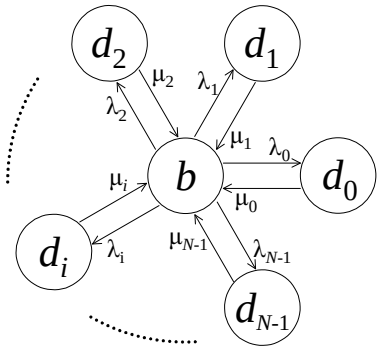


Figure 5: Markov chain model for the user model of input b. State b denotes the situation that no connection is in progress and d_i the situation that class $\rho(b, d_i)$ connection is in progress, $d_i \in \mathcal{J}^K$. $\lambda_{b,i}$ denotes the intensity of call arrivals and $1/\mu_{b,i}$ the mean holding time.

where $d_i = \iota(i^K)$ and $i \in \mathcal{M}$.

The distributions g_b are derived as follows: the detailed balance equation gives condition

$$p_b^\phi \lambda_i = p_{b,d_i} \mu_{b,i} \Leftrightarrow p_{b,d_i} = \frac{\lambda_{b,i}}{\mu_{b,i}} p_b^\phi, \tag{4}$$

and the condition that the sum of the probabilities is 1 gives

$$p_b^\phi + \sum_{i=0}^{N-1} p_{b,d_i} = p_b^\phi + \sum_{i=0}^{N-1} \frac{\lambda_{b,i}}{\mu_{b,i}} p_b^\phi = \left(1 + \sum_{i=0}^{N-1} \frac{\lambda_{b,i}}{\mu_{b,i}}\right) p_b^\phi = 1 \tag{5}$$

$$\Leftrightarrow p_b^\phi = \frac{1}{1 + \sum_{i=0}^{N-1} \frac{\lambda_{b,i}}{\mu_{b,i}}}, \tag{6}$$

and

$$p_{b,d_i} = \frac{\lambda_{b,i}}{\mu_{b,i}} \frac{1}{1 + \sum_{i'=0}^{N-1} \frac{\lambda_{b,i'}}{\mu_{b,i'}}}, \tag{7}$$

where $\lambda_{b,i}$ is the intensity and $1/\mu_{b,i}$ the mean holding time of incoming calls of class $\rho(b, d_i)$.

Now let the random variable X_r denote the number of class r connections in progress and the random variable $\mathbf{X} = (X_0\ X_1\ \cdots\ X_{R-1})$ the state of the whole system. Each $\mathbf{z} \in \omega$ and corresponding $\mathbf{x} \in \Omega$ is associated with a probability $\pi(\mathbf{x})$ in such a way that

$$\pi(\mathbf{x}) = \mathrm{P}\{\mathbf{X} = \mathbf{x}\} = \mathrm{P}\{\mathbf{Z} = \mathbf{z}\} = \prod_{b=0}^{N-1} g_b(z_b).$$

Insensitivity property means that the systems steady state distributions do not depend on the connection holding time distributions, only on their means [6].

Truncation principle means that state probabilities $\tilde{\pi}(\mathbf{x})$ of the truncated system can be calculated with the state probabilities $\pi(\mathbf{x})$ of the system with infinite capacities [2]. This means that the probability $\tilde{\pi}(\mathbf{x})$ is given by probability $\pi(\mathbf{x})$ normalized with the sum of the probabilities $\pi(\mathbf{x})$ over all states $\mathbf{x}$ of the truncated state space.

It can be shown that the insensitivity property applies and distributions are valid when the connection holding times and the user idle times are generally distributed with means $1/\mu_{b,i}$ and $1/\lambda_{b,i}$ respectively. It can also be shown that truncation principle is applicable in this system. For an analogous system these proofs are derived in [5]. The state probabilities of the truncated system are

$$\tilde{\pi}(\mathbf{x}) = \mathrm{P}\{\mathbf{X} = \mathbf{x} \mid \mathbf{X} \in \Omega_S\} = \frac{\pi(\mathbf{x})}{\sum_{\mathbf{x}\in\Omega_S} \pi(\mathbf{x})} 1_{\mathbf{x}\in\Omega_S}.$$

The time blocking probability of an arbitrary class r is the probability that a valid state belongs to the set of blocking states of class r. This probability B_r is calculated by adding up state probabilities $\tilde{\pi}(\mathbf{x})$ of all states that belong to the set $\mathcal{B}_r$, i.e.

$$B_r = \sum_{\mathbf{x}\in\mathcal{B}_r} \tilde{\pi}(\mathbf{x}) = \frac{\sum_{\mathbf{x}\in\mathcal{B}_r} \pi(\mathbf{x})}{\sum_{\mathbf{x}\in\Omega_S} \pi(\mathbf{x})} = \frac{\beta_r}{\gamma_r}. \tag{8}$$

The call blocking of an arbitrary class r is the probability that a valid state belongs to the set of blocking states of class r in such a way that no connection is produced by the user of input b, i.e. $z_b = -1 \Leftrightarrow x_{\rho(b,d)} = 0$ for all outputs d. We constitute the set of valid states Ω_r^C and the set of call blocking states $\mathcal{B}_r^C$ with equations (1) and (3):

$$\Omega_r^C = \{\mathbf{x} \in \Omega_S \mid x_{\rho(b,d)} = 0,\ \forall d \in \mathcal{J}^K\}, \tag{9}$$

and

$$\mathcal{B}_r^C = \mathcal{B}_r \cap \Omega_r^C. \tag{10}$$

Finally, the call blocking probability B_r^C is

$$B_r^C = \frac{\sum_{\mathbf{x}\in\mathcal{B}_r^C} \pi(\mathbf{x})}{\sum_{\mathbf{x}\in\Omega_r^C} \pi(\mathbf{x})} = \frac{\beta_r^C}{\gamma_r^C}. \tag{11}$$

The exact blocking probabilities can be calculated as described above. However, the larger the net the greater the number of the states becomes and it is not effective to go through them all. Thus, it is reasonable to use simulation to get estimates for the blocking probabilities. In the following section we derive the estimators for the probabilities.

