Practical Algorithm for Calculating Blocking Probabilities in Multicast Networks

Samuli Aalto & Jorma Virtamo
Laboratory of Telecommunications Technology
Helsinki University of Technology

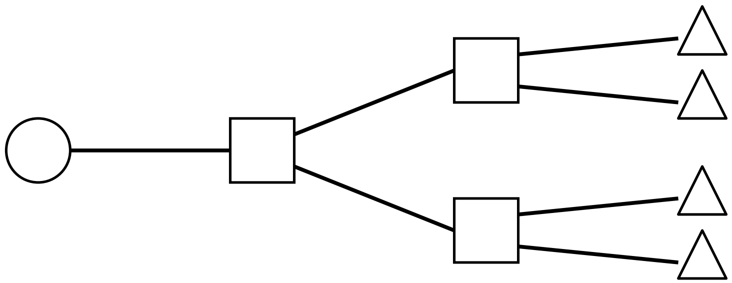
samuli.aalto@hut.fi

Contents

- Introduction: dynamic multicast network model
- Preliminaries: a network with infinite link capacities
- Calculating blocking probabilities
- Practical algorithm
- Numerical results

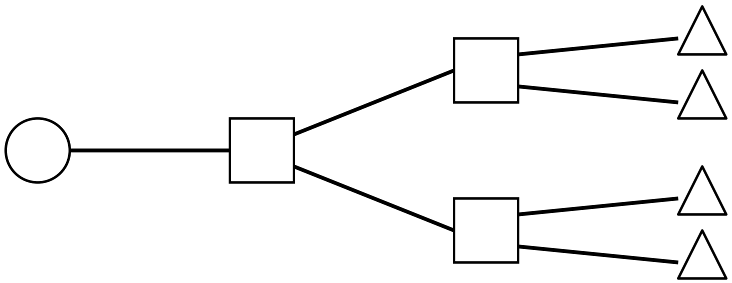
Dynamic multicast network model (1)

- Setup (as in [TD(97)46,TD(99)26]):
  - Unique **service center** offers a variety of **channels**
  - Each channel is delivered by a **dynamic multicast connection**
  - Each multicast connection uses the same **multicast tree**
    $\Rightarrow$ **fixed routing** of these multicast connections
  - Service center located at the **root node** of the multicast tree
  - Users located at the **leaf nodes** of the multicast tree
- Possible application:
  - TV or radio delivery via a telecommunication network



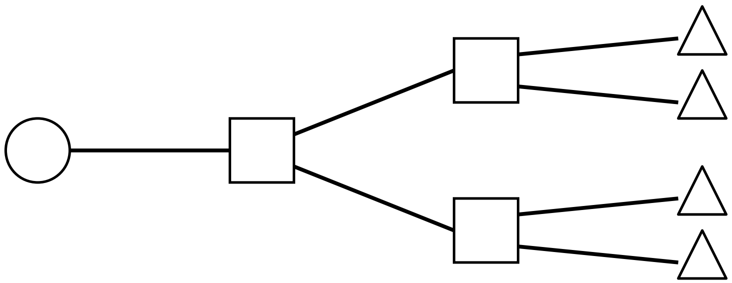
Dynamic multicast network model (2)

- Notation:
  - J = set of links (indexed by j)
  - C_j = capacity of link j
  - $U \subset J$ = set of leaf links = set of user populations (indexed by u)
  - I = set of channels (indexed by i)
  - d_i = capacity requirement of channel i



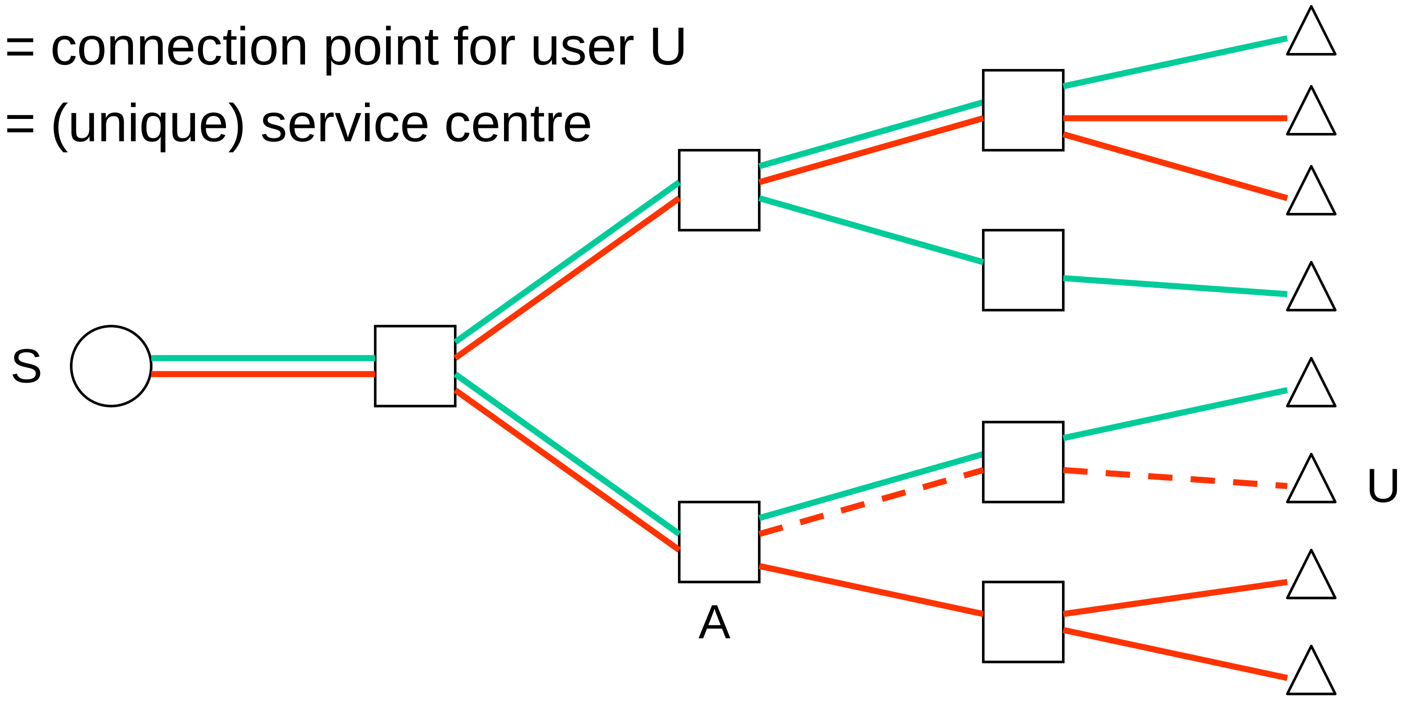
Dynamic multicast network model (3)

