Retrial Queues and their Application in Performance Modelling of Communication Networks

PhD dissertation

JÁNOS ROSZIK

Faculty of Informatics, University of Debrecen

Debrecen, 2006

Ezen értekezést a Debreceni Egyetem Matematika- és Számítástudományok Doktori Iskola Informatikai rendszerek és hálózatok programja keretében készítettem a Debreceni Egyetem doktori (PhD) fokozatának elnyerése céljából.

Debrecen, 200.. ... ...

. . . . . .
Roszik János
jelölt

Tanúsítom, hogy Roszik János doktorjelölt 2003–2006 között a fent megnevezett Doktori Iskola Informatikai rendszerek és hálózatok programjának keretében irányításommal végezte munkáját. Az értekezésben foglalt eredményekhez a jelölt önálló alkotó tevékenységével meghatározóan hozzájárult. Az értekezés elfogadását javasolom.

Debrecen, 200.. ... ...

. . . . . .
Dr. Sztrik János
témavezető

Acknowledgements

I would like to thank my supervisor Prof. János Sztrik for providing me an interesting subject of research, as well as for his guidance and support throughout this work. I am also grateful to Dr. Béla Almási for his helpful discussions over the years, and to György Marcsek for his valuable linguistic suggestions to the manuscript.

Contents

Chapter 1

Introduction

Performance evaluation plays an important role in the design, analysis and development of practical systems, like computer and telecommunication systems and networks. Queueing models are often used for the performance and reliability modelling of these systems, and retrial queues are more and more frequently applied to certain types of them. The reason is that the return of customers plays a special role in many of these systems as well as in other practical applications, and it often has a non-neglectable negative effect on the performance measures.

Another important characteristic of real-life systems is non-reliability, which also has a negative influence on these measures, because most of the components of the systems are subject to random breakdowns and require repairs. Non-reliability has been extensively studied for traditional queues with waiting lines, but only for infinite-source queues with returning customers.

In the first part of the dissertation, some non-reliable finite-source retrial models and a reliable retrial queue in random environment (which also can be applied in the performance analysis of non-reliable systems) are analyzed. These models have not been treated in the literature before. In the second part, a real-life system is modelled using a retrial queueing model. A modelling way of the GSM system (Global System for Mobile Communications) is treated with the MOSEL (Modeling, Specification and Evaluation Language) tool. This is based on previous works of various authors and generalized with some model extensions.

Part I

Retrial Queues

Chapter 2

Introduction to Retrial Queues

Retrial queues (queueing systems with repeated attempts, or queues with returning customers) are characterized by the following feature: a request finding all servers busy upon arrival leaves the service area but after some (random) time repeats his demand.

Queueing models are often used for the performance analysis of computer and communication systems. In case of many real-life systems, retrial queues can be applied in the performance modelling, for example, in modelling magnetic disk memory systems [44], cellular mobile networks [56], computer networks [30], and local-area networks with non-persistent CSMA/CD protocols [38], with star topology [32; 42], with random access protocols [33], and with multiple-access protocols [34]. For more detailed information and results on this type of queueing systems, see for example [13; 23; 27], and a complete survey can be found on queueing systems without retrials in [55]. Further recent results with finite-source of primary requests can be found in [12; 14; 15; 22; 26; 29; 35; 38; 51].

In the next sections, two types of retrial queues are introduced from [27] (with some modifications and extensions). The content of these sections in their original form can be found in [27], on pages 268–269, 95 and 108–111. The first one is a finite-source retrial queue, on which the analysis is based in Part I. The second one is a truncated infinite source queue with returning customers. An extension of this model is applied in Part II to analyze the GSM system.

2.1 A Retrial Queueing Model with Finite Number of Sources

Consider a c-server queueing system where primary calls are generated by $K, c < K < \infty$, sources. Each source can be in one of three states

- under service
- sending repeated calls (i.e. waiting for service)
- free

If a source is free at time t (i.e. if it is not being served and is not waiting for service) then it may generate a primary call during interval $(t, t + dt)$ with probability λdt.

If there is a free server at the time of arrival of a primary call then the call (or equivalently the source which produced the call) starts to be served. During service the source cannot generate new primary calls. After service the source moves into the free state and can generate a new primary call.

If all servers are busy at time of arrival of a primary call, then the source starts generation of repeated calls at exponential intervals with mean $1/\nu$ until it finds a free server, at which time the source starts to be served. As before, after service the source becomes free and can generate a new primary call.

The service time has an exponential distribution with a finite mean $1/\mu = 1$ both for primary calls and repeated calls.

The functioning of the system can be described by means of process $(C(t), N(t))$, where $C(t)$ is the number of busy servers and $N(t)$ is the number of sources of repeated calls (queue length) at time t. Under the above assumptions process $(C(t), N(t))$ is Markovian with finite state space $S = \{0, 1, ..., c\} \times \{0, 1, ..., K - c\}$. Its infinitesimal transition rates $q_{(ij)(nm)}$ are given by:

1. for $0 \leq i \leq c - 1$

$$q_{(ij)(nm)} = \begin{cases} (K - i - j)\lambda, & \text{if } (n, m) = (i + 1, j) \\ i, & \text{if } (n, m) = (i - 1, j) \\ j\nu, & \text{if } (n, m) = (i + 1, j - 1) \\ -((K - i - j)\lambda & \\ \quad +i + j\nu), & \text{if } (n, m) = (i, j) \\ 0 & \text{otherwise.} \end{cases}$$

2. for $i = c$

$$q_{(cj)(nm)} = \begin{cases} (K - c - j)\lambda, & \text{if } (n, m) = (c, j + 1) \\ c, & \text{if } (n, m) = (c - 1, j) \\ -((K - c - j)\lambda + c), & \text{if } (n, m) = (c, j) \\ 0 & \text{otherwise.} \end{cases}$$

Since the state space of the process $(C(t), N(t))$ is finite, the process is ergodic for all values of the rate of generation of new primary calls, and from now on we will assume that the system is in the steady state.

From a practical point of view the most important characteristics of the quality of service to subscribers are the following. (Note: ∞ denotes the steady state in the definitions of the performance measures.)

- *Mean number of sources of repeated calls*

$$N = \mathrm{E}N(\infty) = \sum_{i=0}^{c} \sum_{j=0}^{K-c} jp_{ij}.$$

- *Mean number of busy servers*

$$Y = \mathrm{E}C(\infty) = \sum_{i=0}^{c} \sum_{j=0}^{K-c} ip_{ij}.$$

- *Mean rate of generation of primary calls*

$$\overline{\lambda} = \lambda \mathrm{E}(K - C(\infty) - N(\infty)) = \lambda(K - Y - N).$$

- *Fraction of primary calls which were blocked (i.e. met all servers busy)*

$$B = \frac{\lambda \mathrm{E}(K - C(\infty) - N(\infty); C(\infty) = c)}{\lambda \mathrm{E}(K - C(\infty) - N(\infty))}.$$

- *Mean waiting time*

$$W = \frac{N}{\overline{\lambda}}.$$

- *Mean response time*

$$T = W + \frac{1}{\mu}.$$

- *Utilization of the sources*

$$U_{SO} = \frac{\mathrm{E}(K - C(\infty) - N(\infty))}{K} = \frac{K - N - Y}{K} = 1 - \frac{N + Y}{K}.$$

- *Overall utilization (i.e. the sum of the utilization of the components of the system)*

$$U_O = Y + KU_{SO}.$$

2.2 A Truncated Infinite-source Retrial Queueing Model

Consider a group of c fully available servers in which a Poisson flow of primary calls with rate λ arrives.

If an arriving primary call finds some server free it immediately occupies a server and leaves the system after service. Otherwise, if all servers are engaged, it produces a source of repeated calls. Every such source after some delay produces repeated calls until after one or more attempts it finds a free server, in which case the source is eliminated and the call receives service and then leaves the system.

We assume that periods between successive retrials are exponentially distributed with parameter ν, and service times are exponentially distributed with parameter μ. Without loss of generality we may assume that $\mu = 1$. Furthermore, we suppose that interarrival periods, retrial times and service times are mutually independent.

In this model, the orbit size (i.e. the number of sources of repeated calls) is bounded by a given constant M. If the number of sources equals M then the blocked calls are lost and have no influence on the functioning of the system. The stochastic dynamics of the system can be described by means of a bivariate process $(C^{(M)}(t), N^{(M)}(t))$, where $C^{(M)}(t)$ is the number of busy servers and $N^{(M)}(t)$ is the number of sources of repeated calls at time t. Under the above assumptions the process $(C^{(M)}(t), N^{(M)}(t))$ is Markovian with the finite lattice semi-strip $S^{(M)} = \{0, 1, ..., c\} \times \{0, 1, ..., M\}$ as the state space. Its infinitesimal transition rates $q^{(M)}_{(ij)(nm)}$ are given by:

1. for $0 \le i \le c-1, 0 \le j \le M$

$$q^{(M)}_{(ij)(nm)} = \begin{cases} \lambda, & \text{if } (n,m) = (i+1, j) \\ i, & \text{if } (n,m) = (i-1, j) \\ j\nu, & \text{if } (n,m) = (i+1, j-1) \\ -(\lambda + i + j\nu) & \text{if } (n,m) = (i,j) \\ 0 & \text{otherwise.} \end{cases}$$

2. for $i = c, 0 \le j \le M-1$

$$q^{(M)}_{(cj)(nm)} = \begin{cases} \lambda, & \text{if } (n,m) = (c, j+1) \\ c, & \text{if } (n,m) = (c-1, j) \\ -(\lambda + c), & \text{if } (n,m) = (c,j) \\ 0 & \text{otherwise.} \end{cases}$$

3. for $i = c, j = M$

$$q^{(M)}_{(cM)(nm)} = \begin{cases} c, & \text{if } (n,m) = (c-1, M) \\ -c, & \text{if } (n,m) = (c, M) \\ 0 & \text{otherwise.} \end{cases}$$

Since the state space of the process $(C^{(M)}(t), N^{(M)}(t))$ is finite, the process is always ergodic. Its stationary distribution $p^{(M)}_{ij} = \mathrm{P}\{C^{(M)}(t) = i, N^{(M)}(t) = j\}$ may be found as a solution of the following set of linear equations:

$$(\lambda + i + j\nu)p^{(M)}_{ij} = \lambda p^{(M)}_{i-1,j} + (j+1)\nu p^{(M)}_{i-1,j+1} + (i+1)p^{(M)}_{i+1,j},$$

$$0 \le i \le c-1, 0 \le j \le M-1, \tag{2.1}$$

$$(\lambda + i + M\nu)p_{iM}^{(M)} = \lambda p_{i-1,M}^{(M)} + (i+1)p_{i+1,M}^{(M)},$$
$$0 \le i \le c-1, \tag{2.2}$$

$$(\lambda + c)p_{cj}^{(M)} = \lambda p_{c-1,j}^{(M)} + (j+1)\nu p_{c-1,j+1}^{(M)} + \lambda p_{c,j-1}^{(M)}$$
$$0 \le j \le M-1, \tag{2.3}$$

$$cp_{cM}^{(M)} = \lambda p_{c-1,M}^{(M)} + \lambda p_{i,M-1}^{(M)}, \tag{2.4}$$

which satisfies the normalizing condition

$$\sum_{i=1}^{c}\sum_{j=0}^{M} p_{ij}^{(M)} = 1. \tag{2.5}$$

Explicit formulas for the main performance characteristics

For generating functions

$$p_i^{(M)}(z) = \sum_{j=0}^{M} z^j p_{ij}^{(M)}, \;\; 0 \le i \le c$$

equations (2.1)-(2.4) become

$$(\lambda + i)p_i^{(M)}(z) + \nu z\frac{dp_i^{(M)}(z)}{dz} = \lambda p_{i-1}^{(M)}(z) + \nu\frac{dp_{i-1}^{(M)}(z)}{dz} + p_{i+1}^{(M)}(z),$$
$$0 \le i \le c-1, \tag{2.6}$$

$$(\lambda + c)p_c^{(M)}(z) - \lambda z^M p_{cM}^{(M)} = \lambda p_{c-1}^{(M)}(z) + \nu\frac{dp_{c-1}^{(M)}(z)}{dz} + \lambda z^{(M)}p_c(z)$$
$$-\lambda z^{M+1}p_{cM}^{(M)}. \tag{2.7}$$

Now introduce the generating function

$$p^{(M)}(x,z) = \sum_{i=0}^{c} x^i p_i^{(M)}(z).$$

Then equations (2.6), (2.7) become:

$$\lambda(1-x)p^{(M)}(x,z) + \nu(z-x)\frac{\partial p^{(M)}(x,z)}{\partial z}$$
$$+(x-1)\frac{\partial p^{(M)}(x,z)}{\partial x} + \lambda x^c(x-z)p_c^{(M)}(z)$$
$$+\nu x^c(x-z)\frac{dp_c^{(M)}(z)}{dz} + \lambda z^M x^c(z-1)p_{cM}^{(M)} = 0.$$

Differentiating this equation with respect to z, x, xx, xz, zz at the point $x = 1$, $z = 1$ we get the following equations:

$$\begin{aligned}
\nu N^{(M)} - \lambda B^{(M)} - \nu N_c^{(M)} + \lambda p_{cM}^{(M)} &= 0,\\
\lambda + \nu N^{(M)} - Y^{(M)} - \lambda B^{(M)} - \nu N_c^{(M)} &= 0,\\
\nu \frac{\partial^2 p^{(M)}(1,1)}{\partial z^2} - \lambda N_c^{(M)} - \nu \frac{d^2 p_c^{(M)}(1)}{dz^2} + \lambda M p_{cM}^{(M)} &= 0,\\
-\lambda N^{(M)} - \nu \frac{\partial^2 p^{(M)}(1,1)}{\partial z^2} + (1+\nu)\frac{\partial^2 p^{(M)}(1,1)}{\partial x \partial z}&\\
+\lambda N_c^{(M)} - \nu c N_c^{(M)} - \lambda c B^{(M)} + \nu \frac{d^2 p_c^{(M)}(1)}{dz^2} + \lambda c p_{cM}^{(M)} &= 0,\\
\lambda Y^{(M)} + \nu \frac{\partial^2 p^{(M)}(1,1)}{\partial x \partial z} - \frac{\partial^2 p^{(M)}(1,1)}{\partial x^2} - \lambda c B^{(M)} &= \nu c N_c^{(M)},
\end{aligned}$$

where

$$\begin{aligned}
N^{(M)} &\equiv \mathrm{E}N^{(M)}(\infty) = \frac{\partial p^{(M)}(1,1)}{\partial z},\\
B^{(M)} &\equiv \mathrm{P}\{C^{(M)}(\infty) = c\} = p_c^{(M)}(1),\\
Y^{(M)} &\equiv \mathrm{E}C^{(M)}(\infty) = \frac{\partial p^{(M)}(1,1)}{\partial x},\\
N_c^{(M)} &\equiv \mathrm{E}\{N^{(M)}(\infty); C^{(M)}(\infty) = c\} = \frac{dp_c^{(M)}(1)}{dz}.
\end{aligned}$$

Eliminating from these equations variables

$$N_c^{(M)}, \frac{\partial^2 p^{(M)}(1,1)}{\partial x \partial z}, \frac{\partial^2 p^{(M)}(1,1)}{\partial z^2}, \frac{d^2 p_c^{(M)}(1)}{dz^2}$$

and taking into account that

$$\frac{\partial^2 p^{(M)}(1,1)}{\partial x^2} = \mathrm{Var}C^{(M)}(\infty) + \left(\mathrm{E}C^{(M)}(\infty)\right)^2 - \mathrm{E}C^{(M)}(\infty)$$

we get:

$$Y^{(M)} = \lambda - \lambda p_{cM}^{(M)} \tag{2.8}$$

$$N^{(M)} = \frac{1+\nu}{\nu}\frac{\lambda + \lambda^2 - \mathrm{E}\left(C^{(M)}(\infty)\right)^2}{c - \lambda} - \frac{\lambda}{\nu}\frac{(c+1+\lambda)(1+\nu) + M\nu}{c-\lambda} p_{cM}^{(M)} \tag{2.9}$$

Equation (2.8) can be thought of as a variant of Little's formula and represents a balance between offered, carried and lost traffic. Equation (2.9) is much more interesting. It gives a partial description of the dependence of the mean queue length upon the system parameters, and reduces calculation of the mean queue length to the calculation of the characteristics of the number of busy servers and the rate of lost traffic, which is a simpler problem.

Chapter 3

The MOSEL Tool

MOSEL (Modeling, Specification and Evaluation Language) [16] is a modelling environment with a high-level modelling language which allows us to describe complex real-world systems and to calculate their system measures using other performance evaluation tools. The MOSEL description can be translated automatically into the language of various performance tools and then analyzed by the appropriate tool (at present SPNP – Stochastic Petri Net Package and TimeNET are supported and suitable for the investigated models) to get these measures.

Because of the fact that the state space of the underlying Markov chains of the investigated queueing models is very large and the functioning of the systems is complex, it is quite difficult to calculate the steady state probabilities in the traditional way of solving the system of steady-state equations. To simplify these calculations and to make these studies more usable in practice, the tool MOSEL was used to formulate the models and to calculate the performance measures. With the tool, we can perform two steps in one, so we do not need to write down and somehow solve the set of steady-state equations. The difficulty of modelling lies in the description of the system and its behavior for the performance tool.

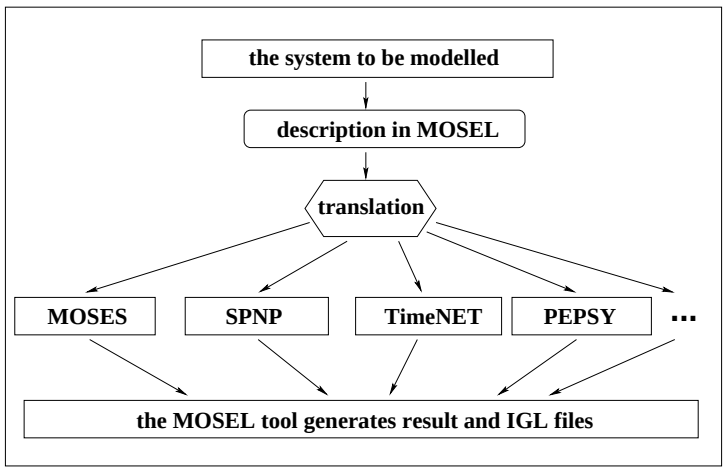


Figure 3.1: The modelling process in the MOSEL environment

MOSEL has already been used, and it has proved its applicability for the modelling of several computer and communication systems. For some examples about computer systems see [5; 8; 61] and in the context of cellular systems [17; 18; 39].

The functioning and usage of MOSEL is illustrated by Figure 3.1. In the modelling process, the user describes the system to be modelled in MOSEL, then the MOSEL description is translated into the language of the chosen performance tool. The tool MOSEL invokes the appropriate tool, parses its results and generates a result file containing the system measures which the user specified in the MOSEL description. If the modeler required graphical representation of the results, an IGL (Intermediate Graphical Language) file is generated, too.

The technical details of programming can be found in [16], and in [19; 17], where the new, revised version of MOSEL, called MOSEL-2, is introduced. Another language extensions supporting non-Markovian distributions, that can be evaluated with the help of the tool SPNP are provided in [59; 60].

In Part I, the original tool is used for the performance evaluation of finite-source retrial queueing systems. Because of page limitations, only the simplest MOSEL description was included and discussed.

In Part II, the GSM system is modelled with the revised modelling language and detailed comments are provided about MOSEL-2 programming.

Chapter 4

Analysis of Single-server Non-reliable Finite-source Retrial Queues

The components of the real systems may be subject to random breakdowns (see for example [36; 46; 57]), so it is important to investigate non-reliable queues because of limited ability of repairs and heavy influence of the breakdowns on the performance measures of the systems (see [3; 9; 53]). Besides this, the analysis of non-reliable retrial queues (where the sources and the server may be subject to random breakdowns and repairs) is also important. For related literature the reader is referred to the works [11; 1; 37; 58] where infinite-source non-reliable retrial queues are treated.

In this chapter finite-source non-reliable retrial queues are investigated. The purpose is to give the main stationary performance and reliability measures of the non-reliable models described in the next sections, and to illustrate graphically the effect of changing various parameters on them. Section 4.1 is devoted to the model described in [27] with server subject to breakdowns and repairs. In Section 4.2, this is extended with non-reliable sources, and in Section 4.3 with reliable but heterogeneous sources. These models were published in [J1; J3; J4].

Note: Because of the fact that these sections contain the main parts of different papers from the author, the reader will find some similarities between them, as well as in Chapter 5 and Chapter 6.

4.1 The $M/M/1//K$ Model with Non-reliable Server

Consider a single server queueing system, where the primary calls are generated by K, $1 < K < \infty$ homogeneous sources. The server can be in operational (up) or non-operational (down) states, and it can be idle or busy. If the server is idle and up, it can serve the calls of the sources. Each of the sources can be in three states: free, sending repeated calls and under service. If a source is free at time t it can generate a primary call during interval $(t, t + dt)$ with probability $\lambda dt + o(dt)$. If the server is free at the time of arrival of a call then the call starts to be served immediately, the source moves into the under service state and the server moves into busy state. The service is finished during the interval $(t, t + dt)$ with probability $\mu dt + o(dt)$ if the server is available. If the server is busy, then the source starts generation of a Poisson flow of repeated calls with rate ν until it finds the server free. After service the source becomes free, and it can generate a new primary call, and the server becomes idle so it can serve a new call. The server can fail during the interval $(t, t + dt)$ with probability $\delta dt + o(dt)$ if it is idle, and with probability $\gamma dt + o(dt)$ if it is busy. If $\delta = 0, \gamma > 0$ or $\delta = \gamma > 0$ *active or independent breakdowns* can be discussed, respectively. If the server fails in busy state, it either *continues servicing* the interrupted call after it has been repaired or the interrupted request *returns to the orbit* (i.e. the source starts generation of repeated calls). The repair time is exponentially distributed with a finite mean $1/\tau$. If the server is failed two different cases can be treated. Namely, *blocked sources* case when all the operations are stopped, that is neither new primary calls nor repeated calls are generated. In the *unblocked (intelligent) sources* case only service is interrupted but all the other operations are continued (new and repeated calls can be generated). All the times involved in the model are assumed to be mutually independent of each other.

This model is another extension of investigations for homogeneous finite-source queueing systems without retrials but with server's breakdowns which were treated in [53]. Similarly, it generalizes the results of [27] where homogeneous systems with reliable servers were analyzed. As it can be seen, this system is more complicated than in the reliable case, since it involves two types of failures, continued or repeated service and blocked or unblocked operations during breakdowns.

In the next subsection the full description of the model by the help of the corresponding multi-component Markov chain is given. Then, the main performance and reliability measures of the system are derived that can be obtained using the MOSEL tool. Finally, the validation of the results and several numerical examples are presented and some comments are made.

4.1.1 The Underlying Markov Chain

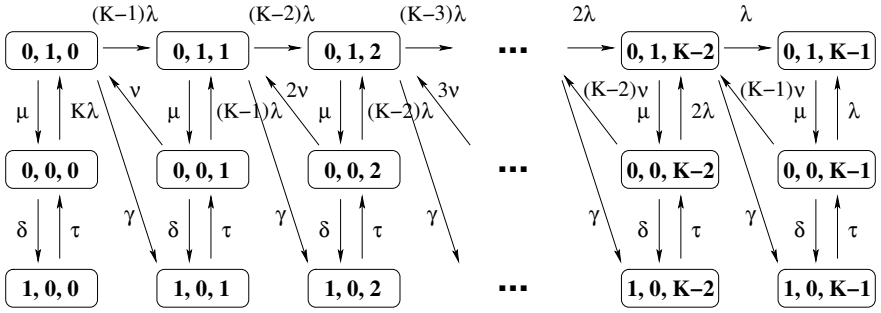
The system state at time t can be described with the process $X(t) = (Y(t); C(t); N(t))$, where $Y(t) = 0$ if the server is up, $Y(t) = 1$ if the server is failed, $C(t) = 0$ if the server is idle, $C(t) = 1$ if the server is busy and $N(t)$ is the number of sources of repeated calls at time t. Because of the exponentiality of the involved random variables this process is a Markov chain with a finite state space. Since the state space of the process $(X(t), t \geq 0)$ is finite, the process is ergodic for all values of the rate of generation of primary calls, and from now on we will assume that the system is in the steady state.

