PRACTICAL ALGORITHM FOR CALCULATING BLOCKING PROBABILITIES IN MULTICAST NETWORKS

SAMULI AALTO AND JORMA VIRTAMO

ABSTRACT. We consider a multicast network model, where a single server node is connected with a collection of user sites by a transport network with tree topology. A predefined set of dynamic multicast (i.e. point-to-multipoint) connections use this transport network, with the server acting as the source for all these connections and the receivers located at the user sites. Earlier analytic results concerning the calculation of blocking probabilities in this kind of multicast setting have been applicable, due to the so called state space explosion, only for networks with few multicast connections. In this paper we show that, by making certain restrictive assumptions, the exact calculation of the end-to-end blocking probability becomes feasible even for large networks with many multicast connections. The main assumptions are as follows: (i) all receivers have a uniform preference distribution when making a choice (to join) between the multicast connections, (ii) the mean holding time for any receiver to be joined to any connection is the same, and (iii) the capacity needed to carry any multicast connection on any link is the same.

1. INTRODUCTION

Consider the following model of a multicast network. There is a single server node connected with a collection of user sites by a transport network with tree topology. A predefined set of dynamic multicast connections use this transport network, with the server acting as the source for all these connections and the receivers located at the user sites. All the multicast connections are available for all receivers. As an example, one can think of the delivery of TV or radio channels by a multicast capable telecommunication network, each channel requiring its own multicast connection. Requests from the receivers to join these multicast connections arrive randomly. If, upon the arrival of such a request, there are not enough resources available to establish a new leg from the receiver to the nearest node that already carries this connection, the request is blocked and lost. Otherwise the new leg is established, in which case we say that the corresponding multicast connection becomes *active* on the links belonging to this leg. Whenever a receiver disconnects, the branch of the connection tree dedicated to this specific receiver is dropped, i.e. the connection becomes *inactive* on the links belonging to this branch, and the corresponding resources are released.

It is clear that one interesting performance measure for this kind of systems is the call blocking probability, which in this dynamic multicast setting refers to the probability that an arbitrary request from a certain receiver to join a certain multicast connection is blocked and lost. Note that the blocking probability is both connection-specific and receiver-specific. Further we remark that the traditional results from the theory of loss networks are not applicable as such in our case of *dynamic* multicast connections (in contrast to the case of *static* multicast connections).

Only recently there has been some progress in the analysis of the blocking probability in this dynamic multicast setting. The first results concerned the case where new connection requests arrive according to a Poisson process. In [1] it was shown how to calculate the call

Date: May 16, 2000.
1991 *Mathematics Subject Classification.* 60K25, 60K30, 68M10.
Key words and phrases. Blocking, multicast, point-to-multipoint connection.

blocking probability for any of the multicast connections in a specific link with finite capacity assuming that all the other links have infinite capacity. The idea was to transform this problem to the problem of calculating the call blocking probability in a so called generalized Engset system, the solution of which is known. The general case with any number of finite capacity links was first treated in [2], where the end-to-end blocking probabilities were estimated by applying the (known) Reduced Load Approximation (RLA) method. The accuracy of the approximation was investigated by simulations. However, the simulations were extremly processing-intensive allowing only the evaluation of rather small networks. An exact algorithm for calculating the end-to-end blocking probabilities in the general case with any number of finite capacity links was presented in [3]. The algorithm was later [4] extended to cover the case where the connection requests are generated by finite user populations (resulting, thus, in a non-Poissonian arrival process).

Let us briefly consider the algorithms presented in [3, 4]. The purpose is to calculate the end-to-end call blocking probability for any receiver of any multicast connection. As in the case of ordinary loss models, the blocking probability can be defined by means of the stationary distribution of the state of the system. In this multicast setting, the state of the system is described by a collection of vectors, each of them consisting of two-state elements. Element i of vector u tells whether any of the receivers located at user site u is connected to multicast connection i, or not. (Note that the state space grows exponentially as a function of the number of multicast connections.) It is observed in the papers that the stationary distribution of the state of the system is just a truncated form of the corresponding distribution in a system with infinite link capacities. However, calculating blocking probabilities in the resulting closed form expression is practically possible only for extremely small networks with very few multicast connections. The main result of the papers is therefore the development of a faster algorithm for the calculation of exact blocking probabilities. The algorithm is similar to the known convolution algorithm used for calculating blocking probabilities of ordinary point-to-point connections in hierarchial multiservice access networks. The modification required is the use of a new type of convolution, the so called OR-convolution. However, the main problem remaining is that even these algorithms are only applicable to networks with few multicast connections (due to the state space explosion).

In this paper we show that, by making certain assumptions, the exact calculation of the end-to-end blocking probability becomes feasible even for large networks with many multicast connections. The main assumptions are as follows: (i) all receivers have a uniform preference distribution when making a choice (to join) between the multicast connections, (ii) the mean holding time for any receiver to be joined to any connection is the same, and (iii) the capacity needed to carry any multicast connection on any link is the same. The key point is that, due to these assumptions, it is not necessary to keep track of the state of each individual connection separately (as in the general case) but just the *number* of active connections on each link, which yields a huge reduction in dimensionality and, therefore, in complexity of the analysis.

The structure of the paper is as follows. In section 2 the notation and the mathematical model are introduced. An efficient algorithm to calculate the time blocking probabilities is developed in section 3. The (more interesting) call blocking probabilities are considered in section 4, where three different user population models, together with the corresponding efficient algorithms to calculate the call blocking probabilities, are presented. Section 5 contains some numerical experiments. The results are summarized and some conclusions are drawn in section 6.

2. Preliminaries

Consider the tree-structured multicast network model introduced in the previous section. Let $\mathcal{J}$ denote the whole set of links j. The subset of leaf links u is denoted by $\mathcal{U}$. The server is located at the root node, which is connected with the rest of the network via a single link indexed by J. Each leaf link u connects a group of users (which is also called user population u) to the network. The set of links on the route from user population u to the server (including link J) is denoted by $\mathcal{R}_u$. The set of multicast connections i (e.g. TV channels) is denoted by $\mathcal{I} = \{1, 2, \dots, I\}$. Each connection needs one capacity unit on any link. The links may have different capacities (typically the more capacity the nearer the server the link resides), C_j denoting the capacity of link j.

For each link j, we further need the following notation. Let $\mathcal{M}_j$ denote the set of all links downstream link j (including link j). Let $\mathcal{N}_j$ denote the set of neighbouring links downstream link j (excluding link j), and $\mathcal{U}_j$ the set of leaf links downstream link j (including just link j if it is a leaf link). Note that, for any i and j, connection i is active on link j if and only if connection i is active on (at least) one of the leaf links $u \in \mathcal{U}_j$. On the other hand, when this happens, connection i is also active on any link k upstream link j (i.e. k s.t. $j \in \mathcal{M}_k$).