- Assumptions concerning user populations:
  - behave independently
  - may consist of
    - an infinite user population (Poisson arrivals of connection requests) as in [TD(97)46,TD(99)26], or
    - a single user (semi-Markov model of a user)



Dynamic multicast network model (4)

- U = new user of channel ‘red’
- A = connection point for user U
- S = (unique) service centre



Contents

- Introduction: dynamic multicast network model
- Preliminaries: a network with infinite link capacities
- Calculating blocking probabilities
- Practical algorithm
- Numerical results

Preliminaries (1)

- Consider first a network with infinite link capacities
- Let

$$Y_{ji} = 1\{\text{connection } i \text{ active on link } j\}$$

- Link state (for any link *j*)

$$N_j = \sum_{i \in I} Y_{ji} \in \{0,1,\ldots,I\}$$

- Detailed link state (for any link *j*)

$$\mathbf{Y}_j = (Y_{ji}; i \in I) \in S := \{0,1\}^I$$

- Detailed network state

$$\mathbf{X} = (Y_{ui}; u \in U, i \in I) \in \Omega := \{0,1\}^{U \times I}$$

Preliminaries (2)

- Assume that the probabilities of the detailed leaf link states (which depend on the user population model adopted) are known, and denote them by

$$\pi_u(\mathbf{y}) := P\{\mathbf{Y}_u = \mathbf{y}\}$$

  - where $\mathbf{y} \in S$
- Due to infinite link capacities and independent behaviour of the user populations, it follows that the probabilities of the detailed network states are also known:

$$\pi(\mathbf{x}) := P\{\mathbf{X} = \mathbf{x}\} = \prod_{u \in U} P\{\mathbf{Y}_u = \mathbf{y}_u\} = \prod_{u \in U} \pi_u(\mathbf{y}_u)$$

  - where $\mathbf{x} = (\mathbf{y}_u;\, u \in U) \in \Omega$

Contents

- Introduction: dynamic multicast network model
- Preliminaries: a network with infinite link capacities
- Calculating blocking probabilities
- Practical algorithm
- Numerical results

Calculating blocking probabilities (1)

- B^t_{ui} = time blocking for user population u and connection i
  = stationary probability of such network states in which
  a new request originating from user population u to join
  connection i would be rejected due to lack of link capacity

- How to calculate B^t_{ui}?

Calculating blocking probabilities (2)

- 1st possibility: closed form expression [TD(99)26]

$$B_{ui}^{t} = 1 - \frac{\sum_{\mathbf{x} \in \tilde{\Omega}_{ui}} \pi(\mathbf{x})}{\sum_{\mathbf{x} \in \tilde{\Omega}} \pi(\mathbf{x})}$$

  – where

$$\tilde{\Omega}_{ui} = \text{set of nonblocking states for class } (u,i)$$

$$\tilde{\Omega} = \text{set of allowed states}$$

- Problem: computationally extremely complex
  – exponential growth both in U and I

Calculating blocking probabilities (3)

- 2nd possibility: recursive algorithm **exact** [TD(99)26]

$$B_{ui}^{t} = 1 - \frac{\sum_{\mathbf{y} \in S} Q_J^{ui}(\mathbf{y})}{\sum_{\mathbf{y} \in S} Q_J(\mathbf{y})}$$

  – where probabilities $Q_j^{ui}(\mathbf{y})$ and $Q_j(\mathbf{y})$ can be calculated recursively (from the common link J back to leaf links u)

- Problem: computationally complex
  – linear growth in U but (still) exponential growth in I

Calculating blocking probabilities (4)

- 3rd possibility: recursive algorithm **pract** [the new one]

$$B_{ui}^{t} = 1 - \frac{\sum_{n=0}^{C_j - 1} Q_J^{ui}(n)}{\sum_{n=0}^{C_j} Q_J(n)}$$

– where probabilities $Q_j^{ui}(n)$ and $Q_j(n)$ can be calculated recursively (from the common link J back to leaf links u)

- Problem: computationally resonable but …
… restrictive assumptions have to be made!