We define the stationary probabilities as follows:

$$P(q,r,j) = \lim_{t\to\infty} P(Y(t)=q, C(t)=r, N(t)=j),$$
$$q=0,1, \quad r=0,1, \quad j=0,...,K^*, \quad \text{where}$$

$$K^* = \begin{cases} K-1 & \text{for blocked case,} \\ K-r & \text{for unblocked case.} \end{cases}$$

Based on the following state transition diagram (which belongs to the simplest case, i.e. the request under service returns to the orbit in case of server breakdowns, and neither new primary calls nor repeated calls are generated during server failure)



the stationary probabilities can be found as a solution of the following set of steady state equations:

$$\begin{aligned}
((K-1)\lambda+\mu+\gamma)P(0,1,0) &= \nu P(0,0,1) + K\lambda P(0,0,0) \\
((K-i-1)\lambda+\mu+\gamma)P(0,1,i) &= (K-i)\lambda P(0,1,i-1) + i\nu P(0,0,i+1) \\
&\quad + (K-i)\lambda P(0,0,i), \;\; i=1,...,K-1 \\
((K-i)\lambda+\delta+i\nu)P(0,0,i) &= \mu P(0,1,i) + \tau P(1,0,i), \;\; i=0,...,K-1 \\
\tau P(1,0,0) &= \delta P(0,0,0) \\
\tau P(1,0,i) &= \delta P(0,0,i) + \gamma P(0,1,i-1), \;\; i=1,...,K-1
\end{aligned}$$

which satisfies the normalizing condition

$$\sum_{r=0}^{1}\sum_{j=0}^{K-1} P(0,r,j) + \sum_{j=0}^{K-1} P(1,0,j) = 1.$$

Knowing the stationary probabilities the main performance measures can be obtained as follows:

- *Utilization of the server*

$$U_S = \sum_{j=0}^{K-1} P(0,1,j).$$

- *Utilization of the repairman*

$$U_R = \sum_{r=0}^{1} \sum_{j=0}^{K^*} P(1, r, j).$$

- *Availability of the server*

$$A_S = \sum_{r=0}^{1} \sum_{j=0}^{K^*} P(0, r, j) = 1 - U_R.$$

- *Mean number of sources of repeated calls*

$$N = E[N(\infty)] = \sum_{q=0}^{1} \sum_{r=0}^{1} \sum_{j=0}^{K^*} jP(q, r, j).$$

- *Mean number of calls staying in the orbit or in service*

$$M = E[N(\infty) + C(\infty)] = N + \sum_{q=0}^{1} \sum_{j=0}^{K-1} P(q, 1, j).$$

- *Utilization of the sources*

$$U_{SO} = \begin{cases} \frac{K-M}{K} A_S & \text{for blocked case,} \\ \frac{K-M}{K} & \text{for unblocked case.} \end{cases}$$

- *Overall utilization*

$$U_O = U_S + KU_{SO} + U_R.$$

- *Mean rate of generation of primary calls*

$$\overline{\lambda} = \begin{cases} \lambda E[K - C(\infty) - N(\infty); Y(\infty) = 0], & \text{for blocked case,} \\ \lambda E[K - C(\infty) - N(\infty)], & \text{for unblocked case.} \end{cases}$$

- *Blocking probability of a primary call*

$$B = \begin{cases} \frac{\lambda E[K - C(\infty) - N(\infty); Y(\infty) = 0; C(\infty) = 1]}{\overline{\lambda}}, & \text{for blocked case,} \\ \frac{\lambda E[K - C(\infty) - N(\infty); C(\infty) = 1]}{\overline{\lambda}}, & \text{for unblocked case.} \end{cases}$$

- *Mean response time*

$$E[T] = M/\overline{\lambda}.$$

- *Mean waiting time*

$$E[W] = N/\overline{\lambda}.$$

To simplify this procedure and to make our study more usable in practice, we use the software tool MOSEL to formulate the model and to calculate the main performance measures.

4.1.2 The MOSEL Implementation

In this subsection the base MOSEL program and the explanation of its main parts are introduced without the technical details of programming. This program belongs to the case of continued service after server's repair and requests' generation is blocked during the server repairing. It does not contain the picture section, which is needed to generate various graphical representations of the measures. The figures in the next section are automatically generated by the tool with the corresponding picture part. In the MOSEL program the following terminology is used: The server and the sources are referred to as a CPU and terminals, respectively.

```
/*------------------------------------------ Declarations------*/
#define NT 3
VAR double prgen;
VAR double prretr;
VAR double prrun;
VAR double cpubreak_idle;
VAR double cpubreak_busy;
VAR double cpurepair;
enum cpu_states {cpu_busy, cpu_idle};
enum cpu_updown {cpu_up, cpu_down};
/*------------------------------------------------ Nodes ------*/
NODE busy_terminals[NT] = NT;
NODE retrying_terminals[NT] = 0;
NODE waiting_terminals[1] = 0;
NODE cpu_state[cpu_states] = cpu_idle;
NODE cpu[cpu_updown] = cpu_up;
/*---------------------------------------- Transitions ------*/
IF cpu==cpu_up FROM cpu_idle, busy_terminals
   TO cpu_busy, waiting_terminals W prgen*busy_terminals;
IF cpu==cpu_up AND cpu_state==cpu_busy FROM busy_terminals
   TO retrying_terminals W prgen*busy_terminals;
IF cpu==cpu_up FROM cpu_idle, retrying_terminals
   TO cpu_busy, waiting_terminals W prretr*retrying_terminals;
IF cpu==cpu_up FROM cpu_busy, waiting_terminals
   TO cpu_idle, busy_terminals W prrun;
IF cpu_state==cpu_idle FROM cpu_up TO cpu_down W cpubreak_idle;
IF cpu_state==cpu_busy FROM cpu_up TO cpu_down W cpubreak_busy;
FROM cpu_down TO cpu_up W cpurepair;
/*------------------------------------------- Results ------*/
RESULT>> if(cpu==cpu_up AND cpu_state==cpu_busy) cpuutil += PROB;
RESULT>> if(cpu==cpu_up) goodcpu += PROB;
RESULT if(cpu==cpu_up) busyterm += (PROB*busy_terminals);
RESULT>> termutil = busyterm / NT;
RESULT>> if(cpu==cpu_up) retravg += (PROB*retrying_terminals);
RESULT if(waiting_terminals>0) waitall += (PROB*waiting_terminals);
RESULT if(retrying_terminals>0)retrall += (PROB*retrying_terminals);
RESULT>> resptime = (retrall + waitall) / NT / (prgen * termutil);
RESULT>> overallutil = cpuutil + busyterm + (1 - goodcpu);
```

In the *declarations part* we define the number of terminals (NT), this is the only program code line that must be modified when modelling larger systems. The terminals have three states: busy (primary call generation), retrying (repeated call generation) and waiting (job service at the CPU). The CPU has two states: idle and busy, and it can be up or failed in both states. We define the three parameters for the terminals: $prgen$ denotes the rate of primary call generation, $prretr$ references to the rate of repeated call generation and $prrun$ denotes the service rate. The $cpubreak\_idle$, $cpubreak\_busy$ and $cpurepair$ variables denote the failure rate in the two CPU states and the repair rate.

The *nodes part* defines the nodes of the system. Our queueing network contains 5 nodes: one node for the number of busy, retrying and waiting terminals, respectively, and two nodes for the CPU. The CPU is idle and up and all the terminals are busy at the starting time.

The *transitions part* describes how the system works. The first transition rule defines the successful primary call generation: the CPU moves from the idle state to busy and the terminal from busy to waiting. The second rule shows an unsuccessful primary call generation: if the CPU is busy when the call is generated then the terminal moves to state retrying. The third rule handles the case of a successful repeated call generation: the CPU moves from the idle state to busy and the terminal from retrying to waiting. The fourth rule describes the request service at the CPU. The fifth and sixth rules describe the CPU fail in idle and busy state. The last rule shows the CPU repair.

Finally, the *results part* calculates the requested output performance measures.

4.1.3 Validation of Results

The results in the reliable case (with very low failure rate and very high repair rate) were validated by the (a little modified) Pascal program for the reliable case given in [27], on pages 272–274. See Table 4.1 for some test results.

In Table 4.2 the results were tested by the help of non-reliable FIFO (First In First Out) model, since if the retrial rate in the repeated calls model tends to infinity, the measures should approach the corresponding ones in FIFO discipline. The derived results are the same up to the 6th decimal digit.

	non-rel. retr. (cont.)	non-rel. retr. (orbit)	reliable [27]
Number of sources:	5	5	5
Request's generation rate:	0.2	0.2	0.2
Service rate:	1	1	1
Retrial rate:	0.3	0.3	0.3
Utilization of the server:	0.5394868123	0.5394867440	0.5394867746
Mean response time:	4.2680691205	4.2680667075	4.2680677918

Table 4.1: Validations in the reliable case

4.1.4 Numerical Examples

In this subsection some sample numerical results are considered to illustrate graphically the influence of the non-reliable server on the mean response time $E[T]$ and on the overall utilization of the system.

	non-rel. retr. (cont.)	non-rel. retr. (orbit)	non-rel. FIFO
Number of sources:	3	3	3
Request's generation rate:	0.1	0.1	0.1
Service rate:	1	1	1
Retrial rate:	1e+25	1e+25	–
Server's failure rate:	0.01	0.01	0.01
Server's repair rate:	0.05	0.05	0.05
Utilization of the server:	0.2232796561	0.2232796553	0.2232796452
Mean response time:	1.4360656331	1.4360656261	1.4360655471

Table 4.2: Validations in the non-reliable case

In the first 3 cases the independent breakdowns are treated, then in the next 3 cases the state dependent and independent ones are considered. In each case different comparisons are made according to the breakdowns *(dependent, independent)*, service continuation *(continued, repeated)* and system operations *(blocked, unblocked)*.

In the last 4 figures, the independent failure case is considered and different comparisons are made according to service continuation and system operations.

The tool SPNP was used which was able to handle the model with up to 126 sources. In this case, on a computer containing a 950 MHz processor and 512 MB RAM, the running time was approximately 1 second.

	K	λ	μ	ν	δ	γ	τ
Figure 4.1	6	x axis	4	0.4	0.05	0.05	0.1
Figure 4.2	6	5	10	x axis	0.05	0.05	0.1
Figure 4.3	6	0.1	x axis	0.4	0.05	0.05	0.1
Figure 4.4	6	x axis	4	0.4	0.005(0.05)	0.05	0.1
Figure 4.5	6	5	10	x axis	0.005(0.05)	0.05	0.1
Figure 4.6	6	0.1	x axis	0.4	0.005(0.05)	0.05	0.1
Figure 4.7	6	0.8	4	0.5	x axis		0.1
Figure 4.8	6	0.1	0.5	0.5	x axis		0.1
Figure 4.9	6	0.8	4	0.5	0.05	0.05	x axis
Figure 4.10	6	0.05	0.3	0.2	0.05	0.05	x axis

Table 4.3: System input parameters

Comments

- In Figures 4.1–4.3 we can see the mean response time $E[T]$ for the reliable and the non-reliable retrial systems with continuous, non-continuous service after repair, with blocked and unblocked operations during service failure when the primary request generation rate, retrial rate and service rate increase. In these cases, the server's failure rate is independent of the state of the server. Figure 4.1 demonstrates a surprising phe-

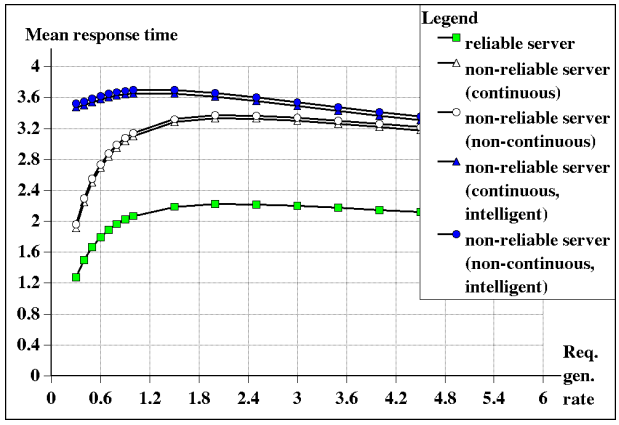


Figure 4.1: $E[T]$ versus primary request generation rate

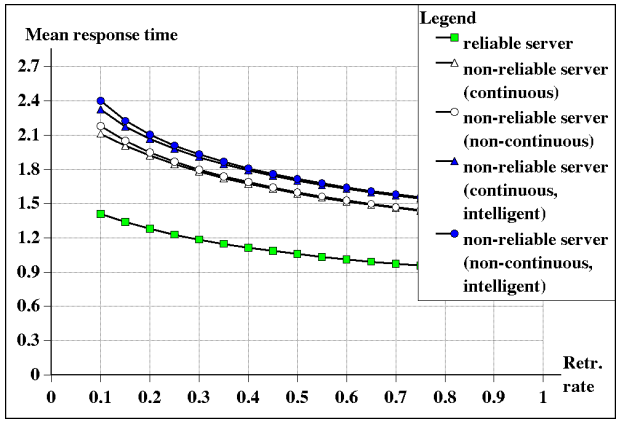


Figure 4.2: $E[T]$ versus retrial rate

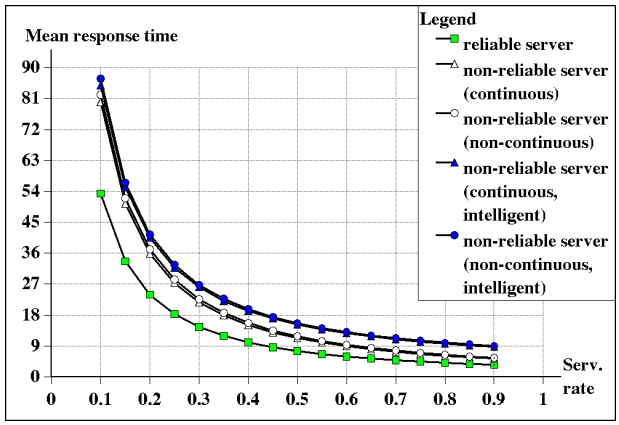


Figure 4.3: $E[T]$ versus service rate

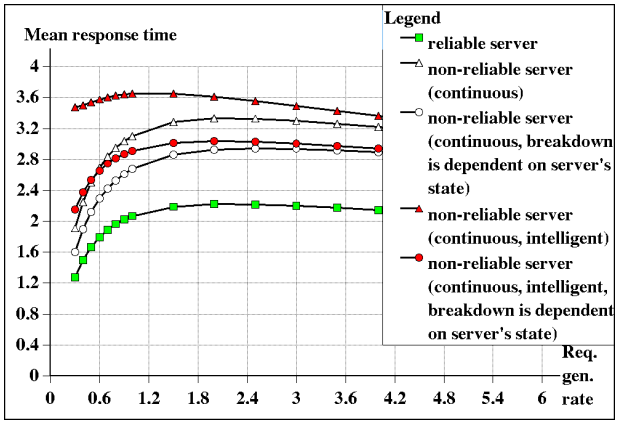


Figure 4.4: $E[T]$ versus primary request generation rate

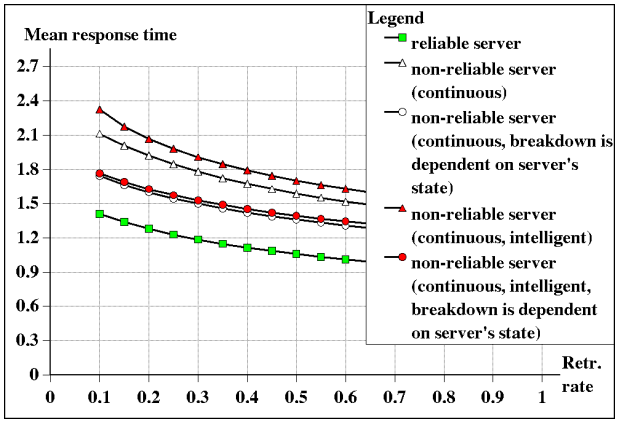


Figure 4.5: $E[T]$ versus retrial rate

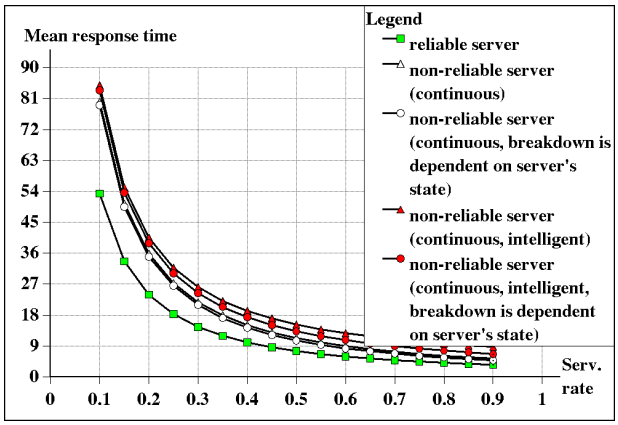


Figure 4.6: $E[T]$ versus service rate

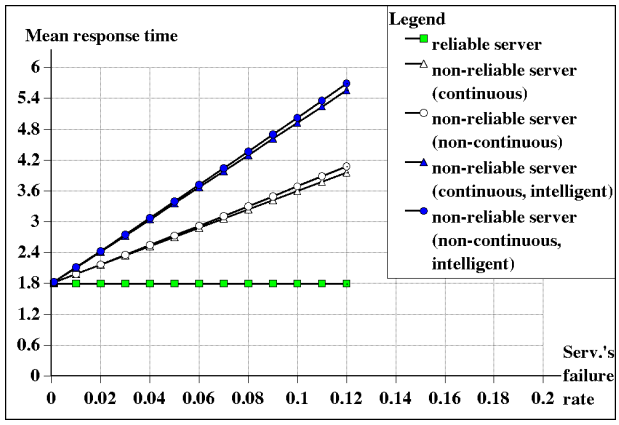


Figure 4.7: $E[T]$ versus server's failure rate

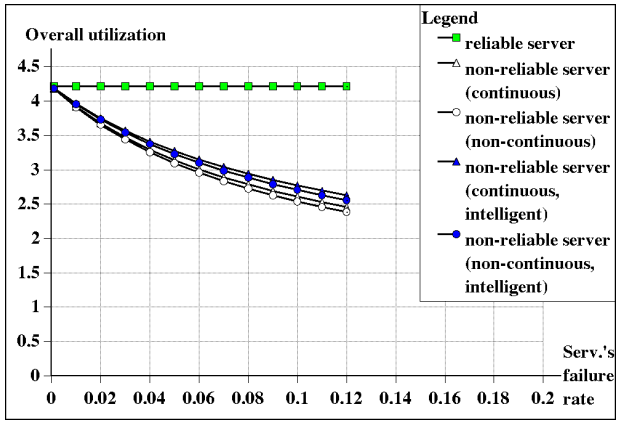


Figure 4.8: U_O versus server's failure rate

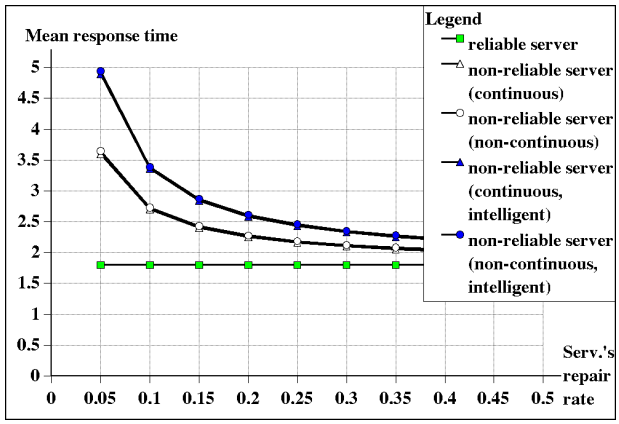


Figure 4.9: $E[T]$ versus server's repair rate

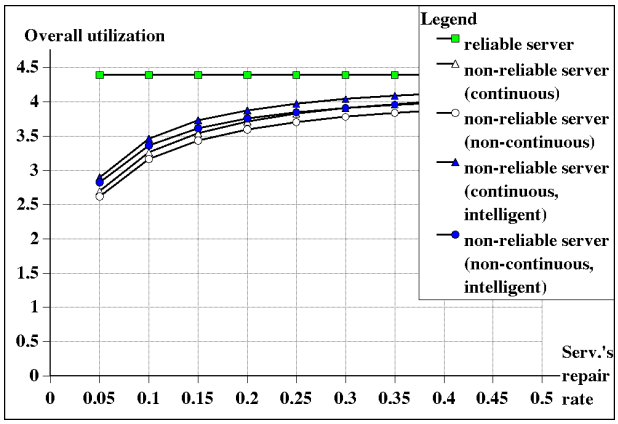


Figure 4.10: U_O versus server's repair rate

nomenon of retrial queues having a maximum of $E[T]$ which was noticed in [25], too. The difference between continuous, non-continuous service, moreover blocked, unblocked systems' operations is clearly shown. However, if the retrial rate increases the continuous and non-continuous service result in the same measure, as it was expected.

- In Figures 4.4–4.6 the mean response time $E[T]$ is displayed with continuous service after repair but the server's failure rate depends on its state. The system operation is either blocked or unblocked. In Figure 4.4 we can see that the curves of independent failure with blocked operations and dependent failures with unblocked operations intersect each other. In each case the difference between the independent and dependent failures is clearly demonstrated.

- In Figure 4.7, we can see that when the request returns to the orbit at the breakdown of the server, the sources will have always longer response times. Although the difference is not considerable it increases as the failure rate increases. The almost linear increase in $E[T]$ can be explained as follows. In the blocked (non-intelligent) case the failure of the server blocks all the operations and the response time is the sum of the down time of the server, the service and repeated call generation time of the request (which does not change during the failure) thus the failure has a linear effect on this measure. In the intelligent case the only difference is that the sources send repeated calls during the server is unavailable, so this is not an additional time.

- In Figure 4.8 and Figure 4.10 it is shown how much the overall utilization is higher in the intelligent case with the given parameters. It is clear that the continued cases have better utilizations, because a request will be at the server when it has been repaired.

- In Figure 4.9, we can see that if the request returns to the orbit at the breakdown of the server, the sources will have longer response times like in Figure 4.7. The difference is not considerable too, and as it was expected, the curves converge to the reliable case.

4.2 The $M/M/1//K$ Model with Non-reliable Server and Non-reliable Sources

Consider a finite-source single server retrial queueing system, where the primary calls are generated by K, $1 < K < \infty$ homogeneous sources. The server can be in operational or non-operational states, and it can be idle or busy. Each of the sources can be in four states: generating a primary call (free), sending repeated calls, under service and failed. If a source is free at time t it can generate a primary call during interval $(t, t + dt)$ with probability $\lambda dt + o(dt)$. If the server is up and idle at the time of the arrival of a call then the call starts to be served immediately, the source moves into the under service state and the server moves into busy state. The service is finished during the interval $(t, t + dt)$ with probability $\mu dt + o(dt)$ if the server is available.

The server can fail during the interval $(t, t + dt)$ with probability $\delta dt + o(dt)$ if it is idle, and with probability $\gamma dt + o(dt)$ if it is busy. Like in the previous section, we treat four types of the model. If the server fails in busy state, it either *continues servicing* the interrupted call after it has been repaired or the interrupted request *returns to the orbit*. The repair time of the server is exponentially distributed with a finite mean $1/\tau$. If the server is failed, two different cases can be treated. Namely, *blocked sources* case when all the operations are stopped expect from the repair of the server. In the *unblocked (intelligent) sources* case only service is interrupted but all the other operations are continued.

If the server is busy (or failed in the unblocked case) at the time of the arrival of a call then the source starts generation of a Poisson flow of repeated calls with rate ν until it finds the server free and up. After service the source becomes free, and it can generate a new primary call, and the server becomes idle so it can serve a new call.