5 Simulation methods

Monte Carlo methods are static simulation methods that are used to estimate parameters of stochastic time invariant systems. With the Monte Carlo methods the system is observed at an arbitrary instant of time. Following Sections 5.1 and

5.2 present two Monte Carlo methods for estimating the blocking probabilities, static Monte Carlo method and inverse convolution method.

Confidence of the estimator is measured as follows: let assume that the number of samples in a single simulation run is N'. To get an estimate for the variance, these N' samples are divided into M batches and thus every batch contains $N = \frac{N'}{M}$ samples. Every batch yields to an estimate of the blocking and with these M estimates, say B_i, we can calculate the sample variance S_m^2 of the simulation run:

$$S_m^2 = \frac{1}{M-1}\sum_{i=1}^{M}(B_i - \bar{B}_m)^2,$$

where $\bar{B}_m$ denotes the sample average of the estimates B_i. The sample deviation S_m is the square root of the sample variance

$$S_m = \sqrt{S_m^2}.$$

On the other hand when the estimates B_i are small, the sample variance becomes also small and the absolute value of the sample variance does not give distinct impression of the accuracy of the estimator. When measuring the estimator with the relative deviation S_{SD} this effect is eliminated. The relative deviation is the ratio of the sample deviation and the corresponding estimate of the blocking:

$$S_{\mathrm{SD}} = \frac{S_m}{\bar{B}_m}.$$

Benefit in using relative deviation is that the magnitude of the value in focus is eliminated and results achieved with different parameter values can be compared. With the sample average and the sample deviation the confidence interval of the estimator at the confidence level $1-\beta$ can be constituted:

$$\left[\bar{B}_m - t_{M-1,1-\beta/2}\frac{S_m}{\sqrt{M}}\ ,\ \bar{B}_m + t_{M-1,1-\beta/2}\frac{S_m}{\sqrt{M}}\right],$$

where $t_{m',\beta'}$ is the β'-fractile of the t-distribution with m' degrees of freedom.

Furthermore, the sample variance can be examined in same way by making repeated simulation runs and then estimating the sample variance as described above.

In Section 5.2 we concentrate on deriving the estimator for β_r in equation (8) and omit the examination of γ_r on the blocking probabilities.

5.1 Static Monte Carlo method

Estimating the time blocking probability B_r via static Monte Carlo method is straightforward. First class r in focus has to be chosen.

Samples $\mathbf{Z}$ of the states of the inputs are generated from distribution $\mathbf{g}$ and thereby we get also the corresponding state $\mathbf{X}$ of the system. Estimators for β_r and γ_r in equation (8) are

$$\hat{\beta}_r = \frac{1}{N}\sum_{i=1}^{N} 1_{\mathbf{X}_i \in \mathcal{B}_r},$$

and

$$\hat{\gamma}_r = \frac{1}{N}\sum_{i=1}^{N} 1_{\mathbf{X}_i \in \Omega_S},$$

where N is the number of samples. Thus the estimator for B_r is

$$\hat{B}_r = \frac{\hat{\beta}_r}{\hat{\gamma}_r}.$$

5.2 Inverse convolution method

The following derivation is based on the development and presentation of the inverse convolution method in [4] and applying in case of multicast loss system [3].

The basic idea of inverse convolution method is to power estimating β_r in equation (8) by generating states directly to the set of blocking states of class $r = \rho(b, d)$ in focus. This is implemented by dividing estimation of β_r into simpler sub-problems and solving them efficiently.

For the first step the set $\mathcal{B}_r$ can be divided into point wise sub-sets in following way: let $\mathcal{D}_r^j$ denote the set of states that cause blocking in link j on level $k = \kappa(j)$ excluding states causing blocking on links j' that belong to the same class r and are situated between input b and the link j, i.e.

$$\begin{aligned}
\mathcal{D}_r^b &= \Omega_S \cap \mathcal{B}_r^b \\
\mathcal{D}_r^j &= \left(\Omega_S \cap \mathcal{B}_r^j\right) \setminus \bigcup_{\substack{\kappa(j')<\kappa(j) \\ j' \in \mathcal{R}(r)}} \mathcal{D}_r^{j'} \\
\mathcal{B}_r &= \bigcup_{j \in \mathcal{R}(r)} \mathcal{D}_r^j.
\end{aligned}$$