Contents

- Introduction: dynamic multicast network model
- Preliminaries: a network with infinite link capacities
- Calculating blocking probabilities
- Practical algorithm
- Numerical results

Restrictive 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
- (iii) The capacity needed to carry any multicast connection in any link is the same
  - Make connections symmetrical!
  - Users and network may still be “unsymmetrical”

Corollaries for infinite capacity link case

- (i) and (ii) $\Rightarrow$
  - Whenever there are n connections active on any leaf link $u \in U$, each possible index combination $\{i_1,\ldots,i_n\}$ is equally probable

- This and the independence of the user populations $\Rightarrow$
  - Whenever there are n connections active on any link $j \in J$, each possible index combination $\{i_1,\ldots,i_n\}$ is equally probable

Algorithm (1)

- Define (for all $j \in J$):

$$Q_j(n) = P\{N_j = n; N_k \leq C_k, \forall k \in M_j\}$$

$$Q_j^{ui}(n) = P\{N_j^{(i)} = n; N_k^{(i)} \leq C_k - 1, \forall k \in M_j \cap R_u;$$
$$N_k \leq C_k, \qquad \forall k \in M_j \setminus R_u\}$$

- Then time blocking probability for class (u,i) is

$$B_{ui}^t = 1 - \frac{P\{\mathbf{X} \in \tilde{\Omega}_{ui}\}}{P\{\mathbf{X} \in \tilde{\Omega}\}} = 1 - \frac{\sum_{n=0}^{C_J - 1} Q_J^{ui}(n)}{\sum_{n=0}^{C_J} Q_J(n)}$$

Algorithm (2)

- Recursion 1 to calculate $Q_j(n)$ [assumptions needed here]:

$$Q_j(n) = \begin{cases} T_j[\pi_j](n), & j \in U \\ T_j[\bigotimes_{k \in N_j} Q_k](n), & j \notin U \end{cases}$$

  - where $\pi_j(n) = P\{N_j = n\}$ depend on the chosen user population model
- Truncation operator 1:
  - Let f be any real-valued function defined on $\{0,1,\ldots,I\}$
  - Then define

$$T_j[f](n) = f(n) \cdot 1\{n \le C_j\}$$

Algorithm (3)

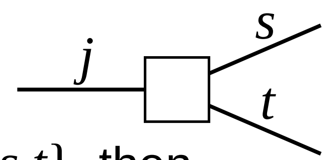
- Definition of operator $\otimes$:
  - Let f and g be any real-valued function defined on $\{0,1,\ldots,I\}$
  - Then define

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

    - where

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

Algorithm (4)



- Key result:
  - If link j has two downstream neighbouring links, $N_j = \{s,t\}$, then

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

  - In other words,

$$\pi_j(n) = [\pi_s \otimes \pi_t](n)$$

    - where $\pi_j(n) = P\{N_j = n\}$
  - Proved by a “sampling without replacement” argument!

Algorithm (5)

- Recursion 2 to calculate $Q_J^{ui}(n)$ [assumptions needed here]:

$$Q_j^{ui}(n) = \begin{cases} T_u^{\circ}[\pi_u^{(i)}](n), & j = u \\ T_j^{\circ}[Q_{D_u(j)}^{ui} \odot \bigotimes_{k \in N_j \setminus R_u} Q_k](n), & j \in R_u \setminus \{u\} \end{cases}$$

  - where $\pi_j^{(i)}(n) = P\{N_j^{(i)} = n\}$ depend on the chosen user population model

- Truncation operator 2:
  - Let f be any real-valued defined on $\{0,1,\ldots,I\}$
  - Then define

$$T_j^{\circ}[f](n) = f(n) \cdot 1\{n \le C_j - 1\}$$

Algorithm (6)

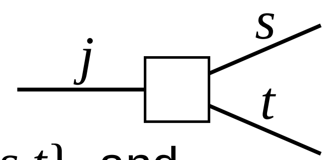
- Definition of operator ⊙:
  - Let f and g be any real-valued function defined on $\{0,1,\ldots,I\}$
  - Then define

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

    - where $p(n) = n/I$ and

$$s^{\circ}(n \mid k,l) = \frac{\binom{\max\{k,l\}}{k+l-n}\binom{I-1-\max\{k,l\}}{n-\max\{k,l\}}}{\binom{I-1}{\min\{k,l\}}}$$

Algorithm (7)



- Key result:
  - If link j has two downstream neighbouring links, $N_j = \{s,t\}$, and link s belongs to the interesting route, i.e. $s = D_u(j)$, then

$$P\{N_j^{(i)} = n\} = \sum_{k=0}^{n} \sum_{l=n-k}^{n} s^{\circ}(n \mid k,l) P\{N_s^{(i)} = k\}$$
$$\times [(1-p(l))P\{N_t = l\} + p(l+1)P\{N_t = l+1\}]$$

  - In other words,

$$\pi_j^{(i)}(n) = [\pi_s^{(i)} \odot \pi_t](n)$$

where $\pi_j^{(i)}(n) = P\{N_j^{(i)} = n\}$

  - Proved by a “sampling without replacement” argument!