Sources can be non-operational only in free state. If a source is free at time t it can fail during the interval $(t, t + dt)$ with probability $\eta dt + o(dt)$ and then it moves to the repairman who follows FIFO discipline for the source breakdowns and gives preemptive priority to the server failure. The repair time of the sources is exponentially distributed with a finite mean $1/\kappa$. All the times involved in the model are assumed to be mutually independent of each other.

4.2.1 The Underlying Markov Chain

The system state at time t can be described with the process $X(t) = (Y(t); C(t); N(t); Z(t))$, where $Y(t) = 0$ if the server is up, $Y(t) = 1$ if the server is failed, $C(t) = 0$ if the server is idle, $C(t) = 1$ if the server is busy, $N(t)$ is the number of sources of repeated calls and $Z(t)$ is the number of failed sources at time t. Because of the exponentiality of the involved random variables this process is a Markov–chain with a finite state space.

Since the state space of the process $(X(t), t \geq 0)$ is finite, the process is ergodic for all values of the rate of generation of primary calls. From now on we will assume that the system is in the steady state.

We define the stationary probabilities:

$$P(q; r; j; k) = \lim_{t \to \infty} P(Y(t) = q, C(t) = r, N(t) = j, Z(t) = k),$$

$$q = 0, 1, \quad r = 0, 1, \quad j = 0, ..., K^*, \quad k = 0, ..., K - r - j, \quad \text{where}$$

$$K^* = \begin{cases} K - 1 & \text{for blocked case,} \\ K - r & \text{for unblocked case.} \end{cases}$$

To obtain the system performance and reliability measures the tool MOSEL is used to get the state probabilities in the equilibrium. Once we have obtained the steady state probabilities, the main system performance measures can be derived in the following way:

- *Availability of the server*

$$A_S = \sum_{r=0}^{1} \sum_{j=0}^{K^*} \sum_{k=0}^{K-r-j} P(0, r, j, k).$$

- *Mean number of sources of repeated calls*

$$N = E[N(\infty)] = \sum_{q=0}^{1} \sum_{r=0}^{1} \sum_{j=0}^{K^*} \sum_{k=0}^{K-r-j} jP(q, r, j, k).$$

- *Mean number of calls staying in the orbit or in service*

$$M = E[N(\infty) + C(\infty)] = N + \sum_{q=0}^{1} \sum_{j=0}^{K-1} \sum_{k=0}^{K-1-j} P(q, 1, j, k).$$

- *Mean number of operational sources*

$$N_O = K - \sum_{q=0}^{1} \sum_{r=0}^{1} \sum_{j=0}^{K^*} \sum_{k=1}^{K-r-j} kP(q, r, j, k).$$

- *Utilization of the server*

$$U_S = \sum_{j=0}^{K-1} \sum_{k=0}^{K-1-j} P(0, 1, j, k).$$

- *Utilization of the repairman*

$$U_R = \sum_{r=0}^{1} \sum_{j=0}^{K^*} \sum_{k=1}^{K-r-j} P(0, r, j, k) + \sum_{r=0}^{1} \sum_{j=0}^{K^*} \sum_{k=0}^{K-r-j} P(1, r, j, k).$$

- *Utilization of the sources*

$$U_{SO} = \begin{cases} \frac{N_O - M}{K} A_S & \text{for blocked case,} \\ \frac{N_O - M}{K} & \text{for unblocked case.} \end{cases}$$

- *Overall utilization*

$$U_O = U_S + KU_{SO} + U_R.$$

- *Mean rate of generation of primary calls*

$$\overline{\lambda} = \begin{cases} \lambda E[K - C(\infty) - N(\infty) - Z(\infty); Y(\infty) = 0] & \text{for blocked case,} \\ \lambda E[K - C(\infty) - N(\infty) - Z(\infty)] & \text{for unblocked case.} \end{cases}$$

- *Blocking probability of a primary call*

$$B = \begin{cases} \frac{\lambda E[K - C(\infty) - N(\infty) - Z(\infty); Y(\infty) = 0; C(\infty) = 1]}{\overline{\lambda}} & \text{for blocked case,} \\ \frac{\lambda E[K - C(\infty) - N(\infty) - Z(\infty); C(\infty) = 1]}{\overline{\lambda}} & \text{for unblocked case.} \end{cases}$$

- *Mean response time*

$$E[T] = M/\overline{\lambda}.$$

- *Mean waiting time*

$$E[W] = N/\overline{\lambda}.$$

4.2.2 Validation of Results

The results in the reliable case were validated by the Pascal program given in [27]. In the case of server's breakdowns and reliable sources the program was tested by the results of [10].

In Table 4.4 some test results are collected when the retrial rates are quite large. The corresponding performance measures should be very close to each other in the case of continued service, restarted repeated call generation after server failure (abbreviated by orbit) and the FIFO discipline which was studied in [5]. As we can see, the results confirm our expectation, the derived results are the same up to the 6th decimal digit.

	non-rel. retr. (cont.)	non-rel. retr. (orbit)	non-rel. FIFO
Number of sources:	3	3	3
Request's generation rate:	0.1	0.1	0.1
Service rate:	1	1	1
Retrial rate:	1e+25	1e+25	–
Server's failure rate:	0.02	0.02	0.02
Server's repair rate:	0.05	0.05	0.05
Sources' failure rate:	0.03	0.03	0.03
Sources' repair rate:	0.05	0.05	0.05
Utilization of the server:	0.0965679029	0.0965679117	0.0965678743
Mean response time:	1.5546014407	1.5546014565	1.5546013953

Table 4.4: Validations

4.2.3 Numerical Examples

In this subsection some sample numerical results are treated to illustrate graphically the influence of the non-reliable server and sources on the mean response time $E[T]$, on the mean number of requests staying at the service facility (in the orbit and at the server), on the overall system utilization, on the mean number of sources of repeated calls and on the mean number of operational sources. The system input parameters of the following figures are collected in Table 4.5.

	K	λ	μ	ν	δ	γ	τ	η	κ
Figure 4.11	5	x axis	4.5	0.5	0.05	0.05	0.1	0.06	0.15
Figure 4.12	5	5	10	x axis	0.05	0.05	0.1	0.06	0.15
Figure 4.13	5	0.1	x axis	0.4	0.05	0.05	0.1	0.06	0.15
Figure 4.14	5	x axis	4.5	0.5	0.005 (0.05)	0.05	0.1	0.06	0.15
Figure 4.15	5	5	10	x axis	0.005 (0.05)	0.05	0.1	0.06	0.15
Figure 4.16	5	0.1	x axis	0.4	0.005 (0.05)	0.05	0.1	0.06	0.15
Figure 4.17	5	0.8	4.5	0.5	0.05	x axis	0.1	0.06	0.15
Figure 4.18	5	0.1	0.5	x axis	0.05	0.05	0.1	0.06	0.15
Figure 4.19	5	0.1	0.5	x axis	0.05	0.05	0.1	0.06	0.15
Figure 4.20	5	0.8	4.5	x 0.5	0.05	0.05	0.1	x axis	0.15

Table 4.5: System input parameters

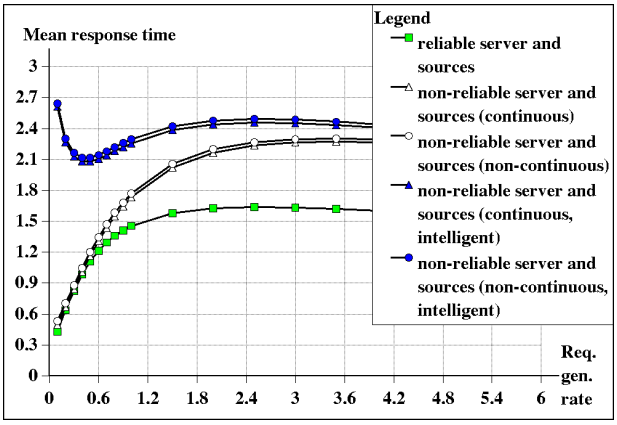


Figure 4.11: $E[T]$ versus primary request generation rate

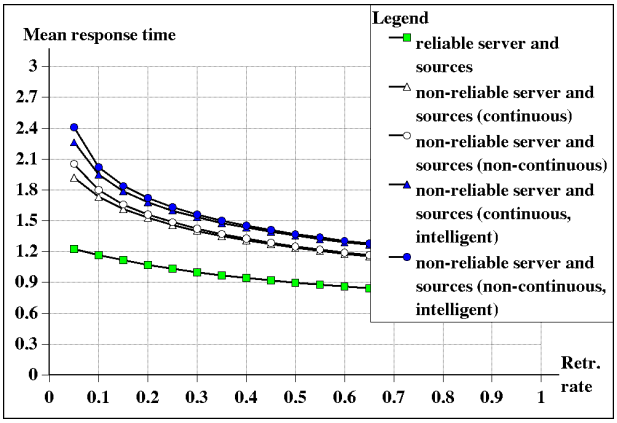


Figure 4.12: $E[T]$ versus retrial rate

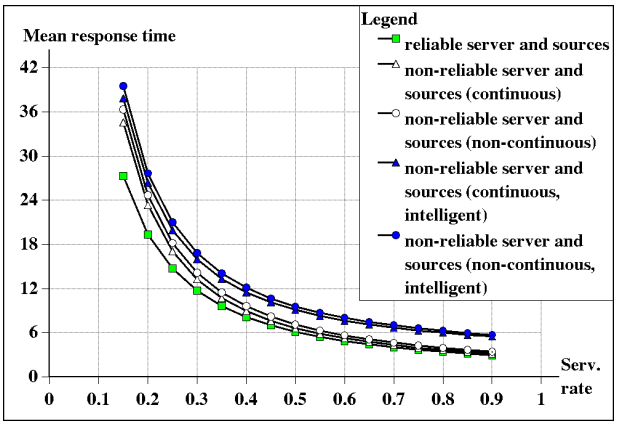


Figure 4.13: $E[T]$ versus service rate

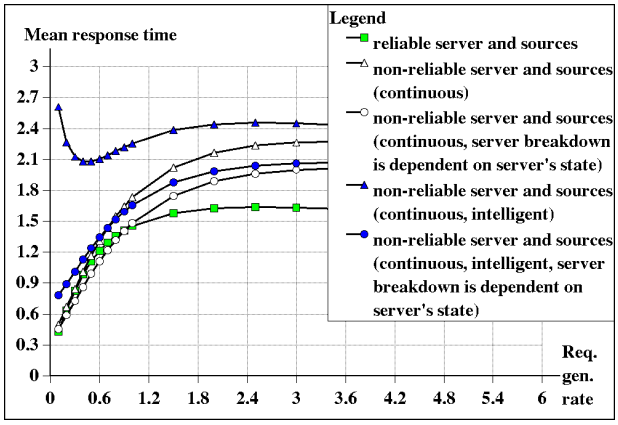


Figure 4.14: $E[T]$ versus primary request generation rate

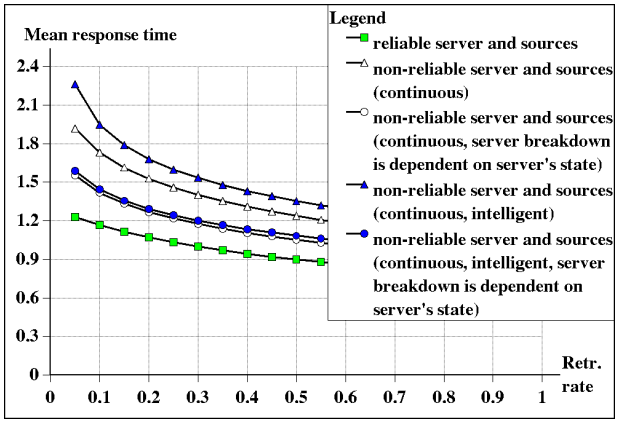


Figure 4.15: $E[T]$ versus retrial rate

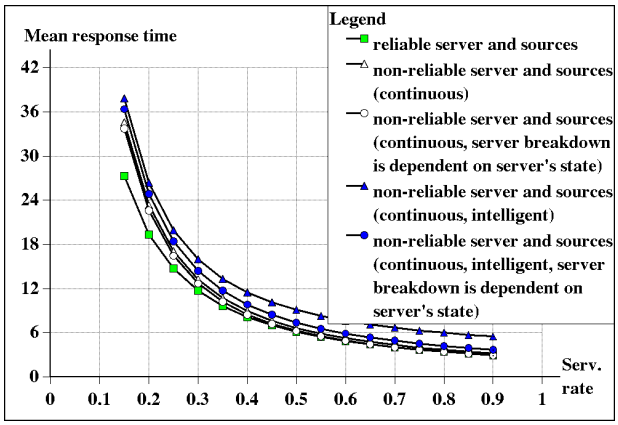


Figure 4.16: $E[T]$ versus service rate

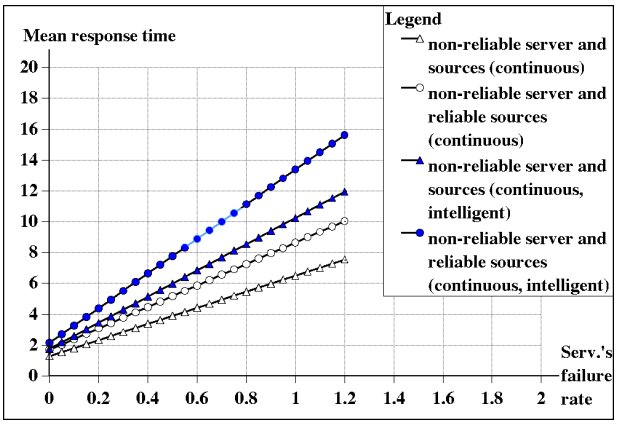


Figure 4.17: $E[T]$ versus server failure rate in busy state

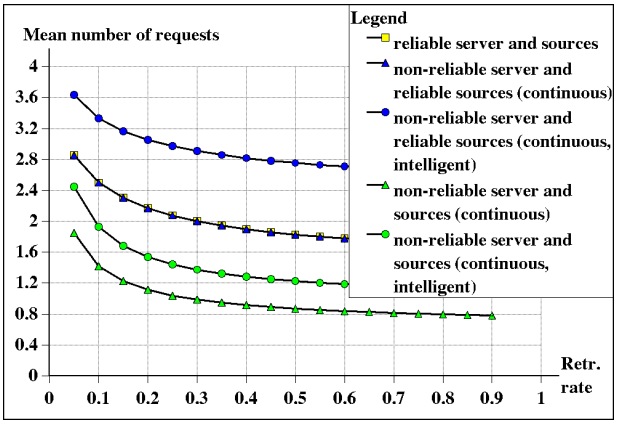


Figure 4.18: Mean number of requests staying in the orbit or in service versus retrial rate

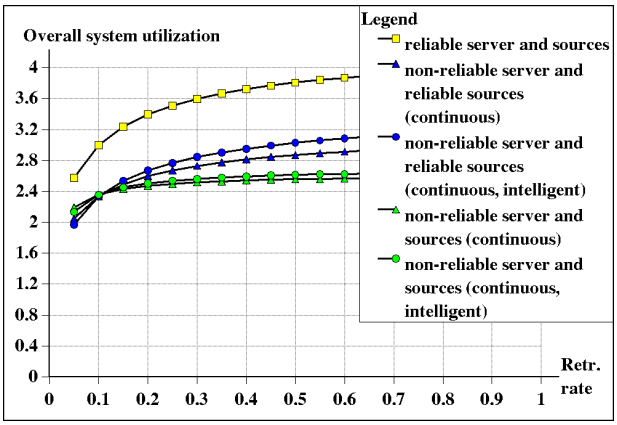


Figure 4.19: Overall system utilization versus retrial rate

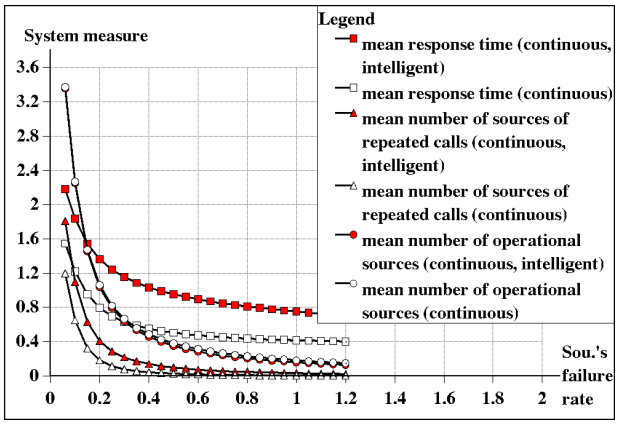


Figure 4.20: System measures versus source failure rate

Comments

- In Figures 4.11–4.13 we can see the mean response time $E[T]$ for the reliable and the non-reliable retrial system with continuous and non-continuous service after repair, with blocked and unblocked operations during server failure when the primary request generation rate, retrial rate and service rate increase. In these cases, the server's failure rate is independent of the state of the server. In Figures 4.14–4.16 the mean response time $E[T]$ is displayed with continuous service after repair but the server's failure rate depends on its state. The difference between continuous, non-continuous service, moreover blocked, unblocked (intelligent) systems' operations is clearly shown. However, if the retrial rate increases the continuous and non-continuous service cases result in the same measure, as it was expected, see Figure 4.12.

- In Figure 4.17 the effect of the server's failure is displayed in the continuous cases. The almost linear increase in $E[T]$ in each case is because the response time is the sum of the down time of the server, the service and repeated call generation time of the request (which do not change during the failure) thus the failure has a linear effect on this measure.

- In Figure 4.18 the effect of the retrial rate on the mean number of requests staying in the orbit or in service is pictured. It is worth pointing out that the values for the reliable case and the blocked case with reliable sources coincide. However, it is not so surprising since during the failure the number of requests remain the same.

- In Figure 4.19 we can see the effect of the retrial rate on the overall system utilization. At the beginning the overall system utilizations are larger for the continuous than the non-continuous cases, then it changes and the difference increases as the retrial rate increases.

- In Figure 4.20 the effect of the sources' failure rate is displayed on the mean response time, on the mean number of sources of repeated calls and on the mean number of operational sources. There is a very slight difference between the continuous and non-continuous cases for the mean number of operational sources, and the mean response

time decreases as the failure rate increases, that is, the mean number of operational sources decreases.

4.3 The $\vec{M}/\vec{M}/1//K$ Model with Non-reliable Server

In this section single server finite-source queueing systems with the following assumptions are investigated. The primary calls are generated by K, $1 < K < \infty$ heterogeneous sources. The server can be in operational or non-operational states, and it can be idle or busy. If it is idle and up, it can serve the calls of the sources. Each of the sources can be in three states: generating a primary call (free), sending repeated calls and under service. The i-th source can generate a primary call during interval $(t, t + dt)$ with probability $\lambda_i dt + o(dt)$. If the server is idle and up at the time of arrival of a call then the call starts to be served immediately, the source moves into the under service state and the server moves into busy state. The service is finished during the interval $(t, t + dt)$ with probability $\mu_i dt + o(dt)$ if the server is available (up). If the server is busy, then the source starts generating a Poisson flow of repeated calls with rate ν_i until it finds the server idle. After service the source can generate a new primary call, and the server becomes idle so it can serve a new call. The server can fail during the interval $(t, t + dt)$ with probability $\delta dt + o(dt)$ if it is idle, and with probability $\gamma dt + o(dt)$ if it is busy. If the server fails in busy state, it either *continues servicing* the interrupted call after it has been repaired or the interrupted request *returns to the orbit*. The repair time is exponentially distributed with a finite mean $1/\tau$. If the server is failed two different cases can be treated. Namely, *blocked sources* case when all the operations are stopped, that is no new primary and repeated calls are generated. In the *unblocked (intelligent) sources* case only service is interrupted but all the other operations are continued (new and repeated calls can be generated). All the times involved in the model are assumed to be mutually independent of each other.

This model generalizes the results of [27] where homogeneous systems with reliable servers were dealt with. Similarly, it is another extension of investigations for heterogeneous finite-source queueing systems without retrials but with server's breakdowns which were treated in [54]. Finally, it can be considered as the natural continuation of [8] in which reliable heterogeneous finite-source retrial systems were analyzed.

4.3.1 The Underlying Markov Chain

Because of the exponentiality of the involved random variables the following process will be a Markov chain. The state of the system at time t can be described by the process

$$X(t) = ((Y(t); \alpha_{C(t)}; \beta_1, ..., \beta_{N(t)}), t \geq 0)$$

where $Y(t) = 0$ if the server is up, $Y(t) = 1$ if the server is down, $C(t) = 0$ if the server is idle, $C(t) = 1$ if the server is busy, and $\alpha_{C(t)}$ is the index of the request under service at time t if the server is busy. Let $N(t)$ be the number of sources of repeated calls at time t, and because of the heterogeneity of the sources we need to identify their indices that are denoted by β_j, $j = 1, ..., N(t)$ if there is a customer in the orbit, otherwise the third component is 0.

Since its state space is finite the process $(X(t), t \geq 0)$ is ergodic for all values of the rate of generation of new primary calls, and from now on we assume that the system is in the steady state.

We define the stationary probabilities

$$P(q;0;0) = \lim_{t\to\infty} P(Y(t) = q; C(t) = 0; N(t) = 0), \qquad q = 0, 1$$

$$P(q;j;0) = \lim_{t\to\infty} P(Y(t) = q; \alpha_1 = j; N(t) = 0),$$
$$q = 0, 1, \qquad j = 1, ..., K,$$

$$P(q;0;i_1, ..., i_k) = \lim_{t\to\infty} P(Y(t) = q; C(t) = 0; \beta_1 = i_1, ..., \beta_k = i_k),$$
$$q = 0, 1, \qquad k = 1, ..., K^*,$$

$$P(q;j;i_1, ..., i_k) = \lim_{t\to\infty} P(Y(t) = q; \alpha_1 = j; \beta_1 = i_1, ..., \beta_k = i_k),$$
$$q = 0, 1, \qquad k = 1, ..., K - 1.$$

where

$$K^* = \begin{cases} K - 1 & \text{for blocked case,} \\ K & \text{for unblocked case.} \end{cases}$$

The traditional way is to derive the related Kolmogorov equations for these probabilities and using the normalizing condition somehow we have to solve the set of equations. In this case it is not so easy, but we perform these two steps by the help of the tool MOSEL.

Once we have obtained these limiting probabilities the main system performance measures can be derived in the following way.

- *Server utilization with respect to source j*

  U_j = P (the server is up and busy with source j)

  that is, we have to summarize all the probabilities where the first component is 0 and the second component is j. Formally,

  $$U_j = \sum_{k=0}^{K-1} \sum_{i_1,...,i_k \neq j} P(0; j; i_1, ..., i_k).$$

  Hence the *server utilization*

  $$U_S = E[Y(\infty) = 0; C(\infty)] = \sum_{j=1}^{K} U_j.$$

- *Utilization of source i*

  $U^{(i)}$ = P (source i generates a new primary call).

  It should be mentioned that in the blocked case the server have to be up, but in the unblocked case the request generation is independent of the server's state.

- *Utilization of the repairman*

$$U_R = E[Y(\infty)] = \sum_{j=0}^{K} \sum_{k=0}^{K^*} \sum_{i_1,...,i_k \neq j} P(1; j; i_1, ..., i_k).$$

- *Availability of the server*

$$A_S = 1 - U_R.$$

Let us denote by $P_O^{(i)}$ the steady state probability that request i is staying in the orbit. It is easy to see that

$$P_O^{(i)} = \sum_{q=0}^{1} \sum_{j=0, j\neq i}^{K} \sum_{k=1}^{K^*} \sum_{i \epsilon (i_1,...,i_k)} P(q; j; i_1, ..., i_k).$$

Similarly, it can easily be seen, that the steady state probability $P_S^{(i)}$ that request i is at the server is given by

$$P_S^{(i)} = \sum_{q=0}^{1} \sum_{k=0}^{K^*} \sum_{i \neq i_1,...,i_k)} P(q; i; i_1, ..., i_k).$$

Hence, the probability $P^{(i)}$ that request i is at the service facility can be obtained by

$$P^{(i)} = P_S^{(i)} + P_O^{(i)}.$$

- *Mean response time of source i*

The derivation of the following formulae are based on [4; 55]. Let us denote by $E[T_i]$ the mean response time of customer i, and by γ_i the *throughput* of request i, that is, the mean number of times that request i is served per unit time. These are related by

$$\gamma_i = \frac{1}{E[T_i] + E[S_i]} = \lambda_i U^{(i)} = \mu_i U_i, \qquad i = 1, ..., K, \tag{4.1}$$

where $E[S_i]$ denotes the mean sojourn time of request i in the source. Since the server is subject to random breakdowns which may stop the operations of the sources, it is clear that $E[S_i] = E[D_i] + 1/\lambda_i \geq 1/\lambda_i$, where $E[D_i]$ denotes the mean delay time due to the failure of the server.