This is illustrated in Figure 6. Now blocking probability B_r of class r can be presented in the form

$$B_r = \frac{\sum_{j \in \mathcal{R}(r)} \sum_{\mathbf{x} \in \mathcal{D}_r^j} \pi(\mathbf{x})}{\sum_{\mathbf{x} \in \Omega_S} \pi(\mathbf{x})} = \frac{\sum_{j \in \mathcal{R}(r)} \delta_r^j}{\gamma_r}, \tag{12}$$

where δ_r^j is the probability that an arbitrary state $\mathbf{X} \in \Omega$ belongs to the set $\mathcal{D}_r^j$. From equations (8), (12) and with the property that the sets $\mathcal{D}_r^j$ are disjoint for all $j \in \mathcal{R}(r)$, we get

$$\beta_r = \sum_{j \in \mathcal{R}(r)} \delta_r^j. \tag{13}$$

To estimate β_r we have to estimate each δ_r^j separately. This estimation can also be divided into parts noticing that

$$\delta_r^j = \mathrm{P}\{\mathbf{X} \in \mathcal{D}_r^j\} = \mathrm{P}\{\mathbf{X} \in \mathcal{D}_r^j \mid \mathbf{X} \in \mathcal{B}_r^j\}\mathrm{P}\{\mathbf{X} \in \mathcal{B}_r^j\}. \tag{14}$$

This follows from the fact that

$$\underbrace{\mathrm{P}\{\mathbf{X} \in \mathcal{B}_r^j \mid \mathbf{X} \in \mathcal{D}_r^j\}}_{=1}\mathrm{P}\{\mathbf{X} \in \mathcal{D}_r^j\} = \mathrm{P}\{\mathbf{X} \in \mathcal{D}_r^j \mid \mathbf{X} \in \mathcal{B}_r^j\}\mathrm{P}\{\mathbf{X} \in \mathcal{B}_r^j\}.$$

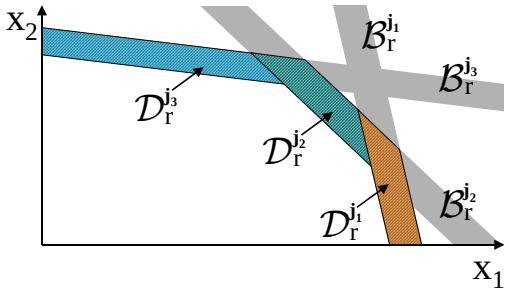


Figure 6: Dividing the set of blocking states.

The probability $\mathrm{P}\{\mathbf{X} \in \mathcal{B}_r^j\}$ can be calculated exactly as follows: a link $j \in \mathcal{R}(r)$ can be reached by a connection of class $s \in \mathcal{L}(j)$, $s = \rho(m,n)$. Let L denote the number of all inputs m from which link j can be reached, i.e. L is the number of sets $\mathcal{L}_m^j = \mathcal{L}(m) \cap \mathcal{L}(j)$ that are not empty. We associate these inputs m with consecutive index $l = 1, \ldots, L \leq N$. Let the random variable Y_l^j denote the occupancy of link j caused by the connections of classes $s \in \mathcal{L}_m^j$ in progress, i.e. $Y_l^j = \sum_{s \in \mathcal{L}_m^j} X_s$. Y_l^j adheres to distribution f_l^j:

$$f_l^j(y) = \mathrm{P}\{Y_l^j = y\} = \begin{cases} \sum_{i=0}^{N-1} 1_{\rho(m,d_i) \notin \mathcal{L}_m^j} g_m(i) + p_m^\phi, & y = 0 \\ \sum_{i=0}^{N-1} 1_{\rho(m,d_i) \in \mathcal{L}_m^j} g_m(i), & y = 1, \end{cases}$$

where $y \in \{0,1\}$ and $d_i = \iota(i^K) \in \mathcal{J}^K$.

Let S_l^j denote the occupancy of link j caused by connections of first l inputs of m. S_l can be constituted recursively. Let $S_0^j = 0$, and thus S_1^j can be presented with Y_1^j and S_0^j, $S_1^j = S_0^j + Y_1^j$. Respectively we get $S_2^j = S_1^j + Y_2^j$ and for an arbitrary $l \leq L$

$$S_l^j = S_{l-1}^j + Y_l^j.$$

S_l^j adheres to distribution q_l^j:

$$q_l^j(x) = \mathrm{P}\{S_l^j = x\},$$

where $x \in \{0,1\}$. Distribution q_l^j can be derived recursively with convolution as follows: obviously $q_0^j = (1\ 0)$ and $q_1^j = f_1^j = T^2(q_0^j \otimes f_1^j)$, where $T^i(\cdot)$ denotes the truncation operation that takes from a vector its first i components and $\otimes$ denotes the convolution operation. Truncation is made because link j has capacity of 1 resource unit and thus there are two possible occupancies (no connection vs. one connection). Respectively, $q_2^j = T^2(q_1^j \otimes f_2^j)$ and further for an arbitrary $l \leq L$

$$q_l^j = T^2(q_{l-1}^j \otimes f_l^j).$$

Finally, $\mathrm{P}\{\mathbf{X} \in \mathcal{B}_r^j\} = q_L^j(1)$. The previous process is illustrated in Figure 7.