Calculating call blocking probabilities

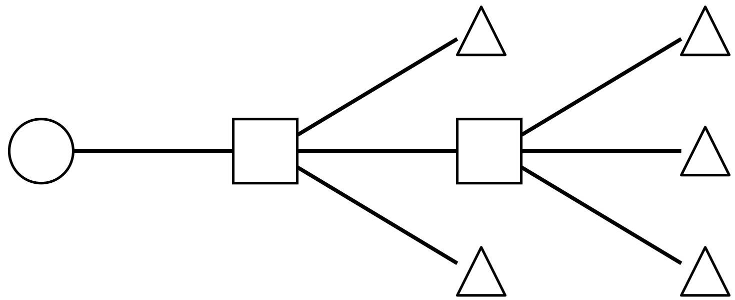
- In the paper, a similar algorithm is derived for calculating call blocking probabilities

- Dependence on the user population model has to be taken carefully into account
  - Infinite user population model:
    - call blocking $B^c{}_{ui}$ equals time blocking $B^t{}_{ui}$
  - Single user model:
    - call blocking $B^c{}_{ui}$ equals time blocking in a modified network where user u is removed

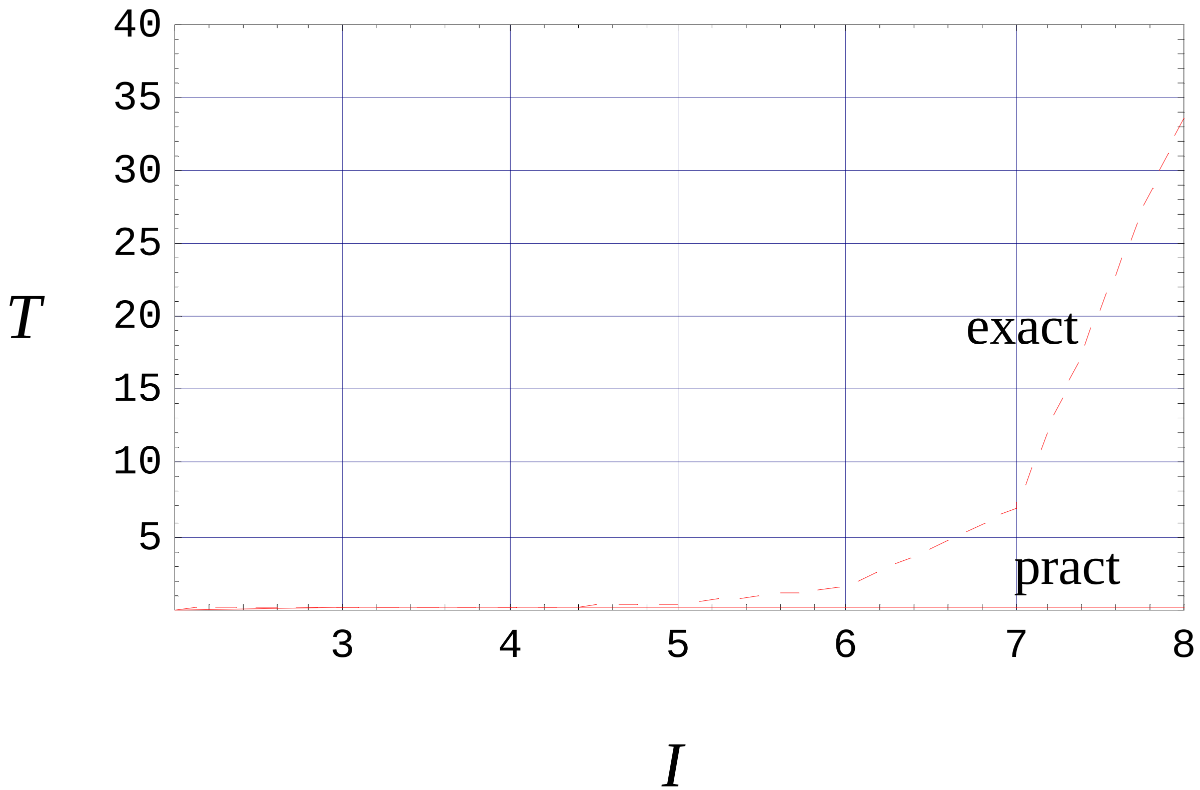
Contents

- Introduction: dynamic multicast network model
- Preliminaries: a network with infinite link capacities
- Calculating blocking probabilities
- Practical algorithm
- Numerical results