Hence, with the aid of (4.1) for $U^{(i)}$ we get

$$U^{(i)} = \frac{1/\lambda_i}{E[T_i] + E[S_i]} = \frac{\mu_i U_i}{\lambda_i} \leq 1 - P^{(i)}, \qquad i = 1, ..., K,$$

and for $P^{(i)}$ we have

$$P^{(i)} = \frac{E[T_i]}{E[T_i] + E[S_i]} = \gamma_i E[T_i] = \lambda_i U^{(i)} E[T_i], \qquad i = 1, ..., K, \tag{4.2}$$

which represents Little's theorem for request i at the service facility.

By the help of (4.2) we can express the mean response time $E[T_i]$ for request i as

$$E[T_i] = \frac{P^{(i)}}{\lambda_i U^{(i)}} = \frac{P^{(i)}}{\mu_i U_i}, \qquad i = 1, ..., K.$$

- *Mean waiting time of source i*

  The mean waiting time $E[W_i]$ of request i is due to the time spent in the orbit (irrespective of whether the server is up or down), and the delay time because of the server's failure. It is easy to see that $E[W_i]$ is given by

$$E[W_i] = E[T_i] - 1/\mu_i = \frac{P^{(i)} - U_i}{\mu_i U_i}, \qquad i = 1, ..., K.$$

- *Mean number of sources of repeated calls*

$$N = E[N(\infty)] = \sum_{i=1}^{K} P_O^{(i)}.$$

- *Mean number of calls staying at the service facility*

$$M = E[C(\infty) + N(\infty)] = \sum_{i=1}^{K} P^{(i)} = \sum_{i=1}^{K} (P_S^{(i)} + P_O^{(i)}) = \sum_{i=1}^{K} P_S^{(i)} + \sum_{i=1}^{K} P_O^{(i)}.$$

- *Mean rate of generation of primary calls*

$$\overline{\lambda} = \sum_{i=1}^{K} \gamma_i = \sum_{i=1}^{K} \lambda_i U^{(i)} = \sum_{i=1}^{K} \mu_i U_i.$$

- *Blocking probability of primary call i*

$$B_i = \begin{cases} \frac{\lambda_i \sum_{j=1, j\neq i}^{K} \sum_{k=0}^{K-1} \sum_{i\neq i_1,...,i_k} P(0;j;i_1,...,i_k)}{\overline{\lambda}} & \text{for blocked case,} \\ \frac{\lambda_i \sum_{j=1, j\neq i}^{K} \sum_{k=0}^{K-1} \sum_{i\neq i_1,...,i_k} (P(0;j;i_1,...,i_k) + P(1;j;i_1,...,i_k))}{\overline{\lambda}} & \text{for unblocked case.} \end{cases}$$

Hence *blocking probability of primary calls*

$$B = \sum_{i=1}^{K} B_i$$

which is the fraction of primary calls which were blocked (i.e. met the server busy).

It is easy to see that in the case of unblocked operations (intelligent sources) with non-reliable server, $U^{(i)} = 1 - P^{(i)}, i = 1, ..., K$, and we get the same formulae derived in [8] that is, most performance measures can be expressed in the terms of the corresponding utilizations U_i as it was stated in [25].

4.3.2 Validation of Results

In the cases when the server's failure rate is very small and the repair rate is large the non-reliable model should be very close to the reliable system. The results in the homogeneous case were validated by the Pascal program given in [27]. For the heterogeneous case the calculations were checked by the numerical results of [8]. In the case of homogeneous sources but with server's breakdowns the program was tested by the examples of [10].

In Table 4.6 some test results are collected when the server's failure rate is quite small and the requests' retrial rates are quite large. Hopefully the corresponding performance measures should be very close to each other in the case of continued, restarted (abbreviated by orbit) service after repair and FIFO disciplines. As we can see, the results confirm our expectation.

	non-rel. retr. (cont.)	non-rel. retr. (orbit)	non-rel. FIFO
Number of sources:	3	3	3
Request's generation rate:	0.2, 0.3, 0.5	0.2, 0.3, 0.5	0.2, 0.3, 0.5
Service rate:	1, 1.2, 1.1	1, 1.2, 1.1	1, 1.2, 1.1
Retrial rate:	1e+20	1e+20	-
Server's failure rate:	0.002	0.002	0.002
Server's repair rate:	0.04	0.04	0.04
Utilization of the server:	0.5785930082	0.5785934601	0.5785951436
Mean response time			
Source 1:	1.6101659841	1.6102714374	1.6109393482
Source 2:	1.4136508315	1.4135712959	1.4128700761
Source 3:	1.3536212335	1.3536213788	1.3537299921

Table 4.6: Validations

4.3.3 Numerical Examples

In this subsection we consider some sample numerical results to illustrate the influence of the non-reliable server on the mean response time $E[T_i]$ and the mean number of request

M staying at the service facility (in the orbit and at the server). The system input parameters are collected in Table 4.7. In each case independent breakdowns were treated. In the homogeneous cases the parameters are the arithmetic means of the corresponding values.

In Figures 4.21 - 4.26 the mean response time $E[T_i]$ is displayed in continuous and non-continuous service after repair in the case of blocked, unblocked (intelligent) operations, as the function of primary request generation, retrial and service rates, respectively.

In Figures 4.27 - 4.28 also $E[T_i]$ is pictured in the case of blocked operations with continuous and non-continuous service after repair as the function of the server's failure rate in busy state, respectively.

Finally, in Figures 4.29 - 4.30 the mean number of request M staying at the service facility can be seen in reliable and non-reliable case with homogeneous, heterogeneous sources under blocked, unblocked operations combined with continuous and non-continuous service after repair as the function of the retrial rate of repeated calls.

	K	$\lambda_1, ..., \lambda_K$	$\mu_1, ..., \mu_K$	$\nu_1, ..., \nu_K$	δ	γ	τ
Figure 4.21, Figure 4.22	5	x axis	4.1,4.3,4.5, 4.7,4.9	0.35,0.4,0.45, 0.6,0.7	0.05	0.05	0.1
Figure 4.23, Figure 4.24	5	2.5,3,4, 6.5,9	6,7,8, 13,16	x axis	0.05	0.05	0.1
Figure 4.25, Figure 4.26	5	0.04,0.06,0.1, 0.14,0.16	x axis	0.2,0.25,0.3, 0.55,0.7	0.05	0.05	0.1
Figure 4.27, Figure 4.28	5	0.6,0.7,0.8, 0.9,1	4.1,4.3,4.5, 4.7,4.9	0.35,0.4,0.45, 0.6,0.7	0.05	x axis	0.1
Figure 4.29	5	0.1	0.5	x axis	0.05	0.05	0.1
Figure 4.30	5	0.06,0.08,0.1, 0.12,0.14	0.3,0.4,0.5, 0.6,0.7	x axis	0.05	0.05	0.1

Table 4.7: System input parameters

Comments

- In Figures 4.21-4.22 we can see the mean response time $E[T_i]$ for the non-reliable system with continuous, non-continuous service after repair, with blocked and unblocked operations during service failure when the primary request generation rate increases. The difference between continuous, non-continuous service, moreover blocked, unblocked (intelligent) systems' operations is clearly shown. The non-continuous case always have longer response times. Similarly, the intelligent sources suffer from longer times, too. However, their curves are very interesting, for some sources at the beginning decrease, then increase, finally decrease again. This shows that the systems are very complex and at different parameter setups we can see different effects.

- In Figures 4.23,4.24, and similarly in Figures 4.25, 4.26 the effect of retrial rate, service rate is demonstrated on $E[T_i]$, respectively. Again the non-continuous cases always have longer response times, and the intelligent sources suffers from longer times, too. However, the difference between the continuous and non-continuous case decreases

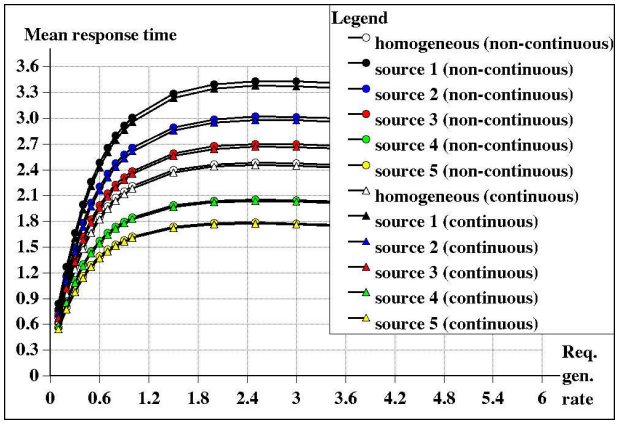


Figure 4.21: $E[T]$ versus primary request generation rate

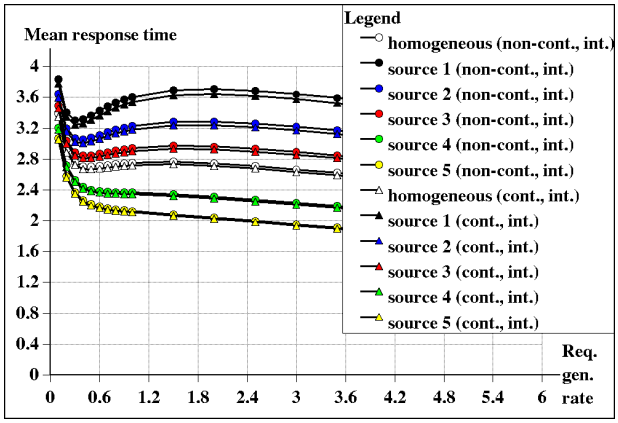


Figure 4.22: $E[T]$ versus primary request generation rate

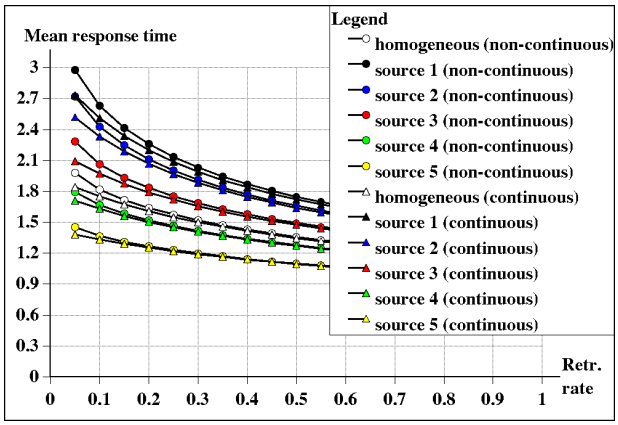


Figure 4.23: $E[T]$ versus retrial rate

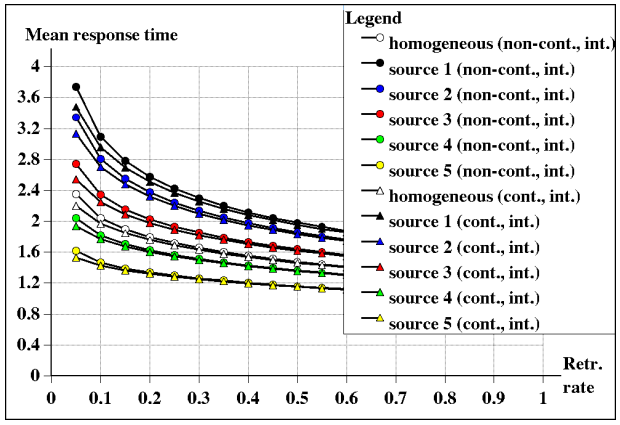


Figure 4.24: $E[T]$ versus retrial rate

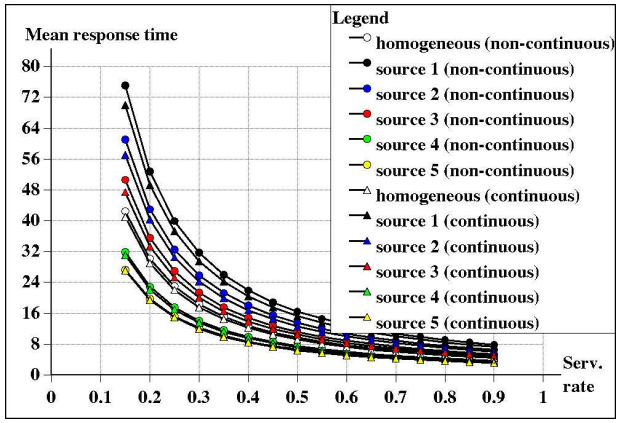


Figure 4.25: $E[T]$ versus service rate

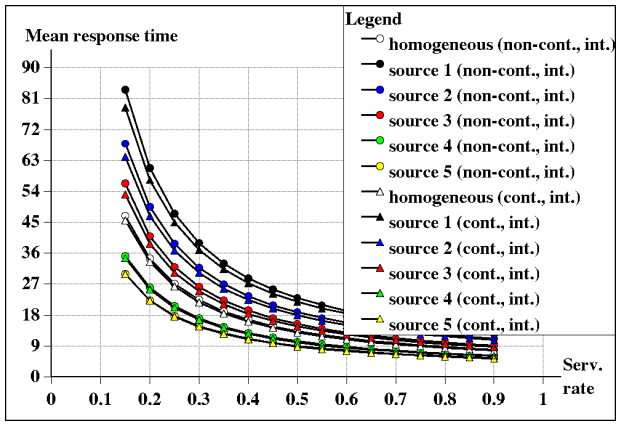


Figure 4.26: $E[T]$ versus service rate

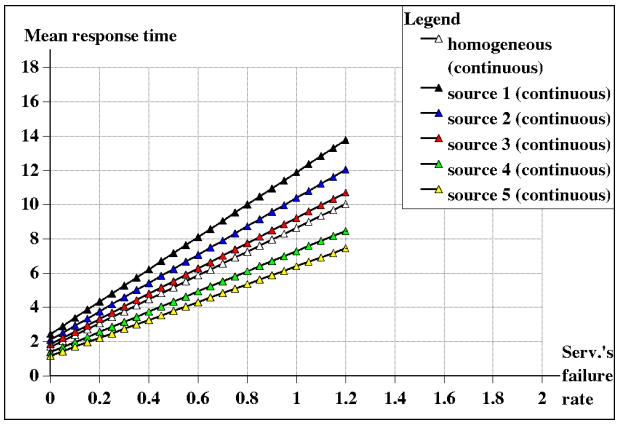


Figure 4.27: $E[T]$ versus server failure rate in busy state

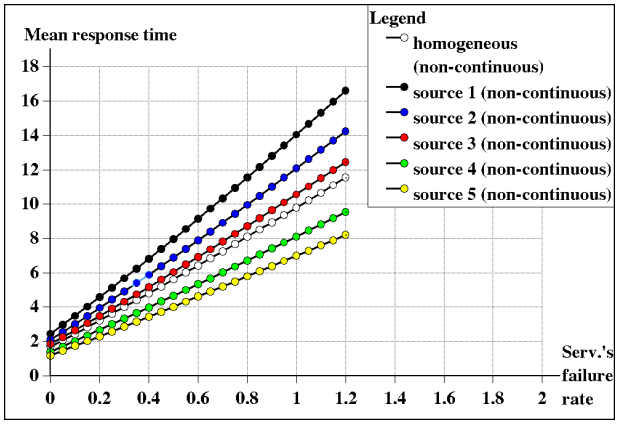


Figure 4.28: $E[T]$ versus server failure rate in busy state

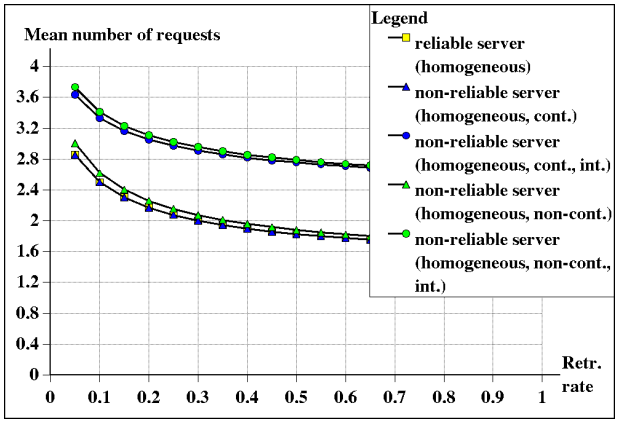


Figure 4.29: M versus retrial rate

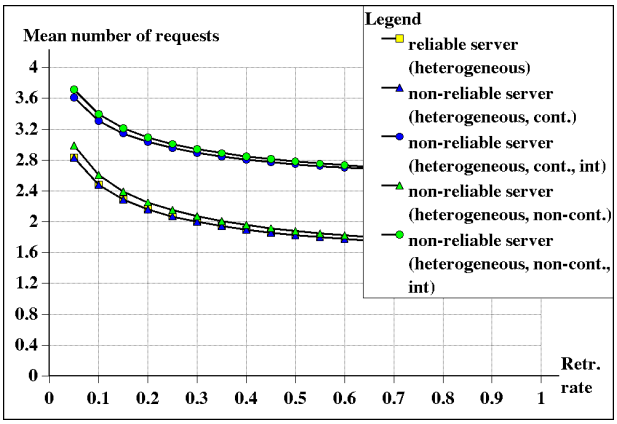


Figure 4.30: M versus retrial rate

as the corresponding rates increase. Furthermore, in each case $E[T_i]$ decreases as we expected.

- In Figures 4.27, 4.28 the effect of server's failure is displayed in the blocked operations case. Again the non-continuous case always has longer response times, which is clear. Moreover, for reliable busy server we have the same results, which was expected, too.
- In Figures 4.29, 4.30 the effect of the retrial rate on the mean number of request M staying at the service facility is pictured. The curves confirm our expectation with respect to the service after repair and blocked operations during the failure. M decreases as the retrial rate increases, which is clear. It is also worth pointing out that the values for the reliable case and non-reliable blocked case coincide.

Chapter 5

Analysis of Multiserver Non-reliable Finite-source Retrial Queues

In this chapter multiserver non-reliable finite-source retrial queues are investigated. The purpose is to generalize the models of [27] and [10]. The novelty of this investigation is the different service rates and different service policies with the non-reliability of the servers.

Section 5.1 is devoted to the extension of the models described in [27] and [10]. The finite-source retrial queue is analyzed with non-reliable heterogeneous (asymmetric) servers, that is the servers have different parameters in service, failure and repair rates. In Section 5.2 two service policies are compared in this model. These results will be published in [J5; J6].

5.1 The $M/\vec{M}/c//K$ Model with Non-reliable Servers

Consider a finite-source retrial queueing system with c servers, where the primary calls are generated by K, $c < K < \infty$ sources. Each server can be in operational (up) or non-operational (down) states, and it can be idle or busy. Each source can be in three states: generating a primary call (free), sending repeated calls and under service by one of the servers. If a source is free at time t, it can generate a primary call during the interval $(t, t + dt)$ with probability $\lambda dt + o(dt)$. If one of the servers is up and idle at the moment of the arrival of the call then the service of the call starts. At the arrival of the calls the availability and idleness of the servers are always examined according to the increasing order of the servers' indices, resulting different load to the servers. The service is finished during the interval $(t, t + dt)$ with probability $\mu_i dt + o(dt)$ if the ith server is available.

Server i can fail during the interval $(t, t+dt)$ with probability $\delta_i dt+o(dt)$ if it is idle, and with probability $\gamma_i dt + o(dt)$ if it is busy. If the server fails in busy state, it either continues servicing the interrupted call after it has been repaired or the interrupted request returns to the orbit. In this section we only investigate the case when the source moves into the sending repeated calls state at the moment of server's failure. The repairman follows FIFO discipline for the servers' breakdowns, and the repair time of the ith server is exponentially distributed with a finite mean $1/\tau_i$. If all the servers fail we treat two different cases. Namely, *blocked sources* case when all the operations are stopped expect from the repair of the servers. In the *unblocked (intelligent) sources* case only service is interrupted but all the other operations are continued.

If all the servers are busy (or failed in the unblocked case) at the moment of the arrival of a call the source starts generation of a Poisson flow of repeated calls with rate ν until it finds an available free server. After service the source becomes free, and it can generate a new primary call, and the server becomes idle so it can serve a new call. All the times involved in the model are assumed to be mutually independent of each other.

5.1.1 The Underlying Markov Chain

The state of the system at time t can be described by the process $X(t) = (\alpha_1(t), ..., \alpha_c(t); N(t))$, where $N(t)$ is the number of sources of repeated calls, $\alpha_i(t)$, i=1, ..., c, denotes the state of the ith server at time t. If there is a customer under service at the ith server, $\alpha_i(t) = 1$, if it is operational and idle then $\alpha_i(t) = 0$, otherwise the server is failed and $\alpha_i(t) = -1$.

Because of the exponentiality of the involved random variables this process is a Markov chain with a finite state space. Since the state space of the process $(X(t), t \geq 0)$ is finite, the process is ergodic for all reasonable values of the rates involved in the model construction. From now on we assume that the system is in the steady state. Let us define the stationary probabilities by:

$$P(i_1, ..., i_c, j) = \lim_{t\to\infty} P\{\alpha_1(t) = i_1, ..., \alpha_c(t) = i_c, N(t) = j\},$$

$$i_1, ...i_c = -1, 0, 1; j = 0, ..., K^* \quad \text{where} \quad K^* = K - \sum_{i_k, i_k=1} i_k.$$

Furthermore, let us denote by $C(t)$ the number of busy servers, by $A(t)$ the number of available servers at time t, and denote by $p_{kj} = \lim_{t\to\infty} P\{C(t) = k, N(t) = j\}$ the joint

distribution of the number of busy servers and the number of repeated calls.

Once we have obtained the above defined probabilities the main steady state system performance measures can be derived as follows:

- *Probability that at least one server is available*

$$A_S = P\{\alpha_k > -1\}, k \in \{1, ..., c\} = 1 - \sum_{j=0}^{K} P(-1, ..., -1, j).$$

- *Mean number of sources of repeated calls*

$$N = E[N(\infty)] = \sum_{k=0}^{c} \sum_{j=1}^{K} j p_{kj} = \sum_{i_1,...,i_c} \sum_{j=1}^{K^*} j P(i_1, ..., i_c, j).$$

- *Utilization of the k-th server*

$$U_k = \sum_{i_1,...,i_c,i_k=1} \sum_{j=0}^{K^*} P(i_1, ..., i_c, j), \quad k = 1, ..., c.$$

- *Mean number of busy servers*

$$C = E[C(\infty)] = \sum_{\substack{i_1,...,i_c \\ K^*>0}} \sum_{j=0}^{K^*} K^* P(i_1, ..., i_c, j) = \sum_{k=1}^{c} U_k.$$

- *Mean number of calls staying in the orbit or in service*

$$M = E[N(\infty) + C(\infty)] = N + C.$$

- *Utilization of the repairman*

$$U_R = \sum_{\substack{i_1,...,i_c \\ -1 \in \{i_1,...,i_c\}}} \sum_{j=0}^{K^*} P(i_1, ..., i_c, j).$$

- *Utilization of the sources*

$$U_{SO} = \begin{cases} \frac{E[K - C(\infty) - N(\infty); A(\infty) > 0]}{K} & \text{for blocked case,} \\ \frac{E[K - C(\infty) - N(\infty)]}{K} & \text{for unblocked case.} \end{cases}$$

- *Overall utilization of the system*

$$U_O = C + KU_{SO} + U_R.$$

- *Mean rate of generation of primary calls*

$$\overline{\lambda} = \begin{cases} \lambda E[K - C(\infty) - N(\infty); A(\infty) > 0] & \text{for blocked case,} \\ \lambda E[K - C(\infty) - N(\infty)] & \text{for unblocked case.} \end{cases}$$

- *Mean waiting time*

$$E[W] = N/\overline{\lambda}.$$

- *Mean response time*

$$E[T] = M/\overline{\lambda}.$$

5.1.2 Validation of Results

The results in the reliable case were validated by the Pascal program given in [27]. In Table 5.1 we can see that the corresponding performance measures are very close to each other, they are the same at least up to the 8th decimal digit. In the case of non-reliable servers, the results were tested by the $M/M/1//K$ retrial model with server's breakdowns which was studied in [10].

