Efficient Importance Sampling for Monte Carlo Simulation of Multicast Networks

P. Lassila, J. Karvo and J. Virtamo
Laboratory of Telecommunications Technology
Helsinki University of Technology
P.O.Box 3000, FIN-02015 HUT, Finland
Email: {Pasi.Lassila, Jouni.Karvo, Jorma.Virtamo}@hut.fi

Abstract— **We consider the problem of estimating blocking probabilities in a multicast loss system via simulation, applying the static Monte Carlo method with importance sampling. An approach is introduced where the original estimation problem is first decomposed into independent simpler sub-problems, each roughly corresponding to estimating the blocking probability contribution from a single link. Then we apply importance sampling to solve each sub-problem. The importance sampling distribution is the original distribution conditioned on that the state is in the blocking state region of a single link. Samples can be generated from this distribution using the so called inverse convolution method. Finally, a dynamic control algorithm is used for optimally allocating the samples between different sub-problems. The numerical results demonstrate that the variance reduction obtained with the method is remarkable, between 400 and 36 000 in the considered examples.**

Keywords— **Multicast, loss systems, simulation, Monte Carlo methods, variance reduction, importance sampling.**

I. Introduction

We consider the calculation of blocking probabilities in tree-structured multicast networks with dynamic membership. In these networks, users at the leaf nodes can join or leave any of the several multicast channels offered by one source (i.e. the root of the tree) in the network. The users joining the network form dynamic multicast connections that share the network resources. Blocking occurs when there are not enough resources available in the network to satisfy the resource requirements of a request. Since blocked calls are lost, this system is also called a "multicast loss system". This scheme is applicable to circuit switched systems as well as packet switched systems with strict quality guarantees for multicast flows.

Karvo et al. [1] studied this system under the assumption of having an infinite user population generating call requests at the leaf nodes and a single finite capacity link in the network. An exact algorithm to compute the blocking probabilities was derived. This work was extended by Boussetta and Belyot [2] by adding unicast traffic to the system. Reduced load approximations of the blocking probabilities in a network were derived in [3]. An exact algorithm for the network case has been given in Nyberg et al. [4]. A problem with the exact solution, however, is that it cannot be computed for networks with a large number of channels, I, due to the exponential growth of the size of the state space; the complexity of the algorithm is of order $O(2^{2I})$ (however, the complexity grows only linearly with respect to the number of links in the network). Therefore, to be able to analyse systems with a larger number of channels, new methods need to be developed for estimating the blocking probabilities quickly and reliably.

One possible approach is to use simulations. As the form of the stationary distribution is known, the static Monte Carlo (MC) method can be used. In order to make the simulation more efficient, it is possible to use importance sampling (IS), where one uses an alternative sampling distribution, which makes the interesting samples more likely than under the original distribution. The twist in the distribution is then corrected for by weighting the samples with the so called likelihood ratio.

The use of IS in MC estimation of blocking probabilities has been previously studied in [5], [6], [7], and [8]. However, the particular loss system that has been studied in these works is the so called multiservice loss system. The multicast network studied here is in many ways different from the multiservice loss system, but it possesses sufficiently common features with the multiservice loss system to allow us to apply in this case the so called inverse convolution method developed in Lassila and Virtamo [8].

In this paper we first show how the problem can be decomposed into independent sub-problems. The decomposition corresponds to breaking the blocking probability down into components each of which essentially gives the blocking probability contribution from a single link. Then we present an efficient IS distribution to be used to estimate the blocking probability contribution from each link. The distribution is a conditional distribution and allows one to generate samples directly into the set of blocking states of a given link, assuming that link solely to have a finite capacity. To generate these samples we use the inverse convolution method. We will show via numerical examples the efficiency of the above method when compared with a direct Monte Carlo approach.

The paper is organized as follows. Section 2 presents briefly the multicast loss system and the decomposition approach. In section 3 we discuss the efficient estimation of blocking probabilities when applying the decomposition and importance sampling. Section 4 contains the main result of the paper, describing the inverse convolution method for multicast loss system. In section 5 we describe the dynamic method for optimally allocating the number of samples to be used for each sub-problem and give some numerical examples demonstrating the effectiveness of the method. Section 6 contains our conclusions.

II. The multicast loss system

We define the multicast loss system in the same way as Nyberg et al. [9]. Consider a network consisting of J links, indexed with $j = 1, \ldots, J$, link j having a capacity of C_j resource units. The set of all links is denoted by $\mathcal{J}$. The network is organized as a tree, where the root link is denoted by J. The set $\mathcal{U}$ denotes the set of user populations, located at the leaves of the tree. The leaf links and the user populations connected to them are indexed with the same index $u \in \mathcal{U} = \{1, \ldots, U\}$. The set of links on the route from user u to the root node is denoted by $\mathcal{R}_u$. The user populations which use link j, i.e. for which link $j \in \mathcal{R}_u$, are denoted by $\mathcal{U}_j$. The size of the set $\mathcal{U}_j$ is denoted by U_j. The multicast network supports I channels, indexed with $i \in \mathcal{I} = \{1, \ldots, I\}$. The channels originating from the root node represent different multicast transmissions, from which the users may choose. Each channel has a capacity requirement $d_i \in \mathbf{d} = \{d_i; i \in \mathcal{I}\}$. We assume the d_i and C_j to be integer multiples of a basic resource unit. In this work, we use the same capacity requirement for all links, but it is trivial to generalize this model to link dependent capacity requirements, by using $d_{j,i}$.