Now, to get an estimator for $\mathrm{P}\{\mathbf{X} \in \mathcal{D}_r^j \mid \mathbf{X} \in \mathcal{B}_r^j\}$ in equation (14) we generate states with condition $\mathbf{X} \in \mathcal{B}_r^j$ and check if they belong to $\mathcal{D}_r^j$. Generating states into $\mathcal{B}_r^j$ is done as follows: the probability that the l^{th} input m causes

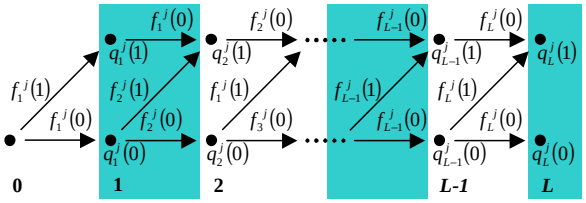


Figure 7: Constituting the state probabilities $q_l^j(\cdot)$ of link j with convolution and truncation. The lower row denotes the occupancy $S_l^j = 0$ and the upper the occupancy $S_l^j = 1$.

occupancy x in link j is

$$\mathrm{P}\{Y_l^j = y \mid S_l^j = x\} = \frac{f_l^j(y) q_{l-1}^j(x-y)}{q_l^j(x)},$$

where $0 \leq y \leq x \leq 1$. Because link capacities are 1, we are interested in which input m generates the connection that causes the blocking. Condition $\mathbf{x} \in \mathcal{B}_r^j$ is equivalent to condition $S_L^j = 1$. We go through inputs m from which there is a connection to the link j and randomly choose one that forces the system into a blocking state. Thus we start from the L^{th} input m, generate a random variable $u \in (0,1)$ from uniform distribution and if $\mathrm{P}\{Y_L^j = 1 \mid S_l^j = 1\} \geq u$, then there is connection in progress in L^{th} input m. Otherwise we move on to the $(L-1)^{th}$ input and do the same examination. This procedure is repeated until the inequality is true and the $\tilde{l}^{th}$ input $\tilde{m}$ that causes blocking is found. After this the class s' of the generated connection has to be found out. The class s' can be generated with condition from the distribution $g_{\tilde{m}}$ as follows:

$$\mathrm{P}\{Z_{\tilde{m}} = z_{\tilde{m}} \mid s' \in \mathcal{L}_{\tilde{m}}^j\} = \frac{g_{\tilde{m}}(z_{\tilde{m}})}{f_{\tilde{l}}^j(1)} 1_{s' \in \mathcal{L}_{\tilde{m}}^j},$$

where $s' = \rho(\tilde{m}, \iota(z_{\tilde{m}}^K))$. Now states of the other inputs m from which there is a connection to the link j are generated also with condition from distributions g_m in such a way that these possible connections do not go through link j, i.e.

$$\mathrm{P}\{Z_m = z_m \mid s \notin \mathcal{L}_m^j\} = \frac{g_m(z_m)}{f_l^j(0)} 1_{s \notin \mathcal{L}_m^j},$$

where $s = \rho(m, \iota(z_m^K))$. Finally, the states of all other inputs b are generated straight from the corresponding distributions g_b. Therefore the state of all inputs $\mathbf{Z}$ is constituted and also the corresponding state of the system $\mathbf{X}$.

With equation (14) we get estimator for δ_r^j:

$$\hat{\delta}_r^j = \hat{P}\{\mathbf{X} \in \mathcal{D}_r^j \mid \mathbf{X} \in \mathcal{B}_r^j\} \underbrace{\hat{P}\{\mathbf{X} \in \mathcal{B}_r^j\}}_{q_L^j(1)} = \frac{q_L^j(1)}{N} \sum_{i=1}^{N} 1_{\mathbf{X}_i \in \mathcal{D}_r^j},$$

where N is the number of samples, and further with equation (13) estimator for β_r is

$$\hat{\beta}_r = \sum_{j \in \mathcal{R}(r)} \hat{\delta}_r^j.$$

6 Simulations

The purpose of the simulations is to compare the efficiency of the inverse convolution method with that of the static Monte Carlo method in estimation for the call blocking probability B_r^C in equation (11). We do the comparison concerning variance of these two methods.

We set $\lambda_{b,i} = \lambda$ and $\mu_{b,i} = \mu$ for all $i \in \mathcal{M}$ and $b \in \mathcal{J}^0$ in equations (4) through (7), and assume probability p that a connection is in progress in a input is known. Thus, given by equation (6)

$$p_b^\phi = 1 - p, \ \forall b \in \mathcal{J}^0.$$

This implies with equation (4) that

$$p_{b,d_i} = \frac{\lambda}{\mu}(1-p) = p_{b,d}, \ \forall i \in \mathcal{M}, d_i \in \mathcal{J}^K$$

where $d_i = \iota(i^K)$. Then, by equation (5)

$$\begin{aligned} p_b^\phi + \sum_{i=0}^{N-1} p_{b,d_i} &= 1 - p + N p_{b,d} = 1 \\ \Rightarrow p_{b,d} &= \frac{p}{N}, \ \forall b \in \mathcal{J}^0 \ \forall d_i \in \mathcal{J}^K \end{aligned}$$

Now distributions g_b are identical for all $b \in \mathcal{J}^0$:

$$g_b = \Big(1 - p \ \underbrace{\frac{p}{N} \ \frac{p}{N} \ \cdots \ \frac{p}{N}}_{N \text{ parts}}\Big), \ \forall b. \tag{15}$$

These assumptions imply that time blocking probabilities and also call blocking probabilities of different classes are identical, i.e. $B_i = B_j$ and $B_i^C = B_j^C$ for all i and j. Thus no class has special status and we choose class $\rho(0,0) = 0$ to be the target of examination.

In a realistic system blocking probabilities are relatively small, i.e. blocking probability is $\lessapprox 5\%$. To achieve that, we assume that probability p is small and thus validity of generated states is high, i.e. $\gamma_0 \approx 1$. Section 6.1 studies more closely how the values of probability p have to be chosen for the simulation so that the validity is high enough and effect of the denominator in equation (11) can be omitted.