	MOSEL	Pascal program
Number of servers:	2	2
Number of sources:	5	5
Request's generation rate:	0.1	0.1
Service rate:	1	1
Retrial rate:	1.1	1.1
Servers' failure rate:	1e-25	–
Servers' repair rate:	1e+25	–
Mean waiting time:	0.0653833701	0.0653833729
Mean number of busy servers:	0.4518596260	0.4518596267
Mean number of sources of repeated calls:	0.0295441060	0.0295441065

Table 5.1: Validations

5.1.3 Numerical Examples

In this subsection some numerical results are considered in the case of homogeneous servers to illustrate graphically the influence of the non-reliable servers on the mean response time $E[T]$ and on the overall system's utilization U_O. The system input parameters of the figures are collected in Table 5.2.

The tool SPNP was used with MOSEL which was able to handle the model with up to 126 sources. In this case, on a PC containing a 1.1 GHz processor and 512 MB RAM, the running time with one parameter setup with 2 servers was approximately 1 second. With 4 servers and 126 sources, in the blocked case, it was 2 minutes and 25 seconds. The maximum number

	c	K	λ	μ	ν	δ,γ	τ
Figure 5.1	2	5	x axis	1	1.1	0.001	0.01
Figure 5.2	2	5	0.2	1	x axis	0.001	0.01
Figure 5.3	2	5	0.2	x axis	1.1	0.001	0.01
Figure 5.4, 5.5	2	5	0.2	1	1.1	x axis	0.01

Table 5.2: System input parameters

of servers that the program was able to calculate the system measures on this computer in an acceptable time was 6. With 6 servers and 10 sources, the program finished its run after 20 minutes, and with 20 sources after 1 hour and 15 minutes.

In Figures 5.1-5.4 the effects of the primary request generation rate, retrial rate, service rate and servers' failure rate on the mean response time are displayed. In Figure 5.5 we can see the effect of servers' failure rate on the overall utilization. In each Figure, the reliable case, the blocked and unblocked (intelligent) cases are illustrated.

Comments

- In Figure 5.1 we can see that with these parameter setup the difference is very small between the non-intelligent and intelligent cases. The interesting phenomenon, which was mentioned in [25], too, that is retrial queues have a maximum of E[T] is also noticed.
- In Figure 5.2 it is demonstrated how long the retrial rate has a significant influence on the mean response time.
- In Figure 5.3 we can see that the increase of the service rate has almost the same influence on the reliable and on the non-reliable systems.
- In Figure 5.4 it can be observed that the increase of the servers' failure rate can have a heavy impact on the mean response time, and as it increases the difference between the two non-reliable models increases significantly.
- In Figure 5.5 it is shown that the overall utilization can be very low if the servers' failure rate increases and the repair rate is not high enough.

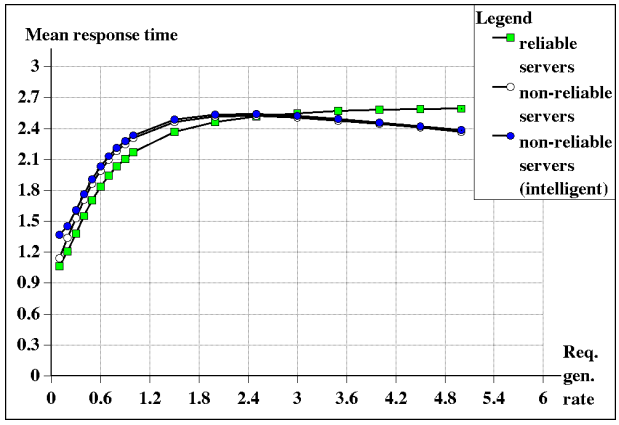


Figure 5.1: $E[T]$ versus primary request generation rate

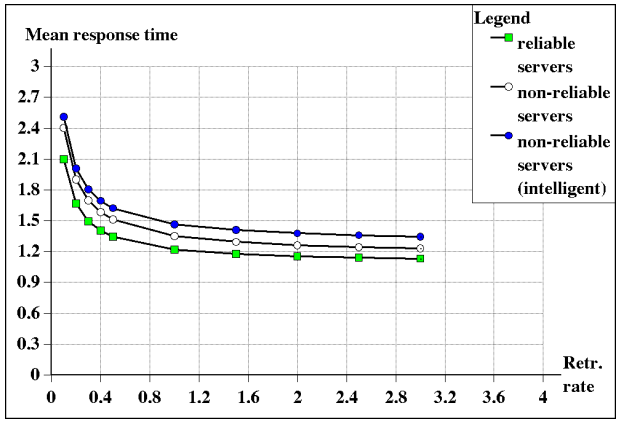


Figure 5.2: $E[T]$ versus retrial rate

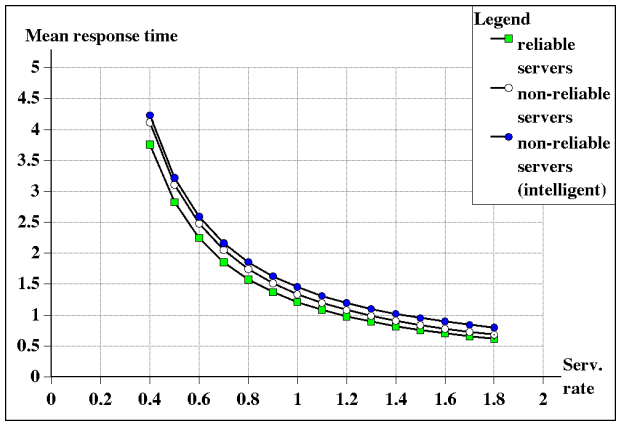


Figure 5.3: $E[T]$ versus service rate

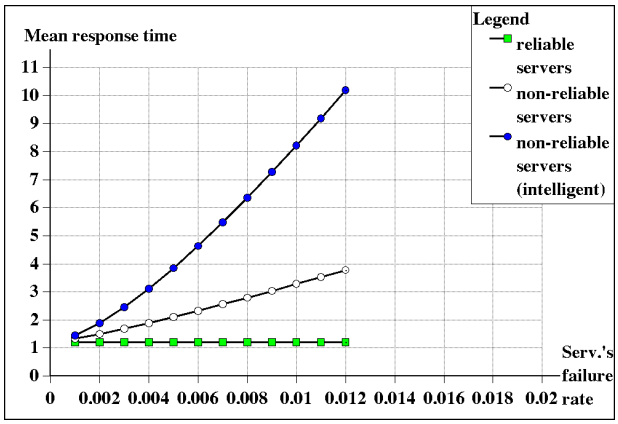


Figure 5.4: $E[T]$ versus servers' failure rate

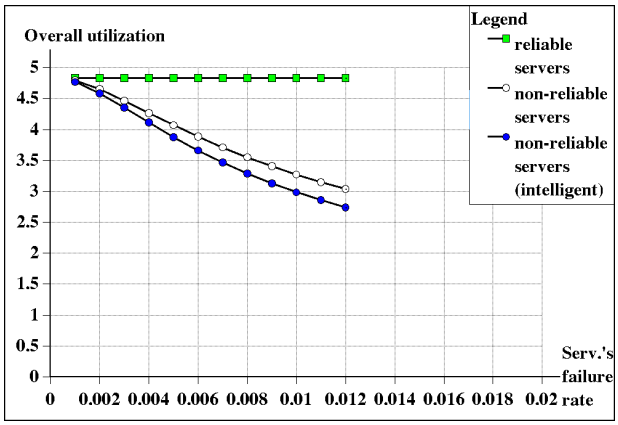


Figure 5.5: U_O versus servers' failure rate

5.2 Comparison of Different Service Policies

In this section we analyze the finite-source retrial queue with non-reliable heterogeneous (asymmetric) servers from Section 5.1. In this study, the most important heterogeneous characteristic is the service rate, since we compare two service policies, namely Random and Fastest Free Server (FFS). In the case of Random service discipline, the requests are assigned to the idle servers randomly, and in the other case, the requests are assigned to the fastest available free server, i.e. the Fastest Free Server case the availability and idleness of the servers are always examined according to the increasing order of the servers indices.

In this study, we compare the service disciplines in the unblocked sources case with independent server breakdowns.

5.2.1 Validation of Results

The results of the tool in the reliable case were validated by the Pascal program given in [27]. The service rates are the same for all servers in each case. In Table 5.3 we can see that the corresponding performance measures are very close to the reliable case and to each other with Random and Fastest Free Server (FFS) disciplines with very low failure and very high repair rates. The results are the same up to the 6th decimal digit.

In the non-reliable single server case, the results were tested by the $M/M/1//K$ retrial model with server breakdowns which was studied in [10].

	Pascal [27]	Random	FFS
Number of servers:	4	4	4
Number of sources:	20	20	20
Request's generation rate:	0.1	0.1	0.1
Service rate:	1	1	1
Retrial rate:	1.2	1.2	1.2
Servers' failure rate:	–	1e-25	1e-25
Servers' repair rate:	–	1e+25	1e+25
Mean waiting time:	0.1064954794	0.1064959317	0.1064959929
Mean number of busy servers:	1.8007480431	1.8007485102	1.8007485548
Mean no. sources of repeated calls:	0.1917715262	0.1917717923	0.1917718470

Table 5.3: Validations in the reliable case

5.2.2 Numerical Examples

In this subsection some numerical results are presented to illustrate graphically the differences between the service disciplines in the mean response time, in the utilizations of the servers and in the overall system's utilization. In the legends of the figures, the Fastest Free Server policy is referenced as *ordered*, and the random case where the service rate of the servers is the average of the rates of the heterogeneous cases is referred to as *averaged random*.

The system input parameters of the figures are collected in Table 5.4.

	c	K	λ	$\mu_1, ..., \mu_c - \mu_{avg}$	ν	δ,γ	τ
Figure 5.6, 5.10	4	20	x axis	8,5,4,1 – 4.5	4	0.01	0.2
Figure 5.7, 5.11	4	20	4	8,5,4,1 – 4.5	x axis	0.01	0.2
Figure 5.8, 5.9, 5.12	4	20	1	8,5,4,1 – 4.5	4	x axis	0.2

Table 5.4: System input parameters

Comments

- In Figure 5.6 we can see the difference between the three cases in the mean response time depending on the primary request generation rate. The difference between the two random cases is not too significant, but the Fastest Free Server (ordered) case always has better response time, especially when more and more requests arrive.
- In Figure 5.7 it is demonstrated how long the retrial rate has a significant influence on the mean response time, after that the decrease is not considerable.
- In Figure 5.8 it is shown how the increase of the servers' failure rate affects the mean response time. The averaged random case has a little better response time than the not averaged random case like in the former figures. The surprising decrease in the mean response time of the Fastest Free Server case can be explained by the help of Figure 5.9.
- In Figure 5.9 we can see the server utilizations versus the servers' failure rate with the same parameter setup as in Figure 5.8. In the random case, the slowest server has the highest utilization and the fastest has the lowest, since it services the request much faster and the requests are assigned to the available and free servers with the same probability. In the beginning of the ordered case, the slowest server has the highest utilization too, but as it fails more often, its service is interrupted more often and looses from its utilization much faster than the faster servers, since it gets requests to serve only if all the other servers are busy or failed.
- In Figure 5.10 the overall utilizations of the systems are displayed versus the primary request generation rate. We can see that the random cases have almost the same while the ordered case has higher utilization.
- In Figure 5.11 we can see the overall utilization versus the retrial rate. Similarly to Figure 5.7, after a time the increase of the retrial rate does not affect this measure significantly.
- In Figure 5.12 the overall utilization is displayed versus the servers' failure rate. Like in Figure 5.8 the mean response time, the overall utilization is getting better for a while in the FFS case as the servers' failure rate increases.

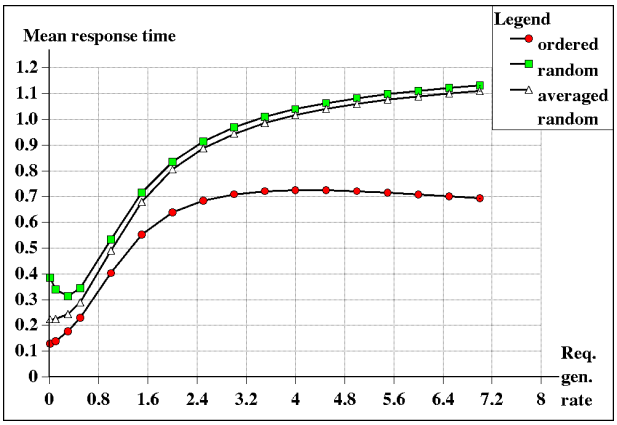


Figure 5.6: Mean response time versus primary request generation rate

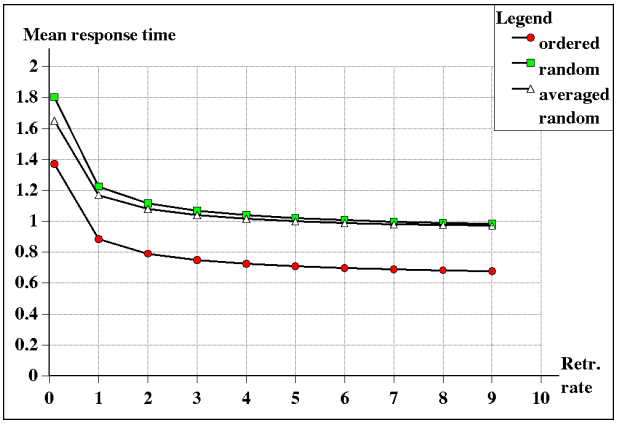


Figure 5.7: Mean response time versus retrial rate

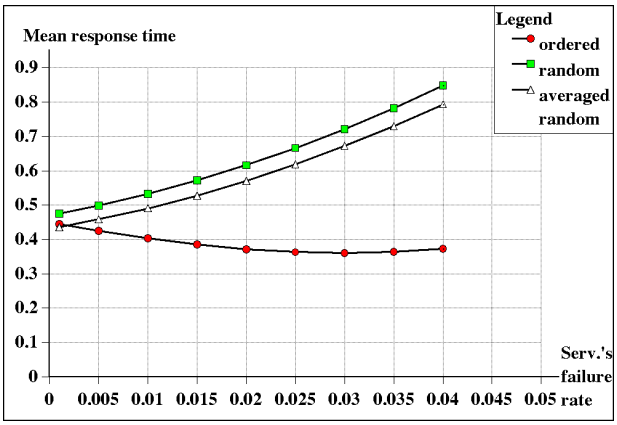


Figure 5.8: Mean response time versus servers' failure rate

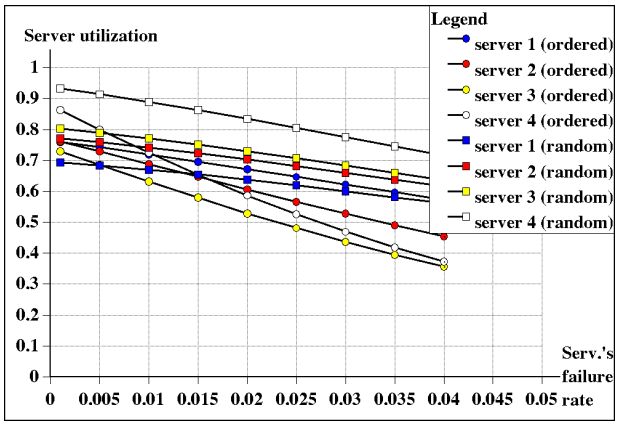


Figure 5.9: Server utilization versus servers' failure rate

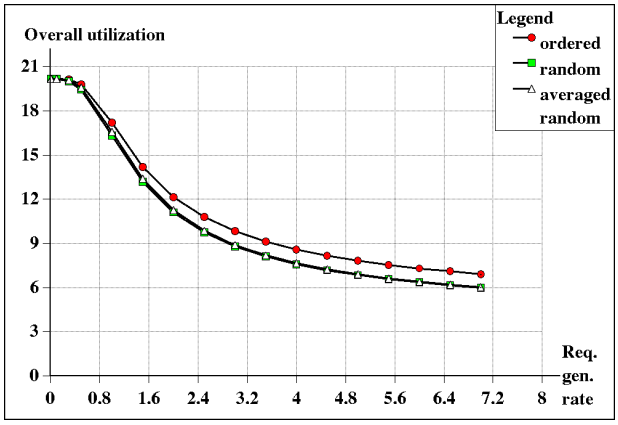


Figure 5.10: Overall utilization versus primary request generation rate

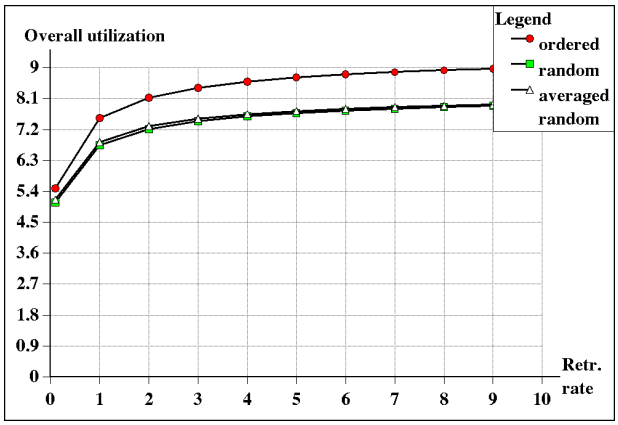


Figure 5.11: Overall utilization versus retrial rate

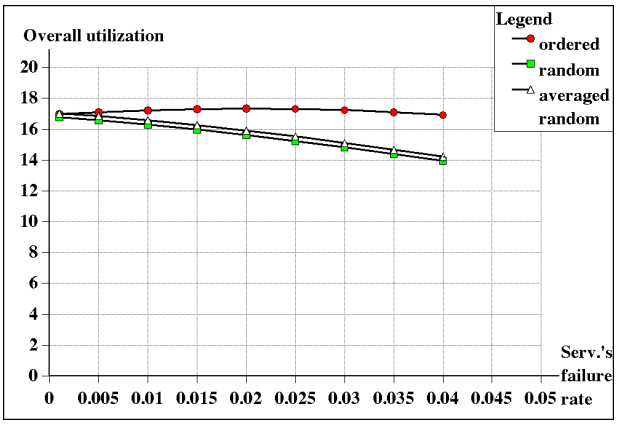


Figure 5.12: Overall utilization versus servers' failure rate

Chapter 6

Retrial Queues in Random Environments

Finite-source queueing systems operating in random environments, sometimes called Markov-modulated queues have been the interest of recent research, see for example [6; 7; 28].

This chapter deals with a finite-source retrial queueing system with heterogeneous sources operating in random environment, that is, the system parameters are subject to randomly occurring fluctuations. Furthermore, it is shown how this type of queueing model can be applied in the analysis of non-reliable retrial systems. Similar models without repeated attempts were treated in [6; 7].

This model was published in [J7].

6.1 The $\vec{M}/\vec{M}/1//K$ Model in Random Environment

Consider a finite-source queue with K sources and a single server, where each source has different parameters and the operation of the sources and the server is influenced by the state of a given random environment.

The server and the sources are collected into M independent groups ($1 \leq M \leq K+1$). The members of a group operate in a common random environment. The environmental changes are reflected in the values of the new and repeated call generation and in the values of the service rates. The members of group m are assumed to operate in a random environment governed by an ergodic Markov chain $(\xi_m(t); t \geq 0)$ with state space $(1, ..., r_m)$ and with transition density matrix

$$\left(\tau_{i_m j_m}^{(m)}\right), \quad i_m, j_m = 1, ..., r_m, \quad \text{where } \tau_{i_m i_m}^{(m)} = -\sum_{k \neq i_m} \tau_{i_m k}^{(m)}.$$

The server can be in two states: idle and busy, and each of the sources can be in free, sending repeated calls and under service states. If source i (which is a member of group m) is free at time t and the environmental process $\xi_m(t)$ is in state j_m the probability that this source generates a new request during the time interval $(t, t+dt)$ is $\lambda_i(j_m)dt + o(dt)$, m=$1, ..., M$. If the server is idle at the time of arrival of a call then the call starts to be served, that is the source moves into the under service state and the server moves into the busy state. Assuming that the server belongs to group 1 and the environmental process $\xi_1(t)$ is in state j_1 the probability that the service of the request originating from client i is completed in time interval $(t, t+dt)$ is $\mu_i(j_1)dt + o(dt)$. If the server is busy on arrival, then the source starts generation of a Poisson flow of repeated calls with rate $\nu_i(j_m)$ until it finds the server free. After service the source becomes free, and it can generate a new primary call, and the server becomes idle and it can serve a new call. All random variables and the random environments are supposed to be independent of each other.

6.1.1 The Underlying Markov Chain

Because of the exponentiality of the involved random variables the following process will be a Markov chain. The state of the system at time t can be described by the process

$$X(t) = (\xi_1(t), ..., \xi_M(t), \alpha(t), \beta_1(t), ..., \beta_{N(t)}(t)),$$

where $\xi_m(t)$ denotes the states of the background processes (m=$1, ..., M$), and $N(t)$ is the number of sources of repeated calls at time t. The index of the source at the server is denoted by $\alpha(t)$, if there is a customer under service, otherwise this value is 0. Because of the heterogeneity of the sources we need to identify the sources in the sending repeated calls state, so we denote their indices by $\beta_k(t)$, k=$1, ..., N(t)$, if there is a customer in this state, otherwise this last component is 0.

Since its state space is finite the process $(X(t), t \geq 0)$ is ergodic with the following steady state probabilities.

$$P(j_1, ..., j_M, j, 0) = \lim_{t \to \infty} \mathrm{P}\{\xi_1(t) = j_1, ..., \xi_M(t) = j_M, \alpha(t) = j, N(t) = 0\}$$

$$P(j_1,...,j_M,j,i_1,...,i_k) =$$
$$\lim_{t\to\infty} \mathrm{P}\{\xi_1(t)=j_1,...,\xi_M(t)=j_M,\alpha(t)=j,\beta_1(t)=i_1,...,\beta_k(t)=i_k\},\ \ k=1,...,K-1.$$

Based on the steady state probabilities the system performance measures can be obtained as:

- *Utilization of the server with respect to source i*

$$U_{Si} = \sum_{j_1,...,j_M} P(j_1,...,j_M,i,0) + \sum_{j_1,...,j_M}\sum_{k=1}^{K-1}\sum_{i_1,...,i_k\neq i} P(j_1,...,j_M,i,i_1,...,i_k),$$
$$i=1,...,K.$$

- *Utilization of the server*

$$U_S = \sum_{i=1}^{K} U_{Si}.$$

- *Probability of source i is sending repeated calls*

$$N_i = \sum_{j_1,...,j_M}\sum_{j=1}^{K}\sum_{k=1}^{K-1}\sum_{\substack{i_1,...,i_k\neq j\\ i\in\{i_1,...,i_k\}}} P(j_1,...,j_M,j,i_1,...,i_k), \quad i=1,...,K.$$

- *Mean number of repeated calls*

$$N = \sum_{i=1}^{K} N_i.$$

- *Utilization of source i*

$$U_i = 1 - U_{Si} - N_i, \quad i=1,...,K.$$

- *Probability of source i is free and its background process is in state j_l*

$$F_i(j_l) = \sum_{\substack{p_1,...,p_M\\ p_l=j_l}}\sum_{\substack{j=1\\ j\neq i}}^{K}\sum_{k=1}^{K-1}\sum_{\substack{i_1,...,i_k\neq j\\ i\notin\{i_1,...,i_k\}}} P(p_1,...,p_M,j,i_1,...,i_k), \quad i=1,...,K.$$

- *Throughput of source i*

$$\gamma_i = \sum_{j_l=1}^{r_l} F_i(j_l)\lambda_i(j_l), \quad i=1,...,K.$$

- *Mean response time of source i*

$$E[T_i] = \frac{1-U_i}{\gamma_i}, \quad i=1,...,K.$$

6.1.2 Validation of Results

The calculated performance measures were validated by the results of [6], where an FCFS (First-Come, First-Served) queueing model is studied. Some performance measures are collected in Table 6.1 and can be compared. We can see that we get back the results of the corresponding queueing model with waiting line, since with very high retrial rates and few sources the difference between the two models is negligible.