Throughout the paper we need the following two assumptions.

Assumption 2.1. *User populations $u \in \mathcal{U}$ behave independently of each other.*

Assumption 2.2. *Whenever there are n connections active on any leaf link $u \in \mathcal{U}$, each possible index combination $\{i_1, \dots, i_n\}$ is equally probable.*

In the rest of *this* section, we further assume that all the link capacities are infinite.

The state of connection i on link j is denoted by Y_{ji}, being 1 (0) whenever active (inactive). The detailed state of link j is denoted by $\mathbf{Y}_j$,

$$\mathbf{Y}_j = (Y_{ji}; i \in \mathcal{I}).$$

As explained in [3, 4],

$$Y_{ji} = \bigoplus_{j' \in \mathcal{N}_j} Y_{j'i} = \bigoplus_{u \in \mathcal{U}_j} Y_{ui}, \tag{2.1}$$

where $\oplus$ refers to the OR-operation. The following two corollaries follow easily from (2.1) and Assumptions 2.1 and 2.2.

Corollary 2.3. *For any links $j, j' \in \mathcal{J}$ such that $\mathcal{M}_j \cap \mathcal{M}_{j'} = \emptyset$, the detailed link states are independent of each other.*

Corollary 2.4. *Whenever there are n connections active on any link $j \in \mathcal{J}$, each possible index combination $\{i_1, \dots, i_n\}$ is equally probable.*

The detailed state $\mathbf{X}$ of the whole network depends on the detailed states of the *leaf* links,

$$\mathbf{X} = (Y_{ui}; u \in \mathcal{U}, i \in \mathcal{I}) \in \Omega,$$

where $\Omega := \{0, 1\}^{\mathcal{U} \times \mathcal{I}}$ denotes the network state space.

Let then N_j denote the number of active connections on link j,

$$N_j = \sum_{i \in \mathcal{I}} Y_{ji} \in \mathcal{S},$$

where $\mathcal{S} := \{0, 1, \dots, I\}$. In the present paper this is called the state of link j. The stationary link state probabilities are denoted by $\pi_j(n)$, $n \in \mathcal{S}$,

$$\pi_j(n) := P\{N_j = n\} = P\{\sum_{i \in \mathcal{I}} Y_{ji} = n\}. \tag{2.2}$$

Next we show how these probabilities can be calculated recursively when the leaf link state probabilites $\pi_u(n)$ are known for all $u \in \mathcal{U}_j$ and $n \in \mathcal{S}$. Note that the leaf link state probabilites $\pi_u(n)$ depend on the chosen user population model. Three different models are considered later in section 4.

1° Assume first that $\mathcal{N}_j$ consists of two links, $\mathcal{N}_j = \{s, t\}$. It is easy to see that

$$\max\{N_s, N_t\} \le N_j \le \min\{N_s + N_t, I\} \tag{2.3}$$

Let then $N_s = k$ and $N_t = l$, and assume, for a while, that $k \ge l$ (implying that $N_j \ge k$). Due to Corollaries 2.3 and 2.4, the event $\{N_j = k + m\}$ can be interpreted as follows. Consider random sampling without replacement from the population of all connections (the size of which is I) including k "marked" (i.e. active on link s) and $I - k$ "unmarked" (i.e. inactive on link s) connections. Now the event given above corresponds to such a result of random sampling that there are m "unmarked" connections in a sample of size l. Therefore, for $m \in \{0, 1, \dots, \min\{l, I - k\}\}$,

$$P\{N_j = k + m \mid N_s = k, N_t = l\} = \frac{\binom{k}{l-m}\binom{I-k}{m}}{\binom{I}{l}}.$$

A corresponding result is, of course, valid if $l \ge k$. Therefore, we have generally, for $m \in \{0, 1, \dots, \min\{\min\{k, l\}, I - \max\{k, l\}\}$,

$$P\{N_j = \max\{k, l\} + m \mid N_s = k, N_t = l\} = \frac{\binom{\max\{k,l\}}{\min\{k,l\}-m}\binom{I-\max\{k,l\}}{m}}{\binom{I}{\min\{k,l\}}}.$$

This can be written as follows. For $n \in \{\max\{k, l\}, \max\{k, l\} + 1, \dots, \min\{k + l, I\}\}$,

$$P\{N_j = n \mid N_s = k, N_t = l\} = s(n \mid k, l),$$

where $s(n \mid k, l)$ is defined as follows:

$$s(n \mid k, l) := \frac{\binom{\max\{k,l\}}{k+l-n}\binom{I-\max\{k,l\}}{n-\max\{k,l\}}}{\binom{I}{\min\{k,l\}}}. \tag{2.4}$$

On the other hand, if $N_j = n$ is fixed, then the pair (N_s, N_t) shall belong to the set

$$\{(k, l) \mid 0 \le k \le n, n - k \le l \le n\}$$

in order that inequation (2.3) be fulfilled. It follows that

$$P\{N_j = n\} = \sum_{k=0}^{n} \sum_{l=n-k}^{n} s(n \mid k, l) P\{N_s = k\} P\{N_t = l\}.$$

Thus,

$$\pi_j(n) = [\pi_s \otimes \pi_t](n), \tag{2.5}$$

where the operator $\otimes$ is defined as

$$[f \otimes g](n) := \sum_{k=0}^{n} \sum_{l=n-k}^{n} s(n \mid k,l) f(k) g(l) \tag{2.6}$$

for all real valued functions $f, g : \mathcal{S} \to \mathbb{R}$. Note that this is *not* the OR-convolution operator defined in [3, 4] but a related operator operating on a different (reduced) space. We further remark that the operator $\otimes$ is associative, i.e. $(f \otimes g) \otimes h = f \otimes (g \otimes h)$.