In simulations there were two parameter values to vary, the number of levels and the probability p. Values $2, \ldots, 6$ were used for the number of levels , i.e. nets of sizes $4 \times 4, 8 \times 8, \ldots, 64 \times 64$ were examined and probability p was used with values 0.001, 0.005, 0.01, 0.05, 0.1, 0.2 and 0.3. Different number of samples were used to see the behavior of the methods; simulation runs was made with 100, 250, 500, 750, 1000, 2500, 5000, 7500, 10000, 25000, 50000 and with 75000 samples. To get estimates for variance each simulation run was divided to 10 batches. To estimate the variance of the estimated variance each simulation run was repeated 10 times.

Table 1: The validity of the states γ_0^C (%).

p	Number of levels				
	2	3	4	5	6
0.001	100.00	100.00	99.999	99.995	99.991
0.005	100.00	99.989	99.964	99.906	99.772
0.010	99.990	99.957	99.867	99.648	99.100
0.050	99.763	98.918	96.782	91.696	81.174

6.1 Validity of the states

The focus is efficient estimation of β_0^C in equation (11) omitting the effect of γ_0^C, thus we have to make simulations with parameter values that give a high validity γ_0^C of the system. To find out that which parameter values produce high validity of the system we estimated $\gamma_0^C = \mathrm{P}\{\mathbf{X} \in \Omega_S^C\}$ via static Monte Carlo simulation with 10^6 samples. Results are shown in Figure 8.

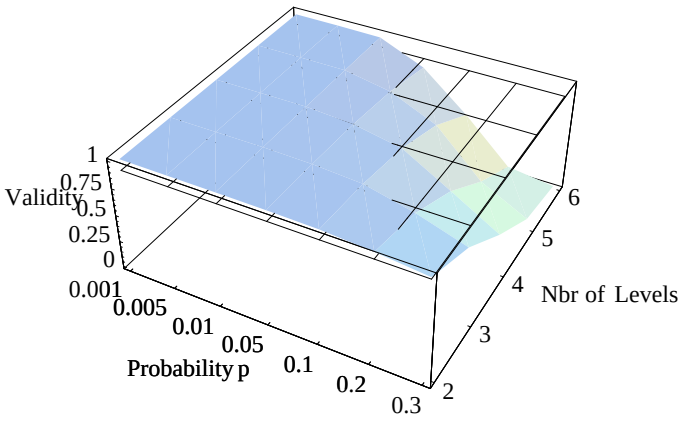


Figure 8: Validity of the states $\mathrm{P}\{\mathbf{X} \in \Omega_S^C\}$, where the estimate is calculated with 10^6 samples. Wire framed plane surface denotes 90% level of validity.

Figure 8 shows clearly the behavior of the system: with number of levels $2, \ldots, 6$ and with small values of the probability, i.e. $0 < p < 0.05$ the validity is high. It seems that with greater values of probability p and number of levels the validity decreases faster as a function of the number of levels. Table 1 shows these values with parameter values $0.001 \leq p \leq 0.05$ and with number of levels $2, \ldots, 6$.

Table 2: The mean ratio of the variances.

p	The number of levels.				
	2	3	4	5	6
0.001	4692	6683	887.2	554.4	563.7
0.005	832.6	1183	169.4	126.0	89.99
0.010	455.6	603.3	86.29	65.32	46.51
0.050	76.85	106.4	10.04	10.22	6.584

6.2 Results

Simulation results of the call blocking probability in case of a 4×4 Banyan net is in Figure 9 and an 8×8 Banyan net in Figure 10. In these cases also the exact blocking probability is known (see Section 6.3) and it is plotted in each subfigure. Figure 11 shows simulation results in case of a 64×64 Banyan net. In all these figures *DMC* denotes the static Monte Carlo method and *IC* the inverse convolution method. Figures of results in cases of 16×16 and 32×32 Banyan nets are excluded; the behavior of the results becomes evident in figures 9 through 11.

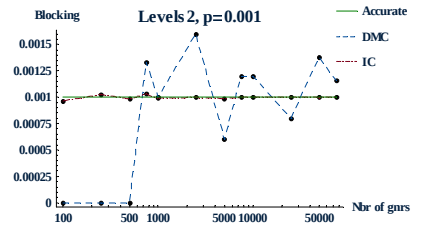
As figures mentioned above show, both the static Monte Carlo method and the inverse convolution method converge to the same value in each case. Cases in which the accurate value of the call blocking probability is known suggest that the estimates are valid. Clearly the inverse convolution method is more effective; the smaller probability p and the number of levels are the faster the convergence is compared with the estimate given by the static Monte Carlo method. When probability p and the size of the net increase, the efficiency of both methods decreases and the difference between the methods becomes smaller. Still, the inverse convolution method is more efficient in the cases in focus.