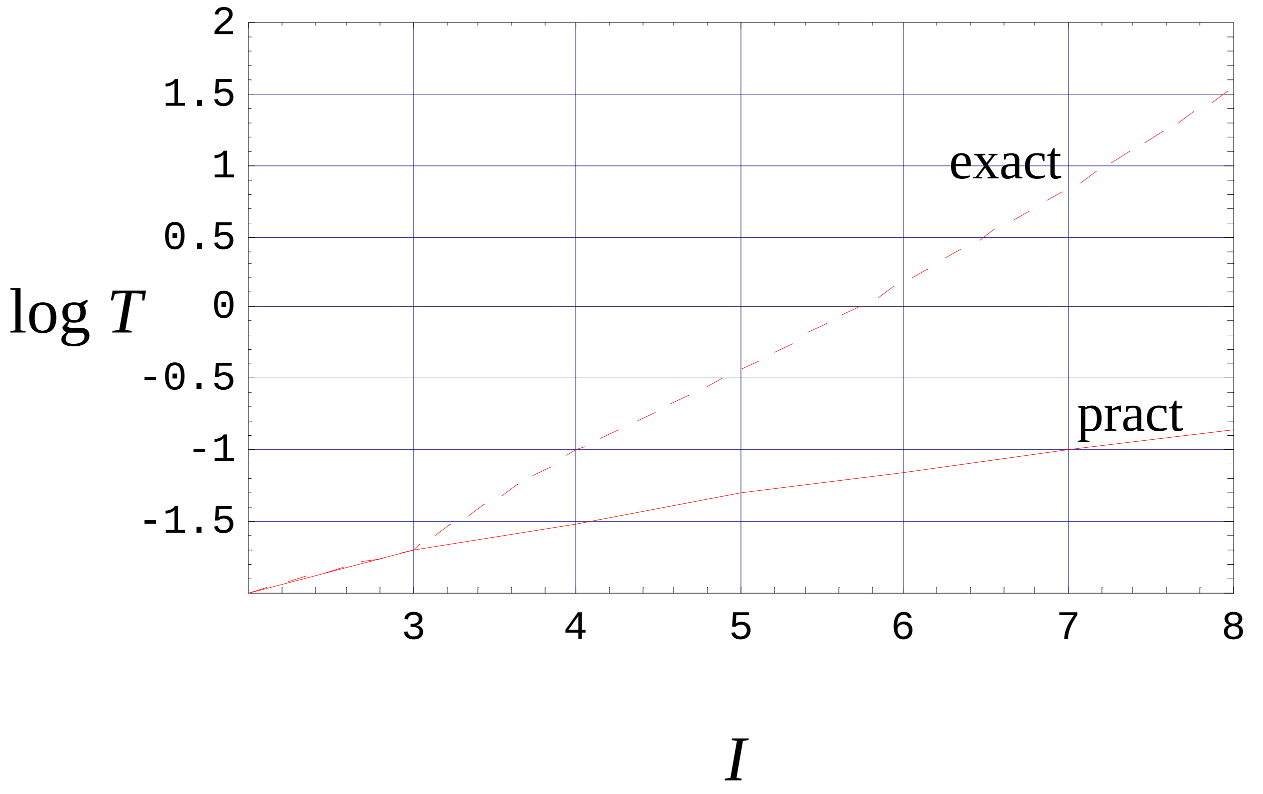
Example network 1 (figure 2 in [3])



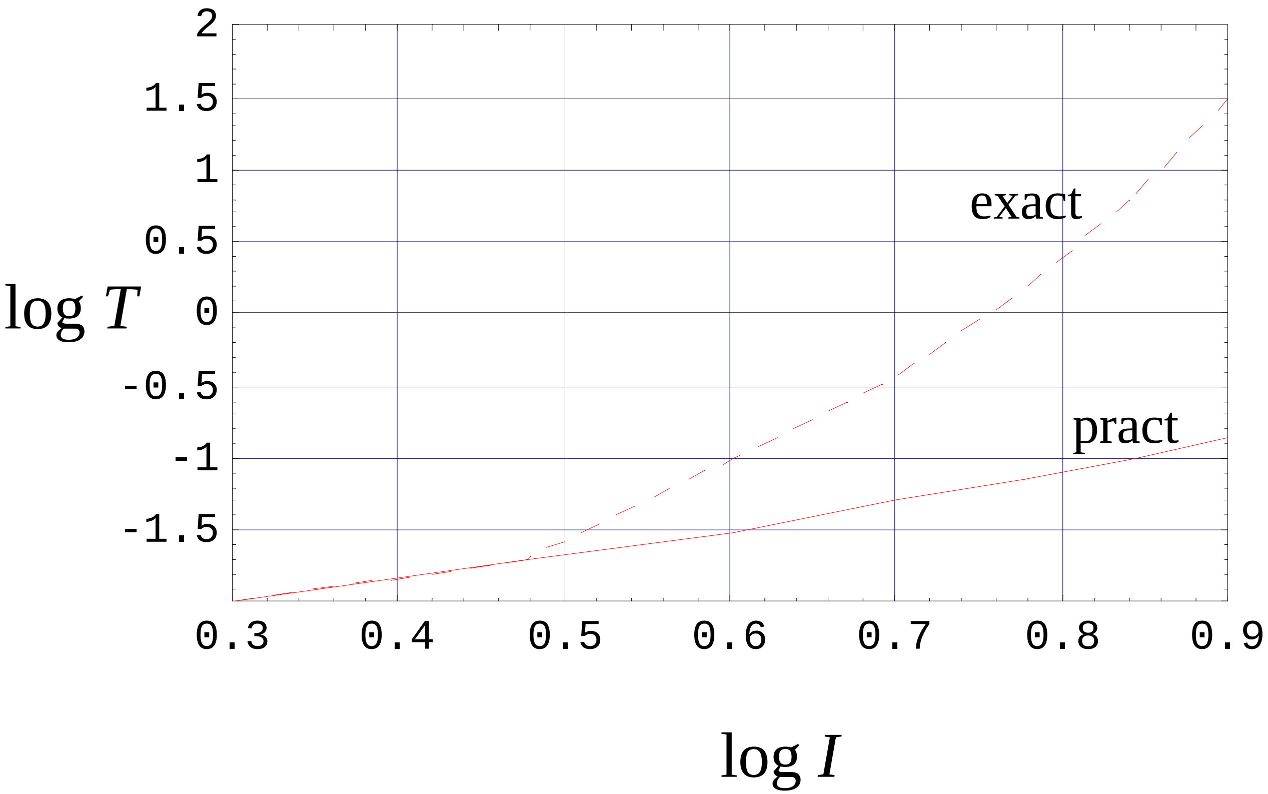
Processing time T vs. nr of multicast connections I (normal scale)