The numerical calculations were checked by the results of the retrial model with a non-reliable server [52], too. The model was used in which the sources are blocked if the server is not operational, and the server continues servicing the interrupted call after it has been repaired. In Table 6.2 we can see that the results are the same. It can easily be seen that a non-reliable model can be considered as a system modulated by a 2-state background process. The system failure can be modelled by setting the rates to 10^{-20} in the second state of the background process.

	FCFS [6]	retrial
Number of sources:	5	5
Request's generation rate:	0.12, 0.06	0.12, 0.06
Service rate:	1, 1	1, 1
Retrial rate:	–	1e+20
Rate of environment's change:	0.5, 1	0.5, 1
Server queue length:	0.6418	–
Requests in the orbit or in service:	–	0.6418567007
Utilization of the server:	0.4342	0.4342057023
Utilization of the sources:	0.8716	0.8716286599
Mean response time:	1.4782	1.4782319316

Table 6.1: Validation by the FCFS model

6.1.3 Numerical Examples

In this subsection some graphically displayed numerical examples are presented. For the easier understanding only simple cases are considered. Only one random environment was used with 2 states. The tool is able to deal with systems with several environments. The system parameters for the figures are given in Table 6.3.

Comments

- In Figures 6.1 and 6.2 the mean response time is displayed as the primary request generation increases. The difference is that in Figure 6.1 all operations are stopped if the background process is in the second state, but in Figure 6.2, only service is interrupted. The results are in agreement with the results of [52], where the same parameters were used.
- In Figures 6.1 and 6.2 we can analyze the same curves as in Figures 6.1 and 6.2 with modified input parameters. We can see the effect of the random environment and the differences between the original and new parameter setups.

	non-reliable retrial [52]	retrial in random env.
Number of sources:	5	5
Request's generation rate:	0.10, 0.15, 0.17, 0.19, 0.21	0.10, 0.15, 0.17, 0.19, 0.21
Service rate:	1.0, 1.1, 1.2, 1.5, 1.6	1.0, 1.1, 1.2, 1.5, 1.6
Retrial rate:	0.15, 0.18, 0.21, 0.22, 0.25	0.15, 0.18, 0.21, 0.22, 0.25
Server's failure/repair rate:	0.1, 1	-
Rate of env. change:	-	0.1, 1
Utilization of the server:	0.404265368271	0.404265368271
Utilization of the sources		
Source 1:	0.673069675362	0.673069675362
Source 2:	0.629766856845	0.629766856845
Source 3:	0.634124904383	0.634124904383
Source 4:	0.622974780687	0.622974780687
Source 5:	0.627325387151	0.627325387151
Mean response time		
Source 1:	5.34303316033	5.34303316033
Source 2:	4.31118758383	4.31118758383
Source 3:	3.73337662001	3.73337662001
Source 4:	3.50379767493	3.50379767493
Source 5:	3.11179039958	3.11179039958

Table 6.2: Validation by the non-reliable retrial model

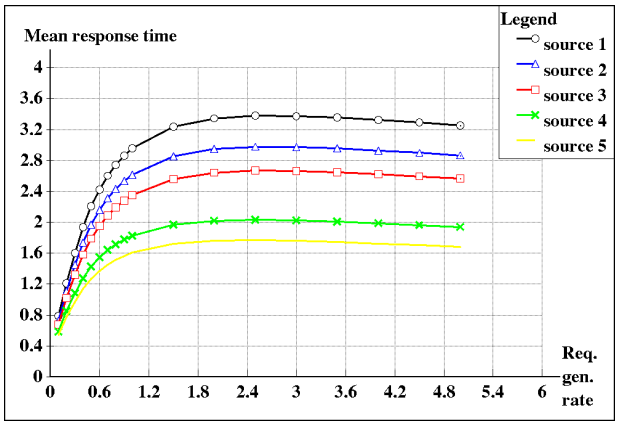


Figure 6.1: Mean response time versus primary request generation rate

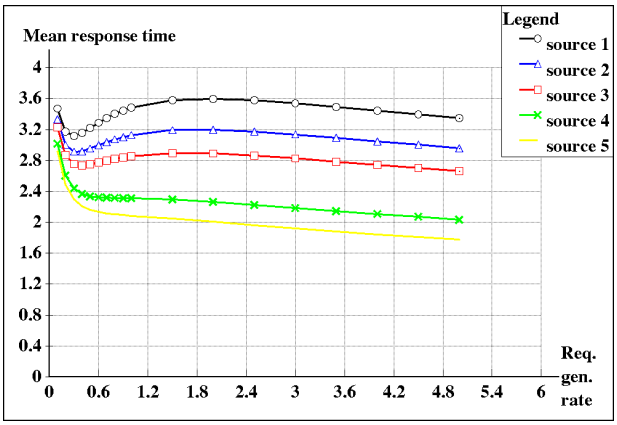


Figure 6.2: Mean response time versus primary request generation rate

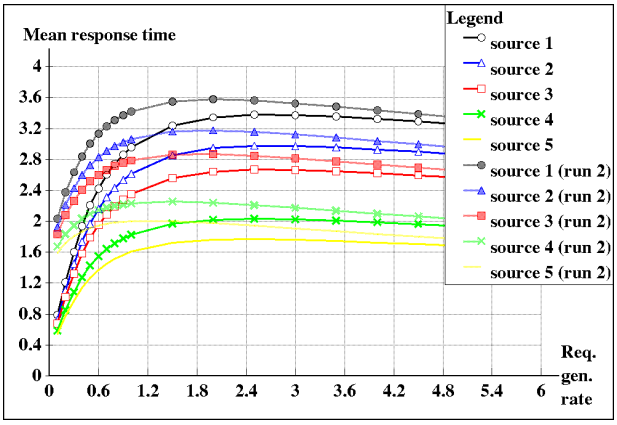


Figure 6.3: Mean response time versus primary request generation rate

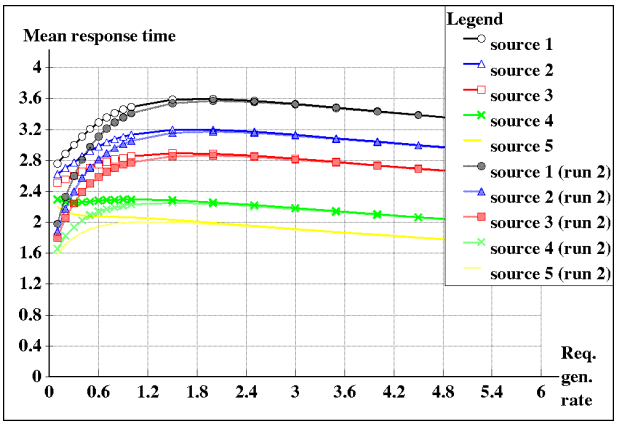


Figure 6.4: Mean response time versus primary request generation rate

	K	$\lambda_1(1)...\lambda_5(1)$ $\lambda_1(2)...\lambda_5(2)$	$\mu_1(1)...\mu_5(1)$ $\mu_1(2)...\mu_5(2)$	$\nu_1(1)...\nu_5(1)$ $\nu_1(2)...\nu_5(2)$	$\tau_{12}^{(1)}$	$\tau_{21}^{(1)}$
Fig. 6.1	5	x axis 1e-20	4.1,4.3,4.5,4.7,4.9 1e-20	0.35,0.4,0.45,0.6,0.7 1e-20	0.05	0.1
Fig. 6.2	5	x axis x axis	4.1,4.3,4.5,4.7,4.9 1e-20	0.35,0.4,0.45,0.6,0.7 0.35,0.4,0.45,0.6,0.7	0.05	0.1
Fig. 6.3	5	x axis 1e-20	4.1,4.3,4.5,4.7,4.9 1e-20	0.35,0.4,0.45,0.6,0.7 1e-20	0.1	0.2
Fig. 6.3 (run 2)	5	x axis $\frac{\lambda_1(1)}{2}...\frac{\lambda_5(1)}{2}$	4.1,4.3,4.5,4.7,4.9 1e-20	0.35,0.4,0.45,0.6,0.7 1e-20	0.1	0.2
Fig. 6.4	5	x axis x axis	4.1,4.3,4.5,4.7,4.9 1e-20	0.35,0.4,0.45,0.6,0.7 0.35,0.4,0.45,0.6,0.7	0.1	0.2
Fig. 6.4 (run 2)	5	x axis $\frac{\lambda_1(1)}{2}...\frac{\lambda_5(1)}{2}$	4.1,4.3,4.5,4.7,4.9 1e-20	0.35,0.4,0.45,0.6,0.7 0.35,0.4,0.45,0.6,0.7	0.1	0.2

Table 6.3: System input parameters

Part II

Application of Retrial Queues in Performance Modelling of Communication Networks

Chapter 7

Retrial Queueing Models of Mobile Communication Networks

Queueing network models are widely used in the traffic modelling of cellular mobile systems, such as GSM (Global System for Mobile Communications), GPRS (General Packet Radio Service) and UMTS (Universal Mobile Telecommunication System). Most of the papers consider queueing systems without retrials (see [40; 21] and references therein for some recent results), but after the study of Tran-Gia and Mandjes [56], which demonstrated in the context of cellular systems that the retrial phenomenon is not neglectable because of the significant negative influence on the system performance measures, authors are more likely to take it into consideration in their cellular mobile network model.

Cellular systems with customer redials are treated in [41], where an approximate technique is proposed for finite and infinite population Markovian models. The authors reduce the state space of the continuous-time Markov chain model by registering only that if there are retrying blocked and dropped customers in the system or not. In the works [45; 2], various infinite-source retrial queueing models are studied. In [45], not only customer redials, but also automatic retrials by the cellular system are taken into consideration, but the dropped customer redials handled as generating new fresh call attempts in the new cell and in case of blocking the call is treated as a blocked fresh call. It is probably less realistic, because an interrupted customer may try to reestablish the call with higher probability in shorter time intervals. In [2], the blocked new and dropped handoff calls are not distinguished, but the involved random variables have general phase type distributions.

7.1 Quality of Service

In cellular networks, the most important quality of service measures are the following:

- the fresh call blocking probability (P_f), i. e. the fraction of new call requests in the cell that cannot be served due to the lack of free channels, and
- the handoff call dropping probability (P_h), that is the average fraction of incoming handoff calls that are terminated because of the lack of free channels.

The grade of service (GoS) is generally defined as the combination of these two probabilities, for example as

$$GoS = \frac{P_f + 10P_h}{11}.$$

Because of the fact, that the handoff call dropping probability has more significant impact on the grade of service, it is important to reduce it even at the expense of increased fresh call blocking probability. In order to prioritize handoff calls, several channel allocation schemes are utilized. One of the most popular policies is the guard channel scheme [21; 56; 41; 2], where some channels are reserved for the calls that move across the cell boundary, that is if there are g reserved channels in the cell, a new fresh call is only accepted if there are at least $g + 1$ available channels. A handoff call is rejected only if all the channels in the cell are occupied.

In Section 7.2, where the GSM system is modelled, this equation is used for calculating the GoS.

7.2 Performance Analysis of GSM

In this section, an infinite-source retrial queueing model of GSM networks is discussed, based upon the ones that were studied by [56; 41; 45; 2]. The blocked and dropped users are treated separately, that is they redial with different probabilities and different rates, like in [41], but the state space is reduced by maximizing the number of redialing customers with appropriately large values (i. e. when the ignored probability mass can be neglected). In [56; 45; 2], these two types of redialing customers were not distinguished. Furthermore, in this model not only the active but also both types of redialing customers are let to depart to other cells, what was not allowed in the previous works. This work can be considered as the initial step towards the analysis of more complex third generation systems focusing on the quality of service issues.

In the next subsections, the accurate description of the cellular model is given along with the underlying Markov chain. It is shown how the model description can be translated into the description language of MOSEL-2. The last part of this section is devoted to some numerical examples, where the analytical results of the calculations are displayed graphically to demonstrate the effect of the changing of various system parameters on the quality of service measures and on the grade of service. These results were published in [J2].

7.2.1 Model Description

In this subsection the following cell model (illustrated by Figure 7.1) is considered in a cellular mobile network.

In this cellular network model only one cell is treated. The cells are considered identical and to have the same traffic parameters, so it is enough to investigate one cell, and the handoff effect from the adjacent cells to this cell and from this cell to adjacent cells is described by handoff processes. Instead of the frequently used single arrival stream model the fresh call and handoff call arrivals are distinguished, what is gainful if we investigate complex call handling policies.

We assume, that the number of channels in the cell is C, and the number of guard channels is g, where $g < C$.

The arrival process of the fresh calls is a Poisson process with rate λ_f. If the number of the active users is smaller than $C - g$, the incoming call starts to be served. Otherwise it is blocked and it starts generation of a Poisson flow of repeated calls (redialing) with probability Θ_1 or leaves the system with probability $1 - \Theta_1$. A blocked customer repeats his call after a random time which is exponentially distributed with mean $1/\nu_{bl}$, and it can be served or blocked again like the fresh calls. The call duration time is exponentially distributed with mean $1/\mu$.

The arrival process of the handoff calls is a Poisson process with rate λ_h. If the number of active users is smaller than C, the incoming call starts to be served. Otherwise it is dropped (handoff failure) and it starts generation of a Poisson flow of repeated calls with probability Θ_2 or leaves the system with probability $1 - \Theta_2$. A dropped customer tries to repeat his call after a random time which is exponentially distributed with mean $1/\nu_{dr}$. If it is blocked it continues redialing with probability Θ_2. The call duration time for handoff calls is also exponentially distributed with mean $1/\mu$.

The active, redialing blocked and dropped customers leave the cell after an exponentially distributed time with mean $1/\mu_a$, $1/\mu_b$ and $1/\mu_d$, respectively.

The number of redialing users because of blocking and dropping is limited to an appropriately large values of N_{bl} and N_{dr} to make the state space finite in order to make the calculations possible by the tools in the steady state.

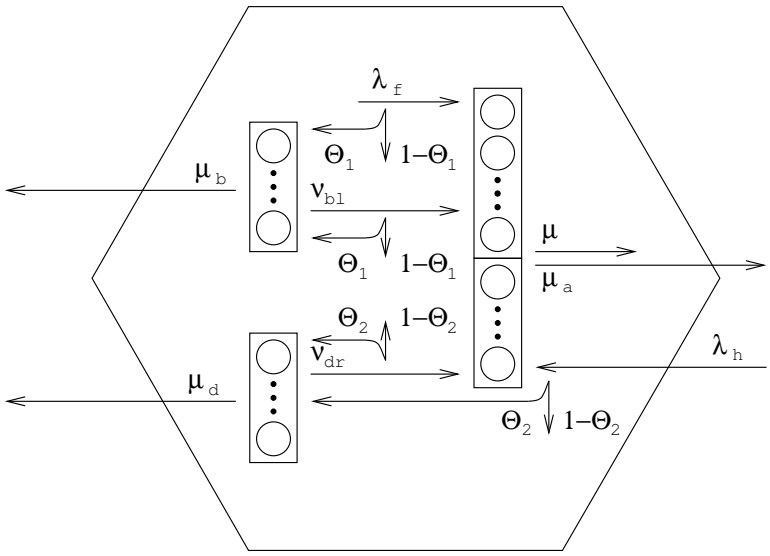


Figure 7.1: Retrial queueing model of a cell

7.2.2 The Underlying Markov Chain

The state of the system can be described with a stochastic process $X(t) = (C(t); N(t); M(t))$, where $C(t)$ is the number of active customers (i. e. the number of busy

channels), $N(t)$ is the number of blocked new customers who are sending repeated calls and $M(t)$ is the number of dropped customers at handoff who are trying to redial at time t.

Because of the exponentiality of the involved random variables the describing process is a Markov chain with a finite state space $S = \{0, ..., C\} \times \{0, ..., N_{bl}\} \times \{0, ..., N_{dr}\}$. Since its state space is finite, the process is ergodic for all values of the rate of the arrival of new and handoff calls, and we can investigate it in the steady state.

Because of the fact that the state space of $(X(t), t \geq 0)$ with sufficiently large N_{bl} and N_{dr} is very large and the functioning of the system is complex, it is very difficult to calculate the steady state probabilities. The tool MOSEL-2 is used to formulate the model and to calculate these probabilities and the system measures.

We define the stationary probabilities:

$$P(i; j; k) = \lim_{t \to \infty} P(C(t) = i, N(t) = j, M(t) = k),$$
$$i = 0, ..., C, \;\; j = 0, ..., N_{bl}, \;\; k = 0, ..., N_{dr}.$$

Knowing the steady state probabilities the system performance and the quality of service measures can be obtained as follows.

- *Mean number of active customers*

$$N_c = \sum_{i=0}^{C} \sum_{j=0}^{N_{bl}} \sum_{k=0}^{N_{dr}} iP(i, j, k).$$

- *Mean number of sources of repeated calls because of the blocking of fresh calls*

$$N_b = \sum_{i=0}^{C} \sum_{j=0}^{N_{bl}} \sum_{k=0}^{N_{dr}} jP(i, j, k).$$

- *Mean number of sources of repeated calls because of the dropping of handoff calls*

$$N_d = \sum_{i=0}^{C} \sum_{j=0}^{N_{bl}} \sum_{k=0}^{N_{dr}} kP(i, j, k).$$

- *Fresh call blocking probability*

$$P_f = \sum_{i=0}^{g} \sum_{j=0}^{N_{bl}} \sum_{k=0}^{N_{dr}} P(C - i, j, k).$$

- *Handoff call dropping probability*

$$P_h = \sum_{j=0}^{N_{bl}} \sum_{k=0}^{N_{dr}} P(C, j, k).$$

7.2.3 Model Conversion to MOSEL-2

In this subsection we discuss the translation of the model into the language of the MOSEL-2 tool. The full MOSEL-2 program can be assembled from the following program parts among the model description in the order of the part numbers.

The number of channels in the cell is C, which is denoted as *N_CHS* in the program, and the number of guard channels is g, which is denoted as *N_G_CHS*.

In the first part of the MOSEL-2 description, we have to define some other system input parameters too, these will be introduced at the appropriate program parts.

```
(1) CONST N_CHS := 15;
    CONST N_G_CHS := 1;
    CONST MAX_BL_USERS := 25;
    CONST MAX_DR_USERS := 25;
    CONST call_arrive := 1.5;
    CONST call_retry_bl := 5;
    CONST call_retry_dr := 6;
    CONST call_duration := 0.05;
    CONST handoff_arrive := 0.4;
    CONST handoff_dep_ac := 1/3;
    CONST handoff_dep_bl := 1/3;
    CONST handoff_dep_dr := 1/3;
    CONST p_retry_bl := 0.7;
    CONST p_retry_dr := 0.9;
```

The state of the system is described by the number of active users, the number of blocked users who redial after some random time, and the number of users whose calls are dropped at handoff and who are redialing. It can be wrote down in MOSEL-2 as defining the nodes of the system. The number of active users is denoted by *active_users*. Its maximum value is the number of channels, and it is 0 at the starting time. The number of redialing users because of blocking and dropping is limited to *MAX_BL_USERS* and *MAX_DR_USERS*, which are defined in (1).

```
(2) NODE active_users[N_CHS] := 0;
    NODE redialing_users_bl[MAX_BL_USERS] := 0;
    NODE redialing_users_dr[MAX_DR_USERS] := 0;
```

The arrival process of the fresh calls is a Poisson process with rate λ_f, that is denoted in the program as *call_arrive*, and defined in (1) like the other parameters. If the number of active users is smaller than $C - g$, the incoming call starts to be served. Otherwise it is blocked and it starts generation of a Poisson flow of repeated calls (redialing) with probability Θ_1 (denoted by p_retry_bl) or leaves the system with probability $1 - \Theta_1$.

```
(3) IF active_users < N_CHS-N_G_CHS
      FROM EXTERN TO active_users
      RATE call_arrive;
    IF active_users >= N_CHS-N_G_CHS
      FROM EXTERN RATE call_arrive THEN {
```

```
    TO redialing_users_bl
                WEIGHT p_retry_bl;
    TO EXTERN WEIGHT 1 - p_retry_bl;
  }
```

The blocked user redials can be handled similar to the fresh call arrivals. If a user is blocked, he repeats his call after a random time which is exponentially distributed with mean $1/\nu_{bl}$. ν_{bl} is denoted as *call_retry_bl*. It can be served or blocked as the fresh calls in the previous part.

```
(4) IF active_users < N_CHS-N_G_CHS
     FROM redialing_users_bl TO active_users
     RATE call_retry_bl*redialing_users_bl;
    IF active_users >= N_CHS-N_G_CHS
     FROM redialing_users_bl
     RATE call_retry_bl*redialing_users_bl THEN {
      TO redialing_users_bl
                WEIGHT p_retry_bl;
      TO EXTERN WEIGHT 1 - p_retry_bl;
    }
```

The call duration time is exponentially distributed with mean $1/\mu$. μ is denoted as *call_duration.*

```
(5) FROM active_users TO EXTERN
     RATE call_duration*active_users;
```

The arrival process of the handoff calls is a Poisson process with rate λ_h. λ_h is denoted in the program as *handoff_arrive*. If the number of active users is smaller than C, the incoming call starts to be served. Otherwise it is dropped and it starts generation of a Poisson flow of repeated calls with probability Θ_2 (denoted by p_retry_dr) or leaves the system with probability $1 - \Theta_2$.

```
(6) IF active_users < N_CHS
     FROM EXTERN TO active_users
     RATE handoff_arrive;
    IF active_users = N_CHS
     FROM EXTERN RATE handoff_arrive THEN {
      TO redialing_users_dr
                WEIGHT p_retry_dr;
      TO EXTERN WEIGHT 1 - p_retry_dr;
    }
```

The dropped user redials can be handled like the blocked fresh call redials. The customer repeats his call after a random time which is exponentially distributed with mean $1/\nu_{dr}$. ν_{dr} is denoted as *call_retry_dr*. If it is blocked it continues retrying with probability Θ_2 (p_retry_dr).

```
(7)  IF active_users < N_CHS-N_G_CHS
       FROM redialing_users_dr TO active_users
       RATE call_retry_dr*redialing_users_dr;
     IF active_users >= N_CHS-N_G_CHS
       FROM redialing_users_dr
       RATE call_retry_dr*redialing_users_dr THEN {
         TO redialing_users_dr
                        WEIGHT p_retry_dr;
         TO EXTERN WEIGHT 1 - p_retry_dr;
     }
```

The active and redialing customers leave the cell after an exponentially distributed time with parameter μ_a, μ_b and μ_d, denoted as *handoff_dep_ac*, *handoff_dep_bl* and *handoff_dep_dr*, respectively.

```
(8)  FROM active_users TO EXTERN
       RATE handoff_dep_ac*active_users;
     FROM redialing_users_bl TO EXTERN
       RATE handoff_dep_bl*redialing_users_bl;
     FROM redialing_users_dr TO EXTERN
       RATE handoff_dep_dr*redialing_users_dr;
```

After describing the system functioning, we can define the system measures we would like to calculate, such as the mean number of active and redialing customers because of blocking and handoff failure, the fresh call blocking and the handoff call dropping probabilities.

```
(9)  PRINT mean_active_users = MEAN(active_users);
     PRINT mn_redialing_users_bl = MEAN(redialing_users_bl);
     PRINT mn_redialing_users_dr = MEAN(redialing_users_dr);
     PRINT call_blocking_prob = PROB(active_users >= N_CHS-N_G_CHS);
     PRINT handoff_call_dropping_prob = PROB(active_users = N_CHS);
```

Finally, we define two pictures that show the changing of the blocking and dropping probabilities depending on the number of channels. If we use *N_CHS* as parameter, we have to define it in (1) as follows: *PARAMETER N_CHS := 6, 7, 8, 9, 10;*

```
(10) PICTURE "Blocking probability vs N_CHS"
       PARAMETER N_CHS
       CURVE call_blocking_prob;
     PICTURE "Dropping probability vs N_CHS"
       PARAMETER N_CHS
       CURVE handoff_call_dropping_prob;
```

7.2.4 Numerical Examples

In this subsection some sample numerical results are presented to illustrate graphically how the quality of service measures depend on variable system parameters.