2° Consider then the general case with $\mathcal{N}_j$ consisting of any number of links. When calculating the state probabilities $\pi_j(n)$, these links (in $\mathcal{N}_j$) can be taken into account one-by-one in the same manner as above. Since the operator $\otimes$ is associative, we can write the final result in the following form:

$$\pi_j(n) = \begin{cases} \pi_j(n), & j \in \mathcal{U}, \\ [\bigotimes_{j' \in \mathcal{N}_j} \pi_{j'}](n), & j \in \mathcal{J} \setminus \mathcal{U}. \end{cases} \tag{2.7}$$

3. Practical algorithm for time blocking

In this section we consider the general case with any number of finite capacity links. Let $\tilde{\mathbf{X}}$ denote the (detailed) state of the network in this case. The network state space Ω is now truncated according to the capacity restrictions on each link j, resulting in

$$\tilde{\Omega} := \{\mathbf{x} \in \Omega \mid \sum_{i \in \mathcal{I}} y_{ji} \le C_j,\ j \in \mathcal{J}\},$$

which is called the truncated state space. In this section we assume that the stationary state probabilities can be calculated using the so called Truncation Principle, i.e. for all $\mathbf{x} \in \tilde{\Omega}$,

$$P\{\tilde{\mathbf{X}} = \mathbf{x}\} = \frac{P\{\mathbf{X} = \mathbf{x}\}}{P\{\mathbf{X} \in \tilde{\Omega}\}}. \tag{3.1}$$

Whether this principle applies or not, depends on the chosen user population model as discussed in the following section.

Consider now a request originating from user population u to join connection i. Our ultimate purpose is to calculate the probability that this request is rejected, i.e. the *call blocking probability* denoted by B^c_{ui}. However, this depends essentially on the chosen user population model. Therefore we postpone the discussion on this subject until the next section, where different user population models are introduced. Instead, we concentrate, in this section, on the *time blocking probability* B^t_{ui}, which, as usually, refers to the stationary probability of such network states in which a new request originating from user population u to join connection i cannot be accepted due to lack of link capacity. The complement of this set of states is denoted by $\tilde{\Omega}_{ui}$,

$$\tilde{\Omega}_{ui} := \{\mathbf{x} \in \tilde{\Omega} \mid \sum_{i' \ne i} y_{ji'} + (y_{ji} \oplus 1_{j \in \mathcal{R}_u}) \le C_j,\ j \in \mathcal{J}\},$$

Due to (3.1),

$$B^t_{ui} = 1 - P\{\tilde{\mathbf{X}} \in \tilde{\Omega}_{ui}\} = 1 - \frac{P\{\mathbf{X} \in \tilde{\Omega}_{ui}\}}{P\{\mathbf{X} \in \tilde{\Omega}\}}. \tag{3.2}$$

Next we present efficient recursive algorithms to calculate the denominator (see subsection 3.1) and the numerator (see subsection 3.2) of the above equation.

3.1. **Denominator**

Define, for $j \in \mathcal{J}$,

$$Q_j(n) := P\{N_j = n; N_{j'} \le C_{j'}, j' \in \mathcal{M}_j\}. \tag{3.3}$$

A key observation is that

$$P\{\mathbf{X} \in \tilde{\Omega}\} = \sum_{n=0}^{C_J} Q_J(n). \tag{3.4}$$

What remains, is to find an efficient method to calculate probabilities $Q_j(n)$. Such a method is presented in the following proposition. However, before that we define, for each real-valued function $f : \mathcal{S} \to \mathbb{R}$, a truncation operator T_j by

$$T_j f(n) := f(n) 1_{n \le C_j}.$$

Proposition 3.1. *Probabilities $Q_j(n)$ can be calculated recursively as follows: for $j \in \mathcal{J}$,*

$$Q_j(n) = \begin{cases} T_j \pi_j(n), & j \in \mathcal{U}, \\ T_j[\bigotimes_{j' \in \mathcal{N}_j} Q_{j'}](n), & j \in \mathcal{J} \setminus \mathcal{U}. \end{cases} \tag{3.5}$$

Proof. The derivation of (3.5) is almost identical to the derivation of (2.7) presented in the previous section. The only amendment concerns the introduction of the truncation operator T_j that quarantees the condition $N_j \le C_j$ to be valid. Recursively, all truncation operators together guarantee that $N_{j'} \le C_{j'}$ for all $j' \in \mathcal{M}_j$ as required. □

3.2. **Numerator**

Let i be the index of the connection the blocking of which we are interested in. Let $N^{(i)}$ denote the number active connections on link j excluding connection i (whether it is active or not),

$$N_j^{(i)} = \sum_{i' \ne i} Y_{ji'}.$$

In addition, define, for all $n \in \mathcal{S}$,

$$\pi_j^{(i)}(n) := P\{N_j^{(i)} = n\} = P\{\sum_{i' \ne i} Y_{ji'} = n\}. \tag{3.6}$$

Note that $\pi_j^{(i)}(I) \equiv 0$.

Assume for a while that $\mathcal{N}_j = \{s, t\}$, and let $N_s^{(i)} = k$ and $N_t^{(i)} = l$. Similarly as in the previous section (see 1° there), we can deduce that, for $n \in \{\max\{k,l\}, \max\{k,l\} + 1, \dots, \min\{k+l, I-1\}\}$,

$$P\{N_j^{(i)} = n \mid N_s^{(i)} = k, N_t^{(i)} = l\} = s^\circ(n \mid k, l), \tag{3.7}$$

where $s^\circ(n \mid k, l)$ is defined as follows:

$$s^\circ(n \mid k, l) := \frac{\begin{pmatrix} \max\{k,l\} \\ k+l-n \end{pmatrix} \begin{pmatrix} I-1-\max\{k,l\} \\ n-\max\{k,l\} \end{pmatrix}}{\begin{pmatrix} I-1 \\ \min\{k,l\} \end{pmatrix}}. \tag{3.8}$$

For future purposes (see Proposition 3.2 below) we need the following operator defined for all real valued functions $f, g : \mathcal{S} \to \mathbb{R}$. Let $[f \odot g](I) = 0$, and, for $n < I$,

$$[f \odot g](n) := \sum_{k=0}^{n} \sum_{l=n-k}^{n} s^{\circ}(n \mid k, l) f(k)[(1 - p(l))g(l) + p(l+1)g(l+1)], \tag{3.9}$$

where $p(n) := n/I$.

Now let u be the index of the user population the blocking of which we are interested in. Define, for $j \in \mathcal{R}_u$,

$$Q_j^{ui}(n) := P\{N_j^{(i)} = n; N_{j'}^{(i)} \le C_{j'} - 1, j' \in \mathcal{M}_j \cap \mathcal{R}_u; N_{j'} \le C_{j'}, j' \in \mathcal{M}_j \setminus \mathcal{R}_u\}. \tag{3.10}$$

A key observation is that

$$P\{\mathbf{X} \in \tilde{\Omega}_{ui}\} = \sum_{n=0}^{C_J - 1} Q_J^{ui}(n). \tag{3.11}$$

The following proposition shows how to calculate the probabilities $Q_j^{ui}(n)$ efficiently. In this case we need another truncation operator. For each real-valued function $f : \mathcal{S} \to \mathbb{R}$, define an operator T_j° by

$$T_j^{\circ} f(n) := f(n) 1_{n \le C_j - 1}.$$

In addition, for each $j \in \mathcal{R}_u \setminus \{u\}$, we denote by $D_u(j)$ the link downstream link j along route $\mathcal{R}_u$. Note that $\mathcal{N}_j \cap \mathcal{R}_u = \{D_u(j)\}$.