The state of a link j is defined by the states of the channels i. Each channel may be either *on* (1) or *off* (0), i.e. the state of channel i on link j is $Y_{j,i} \in \{0, 1\}$. The state of a link is denoted by the vector $\mathbf{Y}_j = \{Y_{j,i}; i \in \mathcal{I}\} \in \mathcal{S}$, where $\mathcal{S}$ is the link state space $\mathcal{S} = \{0, 1\}^I$. A multicast connection is defined by the pair (u, i) of the user u (leaf link) and channel i. Synonymously, we refer to the multicast connection (u, i) as a traffic class. The network state $\mathbf{X}$ is defined by the states of the leaf links $\mathbf{Y}_u \in \mathcal{S}$,

$$\mathbf{X} = (\mathbf{Y}_u; u \in \mathcal{U}) = (Y_{u,i}; u \in \mathcal{U}, i \in \mathcal{I}) \in \Omega, \qquad (1)$$

where $\Omega = \{0, 1\}^{U \times I}$ denotes the network state space. The state of link j is determined by the network state as follows:

$$\mathbf{Y}_j = \begin{cases} \mathbf{Y}_u & \text{if } j = u \in \mathcal{U}, \\ \bigoplus\limits_{u' \in \mathcal{U}_j} \mathbf{Y}_{u'} & \text{otherwise,} \end{cases} \qquad (2)$$

where $\oplus$ denotes an OR-operation between channel states in different links, i.e. channel i is *on* on link j if and only if there is a link $u \in \mathcal{U}_j$ on which the channel is *on*. We denote the link capacity occupancy (or link occupancy for brevity) of link j by S_j,

$$S_j = \mathbf{d} \cdot \mathbf{Y}_j = \sum_{i=1}^{I} d_i Y_{j,i}.$$

Now, in a finite capacity network, the state space is truncated by the capacity constraints of the links,

$$\tilde{\Omega} = \left\{ \mathbf{x} \in \Omega \,\middle|\, S_j \le C_j, \forall j \in J \right\}.$$

A. Probability distributions and blocking

Let us assume that in a system with infinite link capacities, the user populations of the leaf links are independent, and that the leaf link distributions $\pi_u(\mathbf{y}_u) = \mathrm{P}\{\mathbf{Y}_u = \mathbf{y}_u\}, u \in \mathcal{U}$ are known, and represent stationary distributions of reversible Markov processes satisfying the detailed balance equations. Several types of user population models of this kind have been discussed in [9].

In our work, we use a user population model for which traffic classes (u, i) are independent so that

$$\pi_u(\mathbf{y}) = \prod_{i \in \mathcal{I}} p_{u,i}^{y_i} (1 - p_{u,i})^{1 - y_i}, \qquad (3)$$

where $p_{u,i} = \mathrm{P}\{Y_{u,i} = 1\}$ is the probability that channel i is *on* on leaf link u. The probability $p_{j,i}$ of channel i to be in the *on* state on any link j may be calculated from the corresponding known probabilities of the leaf links $\mathcal{U}_j$,

$$p_{j,i} = 1 - \prod_{u \in \mathcal{U}_j} (1 - p_{u,i}).$$

An example of this kind of a process might be an infinite population of users generating calls (connections to the root of the tree) as a Poisson process with intensity λ_u. Arriving calls select channel i with a probability α_i. The channel *on*-probability can be calculated by $p_{u,i} = 1 - \exp(-\lambda_u \alpha_i \tau_i)$, where τ_i denotes the mean holding time of channel i calls. Note that the steady state distribution is insensitive to the call holding time distribution.

The steady state probabilities $\pi(\mathbf{x})$ of the network states in a system with infinite link capacities can then be calculated from

$$\pi(\mathbf{x}) = \mathrm{P}\{\mathbf{X} = \mathbf{x}\} = \prod_{u \in \mathcal{U}} \pi_u(\mathbf{y}_u),$$

since the user populations are independent. Due to the assumed detailed balance, the probabilities $\tilde{\pi}(\mathbf{x})$, $\mathbf{x} \in \tilde{\Omega}$, of states in a system with finite link capacities are now obtained simply by truncation,

$$\tilde{\pi}(\mathbf{x}) = \mathrm{P}\{\mathbf{X} = \mathbf{x} | \mathbf{X} \in \tilde{\Omega}\} = \begin{cases} \dfrac{\pi(\mathbf{x})}{P(\tilde{\Omega})}, & \mathbf{x} \in \tilde{\Omega}, \\ 0, & \text{otherwise,} \end{cases}$$

where $P(\tilde{\Omega}) = \mathrm{P}\{\mathbf{X} \in \tilde{\Omega}\} = \sum_{\mathbf{x} \in \tilde{\Omega}} \pi(\mathbf{x})$.

In a finite capacity network, blocking occurs when a user tries to establish a connection for channel i, and there is at least one link $j \in \mathcal{R}_u$ where the channel is not already *on* and there is not enough spare capacity for setting the channel *on*. Without loss of generality, we assume that the channels are ordered so that the channel for which we are calculating the blocking probability has the greatest index I. With this convention, the channel index is unnecessary and will be omitted for most of the notation. Let S'_j denote link occupancy due to the channels $\{1, \ldots, I-1\}$, i.e.

$$S'_j = \sum_{i=1}^{I-1} d_i Y_{j,i}.$$

Then, the set $\mathcal{B}_u$ of states where connections of traffic class (u, I) are blocked is defined as

$$\mathcal{B}_u = \left\{ \mathbf{x} \in \tilde{\Omega} \,\middle|\, \exists j \in \mathcal{R}_u : S'_j > C_j - d_I, \right\}.$$

We use the expression "link j blocks" if for link j, $S'_j > C_j - d_I$. Thus the set $\mathcal{B}_u$ consists of the states where at least one link blocks. Then the time blocking probability for traffic class (u, I) is

$$B_u = \mathrm{P}\{\mathbf{X} \in \mathcal{B}_u \,|\, \mathbf{X} \in \tilde{\Omega}\} = \frac{\mathrm{P}\{\mathbf{X} \in \mathcal{B}_u\}}{\mathrm{P}\{\mathbf{X} \in \tilde{\Omega}\}} = \frac{P(\mathcal{B}_u)}{P(\tilde{\Omega})}, \quad (4)$$

where $P(\mathcal{B}_u) = \mathrm{P}\{\mathbf{X} \in \mathcal{B}_u\}$.