Figures 9 through 11 show that the estimated variance decreases as function of the number of samples. The trend is the same as with the estimate on the blocking probability: the inverse convolution gives considerably smaller values than the static Monte Carlo method with small parameter values and the difference is smaller when the parameter values are greater. The ratio of the variance estimates is approximately constant in each case. Thus we calculated the mean of the ratios given by different number of samples. Table 2 and figure 12 show mean ratio of estimated variances as function of the number of levels and probability p. The mean ratio is calculated here as follows:

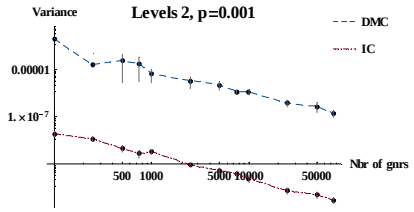
$$\text{ratio} = \frac{1}{N} \sum_{i=1}^{N} \frac{S^2_{\text{DMC},i}}{S^2_{\text{IC},i}},$$

where $S^2_{\text{DMC},i}$ is the sample variance estimated via static Monte Carlo method and $S^2_{\text{IC},i}$ the sample variance estimated via inverse convolution method respectively. The sum is calculated over estimates given by different number of samples.

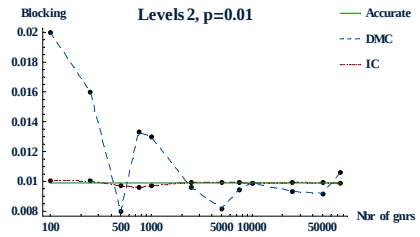
Figure 13 shows the relative deviation of the estimated call blocking probability as a function of probability p and the number of samples. Figure 13(a) shows the relative deviation in case of the 4×4 net and Figure 13(b) in case



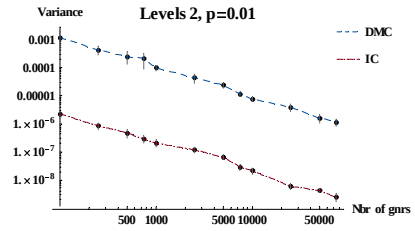
(a) Call blocking probability vs. number of samples, $p = 0.001$.



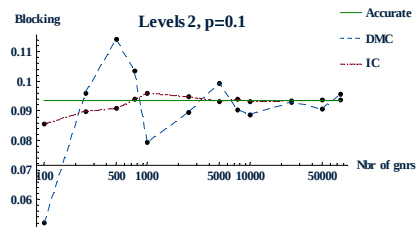
(b) Variance vs. number of samples, $p = 0.001$.



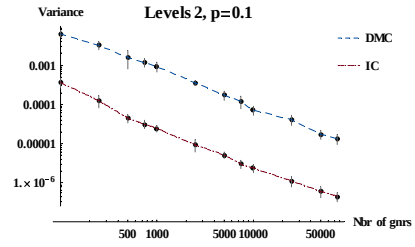
(c) Call blocking probability vs. number of samples, $p = 0.01$.



(d) Variance vs. number of samples, $p = 0.01$.



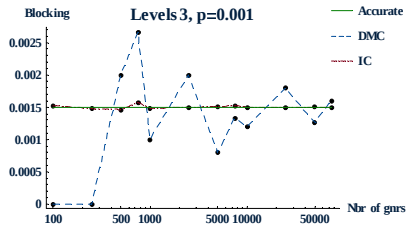
(e) Call blocking probability vs. number of samples, $p = 0.1$.



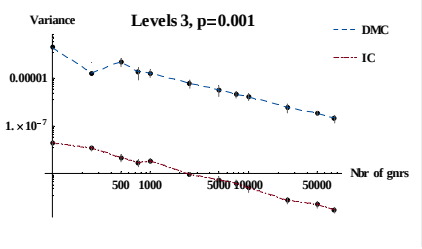
(f) Variance vs. number of samples, $p = 0.1$.

Figure 9: Comparison in a 4×4 Banyan net, where p is the probability that a connection is in progress.

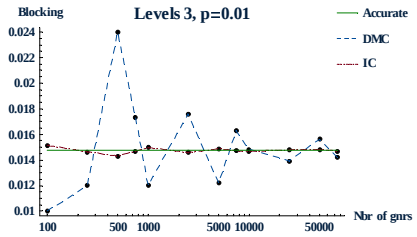
of the 32×32 net. The relative deviation decreases when the number of samples increases in both cases and with both methods. Contrary to the inverse convolution method, the static Monte Carlo gives the better result the greater the probability p becomes. The relative deviation in case of 75000 samples is in Figure 14 and it shows the same result: the inverse convolution method is most efficient when the blocking probability is small. When the possibility of the blocking becomes greater, the efficiency of the static Monte Carlo Method



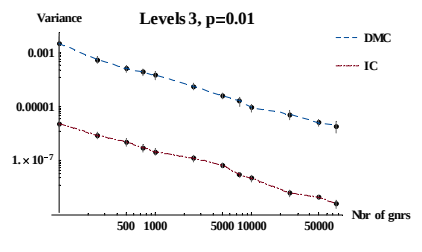
(a) Call blocking probability vs. number of samples, $p = 0.001$.



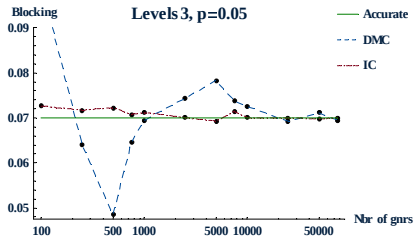
(b) Variance vs. number of samples, $p = 0.001$.



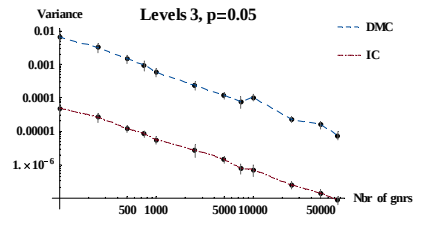
(c) Call blocking probability vs. number of samples, $p = 0.01$.