Proposition 3.2. *Probabilities $Q_j^{ui}(n)$ can be calculated recursively as follows: for $j \in \mathcal{R}_u$,*

$$Q_j^{ui}(n) = \begin{cases} T_u^{\circ} \pi_u^{(i)}(n), & j = u, \\ T_j^{\circ}[Q_{D_u(j)}^{ui} \odot \bigotimes_{j' \in \mathcal{N}_j \setminus \mathcal{R}_u} Q_{j'}](n), & j \in \mathcal{R}_u \setminus \{u\}. \end{cases} \tag{3.12}$$

Proof. For $j = u$, the result is clear by definition. Therefore, we can assume that $j \in \mathcal{R}_u \setminus \{u\}$. For brevity, we denote $D_u(j) = d$. In addition, let

$$N_{(d)} = \sum_{i' \in \mathcal{I}} \bigoplus_{j' \in \mathcal{N}_j \setminus \mathcal{R}_u} Y_{j'i'} \qquad \text{and} \qquad \mathcal{M}_{(d)} = \bigcup_{j' \in \mathcal{N}_j \setminus \mathcal{R}_u} \mathcal{M}_{j'}$$

It is easy to see that (by Corollaries 2.3 and 2.4)

$$P\{N_{(d)} = n; N_{j'} \le C_{j'}, j' \in \mathcal{M}_{(d)}\} = [\bigotimes_{j' \in \mathcal{N}_j \setminus \mathcal{R}_u} Q_{j'}](n).$$

Let then

$$N_{(d)}^{(i)} = \sum_{i' \in \mathcal{I} \setminus \{i\}} \bigoplus_{j' \in \mathcal{N}_j \setminus \mathcal{R}_u} Y_{j'i'}.$$

It is again easy to see that (by Corollaries 2.3 and 2.4)

$$\begin{aligned} & P\{N_{(d)}^{(i)} = n; N_{j'} \le C_{j'}, j' \in \mathcal{M}_{(d)}\} \\ &= (1 - p(n)) P\{N_{(d)} = n; N_{j'} \le C_{j'}, j' \in \mathcal{M}_{(d)}\} \\ &\quad + p(n+1) P\{N_{(d)} = n+1; N_{j'} \le C_{j'}, j' \in \mathcal{M}_{(d)}\} \\ &= (1 - p(n))[\bigotimes_{j' \in \mathcal{N}_j \setminus \mathcal{R}_u} Q_{j'}](n) + p(n+1)[\bigotimes_{j' \in \mathcal{N}_j \setminus \mathcal{R}_u} Q_{j'}](n+1), \end{aligned}$$

where $p(n) = n/I$ (as in (3.9)). Note further that (cf. (3.7))

$$P\{N_j^{(i)} = n \mid N_d^{(i)} = k, N_{(d)}^{(i)} = l\} = s^\circ(n \mid k, l).$$

Thus, for $n \le C_j - 1$,

$$\begin{aligned}
Q_j^{ui}(n) &= P\{N_j^{(i)} = n; N_{j'}^{(i)} \le C_{j'} - 1, j' \in \mathcal{M}_j \cap \mathcal{R}_u; N_{j'} \le C_{j'}, j' \in \mathcal{M}_j \setminus \mathcal{R}_u\} \\
&= \sum_{k=0}^{n} \sum_{l=n-k}^{n} P\{N_j^{(i)} = n \mid N_d^{(i)} = k, N_{(d)}^{(i)} = l\} \\
&\quad \times P\{N_d^{(i)} = k; N_{j'}^{(i)} \le C_{j'} - 1, j' \in \mathcal{M}_d \cap \mathcal{R}_u; N_{j'} \le C_{j'}, j' \in \mathcal{M}_d \setminus \mathcal{R}_u\} \\
&\quad \times P\{N_{(d)}^{(i)} = l; N_{j'} \le C_{j'}, j' \in \mathcal{M}_{(d)}\} \\
&= \sum_{k=0}^{n} \sum_{l=n-k}^{n} s^\circ(n \mid k, l) \\
&\quad \times Q_d^{ui}(k) \\
&\quad \times \{(1 - p(l))[\bigotimes_{j' \in \mathcal{N}_j \setminus \mathcal{R}_u} Q_{j'}](l) + p(l+1)[\bigotimes_{j' \in \mathcal{N}_j \setminus \mathcal{R}_u} Q_{j'}](l+1)\} \\
&= [Q_d^{ui} \odot \bigotimes_{j' \in \mathcal{N}_j \setminus \mathcal{R}_u} Q_{j'}](n).
\end{aligned}$$

Since $Q_j^{ui}(n) = 0$ for all $n \ge C_j$ by definition, equation (3.12) follows. □

4. Practical algorithm for call blocking

In the previous section we found an efficient method to calculate the time blocking probabilities (under Assumptions 2.1 and 2.2) in the general case with any number of finite capacity links in the multicast network. In this section we continue the considertions with regard to this general case. Our purpose is now to find an efficient method to calculate the call blocking probabilities. These probabilities depend on the chosen user population model, which has first to be specified. We will present three different user population models (which already appeared in [4]). They all guarantee Assumption 2.2 to be valid. In addition, as shown in [4], they allow the Truncation Principle (3.1) to be applied as required in the previous section. Each user population model and the corresponding call blocking algorithm is treated in its own subsection below.

4.1. **Infinite user population model**

The first user population model is as follows. Assume that new requests from user site u to join connection i arrive according to a Poisson process with intensity $\nu_u \alpha_i$, where ν_u refers to the total arrival rate (from user site u) and α_i to the probability to choose channel i. The latter probabilities together are called the *preference distribution*. The corresponding holding times are assumed to be generally distributed with a connection-specific (but not user-specific) mean $1/\mu_i$.

It is shown in [3, 4] that for this model the detailed leaf link state probabilities are as follows:

$$P\{\mathbf{Y}_u = \mathbf{y}\} = \prod_{i \in \mathcal{I}} (1 - e^{-a_{ui}})^{y_i} (e^{-a_{ui}})^{1 - y_i},$$

where

$$a_{ui} := \frac{\nu_u \alpha_i}{\mu_i}.$$

In order that Assumption 2.2 be valid, it is required that a_{ui} be constant for all i, i.e.

$$a_{ui} \equiv a_u.$$

This will be the case, for example, if

(i) the preference distribution is uniform, i.e. $\alpha_i \equiv 1/I$, and
(ii) the mean holding times for different connections are equal, i.e. $1/\mu_i \equiv 1/\mu$.