B. Decomposition

In order to divide the task of estimating $P(\mathcal{B}_u)$ to simpler sub-problems, we partition $\mathcal{B}_u$ into sets $\mathcal{E}_u^j$. $\mathcal{E}_u^j$ is defined as the set of points in $\mathcal{B}_u$ where link j blocks but none of the links closer to user u block,

$$\mathcal{E}_u^j = \mathcal{B}_u \cap \left\{ \mathbf{x} \in \Omega \,\middle|\, S'_j > C_j - d_I \wedge S'_{j'} \leq C_{j'} - d_I, \forall j' \in \mathcal{R}_u^j \right\},$$

where $\mathcal{R}_u^j$ denotes the set of links on the path from u to j, including link u but not link j. The $\mathcal{E}_u^j$ obviously form a partitioning of $\mathcal{B}_u$, i.e.

$$\mathcal{B}_u = \bigcup_{j \in \mathcal{R}_u} \mathcal{E}_u^j,$$

and $\mathcal{E}_u^j \cap \mathcal{E}_u^{j'} = \emptyset$, when $j \neq j'$. From this it follows that

$$\mathrm{P}\{\mathbf{X} \in \mathcal{B}_u\} = \sum_{j \in \mathcal{R}_u} \mathrm{P}\{\mathbf{X} \in \mathcal{E}_u^j\}, \quad (5)$$

and we have decomposed the problem into simpler sub-problems of determining the probabilities $\mathrm{P}\{\mathbf{X} \in \mathcal{E}_u^j\}$. The probability $\mathrm{P}\{\mathbf{X} \in \mathcal{E}_u^j\}$ can be thought of as the blocking probability contribution due to link j. It should be noted, however, that blocking in the states where several links block can be arbitrarily attributed to any of the blocking links. We have adopted the convention which attributes it to the blocking link closest to the user.

For later use, we introduce the superset $\mathcal{D}_u^j \supset \mathcal{E}_u^j$, which will have an important role in importance sampling,

$$\mathcal{D}_u^j = \left\{ \mathbf{x} \in \Omega \,\middle|\, C_j - d_I < S'_j \leq C_j \wedge Y_{j,I} = 0 \right\}.$$

This set corresponds to blocking states in a system where link j has a finite capacity C_j but all other links have infinite capacity. However, in real systems, all links have finite capacity, and several links could block simultaneously. Thus, sets $\mathcal{D}_u^j$ are not disjoint unlike their subsets $\mathcal{E}_u^j$.

The sets $\mathcal{D}_u^j$ and $\mathcal{E}_u^j$ are illustrated in Figure 1. The real state space is impossible to draw, since it has $U \times I$ dimensions, and has only two points per each dimension. Thus, the figure only illustrates the principle of decomposition showing an example where the sets consist of elements having two components (x_1, x_2). In the figure, $\mathcal{B}_u$ is represented by the grey area and it consists of the union of three disjoint subsets $\mathcal{E}_u^2, \mathcal{E}_u^3$ (light grey areas) and $\mathcal{E}_u^1$ (dark grey area). Also note that each $\mathcal{E}_u^j$ is a subset of the corresponding $\mathcal{D}_u^j$.

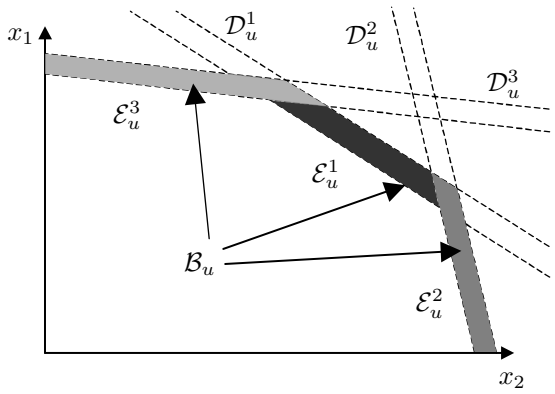

Fig. 1. Decomposition of $\mathcal{B}_u$ into sets $\mathcal{D}_u^j$ and $\mathcal{E}_u^j$.

III. Efficient importance sampling

In what follows we discuss the efficient estimation of the blocking probabilities. As the form of the stationary distribution $\pi(\mathbf{x})$ is known, a natural choice for the simulation method is the static Monte Carlo method. The main problem in the simulation is to quickly get a good estimate for $\mathrm{P}\{\mathbf{X} \in \mathcal{B}_u\}$, i.e., the numerator in Eq. (4), especially in the case when the blocking probability B_u is very small. For completeness, recall that B_u also depends on $\mathrm{P}\{\mathbf{X} \in \tilde{\Omega}\}$ given by the denominator of (4). This probability is usually close to 1 and is easy to estimate using the standard MC method. Therefore, in the rest of this paper we concentrate on efficient methods for estimating $\mathrm{P}\{\mathbf{X} \in \mathcal{B}_u\}$.

As already noted, Eq. (5) allows us to decompose the estimation of $\mathrm{P}\{\mathbf{X} \in \mathcal{B}_u\}$ into independent sub-problems of estimating the $\mathrm{P}\{\mathbf{X} \in \mathcal{E}_u^j\}$. Now, the idea of our importance sampling method is simply based on expressing $\mathrm{P}\{\mathbf{X} \in \mathcal{E}_u^j\}$ as a conditional probability,

$$\mathrm{P}\{\mathbf{X} \in \mathcal{E}_u^j\} = \mathrm{P}\{\mathbf{X} \in \mathcal{E}_u^j \,|\, \mathbf{X} \in \mathcal{D}_u^j\} \mathrm{P}\{\mathbf{X} \in \mathcal{D}_u^j\}. \quad (6)$$

This relation is useful from the simulation point of view since we can compute $\mathrm{P}\{\mathbf{X} \in \mathcal{D}_u^j\}$ exactly (see [1]) and we can efficiently generate points from the original distribution under the condition $\mathbf{X} \in \mathcal{D}_u^j$, as explained later.