(d) Variance vs. number of samples, $p = 0.01$.



(e) Call blocking probability vs. number of samples, $p = 0.05$.



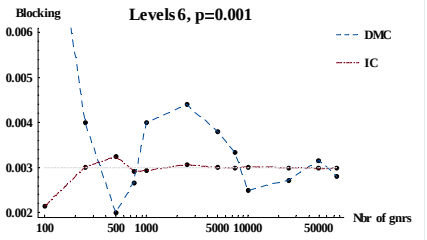
(f) Variance vs. number of samples, $p = 0.05$.

Figure 10: Comparison in an 8×8 Banyan net, where p is the probability that a connection is in progress.

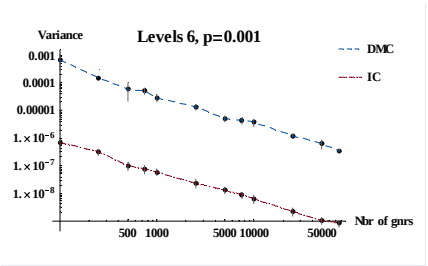
increases in comparison to the inverse convolution method.

6.3 Exact solutions

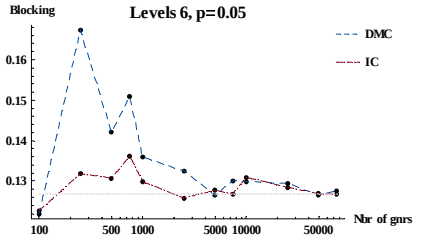
We computed exact solutions of call blocking probability of class $\rho(0,0) = 0$ for small 4×4 and 8×8 Banyan nets with the same assumptions as used in simulations, see equation (15). This was executed simply by going through all



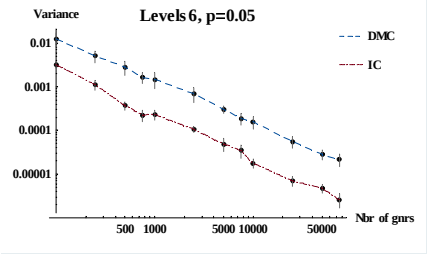
(a) Call blocking probability vs. number of samples, $p = 0.001$.



(b) Number of samples vs. standard deviation, $p = 0.001$.



(c) Call blocking probability vs. number of samples, $p = 0.05$.



(d) Number of samples vs. standard deviation, $p = 0.05$.

Figure 11: Comparison in a 64×64 Banyan net, where p is the probability that a connection is in progress.

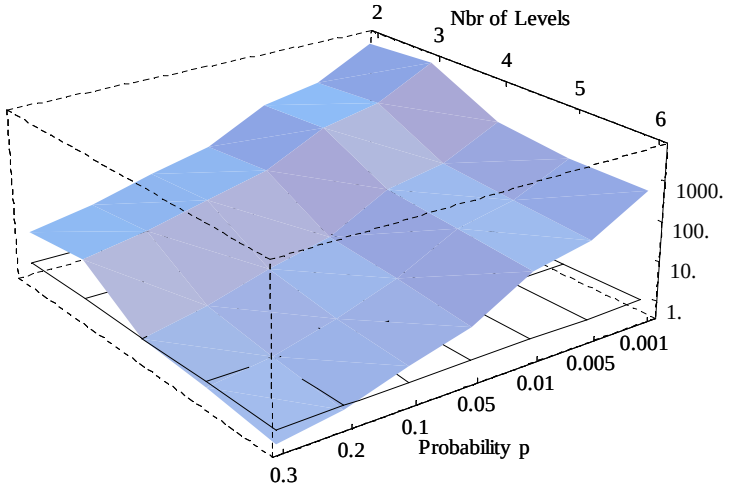
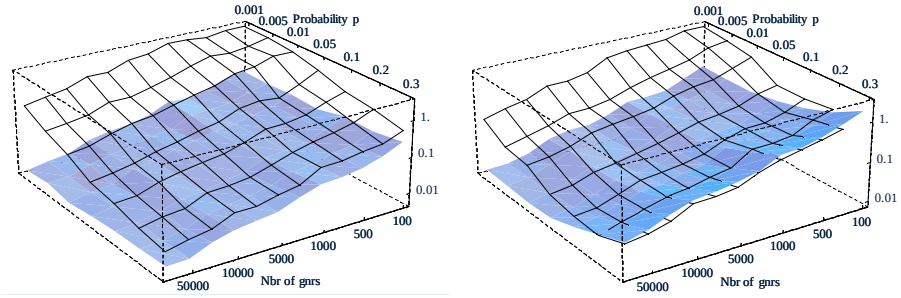


Figure 12: The mean ratio of the estimated variance as a function of probability p and the number of levels. Wire framed surface: rate = 1.



(a) Relative deviation vs. number of samples and probability p, a 4×4 Banyan net.

(b) Relative deviation vs. number of samples and probability p, a 32×32 Banyan net.

Figure 13: Relative deviation comparison. Wire framed surface: direct Monte Carlo method, solid surface: inverse convolution method.