In this case we have, for $n = 0, 1, \ldots, I$,

$$\pi_u(n) := P\{N_u = n\} = \left(\begin{array}{c} I \\ n \end{array} \right) (1 - e^{-a_u})^n (e^{-a_u})^{I-n}. \tag{4.1}$$

In other words, $N_u \sim \mathrm{Bin}(I, 1 - e^{-a_u})$.

Due to Poisson arrivals, the call blocking probability B^c_{ui} is equal to the time blocking probability B^t_{ui}. Therefore, the algorithm presented in the previous section does not need to be modified in any way.

4.2. **Single user model**

The second user population model is as follows. Assume that each user site comprises of just one user, which is able to join any connection (but just one at a time). The behaviour of this user is modelled by a semi-Markov process with $I + 1$ states $\{0, 1, \ldots, I\}$. State 0 refers to idleness, and the other states to active connections. Idle periods are assumed to be generally distributed with a user-specific mean $1/\lambda_u$. After an idle period the user generates a request to join connection i with probability α_i. The corresponding holding times are assumed to be generally distributed with a connection-specific (but not user-specific) mean $1/\mu_i$. After each active period, the user starts a new idle period.

It is shown in [4] that for this model the detailed leaf link state probabilities are as follows: $P\{\mathbf{Y}_u = \mathbf{y}\} = 0$ if $\sum_{i \in \mathcal{I}} y_i > 1$, and otherwise

$$P\{\mathbf{Y}_u = \mathbf{y}\} = (p_u)^{\sum_{i \in \mathcal{I}} y_i} (1 - p_u)^{1 - \sum_{i \in \mathcal{I}} y_i} \prod_{i \in \mathcal{I}} (\beta_i)^{y_i},$$

where

$$p_u := \frac{\sum_{i' \in \mathcal{I}} \alpha_{i'}/\mu_{i'}}{1/\lambda_u + \sum_{i' \in \mathcal{I}} \alpha_{i'}/\mu_{i'}} \quad \text{and} \quad \beta_i := \frac{\alpha_i/\mu_i}{\sum_{i' \in \mathcal{I}} \alpha_{i'}/\mu_{i'}}.$$

In order that Assumption 2.2 be valid, it is required that there is μ such that

$$\frac{\alpha_i}{\mu_i} \equiv \frac{1}{I\mu}.$$

This will be the case, for example, if

(i) the preference distribution is uniform, i.e. $\alpha_i \equiv 1/I$, and
(ii) the mean holding times for different connections are equal, i.e. $1/\mu_i \equiv 1/\mu$.

In this case we have $\pi_u(n) = 0$ for $n > 1$, and, for $n = 0, 1$,

$$\pi_u(n) := P\{N_u = n\} = (p_u)^n (1 - p_u)^{1-n}. \tag{4.2}$$

In other words, $N_u \sim \mathrm{Bernoulli}(p_u)$, where $p_u = \frac{\lambda_u}{\lambda_u + \mu}$.

As stated in [4], call blocking for user u is equal to time blocking in a modified network where user u is removed. In other words, for leaf link u, the probabilities $\pi_u(n)$ should be replaced by the probabilities $\pi^\circ_u(n)$, where

$$\pi^\circ_u(n) = 1_{n=0}. \tag{4.3}$$

This results in the following recursive algorithm:

$$B_{ui}^{c} = 1 - \frac{\sum_{n=0}^{C_j-1} Q_J^{ui}(n)}{\sum_{n=0}^{C_j} Q_J(n)}, \tag{4.4}$$

where, for all $j \in \mathcal{J}$,

$$Q_j(n) = \begin{cases} T_u \pi_u^{\circ}(n), & j = u, \\ T_j \pi_j(n), & j \in \mathcal{U} \setminus \{u\}, \\ T_j[\bigotimes_{j' \in \mathcal{N}_j} Q_{j'}](n), & j \in \mathcal{J} \setminus \mathcal{U}. \end{cases} \tag{4.5}$$

and, for all $j \in \mathcal{R}_u$,

$$Q_j^{ui}(n) = \begin{cases} T_u^{\circ} \pi_u^{\circ}(n), & j = u, \\ T_j^{\circ}[Q_{D_u(j)}^{ui} \odot \bigotimes_{j' \in \mathcal{N}_j \setminus \mathcal{R}_u} Q_{j'}](n), & j \in \mathcal{R}_u \setminus \{u\}. \end{cases} \tag{4.6}$$

4.3. **Connection specific user model**

The third user population model is as follows. Assume that each user site comprises of exactly I users, each corresponding to a specific connection. The behaviour of any user (u, i) is modelled by a two-state semi-Markov process with state space $\{0, 1\}$. State 0 refers to idleness, and state 1 to activity (w.r.t. corresponding connection i). Idle periods are assumed to be generally distributed with a user-specific mean $1/\lambda_{ui}$, and active periods are assumed to be generally distributed with a connection-specific mean $1/\mu_i$.

It is shown in [4] that for this model the detailed leaf link state probabilities are as follows:

$$P\{\mathbf{Y}_u = \mathbf{y}\} = \prod_{i \in \mathcal{I}} (p_{ui})^{y_i} (1 - p_{ui})^{1 - y_i},$$

where

$$p_{ui} := \frac{\lambda_{ui}}{\lambda_{ui} + \mu_i}.$$

In order that Assumption 2.2 be valid, it is required that p_{ui} be constant for all i, i.e.

$$p_{ui} \equiv p_u.$$

This will be the case, for example, if

(i) the mean idle times for each (connection specific) user at the same user site are equal, i.e. $1/\lambda_{ui} \equiv 1/\lambda_u$, and
(ii) the mean holding times for different connections are equal, i.e. $1/\mu_i \equiv 1/\mu$.

In this case we have, for $n = 0, 1, \ldots, I$,

$$\pi_u(n) := P\{N_u = n\} = \binom{I}{n} (p_u)^n (1 - p_u)^{I-n}. \tag{4.7}$$

In other words, $N_u \sim \mathrm{Bin}(I, p_u)$.

As stated in [4], call blocking for user (u, i) is equal to time blocking in a modified network where user (u, i) is removed. In other words, for leaf link u, the probabilities $\pi_u(n)$ should be replaced by the probabilities $\pi_u^\circ(n)$, where

$$\pi_u^\circ(n) = \begin{pmatrix} I-1 \\ n \end{pmatrix} (p_u)^n (1-p_u)^{I-1-n}. \tag{4.8}$$

Formally, the algorithm for this model is equal to the algorithm presented in the previous subsection, see formulas (4.4)–(4.6).