Comments

- In Figures 7.2 and 7.3 the fresh call blocking and handoff call dropping probabilities are displayed versus the number of channels with and without user redials. The system input parameters belonging to the curves without redials are the same as in [21], where a similar model is studied without customer redials ($g = 3$, $\lambda_f = 0.5$, $\mu = 0.05$, $\mu_a = \mu_b = \mu_d = 1/3$, $\lambda_h = 0.4$, $\nu_{bl} = \nu_{dr} = 10^6$, $\Theta_1 = \Theta_2 = 10^{-6}$ and for the other curve $\nu_{bl} = \nu_{dr} = 6$, $\Theta_1 = 0.8$, $\Theta_2 = 0.9$, furthermore the maximum number of redialing customers is 25, respectively). These results are in agreement with theirs in the exponential case.

- In Figures 7.4 and 7.5 the fresh call blocking and handoff call dropping probabilities are displayed versus the mean handoff call arrival rate. The system input parameters are the same as in Figures 7.2 and 7.3, except of that $C = 8$, and λ_h is on the x axis, like in [21].

  The negative influence of the retrial phenomenon is shown in each figures, and we can see that it increases as the handoff call arrival rate increases.

- In Figure 7.6 we can see the fresh call blocking probability, the handoff call dropping probability and the grade of service as the mean fresh call arrival rate increases. The following system input parameters were used: $C = 7$, $g = 1$, $\mu = 0.05$, $\mu_a = \mu_b = \mu_d = 1/3$, $\lambda_h = 0.4$, $\nu_{bl} = 6$, $\nu_{dr} = 7$, $\Theta_1 = 0.8$ and $\Theta_2 = 0.9$.

- In Figure 7.7 the fresh call blocking and handoff call dropping probabilities and the GoS are displayed versus the number of guard channels. We can see that a very few number of guard channels can improve the grade of service significantly, but then only a very small handoff call dropping advance can be achieved on the great expense of fresh call blocking probability, and the GoS declines. The system input parameters are the following: $C = 15$, $\lambda_f = 3$, $\mu = 0.05$, $\mu_a = \mu_b = \mu_d = 1/3$, $\lambda_h = 0.4$, $\nu_{bl} = 6$, $\nu_{dr} = 7$, $\Theta_1 = 0.8$ and $\Theta_2 = 0.9$.

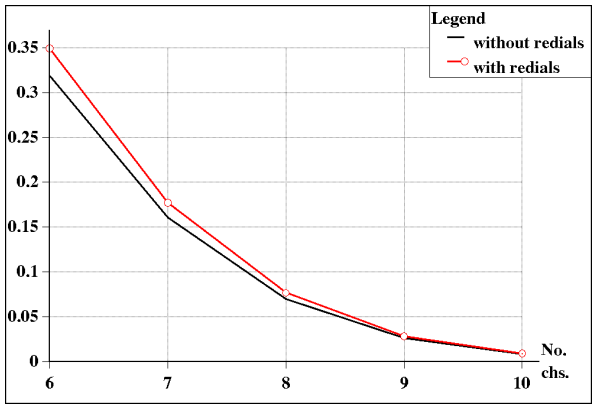


Figure 7.2: Fresh call blocking probability versus number of channels

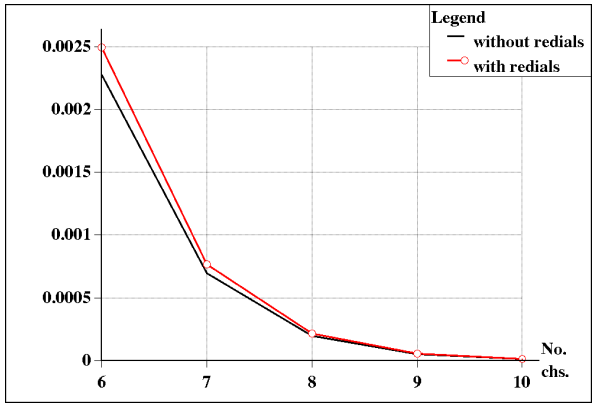


Figure 7.3: Handoff call dropping probability versus number of channels

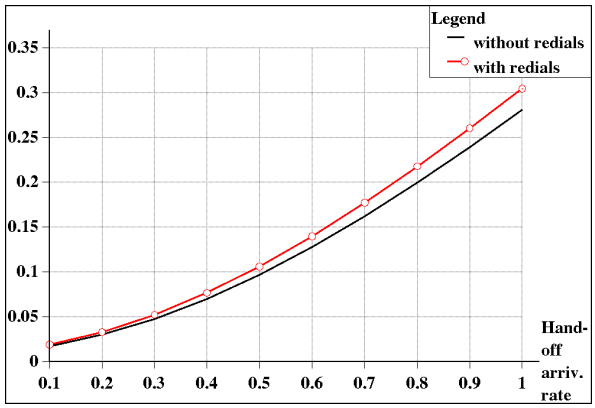


Figure 7.4: Fresh call blocking probability versus mean handoff call arrival rate

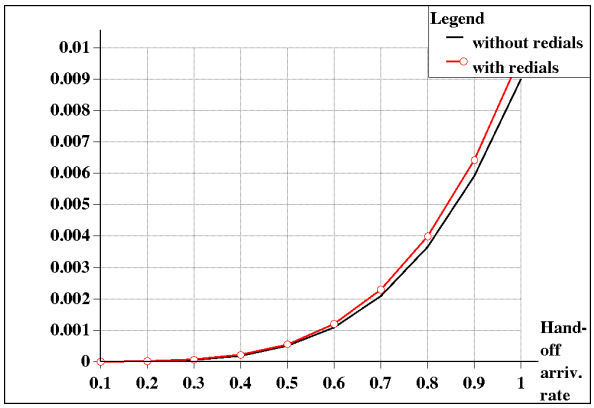


Figure 7.5: Handoff call dropping probability versus mean handoff call arrival rate

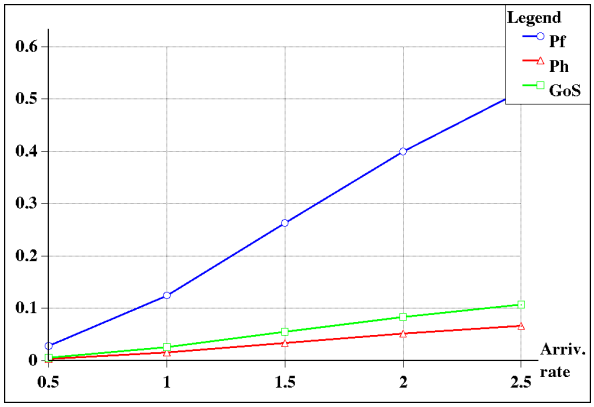


Figure 7.6: System measures versus mean fresh call arrival rate

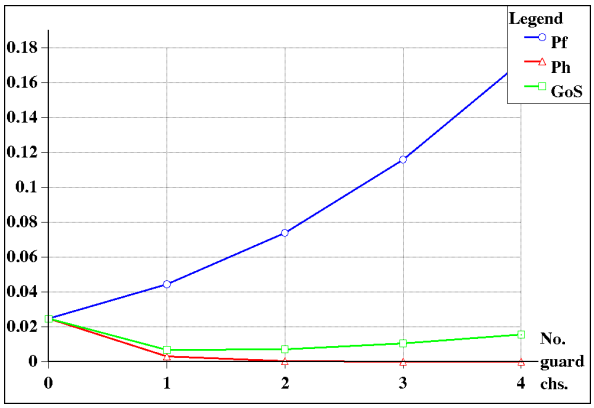


Figure 7.7: System measures versus number of guard channels

Part III

Conclusions

In this dissertation the performance of different types of retrial queues was evaluated, and a retrial queueing model was applied for the performance analysis of a cell based telecommunication network. The efficient tool MOSEL was used, which made it much easier to analyze these complex models.

In Part I, the performance of finite-source retrial queueing systems was investigated with components subject to random breakdowns and repairs. A single-server queue was extended and analyzed with non-reliable server and sources, and the non-reliable server case was generalized by heterogeneous sources, i.e. the sources were allowed to have different parameters. Multiserver retrial queues were treated with non-reliable servers, and different service policies were compared. Finally, a reliable retrial queue was treated in random environment, which also can be applied for the analysis of systems in which the components are subject to random breakdowns. To the best know of the author, these finite-source retrial queueing systems have not been investigated before in the literature.

In Part II, a truncated infinite-source retrial queue was applied for the performance evaluation of the GSM system. This was based on previous works of others in this topic, and some generalizations were made. It was described in details how this system can be modelled by the help of the tool MOSEL-2, which makes the modelling and the further extensions much easier. This model can be developed to a layered one, and the newest version of MOSEL-2 allows us to use other distributions than the exponential, which can be applied and analyzed, too.

Summary

Performance evaluation plays an important role in the design, analysis and development of practical systems, like computer and telecommunication systems and networks. Queueing models are often used for the performance and reliability modelling of these systems, and retrial queues are more and more frequently applied to certain types of them. The reason is that the return of customers plays a special role in many of these systems as well as in other practical applications, and it often has a non-neglectable negative effect on the performance measures.

Another important characteristic of real-life systems is non-reliability, which also has a negative influence on these measures, because most of the components of the systems are subject to random breakdowns and require repairs. Non-reliability has been extensively studied for traditional queues with waiting lines, but only for infinite-source queues with returning customers.

The dissertation consists of two parts. In Part I, some non-reliable finite-source retrial models and a reliable retrial queue in random environment (which also can be applied in the performance analysis of non-reliable systems) are analyzed. To the best know of the author, these models have not been treated in the literature before. In Part II, a real-life system is modelled using a retrial queueing model. A modelling way of the GSM system (Global System for Mobile Communications) is treated with a modelling environment. This is based on previous works of various authors and generalized with some model extensions.

Retrial Queues

Retrial queues (queueing systems with repeated attempts, or queues with returning customers) are characterized by the following feature: a request finding all servers busy upon arrival leaves the service area but after some (random) time repeats his demand. This feature plays a special role in many computer and communication systems and networks as well as in other practical applications. In case of many real-life systems, retrial queues can be applied in the performance modelling, for example in modelling local-area and cellular mobile networks.

In Chapter 2, two types of retrial queues are introduced. The first one is a finite-source retrial queue, on which the analysis is based in Part I. The second one is a truncated infinite source queue with returning customers. An extension of this model is applied in Part II to analyze the GSM system.

The Applied Modelling Tool

MOSEL (Modeling, Specification and Evaluation Language) is a modelling environment with a high-level modelling language which allows us to describe complex real-world systems and to calculate their system measures using other performance evaluation tools. The MOSEL description can be translated automatically into the language of various performance tools and then analyzed by the appropriate tools (at present SPNP – Stochastic Petri Net Package and TimeNET are supported and suitable for the investigated models) to get these measures.

Because of the fact that the state space of the underlying Markov chains of the investigated queueing models is very large and the functioning of the systems is complex, it is quite difficult to calculate the steady state probabilities in the traditional way of solving the system of steady-state equations. To simplify these calculations and to make our studies more usable in practice, the tool MOSEL was used to formulate the models and to calculate the performance measures. This tool has already been used, and it has proved its applicability for the modelling of several computer and communication systems. The functioning and usage of MOSEL is illustrated by Figure 1.

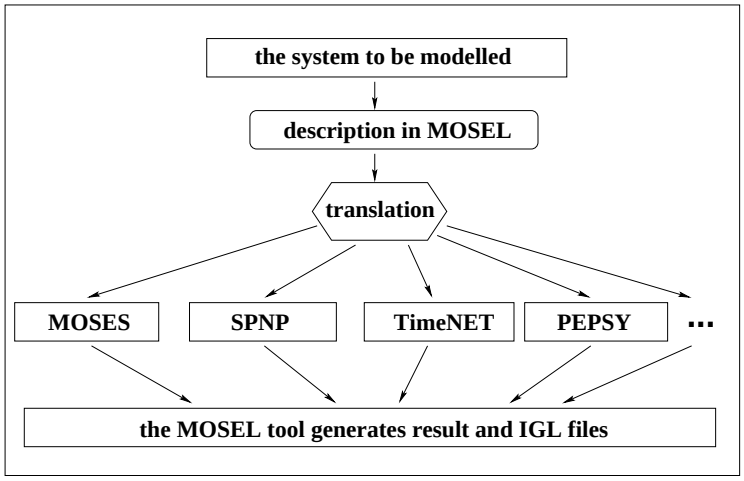


Figure 1: The modelling process in the MOSEL environment

In Part I, the original tool is used for the analysis of finite-source retrial queueing systems. Because of page limitations, only the simplest MOSEL description was included and discussed.

In Part II, the GSM system is modelled with the revised modelling language, called MOSEL-2, and detailed comments are provided about MOSEL-2 programming.

Analysis of Retrial Queueing Systems

The components of the real systems may be subject to random breakdowns so it is important to investigate non-reliable queueing systems, as well as non-reliable retrial queues, because of limited ability of repairs and heavy influence of the breakdowns on the performance measures.

- *Analysis of Single-server Non-reliable Finite-source Retrial Queues*
  In Chapter 4, single-server non-reliable finite-source retrial queues are treated. The purpose is to give the main stationary performance and reliability measures of the non-reliable models, and to illustrate graphically the effect of changing various parameters on them. Section 4.1 is devoted to the model described in the book of Falin and Templeton with server subject to breakdowns and repairs. In Section 4.2 this is extended with non-reliable sources, and in Section 4.3 with reliable but heterogeneous sources.

- *Analysis of Multiserver Non-reliable Finite-source Retrial Queues*
  In Chapter 5, a multiserver non-reliable finite-source retrial queue is investigated. Section 5.1 is devoted to the extension of the model described in the book of Falin and Templeton. The finite-source retrial queue is analyzed with non-reliable heterogeneous (asymmetric) servers, that is the servers have different parameters in service, failure and repair rates. In Section 5.2 two service policies are compared in this model.

- *Analysis of Retrial Queues in Random Environment*
  Chapter 6 deals with the performance analysis of a finite-source retrial queueing system with heterogeneous sources operating in random environment, that is, the system parameters are subject to randomly occurring fluctuations. Besides, the queueing model is applied for the analysis of non-reliable retrial models.

For an example analysis, see Figure 2, where we can see the difference between the Fastest Free Server (referred to as ordered in the legend of the figure), Random and the Averaged Random (where the service rate of the servers is the average of the rates of the heterogeneous case) service policies in the mean response time depending on the primary request generation rate. In the case of Random service disciplines, the requests are assigned to the idle servers randomly. In the other case, the requests are assigned to the fastest available free server. We can see that the difference between the two random case is not too significant, but the Fastest Free Server case always has better response time, especially when more and more requests arrive.

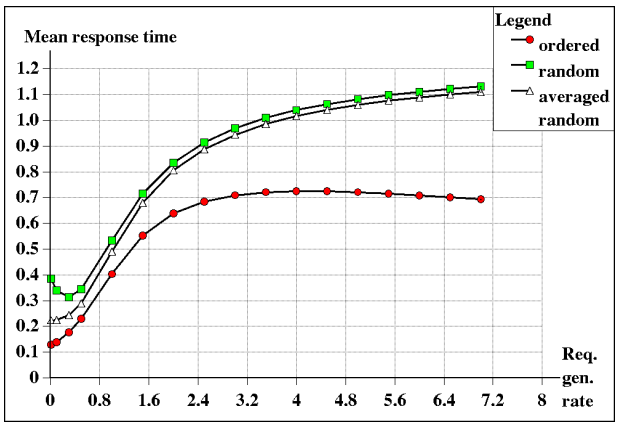


Figure 2: Mean response time versus primary request generation rate

Application of Retrial Queues in Modelling of Communication Networks

Queueing network models are widely used in the traffic modelling of cellular mobile systems, such as GSM (Global System for Mobile Communications), GPRS (General Packet Radio Service) and UMTS (Universal Mobile Telecommunication System). Most of the papers consider queueing systems without retrials, but after the study of Tran-Gia and Mandjes (1997), which demonstrated in the context of cellular systems that the retrial phenomenon is not neglectable because of the significant negative influence on the system performance measures, authors are more likely to take it into consideration.

In Section 7.2, an infinite-source retrial queueing model of GSM networks is discussed, based upon the ones that were studied by others. The blocked and dropped users are treated separately, that is they redial with different probabilities and different rates. The state space is reduced by maximizing the number of redialing blocked and dropped customers with appropriately large values (i. e. when the ignored probability mass can be neglected). Furthermore, in this model not only the active but also both types of redialing customers are let to depart to other cells, what was not allowed in the previous works. This work can be considered as the initial step towards the analysis of more complex third generation systems using the MOSEL-2 tool.

The accurate description of the cellular model is given along with the underlying Markov chain. It is shown how the model description can be translated into the description language of MOSEL-2. Some numerical examples are treated, where the analytical results of the calculations are displayed graphically to demonstrate the effect of the changing of various system parameters on the quality of service measures.

For an example, see Figure 3, where the fresh call blocking and handoff call dropping probabilities and the GoS (Grade of Service) are displayed versus the number of guard channels. In this figure, we can see how many guard channels can improve the grade of service significantly, and then the other reserved channels can achieve only very small handoff call dropping advance on the great expense of fresh call blocking probability, and the GoS declines.

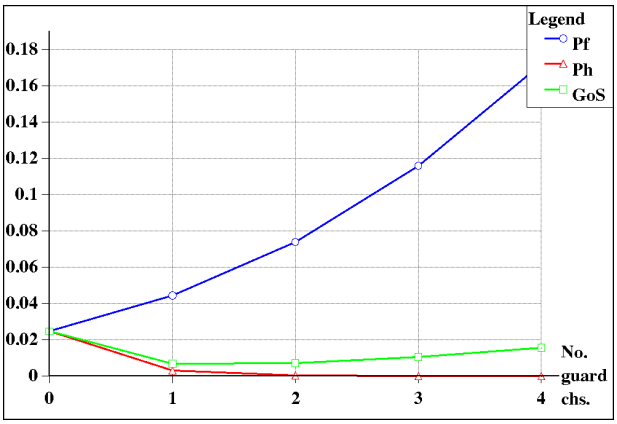


Figure 3: System measures versus number of guard channels

Összefoglaló

A teljesítményelemzés fontos szerepet játszik a gyakorlati rendszerek – mint például a telekommunikációs hálózatok – tervezésében, elemzésében és fejlesztésében. Gyakran alkalmaznak sorbanállási modelleket ezen rendszerek teljesítményének és megbízhatóságának vizsgálatában. Különböző típusaiknál egyre gyakrabban használják a visszatéréses sorbanállási rendszereket is. Ennek az oka az, hogy a visszatérő igények többüknél – akárcsak mint más gyakorlati alkalmazásoknál – gyakran speciális szerepet játszanak, és nem elhanyagolható negatív hatással vannak a teljesítményjellemzőkre.

A gyakorlati rendszerek egy másik fontos jellemzője a meghibásodhatóság, mely, mivel a rendszerkomponensek többsége meghibásodhat és javítást igényel, szintén negatív hatással van ezekre a jellemzőkre. A nem megbízhatóság hatását alaposan tanulmányozták hagyományos, várakozási sorral rendelkező sorbanállási rendszereknél, visszatérő igényekkel viszont csak végtelen forrású modellek esetén.

A disszertáció két részből áll. Az I. részben néhány nem megbízható, véges forrású visszatéréses sorbanállási rendszert vizsgálunk, valamint egy megbízható visszatéréses rendszert véletlen környezetben, ami szintén alkalmazható a nem megbízható modellek elemzésében. A szerző legjobb tudása szerint ezeket a modelleket nem tárgyalták korábbi munkákban. A II. részben egy valós rendszert vizsgálunk egy visszatérő igényeket tartalmazó sorbanállási rendszer segítségével. Itt egy modellezési módot tekintünk át mások korábbi munkáira alapozva a GSM rendszer (Global System for Mobile Communications) teljesítményelemzésére.

Visszatéréses sorbanállási rendszerek

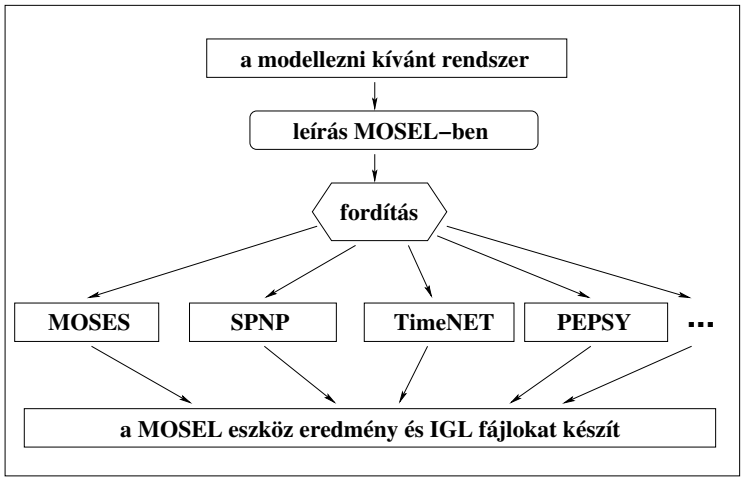
A visszatérő igényeket tartalmazó vagy visszatéréses sorbanállási rendszerek olyan rendszerek, melyekben ha egy beérkező igény minden kiszolgálót foglaltnak talál, akkor elhagyja a kiszolgálókat, majd egy (véletlen) idő eltelte után megismétli a kérést. Ez a tulajdonság más gyakorlati alkalmazások mellett speciális szerepet játszik számos, napjainkban is használt számítógép és kommunikációs rendszerben és hálózatban is. Így több valós rendszer teljesítményelemzése esetén is – mint például a helyi és a celluláris mobil hálózatok – alkalmazhatóak a visszatéréses sorbanállási modellek.

A 2. fejezetben két különböző visszatéréses sorbanállási modellt tekintünk át. Az első az a véges forrású visszatéréses sorbanállási rendszer, amelyen az I. részben lévő elemzések alapszanak. A második pedig egy végtelen forrású modell, melynek egy kiterjesztett változatát alkalmazzuk a II. részben a GSM rendszer hatékonyság-vizsgálatára.

Az alkalmazott modellezési eszköz

A MOSEL (Modeling, Specification and Evaluation Language) egy olyan magasszintű nyelvvel rendelkező modellezési környezet, mely lehetővé teszi valós, bonyolult rendszerek leírását, és ez alapján kiszámítja a keresett rendszerjellemzőket más hatékonyságvizsgálati eszközök felhasználásával. A MOSEL leírás automatikusan lefordítható a különböző eszközök nyelvére (jelenleg az SPNP – Stochastic Petri Net Package és a TimeNET alkalmazható a vizsgált modellek esetén), majd az adott eszközt felhasználva elemezhetjük a modellt és kaphatjuk meg a keresett rendszerjellemzőket.