Then we only need to estimate via MC simulation the conditional probability $p' = \mathrm{P}\{\mathbf{X} \in \mathcal{E}_u^j \mid \mathbf{X} \in \mathcal{D}_u^j\}$ instead of $p = \mathrm{P}\{\mathbf{X} \in \mathcal{E}_u^j\}$ (see Figure 2). The estimation of p' is much more efficient than the estimation of p since typically p' is much greater than p. The efficiency gain obtained with the above can

be shown as follows. When estimating p via standard MC simulation each sample is an independent Bernoulli variable and the relative error (or relative deviation) of the estimate, given by the ratio of the standard deviation and the mean of the estimate, after N samples have been drawn is $\sqrt{(1-p)/(pN)}$. Similarly, when estimating p' the relative deviation is given by $\sqrt{(1-p')/(p'N)}$ Typically, the blocking probabilities are of the order 1% and to illustrate the efficiency gain we can consider e.g. an example where $p = 0.005$. Then we need almost 80 000 samples to get a relative error of 5%. On the other hand, when estimating p' we need less samples to reach the same accuracy level. How much less depends on how big a part of $\mathcal{E}_u^j$ is inside $\mathcal{D}_u^j$. Typically p' is in the range $0.5, \ldots, 1$. Assuming, e.g. that $p' = 0.9$, we only need about 45 samples to reach the same 5% relative error level, giving us a decrease by a factor of almost 2000 in the required sample size. Our numerical results also indicate that variance reductions of this order can indeed be obtained.

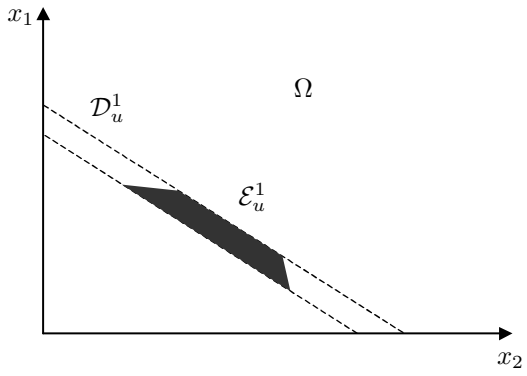

Fig. 2. Estimation of $\mathrm{P}\{\mathbf{X} \in \mathcal{E}_u^j \,|\, \mathbf{X} \in \mathcal{D}_u^j\}$.

From Eq. (6) we have the following estimator, $\widehat{\eta}_u^j$, for $\eta_u^j = \mathrm{P}\{\mathbf{X} \in \mathcal{E}_u^j\}$,

$$\widehat{\eta}_u^j = \frac{v_u^j}{N_j} \sum_{n=1}^{N_j} 1_{\mathbf{X}_n^* \in \mathcal{E}_u^j}, \tag{7}$$

where $v_u^j = \mathrm{P}\{\mathbf{X} \in \mathcal{D}_u^j\}$ and $\mathbf{X}_n^*$ denotes samples drawn from the conditional distribution $p_j^*(\mathbf{x}) = \mathrm{P}\{\mathbf{X} = \mathbf{x} \mid \mathbf{X} \in \mathcal{D}_u^j\}$. Observe that Eq. (7) corresponds to the same estimator for $\hat{\eta}_u^j$ if we take the conditional distribution $p_j^*(\mathbf{x})$ as our importance sampling distribution in the importance sampling estimator

$$\widehat{\eta}_u^j = \frac{1}{N_j} \sum_{n=1}^{N_j} 1_{\mathbf{X}_n^* \in \mathcal{E}_u^j} \, w(\mathbf{X}_n^*),$$

where $w(\mathbf{x}) = \pi(\mathbf{x})/p_j^*(\mathbf{x}) = v_j$ is the so called likelihood ratio, the value of which in our case is constant.

Finally, the estimator for $P(\mathcal{B}_u)$ is simply

$$\widehat{P}(\mathcal{B}_u) = \sum_{j \in \mathcal{R}_u} \widehat{\eta}_u^j.$$

Given the total number of samples N to be used for the estimator, the number of samples N_j allocated to each sub-problem is a free parameter. In section V-A we show how to choose each N_j to minimize the variance of $\widehat{P}(\mathcal{B}_u)$.

IV. Inverse convolution method

In this section, we present the inverse convolution method (IC) for sample generation. We are now only considering the estimation of one η_u^j for fixed $j \in \mathcal{R}_u$ and traffic class (u, I). The following method is based on the observation that it is relatively easy to generate points from the conditional distribution $p_j^*(\mathbf{x}) = \mathrm{P}\{\mathbf{X} = \mathbf{x} \mid \mathbf{X} \in \mathcal{D}_u^j\}$ by reversing the steps used to calculate the occupancy distribution of the considered link. Note that the condition $\mathbf{X} \in \mathcal{D}_u^j$ is a condition expressed in terms of the occupancy, S_j', of the considered link. The idea in the inverse convolution method is to first generate a sample of $\mathbf{Y}_j$ such that the occupancy of the link is in the blocking region. Then, given the state $\mathbf{Y}_j$, the state of the network, i.e. states of the leaf links, is generated. The mapping $\bigoplus : \mathbf{X} \to \mathbf{Y}_j$ is surjective, having several possible network states $\mathbf{X}$ generating the link state $\mathbf{Y}_j$, and we draw one of them according to their probabilities.