5. Numerical results

To get some idea about the efficiency of the new algorithm, we made numerical experiments with two example networks. The first one was already introduced in [3] and the second one in [4]. We implemented both the original algorithm (called here `exact`) presented in [3, 4] and the new algorithm (called here `pract`) presented in this paper with Mathematica (version 3) using similar programming style. The computations were run on a PC equipped with a Pentium Celeron 330 MHz processor. The purpose was to study the time consumed by these algorithms (to compute the blocking probability) as a function of the total number I of multicast connections.

5.1. **Example 1**

First we considered the network depicted in figure 2 of [3]. It was, however, assumed that (i) all receivers have a uniform preference distribution when making a choice (to join) between the multicast connections, (ii) the mean holding time for any receiver to be joined to any connection is the same, and (iii) the capacity needed to carry any multicast connection on any link is the same. Under these assumptions, both algorithms produce, of course, the same results as regards the blocking probabilities themselves. What differs is the time needed to compute these values.

Below in figures 5.1 – 5.3 we have plotted the processing time for both algorithms as a function of I for $I = 2, 3, \ldots, 8$. In figure 5.1 the scaling is normal, whereas in figures 5.2 and 5.3 we have used logarithmic and log-log scales, respectively.

The three figures demonstrate clearly the effectiveness of the new algorithm `pract` (as compared to the original one `exact`). Since the curve corresponding to the original algorithm `exact` is almost linear in the logaritmic scale (see figure 5.2), the processing time for that algorithm grows exponentially as a function of I. As regards the curve corresponding to the new algorithm `pract`, it seems to be strictly concave in the logarithmic scale (see figure 5.2) and almost linear in the log-log scale (see figure 5.3). Consequently, the processing time for the new algorithm seems to grow according to some power law. As estimated from figure 5.3, the processing time is approximately proportional to the square of I.

With the new algorithm (`exact`) we were able to consider higher values of I. Below in figures 5.4 – 5.6 we have plotted the processing time for the new algorithm as a function of I for $I = 2, 3, \ldots, 50$. The three figures correspond to the same three scalings as before.

With the exteneded range, the curve corresponding the new algorithm is still strictly concave in the logarithmic scale (see figure 5.5) but maybe only asymptotic linear in the log-log scale (see figure 5.6). As estimated from figure 5.6, the processing time of the new algorithm seems to be proportional to the cube (rather than the square) of I. Of course, the range of I should still be extended to justify the last statement. This is left for future work.

5.2. **Example 2**

Then we considered the network depicted in figure 5 of [4]. As above in Example 1, we considered the case with the conditions (i) – (iii) satisfied.

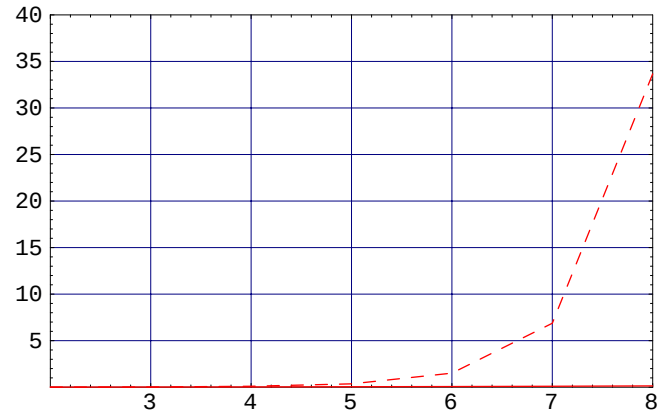

FIGURE 5.1. Processing time T (in seconds) vs. I for $I = 2, 3, \ldots, 8$ in example 1. Normal scaling: x-axis $= I$, y-axis $= T$. Curves: dotted = `exact`, continuous = `pract`.

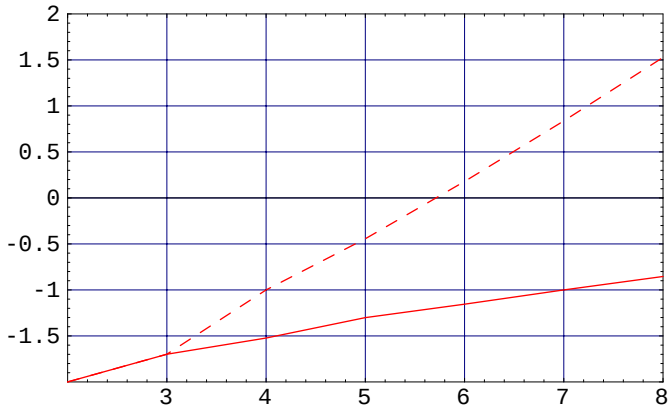

FIGURE 5.2. Processing time T (in seconds) vs. I for $I = 2, 3, \ldots, 8$ in example 1. Logarithmic scaling: x-axis $= I$, y-axis $= \log T$. Curves: dotted = `exact`, continuous = `pract`.

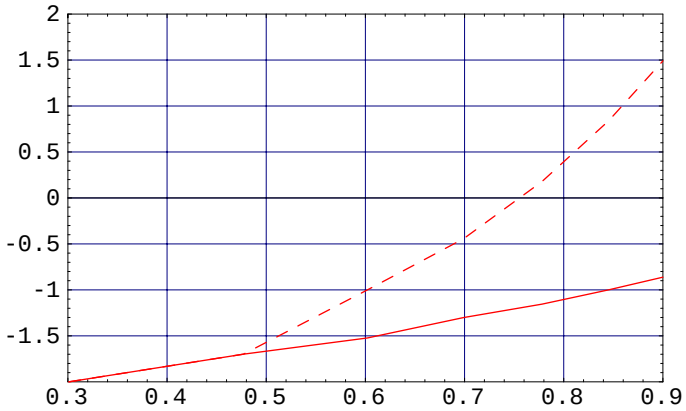

FIGURE 5.3. Processing time T (in seconds) vs. I for $I = 2, 3, \ldots, 8$ in example 1. Log-log scaling: x-axis $= \log I$, y-axis $= \log T$. Curves: dotted = `exact`, continuous = `pract`.