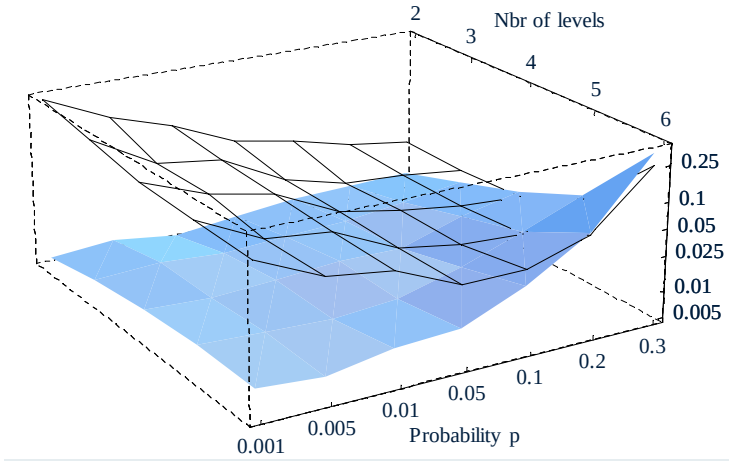


Figure 14: Relative deviation vs. number of levels and probability p. Relative deviation is estimated with 75000 samples in each point. Wire framed surface: direct Monte Carlo method, solid surface: inverse convolution method.

states $\mathbf{z} \in \omega_0 = \{\mathbf{z} \in \omega \mid z_0 = 0\}$, and examining corresponding states $\mathbf{x} \in \Omega$, i.e. does $\mathbf{x}$ belong to the set of allowed states Ω_0^C or is it a state of the set of blocking states $\mathcal{B}_0^C$ and then weighting it with probability $\pi(\mathbf{x})$, which was kept in a symbolic form. In case of a 4×4 Banyan net there are $5^3 = 125$ states and in case of an 8×8 net $9^7 = 4782969$ states in the set ω_0.

From equation (11) we get

$$\beta_0^C(p) = \sum_{\mathbf{x}:\mathbf{z}\in\omega_0} 1_{\mathbf{x}\in\mathcal{B}_0^C} \pi(\mathbf{x}),$$

and

$$\gamma_0^C(p) = \sum_{\mathbf{x}:\mathbf{z}\in\omega_0} 1_{\mathbf{x}\in\Omega_0^C}\pi(\mathbf{x}),$$

and the equation for the call blocking probability is

$$B_0^C(p) = \frac{\beta_0^C(p)}{\gamma_0^C(p)}.$$

Calculation of $\beta_0^C(p)$ and $\gamma_0^C(p)$ was implemented with Mathematica that was used to generate and weight the states, calculate and simplify the sums, together with a C-code that was used to check whether a state belong to the set ω_0 or $\mathcal{B}_0^C$. The interface was implemented and all symbolic calculations were executed with Mathematica.

The exact formulae for the call blocking probabilities are

$$B_{4\times4}(p) = \frac{p}{4}\frac{16-12p-p^2}{4-4p+p^3}$$
$$B_{8\times8}(p) = \frac{p}{8}\frac{6144-9216p-7808p^2+20480p^3-10656p^4+616p^5+447p^6}{512-2304p^2+1920p^3+976p^4-1536p^5+444p^6-11p^7},$$

where p is the probability that a connection of an arbitrary class is in progress.

7 Conclusions

In this paper we have presented a Banyan net and derived the exact expressions for the time and call blocking probabilities. We have also derived two Monte Carlo methods for estimation of these probabilities and presented a specific user model to satisfy the requirements of the use of the simulation methods.

When estimating the blocking probabilities, the confidence of the estimate depends on the number of samples that hit in the blocking region. The probability to hit in the desired region is small if the probabilities that a connection is in progress are small and thereby, with static Monte Carlo method samples are required massively.

In the inverse convolution method the idea is to generate the samples directly into the blocking region. First the blocking region is divided into parts associated with single links and then blocking caused by each link is estimated separately. The benefit in comparison with the static Monte Carlo method was seen in the simulations.

Numerical results show that both methods give the same estimate for the call blocking probability in all cases examined. In terms of variance and relative deviation the inverse convolution method gives remarkably better results than the static Monte Carlo method. Considering the variance, the inverse convolution method can be even 6700 times more effective than the static Monte Carlo method. Nevertheless, it has to be noticed that inverse convolution method performs at most effectively when the load of the system is low, i.e. blocking probability $\ll 5\%$. The inverse convolution method has also a higher memory requirement than the static Monte Carlo method. When the load of the system grows, the efficiency of the inverse convolution method decreases and the the static Monte Carlo method gives results with the same accuracy.

References

[1] Chen, T. M., and Liu, S. S. *ATM Switching Systems.* Artech House, 1995.

[2] Kelly, F. P. *Reversibility and Stochastic Networks.* John Wiley & Sons, 1979.

[3] Lassila, P., Karvo, J., and Virtamo, J. Efficient importance sampling for monte carlo simulation of multicast networks. In *Proc. INFOCOM'01* (Anchorage, Alaska, Apr. 2001), pp. 432–439.

[4] Lassila, P. E., and Virtamo, J. T. Nearly optimal importance sampling for Monte Carlo simulation of loss systems. *ACM Transactions on Modeling and Computer Simulation (TOMACS) 10*, 4 (Oct. 2000), 326–347.

[5] Nyberg, E., Virtamo, J., and Aalto, S. An exact end-to-end blocking probability algorithm for multicast networks. Submitted for publication, Sept. 2000.

[6] Schassberger, R. Two remarks on insensitive stochastic models. *Adv. Appl. Prob. 18* (1986), 791–814.