The main steps of the simulation can be summarized as follows:

1. Generate the states for leaf links u by
   (a) Generate a sample state $\mathbf{Y}_j$ under the condition $C_j - d_I < S_j' \leq C_j \wedge Y_{j,I} = 0$ for link j.
   (b) Generate the leaf link states $\mathbf{Y}_u$, $u \in \mathcal{U}_j$, with the condition that link j state $\mathbf{Y}_j = \oplus_{u \in \mathcal{U}_j} \mathbf{Y}_u$ is given.
   (c) Generate the states $\mathbf{Y}_u$, $u \in \mathcal{U} - \mathcal{U}_j$ for the rest of the leaf links as in the normal Monte Carlo simulation.
2. The sample state of the network $\mathbf{X}_n \in \mathcal{D}_u^j$ consists of the set of all sample states of leaf links generated with step 1.
3. To collect the statistics for estimator (7), check if $\mathbf{X}_n \in \mathcal{E}_u^j$.

The above steps are repeated for generating N_j samples. The method of generating a sample for link j (step 1a) is explained in more detail in section IV-A. The method for generating the leaf link states from the link state (step 1b) is explained in section IV-B. See figure 3.

A. Generating a sample for $\mathcal{D}_u^j$

As already noted, we have partitioned the set of blocking states into disjoint sets $\mathcal{E}_u^j$. However, it is not easy to generate samples directly to these sets. Instead, we generate samples to sets $\mathcal{D}_u^j$ which correspond to the states in which at least link j blocks. After that it is possible to check if the sample belongs to the set $\mathcal{E}_u^j$ to collect the sum in Eq. (7).

First, the link occupancy S_j is easily calculated recursively as follows. Let $S_{j,i}$ denote link occupancy due to the first i channels,

$$S_{j,i} = \sum_{i' \leq i} d_{i'} Y_{j,i'}.$$

Then $S_j = S_{j,I}$ and $S_j' = S_{j,I-1}$. The $Y_{j,i}$ are mutually independent, and we can express $S_{j,i} = S_{j,i-1} + d_i Y_{j,i}$, where

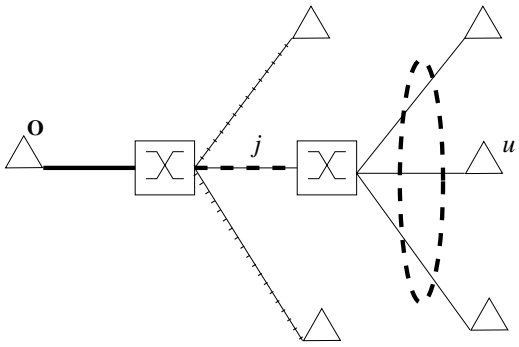

Fig. 3. Example of generating a sample in the set $\mathcal{D}_u^j$. First, a sample state for the link j, denoted by the thick dashed line, is generated by inverse convolution. Given that channel i is *on* in this state, the states of that channel are generated for the links marked by the dashed ellipse by another inverse convolution step (if channel i is *off* on link j then it is *off* on all those links). This is repeated for all i. States for the links denoted by ticks are generated by a simple draw.

$S_{j,i-1}$ and $Y_{j,i}$ are independent. For sample generation, we are only interested in the occupancy generated by the first $I-1$ channels, S'_j, since a call for a channel cannot be blocked if it is already in the *on* state. This is also reflected in the definition of the set $\mathcal{D}_u^j$.

Let $\mathrm{P}\{S'_{j,i} = x\} = q_{j,i}(x)$ denote the probability distribution of $S'_{j,i}$. Now, the probability mass v_j of the set $\mathcal{D}_u^j$, can be calculated as

$$v_j = \mathrm{P}\{\mathbf{X} \in \mathcal{D}_u^j\} = (1-p_{j,I}) \sum_{i=C_j-d_I+1}^{C_j} q_{j,I-1}(i).$$

The link occupancy distribution $q_{j,I-1}$ may be calculated recursively by convolution:

$$q_{j,i}(x) = q_{j,i-1}(x-d_i)p_{j,i} + q_{j,i-1}(x)(1-p_{j,i}), \qquad (8)$$

where the recursion starts with $q_{j,0}(x) = 1_{x=0}$. For interpretation of the convolution step, note that the event $\{S_{j,i} = x\}$ is the union of the events $\{Y_{j,i} = y_{j,i}, S_{j,i-1} = x - d_i y_{j,i}\}$, $y_{j,i} \in \{0,1\}$, where $y_{j,i} = 0$ means that the channel i is in the *off* state, and $y_{j,i} = 1$ means that the channel i is in the *on* state on link j. The corresponding probabilities are $q_{j,i'-1}(x)(1-p_{j,i'})$ and $q_{j,i'-1}(x-d_{i'})p_{j,i'}$, respectively. Conversely, we can infer what is the conditional probability of the event $\{Y_{j,i} = 1,\, S_{j,i-1} = x - d_i\}$ given that $S_{j,i} = x$,

$$\mathrm{P}\{Y_{j,i} = 1,\, S_{j,i-1} = x - d_i \,|\, S_{j,i} = x\} = \frac{q_{j,i-1}(x-d_i)p_{j,i}}{q_{j,i-1}(x-d_i)p_{j,i} + q_{j,i-1}(x)(1-p_{j,i})} = \frac{q_{j,i-1}(x-d_i)p_{j,i}}{q_{j,i}(x)}. \qquad (9)$$

The probability of the event $\{Y_{j,i} = 0,\, S_{j,i-1} = x \,|\, S_{j,i} = x\}$ is then $1 - \mathrm{P}\{Y_{j,i} = 1,\, S_{j,i-1} = x - d_i \,|\, S_{j,i} = x\}$.