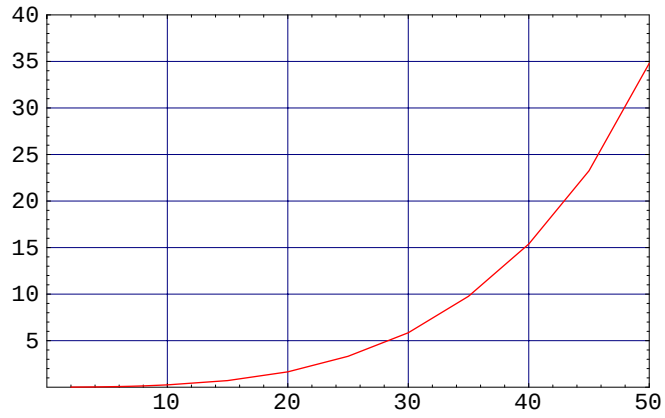

FIGURE 5.4. Processing time T (in seconds) vs. I for $I = 2, 3, \ldots, 50$ in example 1. Normal scaling: x-axis $= I$, y-axis $= T$. Curve: continuous = `pract`.

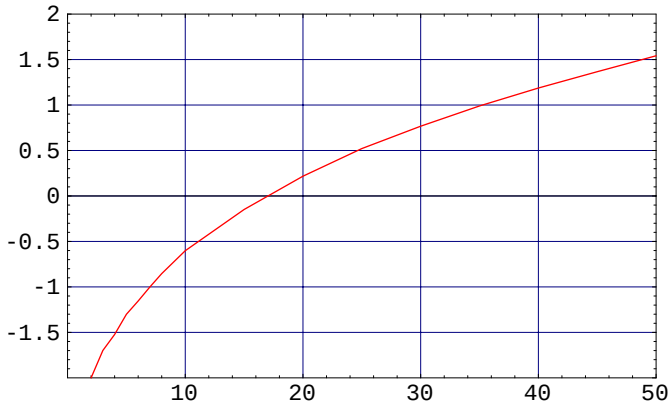

FIGURE 5.5. Processing time T (in seconds) vs. I for $I = 2, 3, \ldots, 50$ in example 1. Logarithmic scaling: x-axis $= I$, y-axis $= \log T$. Curve: continuous = `pract`.

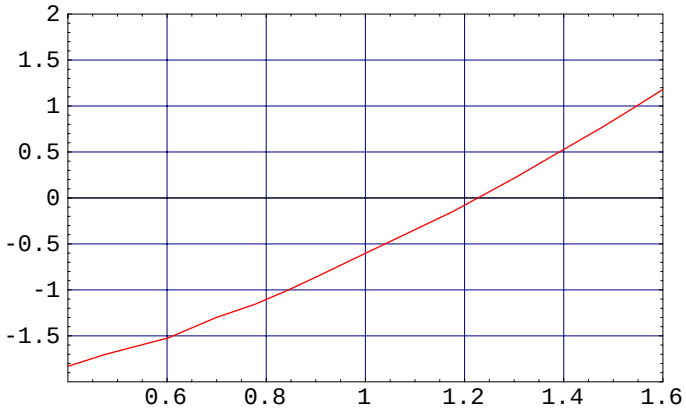

FIGURE 5.6. Processing time T (in seconds) vs. I for $I = 2, 3, \ldots, 50$ in example 1. Log-log scaling: x-axis $= \log I$, y-axis $= \log T$. Curve: continuous = `pract`.

Below in figures 5.7 – 5.9 we have plotted the processing time for both algorithms as a function of I for $I = 2, 3, \ldots, 8$. As before, the three figures correspond to the three different scalings. Furthermore, in figures 5.10 – 5.12 we have plotted the processing time for the new algorithm as a function of I for $I = 2, 3, \ldots, 50$. The three figures correspond again to the three different scalings.

The conclusions in this second example do not differ from those of the first one. The processing time for the original algorithm `exact` seems to grow exponentially as a function of I (corresponding to the linear growth in the logaritmic scale as demonstrated in figure 5.8), whereas the processing time of the new algorithm `pract` seems to be proportional to the cube of I (as estimated from figure 5.12). However, as mentioned already above, the range of I should still be extended to justify the last statement.

6. Summary

In this paper we considered a multicast network model, where a single server node is connected with a collection of user sites by a transport network with tree topology. A predefined set of dynamic multicast connections use this transport network, with the server acting as the source for all these connections and the receivers located at the user sites. The purpose was to develop a practical algorithm for calculating blocking probabilities in this setting. The original algorithm presented in earlier papers [3, 4] had been found to be applicable, due to the state space explosion, only for networks with few multicast connections. Essentially, under the following three assumptions

(i) all receivers have a uniform preference distribution when making a choice (to join) between the multicast connections,
(ii) the mean holding time for any receiver to be joined to any connection is the same, and
(ii) the capacity needed to carry any multicast connection on any link is the same,

we were able to develop such an algorithm. The key point was that, due to these assumptions, it is not necessary to keep track of the state of each individual connection separately (as in the general case) but just the number of active connections on each link, which yielded a huge reduction in dimensionality. The efficiency of the new algorithm compared to the original one was demonstrated by numerical examples. While the processing time of the original algorithm proved out to be exponentially dependent on the total number of multicast connections, the new algorithm seemed to follow a power law.

References

[1] J. Karvo, J. Virtamo, S. Aalto, and O. Martikainen, Blocking of dynamic multicast connections in a single link, in Proc. of Fourth International Conference on Broadband Communications (BC'98), *Broadband Communications, The future of telecommunications*, P.J. Kuhn and R. Ulrich (eds.), Chapman & Hall, London, 1998, pp. 473-483.
[2] J. Karvo, J. Virtamo, S. Aalto, and O. Martikainen, Blocking of dynamic multicast connections, to appear in a special issue of *Telecommunication Systems* (Select Proceedings of the Fourth INFORMS Telecommunications Conference).
[3] E. Nyberg, J. Virtamo and S. Aalto, An exact algorithm for calculating blocking probabilities in multicast networks, to appear in Proc. of Networking 2000, Paris, France, 14-19 May 2000.
[4] E. Nyberg, J. Virtamo and S. Aalto, An exact end-to-end blocking probability algorithm for multicast networks, submitted.

(Samuli Aalto) Helsinki University of Technology, Laboratory of Telecommunications Technology, P.O. Box 3000, FIN-02015 HUT, Finland
E-mail address: `samuli.aalto@hut.fi`

(Jorma Virtamo) Helsinki University of Technology, Laboratory of Telecommunications Technology, P.O. Box 3000, FIN-02015 HUT, Finland
E-mail address: `jorma.virtamo@hut.fi`

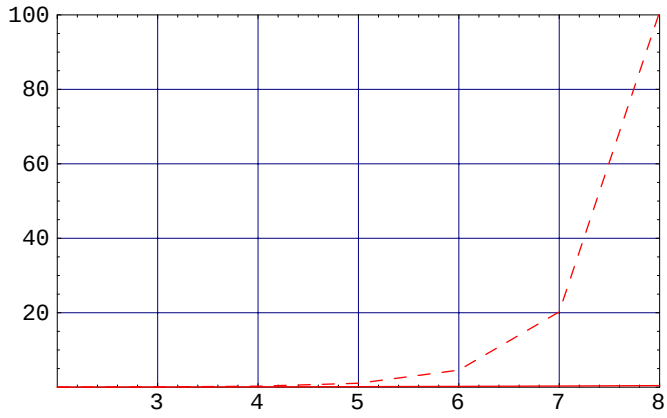

FIGURE 5.7. Processing time T (in seconds) vs. I for $I = 2, 3, \ldots, 8$ in example 2. Normal scaling: x-axis $= I$, y-axis $= T$. Curves: dotted = `exact`, continuous = `pract`.