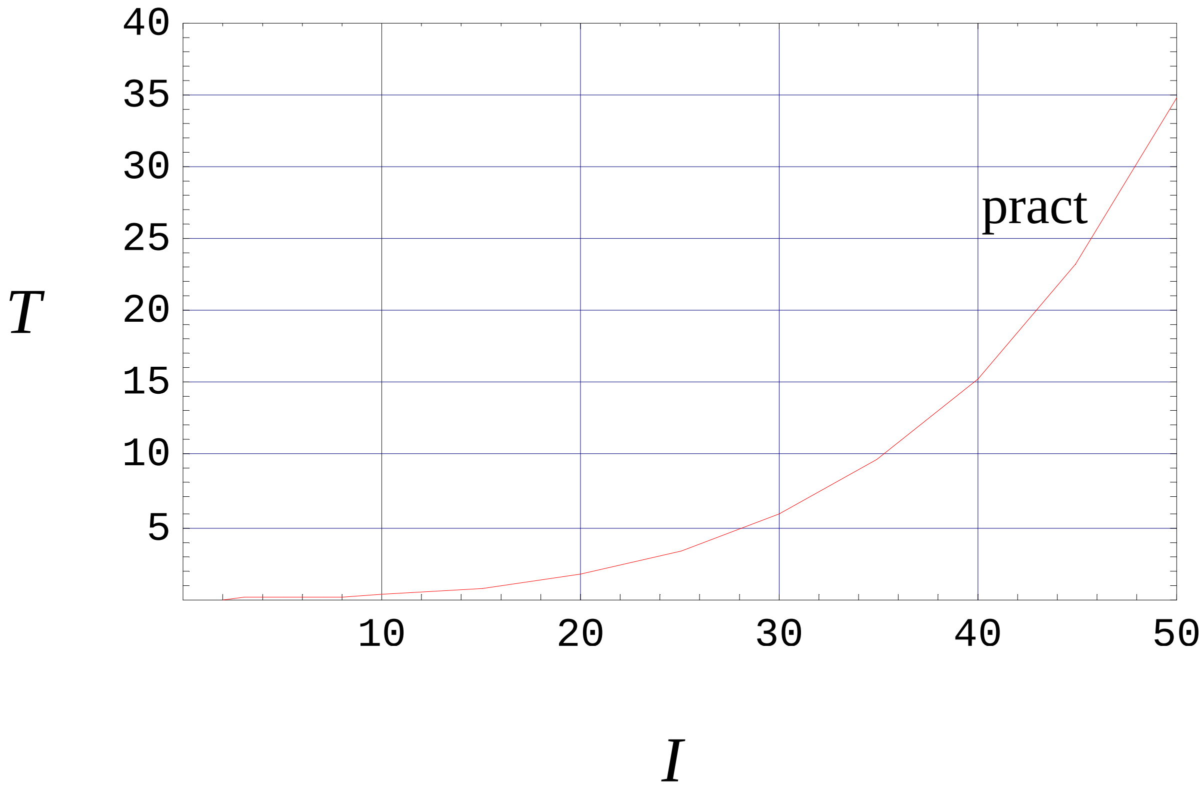
Processing time T vs. nr of multicast connections I (logarithmic scale)



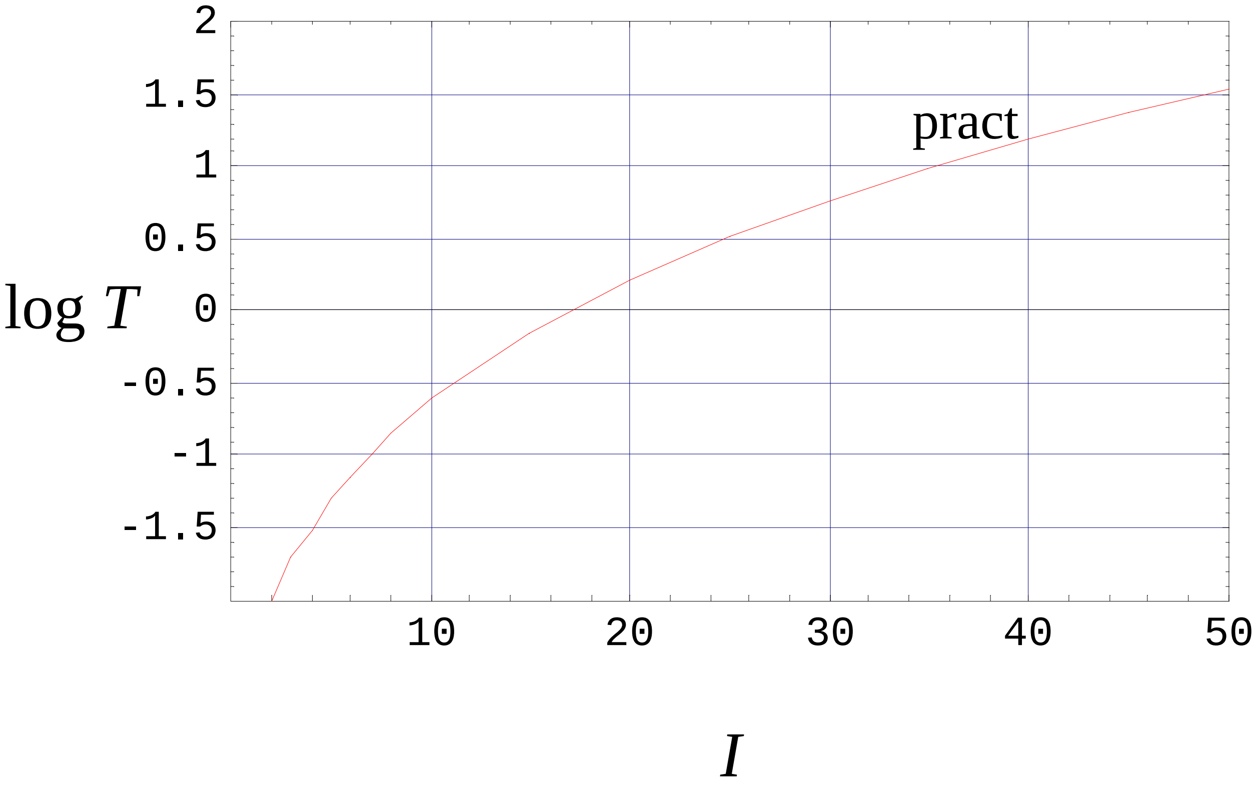
Processing time T vs. nr of multicast connections I (log-log scale)