Having all the necessary tools, we are now able to generate samples to the set $\mathcal{D}_u^j$. This set corresponds to states in which $C_j - d_I < S'_j \le C_j$ and $Y_{j,I} = 0$. Generation of a state starts by drawing a value for $S'_j = S_{j,I-1}$ using the distribution $q_{j,I-1}(\cdot)$ with the condition that $C_j - d_I < S'_j \le C_j$. This conditional distribution can be precomputed and stored.

Then, given the value of $S_{j,I-1}$, the state $Y_{j,i}$ of each channel ($i = I-1, \ldots, 1$) is drawn in turn using probabilities (9). Concurrently with the state $Y_{j,i}$, the value of $S_{j,i-1}$ becomes determined. This is then used as the conditioning value in the next step to draw the value of $Y_{j,i-1}$ (and of $S_{j,i-2}$), etc. Drawing a sample for $Y_{j,i}$ requires just generation of a uniform random number from the interval (0,1) and setting the channel to *on* state if the number is smaller or equal than the probability. Note that for reasonable sizes of links, it is advantageous to store the probabilities for fast generation of samples.

The next subsection presents a method for drawing leaf link states $\mathbf{Y}_u$, given the state $\mathbf{Y}_j$ of link j.

B. Generating leaf link states from a link state

Having drawn a value for state $\mathbf{Y}_j$ of link j, it is possible to draw values of the state vectors $\mathbf{Y}_u$, $u \in \mathcal{U}$ of the leaf links. For $u \in \mathcal{U}_j$, states $\mathbf{Y}_u$ are generated under the condition $\mathbf{Y}_j = \oplus_{u \in \mathcal{U}_j} \mathbf{Y}_u$ using a similar inverse convolution procedure as above. This condition can be broken down into separate conditions for each channel, i.e. for each i we have a separate problem of generating the values $Y_{u,i}$, $u \in \mathcal{U}$, under the condition $Y_{j,i} = \oplus_{u \in \mathcal{U}_j} Y_{u,i}$ with a given $Y_{j,i}$. Note that only channels i which are in the *on* state on link j, $Y_{j,i} = 1$, need to be considered. If $Y_{j,i} = 0$, then necessarily $Y_{u,i} = 0$ for all $u \in \mathcal{U}_j$. The above conditions affect leaf links $u \in \mathcal{U}_j$. For other links $u \in \mathcal{U} - \mathcal{U}_j$, the states $\mathbf{Y}_u$ are independently generated from the distribution (3).

First, let us consider a convolutional approach for generating a link state for channel i and link j *if* we already know the states for each link $u \in \mathcal{U}_j$. In this section, we use an index $u_j \in \{1, \ldots, U_j\} = \mathcal{U}_j$ for the subset of leaf links. Let $S_{u_j,i} = x$ denote the event that the channel is on state x on link j when $u' = 1, \ldots, u_j$ leaf links have been counted for, i.e.

$$S_{u_j,i} = \bigoplus_{u' \le u_j} y_{u',i}.$$

Let $\mathrm{P}\{S_{u_j,i} = 1\} = q_{u_j,i}$ be the probability that the channel is on in at least one of the links $\{1, \ldots, u_j\} \subset \mathcal{U}_j$. These probabilities can be calculated recursively as follows:

$$\begin{aligned} q_{u_j,i} &= 1 - (1-p_{u_j,i})(1-q_{u_j-1,i}) \\ &= (1-q_{u_j-1,i})p_{u_j,i} + q_{u_j-1,i}. \end{aligned}$$

The recursion starts with $q_{0,i} = 0$. If $S_{u_j-1,i} = 1$, then necessarily $S_{u_j,i} = 1$ in any case, see figure 4. Conversely, to generate the state for each leaf link, given the value of $Y_{j,i}$, we first check if the channel would be *off* after $u_j - 1$ leaf links counted for, since if it is, then it has to be *on* on leaf link u_j, and *off* on the rest of the links, which happens with probability

$$\mathrm{P}\{S_{u_j-1,i} = 0 \,|\, S_{u_j,i} = 1\} = \frac{(1-q_{u_j-1,i})p_{u_j,i}}{(1-q_{u_j-1,i})p_{u_j,i} + q_{u_j-1,i}} = \frac{(1-q_{u_j-1,i})p_{u_j,i}}{q_{u_j,i}}.$$

Note that the event $S_{u_j-1,i} = 0$ implies directly that $\{Y_{u',i} = 0, \forall u' < u_j\}$. The probability $\mathrm{P}\{S_{u_j-1,i} = 1 \,|\, S_{u_j,i} = 1\}$ is given by $1 - \mathrm{P}\{S_{u_j-1,i} = 0 \,|\, S_{u_j,i} = 1\}$. In the case of the event $S_{u_j-1,i} = 1$, we need to check if the channel is *on* on the leaf link u, which happens with probability

$$\mathrm{P}\{Y_{u_j,i} = 1 \,|\, S_{u_j-1,i} = 1\} = p_{u_j,i}.$$

Thus we generate a uniform random number first to decide if the leaf link is the last one for which the channel is on. If it is not, we generate yet another uniform random number to decide if the channel should be set on.

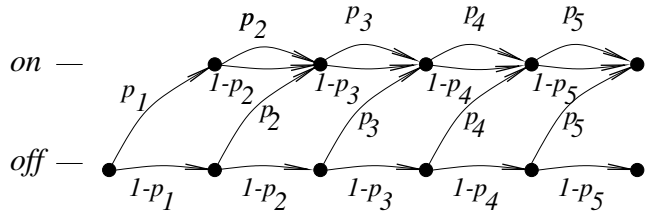

Fig. 4. Convolution approach for generating link state from leaf link states. Each link is convoluted in turn. Note that if the channel is already on, it will be on in the next step also, independently of the state of the convoluted link.

This procedure is repeated for each channel. As a result of this procedure, we have the state vectors of each leaf link $u \in \mathcal{U}_j$. The rest of the leaf link states must be generated as in the normal Monte Carlo simulation using Eq. (3).