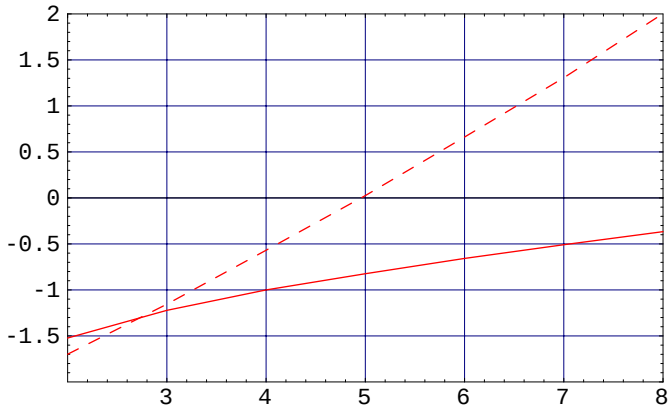

FIGURE 5.8. Processing time T (in seconds) vs. I for $I = 2, 3, \ldots, 8$ in example 2. Logarithmic scaling: x-axis $= I$, y-axis $= \log T$. Curves: dotted = `exact`, continuous = `pract`.

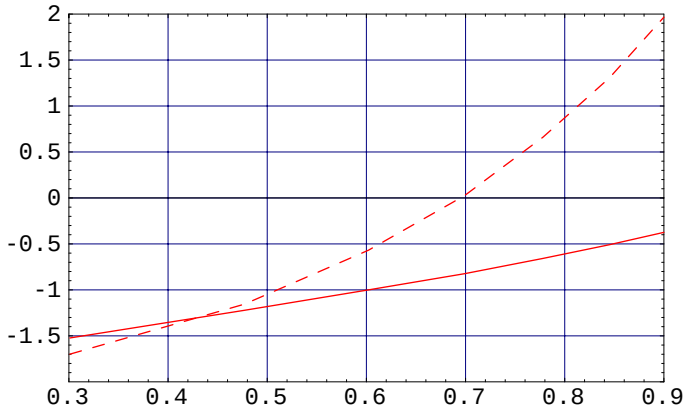

FIGURE 5.9. Processing time T (in seconds) vs. I for $I = 2, 3, \ldots, 8$ in example 2. Log-log scaling: x-axis $= \log I$, y-axis $= \log T$. Curves: dotted = `exact`, continuous = `pract`.

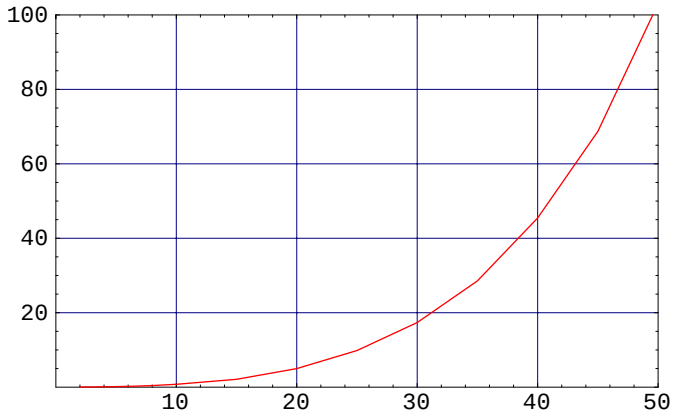

FIGURE 5.10. Processing time T (in seconds) vs. I for $I = 2, 3, \ldots, 50$ in example 2. Normal scaling: x-axis $= I$, y-axis $= T$. Curve: continuous = `pract`.

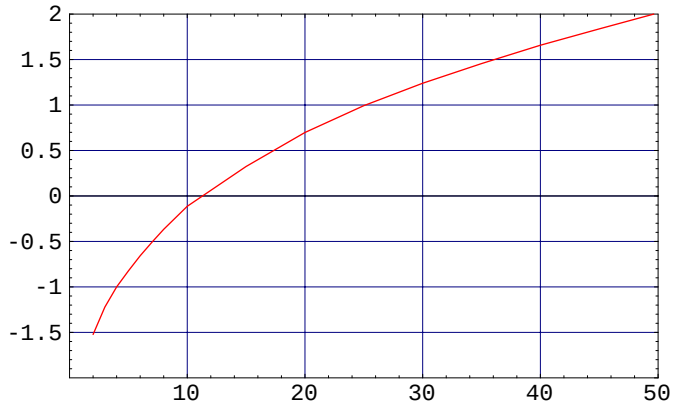

FIGURE 5.11. Processing time T (in seconds) vs. I for $I = 2, 3, \ldots, 50$ in example 2. Logarithmic scaling: x-axis $= I$, y-axis $= \log T$. Curve: continuous = `pract`.

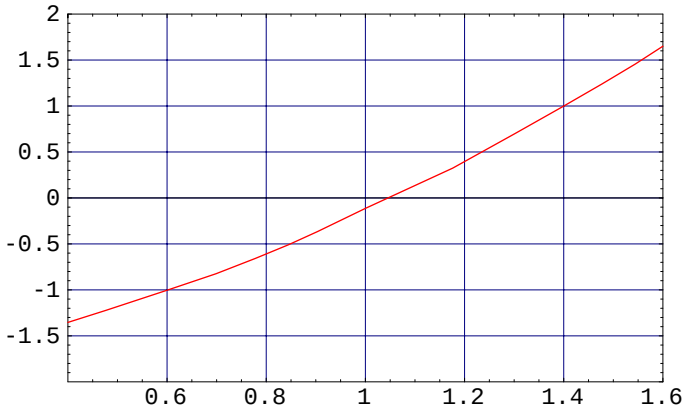

FIGURE 5.12. Processing time T (in seconds) vs. I for $I = 2, 3, \ldots, 50$ in example 2. Log-log scaling: x-axis $= \log I$, y-axis $= \log T$. Curve: continuous = `pract`.