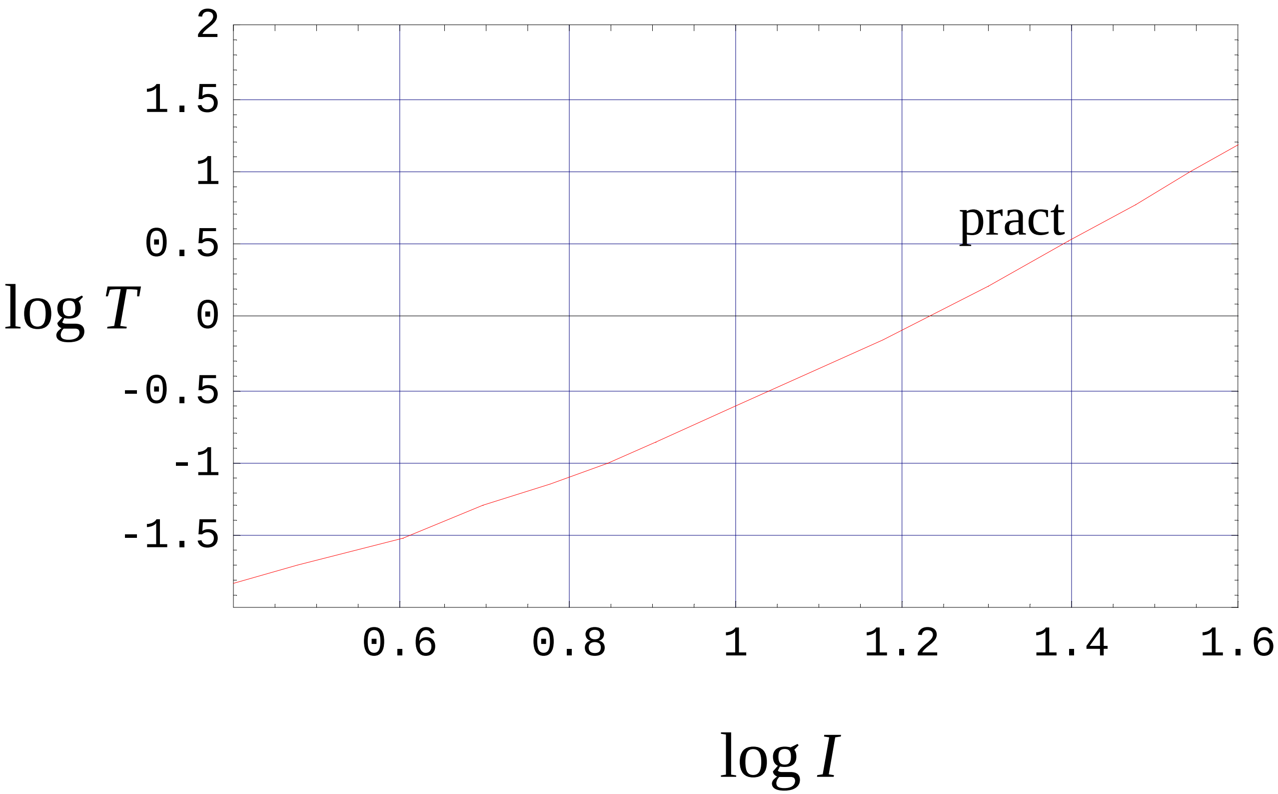
Processing time T vs. nr of multicast connections I (normal scale)