C. Complexity

The generation of samples is almost as fast as in the standard MC method, once the conditional distributions have been computed. Generation of the link state in $\mathcal{D}_u^j$ is as fast as generating a standard link state. Generating leaf link states takes $2(U-1)$ steps at maximum, compared with U steps of the standard MC method (generating U leaf link states). Thus, the worst case time to generate a sample state with the proposed method is twice that of the standard MC method. Furthermore, the memory requirements of the algorithm, i.e. the number of elements in the arrays, are not prohibitive. The number of array elements to be stored can be seen to be $I(C_j + U)$. It should be noted that the dependence on I and U is only linear, in spite of the exponential growth in I of the state space Ω.

Our method scales very favourably with the network size, essentially defined by the number of user populations U. Apparently, the decomposition leads to replication of the simulation for each link on the route of the connection. Per each subtask, the work is about the same as in standard MC. The number of links on the route, however, grows relatively slowly, and in practice will never be very large. Moreover, optimal allocation of samples between the subtasks, discussed in the next section, essentially eliminates the slight disadvantage.

V. Numerical results

A. Allocation of the sample points

We also implemented and tested a scheme of allocating sample points optimally for each estimation of $\widehat{\eta}_u^j$, as explained in [8]. The inverse convolution method with sample allocation (ICSA) is presented in this subsection. The $\widehat{\eta}_u^j$ for different links have differing variances, and contribute differently to the total variance of the estimate $\widehat{P}(\mathcal{B}_u)$. Thus, we may vary sample size for different links j to minimize the total variance. The optimal allocation of samples is

$$N_j = \frac{s_j}{\sum_{i=1}^{J} s_i} N, \quad j = 1, \ldots, J, \tag{10}$$

where we have denoted $s_j^2 = \mathrm{V}[1_{\mathbf{X}^* \in \mathcal{E}_u^j}] = (1-p')/p'$, where $p' = \mathrm{P}\{\mathbf{X}^* \in \mathcal{E}_u^j\}$.

Of course, the s_j are not known before the simulation. Therefore a dynamic sample allocation scheme is needed. One practical solution is to make the simulation in batches, using $J \cdot M$ samples per batch, where M is a suitable integer, for instance $M = 100$. In the first batch, all the samples are distributed evenly between different links, i.e., M samples are used per link. Then initial estimates for the s_j are obtained. Using these estimates, the optimal sample sizes after the second batch, i.e. for $N = 2J \cdot M$, can be calculated from (10). If the calculated N_j is less than the number of samples already used (M samples in the first batch) no samples of the new batch are allocated for that link. Otherwise, the available $J \cdot M$ new samples are distributed between the links in proportion to the deficiencies (deficiency being the difference between the calculated optimal value after the new batch and the actual number of samples used so far). Real numbers are appropriately rounded to integers. After the new batch, new estimators are calculated for the s_j and the procedure is repeated.

B. Numerical examples

Here some numerical examples are presented in order to illustrate the efficiency of the presented method in Monte Carlo simulation of the blocking probabilities. We consider the same network used in [4], [9], for which we know the exact results. The network is shown in figure 3. There is a root node, eight channels, $I = 8$, with $d_i = 1$ for all channels. The capacity of the root link is $C_J = 7$, for the others, $C_j = 6$. Each leaf link has an infinite user population offering traffic to each channel with intensity $a\alpha_i$, where α_i comes from the truncated geometric preference distribution with $p = 0.2$. We simulated blocking for channel I with three values for a: 1.0, 1.3 and 2.0 to compare the simulation methods in light load, moderate load and high load circumstances.

To this end, we estimated the relative deviation of the estimator, given by $(\mathrm{V}[\widehat{P}(\mathcal{B}_u)])^{1/2}/\widehat{P}(\mathcal{B}_u)$, for 10^4 samples (Case 1) and 10^5 samples (Case 2). For classic Monte Carlo (MC), the total number of samples were used, while for Inverse Convolution method (MC-IC), one third of samples was used for each

TABLE I

The relative deviation of the estimates $\widehat{P}(\mathcal{B}_u)$ for the first example. Case 1: 10 000 samples, Case 2: 100 000 samples.

Case	a / B_u	relative deviation		
		MC	MC-IC	MC-ICSA
1	1.0 / 0.56%	0.5883	0.0187	0.0031
	1.3 / 1.7%	0.3493	0.0214	0.0036
	2.0 / 7.2%	0.1755	0.0262	0.0047
2	1.0 / 0.56%	0.1765	0.0058	0.0010
	1.3 / 1.76%	0.1092	0.0069	0.0011
	2.0 / 7.2%	0.0527	0.0085	0.0015

estimate $\widehat{\eta}^j_{u,i}$. For Inverse Convolution with Sample Allocation (MC-ICSA), the total number of samples were allocated for each estimate, according to the algorithm. In these examples, we implemented MC-IC so that it first generates a network state for estimating $P(\tilde{\Omega})$. Then it reuses this state partially for those links not generated with inverse convolution. This induces some dependence between the estimates $\widehat{\eta}^j_u$ in the simulation, but its effect on the variance of the estimator is not too high for these networks. For MC-ICSA this scheme of reducing processing time per sample is not feasible.

As can be seen, the variance reductions obtained with the inverse convolution method are remarkable. For example, for light load ($a = 1.0$), the ratio between the deviations of the standard MC and the inverse convolution method (MC-IC) is about 30 in both cases, and between MC and MC-ICSA, about 190 in case 1 and 180 in case 2, corresponding to a decrease of 900 to 36 000 in the required sample sizes.

Also, we can note here that with the inverse convolution method the estimation of the variance of the estimates is guaranteed to be reliable. In rare event simulation (which is not the main interest here), one problem is that one can get results that appear to be very accurate judging from the estimated variance, but the results can, in reality, be far from the correct value. This can happen, e.g. when using a single heavily twisted IS distribution, and the reason is that the likelihood ratio $w(\cdot)$ can have a huge value at some point in the state space, but under the twisted distribution these points are very rare and we never encounter them during the course of a simulation run. Hence, the estimates, especially for the variance or other higher moments, can be heavily underestimated, as has been rigorously shown in [10]. With the inverse convolution method, however, the estimation is always reliable, since the observed values of the samples are bounded within the interval $[0, 1]$. Thus, the problem of the occurrence of events with a very small probability under the IS distribution but having a significant contribution to the estimate does not occur.

A second example used was a network having a larger number of channels, thus generating a somewhat larger state space, big enough for not being solvable with the exact algorithm [4]. In this case, we have the same network topology as in the first example, but use $I = 50$ channels. The link capacities are also increased: the leaf link capacities $C_u = 30$, root link capacity $C_J = 35$, and the middle link capacity $C_j = 33$. All the other parameters remained the same as in the first example. We estimated the blocking probability of a three link route from the user to the root node. The results are presented in table II. For example, the ratio between the deviations of the standard MC and MC-ICSA is about 20 in both cases for heavy load ($a = 700$), and about 140 for light load ($a = 400$), corresponding to a decrease of 400 to 19 600 in the required sample sizes. To get a feel of what kind of reductions in the run time one can obtain with the inverse convolution method, we also provided the actual CPU-time used by our simulator implemented in Matlab. Note that the classic MC algorithm could be implemented using matrix operations, which in this case increases the speed difference in favour of the classic MC. In spite of this, we see that the time used per sample in MC-IC is only five times as long as for the classic MC. Increasing the number of samples for the classic MC even ten times (case 2), however, does not bring the relative deviation anywhere near to the one of MC-IC. We also see that the added penalty for sample allocation is small enough to justify its use. The added cost for preparing simulation in MC-IC and MC-ICSA was small, about 80 ms even for this larger network.

TABLE II

The relative deviation of the estimates $\widehat{P}(\mathcal{B}_u)$, and CPU time for simulation, for the second example. Case 1: 10 000 samples, Case 2: 100 000 samples.

Case	a / B_u	relative deviation/elapsed time		
		MC	MC-IC	MC-ICSA
1	400 / 0.59%	0.5655/ 7.0 s	0.0230/ 37 s	0.0038/ 43 s
	700 / 10%	0.1245/ 6.9 s	0.0338/ 36 s	0.0063/ 41 s
2	400 / 0.59%	0.1725/ 70 s	0.0072/ 370 s	0.0012/ 420 s
	700 / 10%	0.0426/ 69 s	0.0105/ 360 s	0.0020 400 s

VI. Conclusions

In this paper we have presented a new approach to the problem of estimating blocking probabilities in a multicast network by using the static Monte Carlo simulation method and importance sampling. First we observed that the estimation problem can be decomposed into separate simpler sub-problems; estimation of blocking on each link on the route, attributing each blocking state to a single link, viz. the blocking link closest to the user on the route R_u.

For the solution of the sub-problems, we presented a method which very closely approximates the generation of samples with the ideal IS distribution, and gives a remarkable variance reduction. The idea of the method is to generate samples directly into the set of blocking states of a given link in a system, where all the other links are assumed to have an infinite capacity. This

is achieved by the inverse convolution method presented in the paper. This set of blocking states of course extends beyond the allowed state space of the system, and may generate blocking in other links, as well. Then, simulation is essentially only needed to determine which part of this set is actually inside the allowed state space, and which part of this set adds to the attributed blocking of the link. The inverse convolution method can easily be modified to cover the case studied by Nyberg et al. [4], where the multicast network model was extended to include independent background traffic on the links. This extension includes the from a practical point of view important case where the multicast tree is a part of a larger network carrying also unicast traffic.

Acknowledgement

This research has been funded by the Academy of Finland. The work of the first author has also been financially supported by the Nokia Foundation.

References

[1] J. Karvo, J. Virtamo, S. Aalto, and O. Martikainen, "Blocking of dynamic multicast connections in a single link," in *Proceedings of Broadband Communications'98*, April 1998, pp. 473–483.

[2] K. Boussetta and A. L. Belyot, "Multirate resource sharing for unicast and multicast connections," in *Proceedings of Broadband Communications'99*, November 1999.

[3] J. Karvo, J. Virtamo, S. Aalto, and O. Martikainen, "Blocking of dynamic multicast connections," to appear in Telecommunication Systems.

[4] E. Nyberg, J. Virtamo, and S. Aalto, "An exact algorithm for calculating blocking probabilities in multicast networks," in *Proceedings of Networking 2000*, Paris, May 2000, pp. 275–286.

[5] P. E. Lassila and J. T. Virtamo, "Efficient importance sampling for monte carlo simulation of loss systems," in *Proceedings of the ITC-16*. June 1999, pp. 787–796, Elsevier.

[6] M. Mandjes, "Fast simulation of blocking probabilities in loss networks," *European Journal of Operations Research*, vol. 101, pp. 393–405, 1997.

[7] K. W. Ross, *Multiservice Loss Models for Broadband Telecommunication Networks*, Springer-Verlag, 1995.

[8] P. Lassila and J. Virtamo, "Nearly optimal importance sampling for monte carlo simulation of loss systems," to appear in: ACM Transactions on Modeling and Computer Simulation.

[9] E. Nyberg, J. Virtamo, and S. Aalto, "An exact algorithm for calculating blocking probabilities in multicast networks," submitted for publication.

[10] J. S. Sadowsky, "On the optimality and stability of exponential twisting in monte carlo estimation," *IEEE Transactions on Information Theory*, vol. 39, pp. 119–128, 1993.