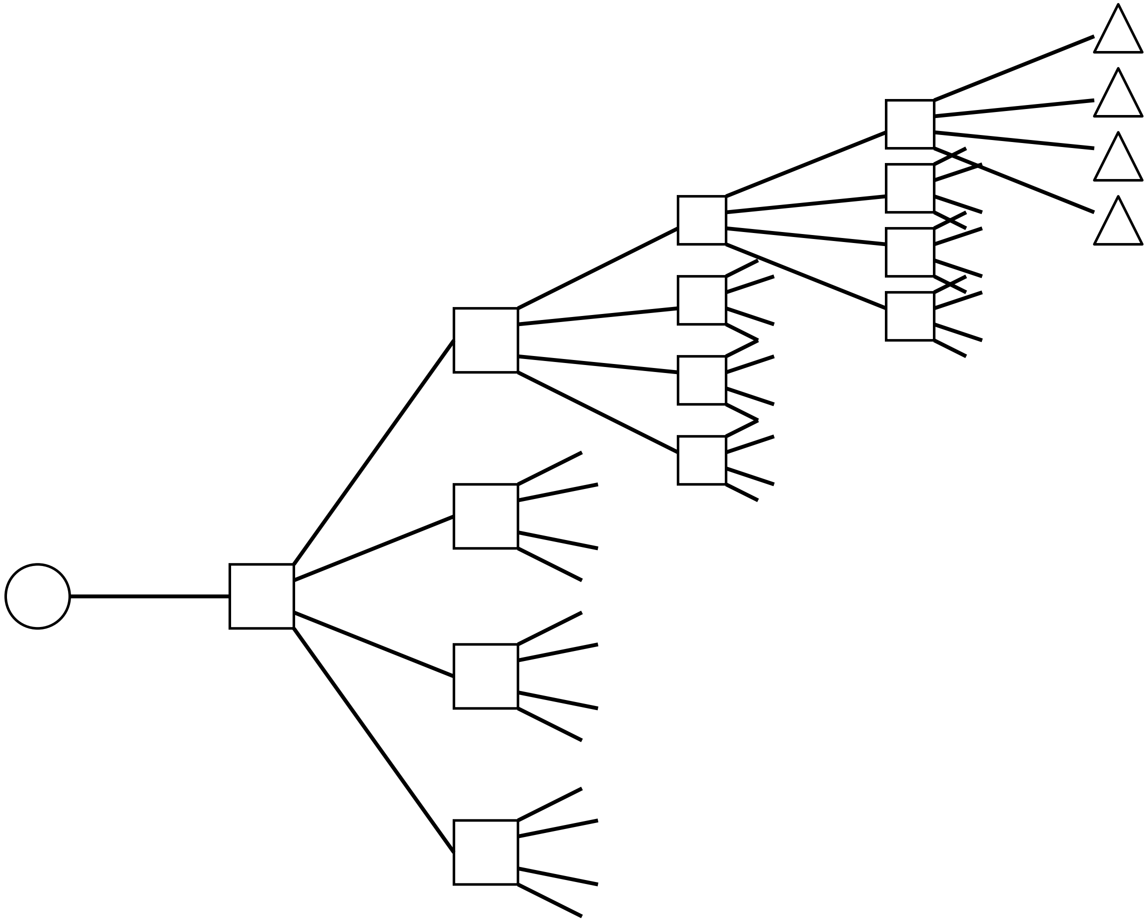
Processing time T vs. nr of multicast connections I (logarithmic scale)



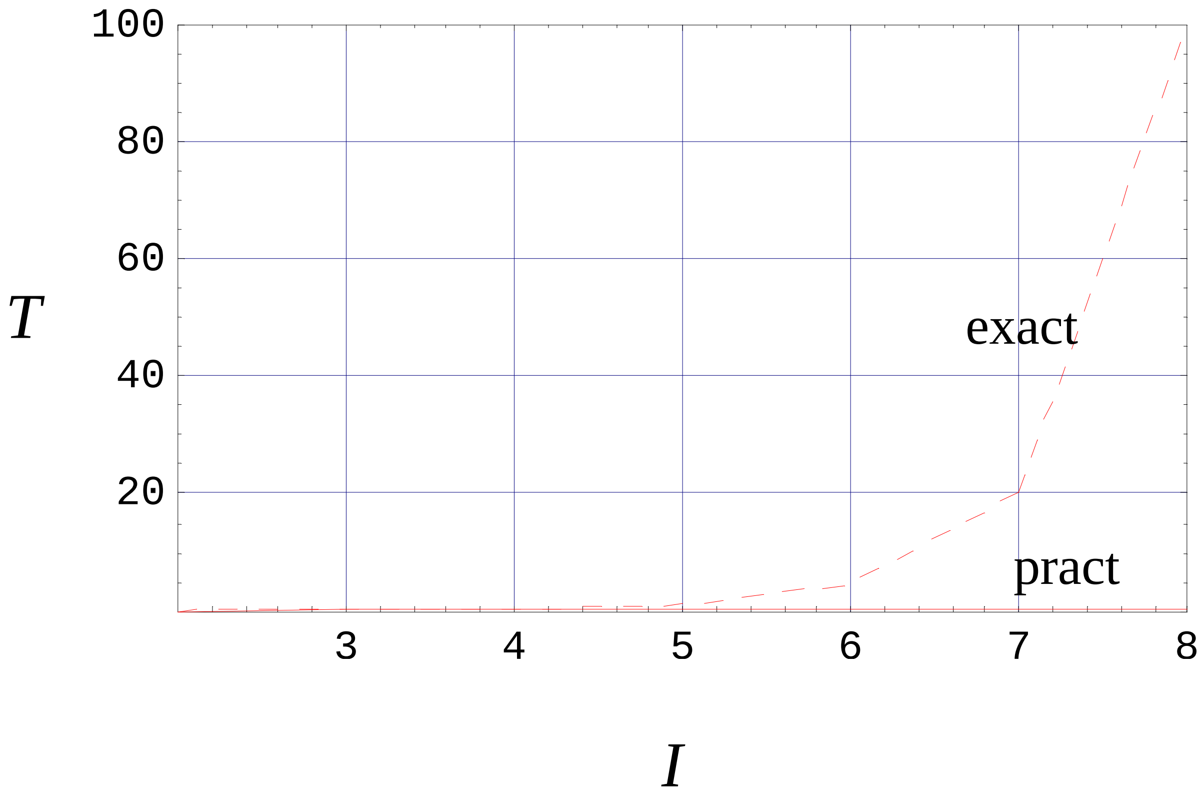
Processing time T vs. nr of multicast connections I (log-log scale)



Example network 2 (figure 5 in [4])



Processing time T vs. nr of multicast connections I (normal scale)



THE END