Mivel a vizsgált sorbanállási modelleket leíró Markov-láncok állapottere nagyon nagy és a rendszerek működése bonyolult, ezért a hagyományos módon, azaz az egyensúlyi állapotegyenlet-rendszer megoldásával meglehetősen nehéz kiszámítani az egyensúlyi állapotvalószínűségeket. Hogy egyszerűsítsük ezeket a számításokat, továbbá a gyakorlatban használhatóbbá tegyük az elemzéseket, a MOSEL eszközt használjuk a modellek leírására és a teljesítményjellemzők kiszámítására, melyet már számos számítógép és kommunikációs rendszer modellezésére használták, és bizonyította alkalmazhatóságát. A MOSEL működését és használatát az itt látható ábra szemlélteti.



4. ábra. A modellezési folyamat a MOSEL környezetben

Az I. részben az eredeti eszközt használjuk véges forrású visszatéréses sorbanállási rendszerek elemzésére. Oldalszám korlátok miatt csak a legegyszerűbb MOSEL leírást tárgyaljuk.

A II. részben az átdolgozott modellezőnyelvet (MOSEL-2) alkalmazzuk – részletes magyarázatokkal a MOSEL-2 prgramozással kapcsolatban – a GSM rendszer modellezésére.

Visszaéréses sorbanállási rendszerek elemzése

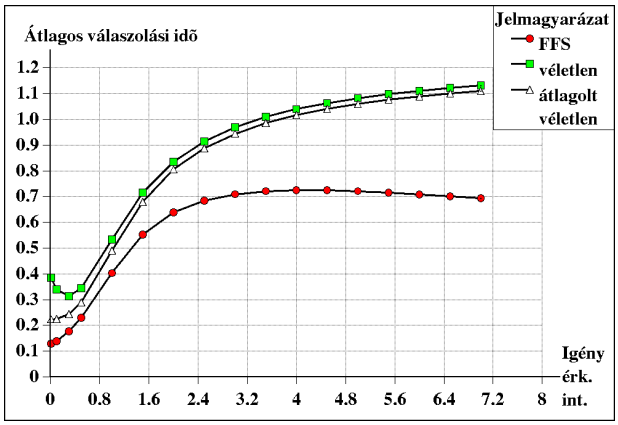
Mivel a valós rendszerek komponensei általában meghibásodhatnak, ezért a korlátozott javítási lehetőségek, és a meghibásodásoknak a teljesítményjellemzőkre gyakorolt jelentős hatása miatt fontos a nem megbízható sorbanállási rendszerek vizsgálata, így a nem megbízható visszatéréses modelleké is.

- *Egykiszolgálós, nem megbízható, véges forrású visszatéréses modellek elemzése*
  A 4. fejezetben egykiszolgálós, nem megbízható, véges forrású visszatéréses sorbanállási rendszereket vizsgálunk. Célunk a nem megbízható modellek legfontosabb egyensúlyi megbízhatósági és teljesítményjellemzőinek megadása, valamint különböző paramétereknek az ezekre gyakorolt hatásának grafikus szemléltetése. A 4.1 alfejezetben a Falin és Templeton könyvében leírt modellt vizsgájuk nem megbízható kiszolgálóval. A 4.2 alfejezetben ezt bővítjük nem megbízható, 4.3-ban pedig megbízható de heterogén forrásokkal.

- *Többkiszolgálós, nem megbízható, véges forrású visszatéréses modellek elemzése*
  A 5. fejezetben egy többkiszolgálós, nem megbízható, véges forrású visszatéréses sorbanállási rendszert vizsgálunk. Az 5.1. alfejezetben a Falin és Templeton könyvében leírt modellt általánosítjuk. Itt a véges forrású visszatéréses modellt heterogén (aszimmetrikus) kiszolgálókkal – melyek különböző kiszolgálási, meghibásodási és javítási paraméterekkel rendelkezhetnek – elemezzük. Az 5.2. alfejezetben két kiszolgálási elvet hasonlítunk össze ennél a modellnél.

- *Visszatéréses sorbanállási rendszerek véletlen környezetben*
  A 6. fejezet egy véletlen környezetben lévő, heterogén forrásokat tartalmazó visszatéréses modell teljesítményelemzésével foglalkozik, azaz egy olyan sorbanállási rendszerrel, melynek paraméterei egy háttérfolyamat alapján változhatnak. Emellett ezt a modellt is alkalmazzuk nem megbízható visszatéréses rendszerek vizsgálatára.

Az 5. ábrán egy elemzési példa látható, ahol a leggyorsabb szabad kiszolgáló (Fastest Free Server – FFS), a véletlen (Random) és az átlagolt véletlen (itt mindegyik kiszolgálási intenzitás a heterogén eset intenzitásainak átlaga) kiszolgálási elvek esetén láthatjuk az átlagos válaszolási időt az új igény érkezési intenzitás függvényében. A véletlen kiszolgálási elvek esetén az igények véletlenszerűen kerülnek az egyes szabad kiszolgálókhoz. A másik elv esetén viszont a beérkező igény a leggyorsabb szabad kiszolgálóhoz kerül. Látható, hogy a különbség a két véletlen eset között nem jelentős, de a leggyorsabb szabad kiszolgálót választó elv mindig jobb átlagos válaszidővel rendelkezik, különösen mikor egyre több és több igény érkezik.



5. ábra. Átlagos válaszolási idő az új igény érkezési intenzitás függvényében

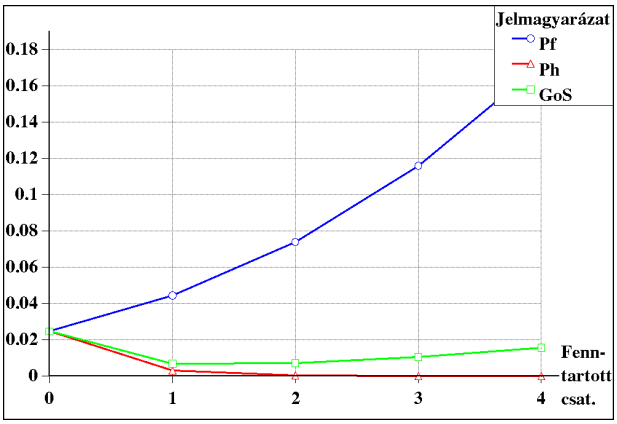
Visszatéréses sorbanállási rendszerek alkalmazása kommunikációs hálózatok modellezésében

A sorbanállási hálózati modelleket gyakran alkalmazzák a celluláris mobil rendszerek, mint például a GSM (Global System for Mobile Communications), a GPRS (General Packet Radio Service) és az UMTS (Universal Mobile Telecommunication System) teljesítményelemzésében. Elsősorban a klasszikus, visszatérő igényeket nem tartalmazó sorbanállási modelleket használják, de Tran-Gia és Mandjes (1997) cikke után egyre többen veszik figyelembe a visszatérő igényeknek a teljesítményjellemzőkre gyakorolt hatását.

A 7.2. fejezetben a GSM hálózatoknak egy végtelen forrású sorbanállási modelljét tárgyaljuk korábbi munkák alapján. A modellben a blokkolt és megszakadt hívások miatti ismétlődő kéréseket külön kezeljük, azaz a hívásismétlési valószínűségek és intenzitások különbözőek lehetnek. Az állapotteret mindkét típus esetén az ismétlődő hívásokat generáló felhasználók maximális számának megfelelően nagy értékekkel való korlátozásával redukáljuk. A vizsgált modellben továbbá nem csak az aktív, hanem mindkét típusú sikertelen hívás miatt újra próbálkozó felhasználó is távozhat másik cellába, ami a korábbi tanulmányokban nem volt megengedve. Ez a munka tekinthető az első lépésnek a bonyolultabb, harmadik generációs rendszereknek a MOSEL-2 eszközzel történő vizsgálata felé.

A modellezés során megadjuk a cella alapú modell pontos leírását az azt leíró Markov-lánccal együtt, majd megmutatjuk, hogyan fordítható le ez a MOSEL-2 leíró nyelvére. Áttekintünk néhány numerikus példát, ahol az analitikus eredményeket grafikusan ábrázolva mutatjuk meg különböző paraméterek változásának hatását a szolgáltatás minőségének jellemzőire.

A 6. ábrán erre látható egy példa, ahol az új hívások blokkolásának és az átadottak megszakadásának valószínűsége, valamint a GoS (Grade of Service) figyelhető meg a fenntartott csatornák számának függvényében. Látható, hogy kis számú csatorna fenntartása jelentősen javíthatja a szolgáltatás minőségi mértékét, de több ilyen csatorna esetén csak nagyon kis átadott hívásmegszakadási valószínűség-csökkenés érhető el jelentős blokkolási valószínűség-növekedés árán, és a GoS romlik.



6. ábra. Rendszerjellemzők a fenntartott csatornák számának függvényében

Bibliography

[1] **Aissani A. and Artalejo J. R.** On the single server retrial queue subject to breakdowns, *Queueing Systems Theory and Applications*, Vol. 30 (1998), 309-321.

[2] **Alfa A. S. and Li W.** PCS networks with correlated arrival process and retrial phenomenon, *IEEE Transactions on Wireless Communications*, Vol. 1 (2002) 630-637.

[3] **Almási B.** A Queueing Model for a Processor-Shared Multi-Terminal System Subject to Breakdowns *Acta Cybernetica*, Vol. 10 (1993) 273-282.

[4] **Almási B.** Response time for finite-source heterogeneous nonreliable queueing systems, *Computers and Mathematics with Applications*, Vol. 31 (1996) 55-59.

[5] **Almási B., Bolch G. and Sztrik J.** Performability Modeling of Non-homogeneous Terminal Systems Using MOSEL, *5th International Workshop on Performnability Modeling of Computer and Communication Systems, Erlangen, Germany, 2001* 15-16.

[6] **Almási B., Bolch G., Sztrik J.** Performability Modeling a Client-Server Communication System with Randomly Changing Parameter Using MOSEL. *5th International Workshop on Performability Modeling of Computer and Communication Systems, Erlangen, Germany, 2001* 37-41.

[7] **Almási B., Bolch G., Sztrik J.** Analysing Markov-modulated finite-source queueing systems. *Annales Univ. Sci. Budapest., Sect. Comp.*, Vol. 22 (2003) 22-33.

[8] **Almási B., Bolch G. and Sztrik J.** Heterogeneous finite-source retrial queues, *Journal of Mathematical Sciences*, Vol. 121 (2004) 2590-2596.

[9] **Almási B., Sztrik J.** Optimization Problems on the Performance of a Non-Reliable Terminal System, *Computers and Mathematics with Applications*, Vol. 38 (1999) 13-21.

[10] **Almási B., Roszik J. and Sztrik J.** Homogeneous finite-source retrial queues with server subject to breakdowns and repairs, *Mathematical and Computer Modelling*, Vol. 42 (2005) 673-682.

[11] **Artalejo J.R.** New results in retrial queueing systems with breakdown of the servers, *Statistica Neerlandica*, Vol. 48 (1994) 23-36.

[12] **Artalejo J.R.** Retrial queues with a finite number of sources, *J. Korean Math. Soc.*, Vol. 35 (1998) 503-525.

[13] **Artalejo J.R.** Accessible bibliography on retrial queues, *Math. Comput. Modeling*, Vol. 30 (1999) 1-6.

[14] **Artalejo, J.R. and Gomez-Corral, A.** Information theoretic analysis for queueing systems with quasi-random input, *Mathematical and Computer Modelling*, Vol. 22 (1995) 65-76.

[15] **Artalejo J.R., Rajagopalan V. and Sivasamy R.** On finite Markovian queues with repeated attemps, *Investigacion Operativa*, Vol. 9 (2000) 83-94.

[16] **Begain K., Bolch G. and Herold H.** *Practical performance modeling, application of the MOSEL language*, Kluwer Academic Publisher, Boston, 2001.

[17] **Begain K., Barner J., Bolch G., Zreikat A. I.** The Performance and Reliability Modelling Language MOSEL and its Application, *International Journal of Simulation*, Vol. 3 (2003) 66-80.

[18] **Begain K., Bolch G., Telek M.** Scalable Schemes for Call Admission and Handover in Cellular Networks with Multiple Services, *Wireless Personal Communications Journal*, Kluwer Academic Publisher, 1999.

[19] **Beutel B.** Integration of the Petri Net tool TimeNET into the MOSEL modelling environment, *MS Thesis*, Department of Computer Science, University of Erlangen, Germany, 2003.

[20] **Daigle J.N.** *Queueing theory for telecommunications*, Addison-Wesley, Nwe York, 1992.

[21] **Dharmaraja S., Trivedi K.S., Logothetis D.** Performance modeling of wireless networks with generally distributed handoff interarrival times, *Computer Communications*, Vol. 26 (2003) 1747-1755.

[22] **Dragieva V.I.** Single-line queue with finite source and repeated calls, *Problems of Information Transmission*, Vol. 30 (1994) 283-289.

[23] **Falin G.I.** A survey of retrial queues, *Queueing Systems*, Vol. 7 (1990) 127-168.

[24] **Falin G.I.** A multiserver retrial queue with a finite number of sources of primary calls, *Mathematical and Computer Modelling*, Vol. 30 (1999) 33-49.

[25] **Falin G.I. and Artalejo J.R.** A finite source retrial queue, *European Journal of Operational Research*, Vol. 108 (1998) 409-424.

[26] **Falin G.I. and Gomez Corral A.** On a bivariate Markov process arising in the theory of single-server retrial queues, *Statistica Neerlandica*, Vol. 54 (2000) 67-78.

[27] **Falin G.I. and Templeton J.G.C.** *Retrial queues*, Chapman and Hall, London, 1997.

[28] **Gaver D.P., Jacobs P.A., Latouche G.** Finite birth-and-death models in randomly changing environments, *Advances in Applied Probability*, Vol. 16 (1984) 715-731.

[29] **Gomez Corral A.** Analysis of a single-server retrial queue with quasi-random input and nonpreemptive priority, *Computers and Mathematics with Applications*, Vol. 43 (2002) 767-782.

[30] **Houck D.J. and Lai W.S.** Traffic modelling and analysis of hybrid fibercoax systems, *Computer Networks and ISDN Systems*, Vol. 30 (1998) 821-834.

[31] **Jain R.** *The art of computer systems performance analysis*, John Wiley and Sons, New York, 1991.

[32] **Janssens G.K.** The quasi-random input queueing system with repeated attempts as a model for collision-avoidance star local area network, *IEEE Transactions on Communications*, Vol. 45 (1997) 360-364.

[33] **Kalmychkov A.I. and Medvedev G.A.** Probability characteristics of Markov local-area networks with random-access protocols, *Automatic Control and Computer Science*, Vol. 24 (1990) 38-45.

[34] **Khomichkov I.I.** Study of models of local networks with multiple-access protocols, *Automation and Remote Control*, Vol. 54 (1993) 1801-1811.

[35] **Kok A.G.** Algorithmic methods for single server systems with repeated attemps, *Statistica Neerlandica*, Vol. 38 (1984) 23-32.

[36] **Kovalenko I.N., Kuznetsov N.Yu. and Pegg P.A.** *Mathematical theory of reliability of time dependent systems with practical applications*, John Wiley and Sons, Chichester, 1997.

[37] **Kulkarni V. G. and Choi B. D.** Retrial queues with server subject to breakdowns and repairs, *Queueing Systems Theory and Applications*, Vol. 7 (1990) 191-208.

[38] **Li Hui and Yang Tao** A single server retrial queue with server vacations and a finite number of input sources, *European Journal of Operational Research*, Vol. 85 (1995) 149-160.

[39] **Li Y., Al-Begain K., Awan I.** Performance Modelling of GSM/GPRS Mobile System with MOSEL, *4th PGNet, Liverpool, June 2003.* 245-250.

[40] **Litjens R. and Boucherie R. J.** Elastic calls in an integrated services network: the greater the call size variability the better the QoS, *Performance Evaluation*, Vol. 52 (2003) 193-220.

[41] **Marsan M. A., Carolis G. D., Leonardi E., Cigno R. L., Meo M.** Efficient estimation of call blocking probabilities in cellular mobile telephony networks with customer retrials, *IEEE Journal on Selected Areas in Communications*, Vol. 19 (2001) 332-346.

[42] **Mehmet-Ali M.K., Hayes J.F. and Elhakeem A.K.** Traffic analysis of a local area network with star topology, *IEEE Transactions on Communications*, Vol. 36 (1988) 703-712.

[43] **Nobel R.D. and Tijms Henk C.** Optimal control of a queueing system with heterogeneous servers and setup costs, *IEEE Trans. Autom. Control*, Vol. 45 (2000) 780-794.

[44] **Ohmura H. and Takahashi Y.** An analysis of repeated call model with a finite number of sources, *Electronics and Communications in Japan*, Vol. 68 (1985) 112-121.

[45] **Onur E., Delic H., Ersoy C. and Caglayan M. U.** Measurement-based replanning of cell capacities in GSM networks, *Computer Networks*, Vol. 39 (2002) 749-767.

[46] **Ravichandran N.** *Stochastic methods in reliability theory*, John Wiley and Sons, New York, 1990.

[47] **Roszik J.** Homogeneous finite-source retrial queues with server and sources subject to breakdowns and repairs, *Annales Univ. Sci. Budapest., Sect. Comp.*, Vol. 23 (2004) 213-227.

[48] **Roszik J., Almási B., Sztrik J.** Multiserver retrial queues with finite number of heterogeneous sources, *Proceedings of 6th International Conference on Applied Informatics*, Eger, Hungary, (2004) Vol. II, 19-26.

[49] **Roszik J., Sztrik J.** The effect of server's breakdown on the performance of finite-source retrial queueing systems, *Proceedings of 6th International Conference on Applied Informatics*, Eger, Hungary, (2004) Vol. II, 221-229.

[50] **Roszik J., Sztrik J., Kim C.S.** Retrial queues in the performance modeling of cellular mobile networks using MOSEL, *International Journal of Simulation: Systems, Science & Technology*, Vol. 6 (2005) 38-46.

[51] **Stepanov S. N.** The analysis of the model with finite number of sources and taking into account the subscriber behaviour, *Automation and Remote Control*, Vol. 55 (1994) 100-113.

[52] **Sztrik J., Almási B., Roszik J.** Heterogeneous finite-source retrial queues with server subject to breakdowns and repairs. *Journal of Mathematical Sciences*, Vol. 132 (2006) 677-685.

[53] **Sztrik J. and Gál T.** A recursive solution of a queueing model for a multi-terminal system subject to breakdowns, *Performance Evaluation*, Vol. 11 (1990) 1-7.

[54] **Sztrik J. and Pósafalvi A.** On the heterogeneous machine interference with limited server's availability, *European Journal of Operational Research*, Vol. 28 (1987) 321-328.

[55] **Takagi H.** *Queueing Analysis, A Foundation of Performance Evaluation, Vol. 2., Finite Systems*, North-Holland, Amsterdam, 1993.

[56] **Tran-Gia P. and Mandjes M.** Modeling of customer retrial phenomenon in cellular mobile networks, *IEEE Journal of Selected Areas in Communications*, Vol. 15 (1997) 1406-1414.

[57] **Trivedi K. S.** *Probability and statistics with reliability, queueing and computer science applications*, Prentice-Hall, Englewood Cliffs, 1982.

[58] **Wang J., Cao J. and Li Q. L.** Reliability analysis of the retrial queue with server breakdowns and repairs, *Queueing Systems Theory and Applications*, Vol. 38 (2001) 363-380.

[59] **Wüchner P.** Extending the interface between the modeling languages MOSEL and CSPL by adding simulation constructs, *Semester Thesis SA-I4-2003-06*, Department of Computer Science, University of Erlangen, Germany, 2003.

[60] **Wüchner P.** Performance modelling of mobile networks using MOSEL-2, *M.S. Thesis*, Department of Computer Science, University of Erlangen, Germany, 2004.

[61] **Zreikat A. I., Bolch G., Sztrik J.** Performance Modeling of Non-homogeneous Unreliable Multi-Server Systems Using MOSEL, *Computers and Mathematics with Applications*, Vol. 46 (2003) 293-312.

Appendix A

Publications of the Author

International Journal Papers

[J1] **Roszik J.** Homogeneous finite-source retrial queues with server and sources subject to breakdowns and repairs, *Annales Univ. Sci. Budapest., Sect. Comp.*, Vol. 23 (2004) 213-227.

[J2] **Roszik J., Sztrik J., Kim C.S.** Retrial queues in the performance modeling of cellular mobile networks using MOSEL, *International Journal of Simulation: Systems, Science & Technology*, Vol. 6 (2005) 38-46.

[J3] **Almási B., Roszik J., Sztrik J.** Homogeneous finite-source retrial queues with server subject to breakdowns and repairs, *Mathematical and Computer Modelling*, Vol. 42 (2005) 673-682.

[J4] **Sztrik J., Almási B., Roszik J.** Heterogeneous finite-source retrial queues with server subject to breakdowns and repairs, *Journal of Mathematical Sciences*, Vol. 132 (2006) 677-685.

[J5] **Roszik J., Sztrik J.** Performance analysis of finite-source retrial queues with non-reliable heterogeneous servers, *Journal of Mathematical Sciences* (submitted for publication)

[J6] **Sztrik J., Roszik J.** Finite-source retrial queueing systems with heterogeneous non-reliable servers and different service policies, *Journal of Mathematical Sciences* (submitted for publication)

[J7] **Roszik J., Sztrik J., Virtamo J.** Performance analysis of finite-source retrial queues operating in random environments, *International Journal of Operations Research* (to appear)

International Conference Papers

[C1] **Roszik J., Almási B., Sztrik J.** Multiserver retrial queues with finite number of heterogeneous sources, *Proceedings of 6th International Conference on Applied Informatics*, Eger, Hungary, (2004) Vol. II, 19-26.

[C2] **Roszik J., Sztrik J.** The effect of server's breakdown on the performance of finite-source retrial queueing systems, *Proceedings of 6th International Conference on Applied Informatics*, Eger, Hungary, (2004) Vol. II, 221-229.

[C3] **Hyytiä E., Lassila P., Penttinen A., Roszik J.** Traffic load in dense wireless multihop network, *Proceedings of The 2nd ACM International Workshop on Performance Evaluation of Wireless Ad Hoc, Sensor, and Ubiquitous Networks (PE-WASUN '05)*, Montreal, Quebec, Canada, (2005) 9-17.

Publications in Hungarian

[H1] **Roszik J.** Visszatéréses sorbanállási rendszerek a telekommunikációs hálózatok modellezésében, *Informatika a Felsőoktatásban 2005*, Debrecen, Hungary, (2005) 6 pages

[H2] **Roszik J.** Számítógép-hálózatok gyakorlati segédanyag, *mobiDIÁK könyvtár*, Debrecen, Hungary, (2005) 43 pages

Appendix B

Conference Presentations

International

[IC1] **Almási B., Roszik J., Sztrik J.** Heterogeneous finite-source retrial queues with server subject to breakdowns and repairs, *XXIII International Seminar on Stability Problems for Stochastic Models*, Pamplona, Spain (2003) (presented by J. Sztrik)

[IC2] **Almási B., Roszik J., Sztrik J.** Homogeneous finite-source retrial queues with server subject to breakdowns and repairs, *5th EURO/INFORMS Joint International Meeting*, Istanbul, Turkey (2003) (presented by J. Sztrik)

[IC3] **Bolch G., Roszik J., Sztrik J.** Heterogeneous finite-source retrial queues in the analysis of communication systems with CSMA/CD protocols, *International Conference on Modern Mathematical Methods of Analysis and Optimization of Telecommunication Networks*, Gomel, Belarus (2003) (presented by J. Sztrik)

[IC4] **Roszik J., Almási B., Sztrik J.** Multiserver retrial queues with finite number of heterogeneous sources, *ICAI '2004 - 6th International Conference on Applied Informatics*, Eger, Hungary (2004) (presented by J. Roszik)

[IC5] **Roszik J., Sztrik J.** The effect of server's breakdown on the performance of finite-source retrial queueing systems, *ICAI '2004 - 6th International Conference on Applied Informatics*, Eger, Hungary (2004) (presented by J. Sztrik)

[IC6] **Roszik J., Sztrik J.** Retrial queues for performance modelling and evaluation of heterogeneous networks, *HET-NETs '04 - The Second International Working Conference on the Performance Modelling and Evaluation of Heterogeneous Networks*, Ilkley, West Yorkshire, U.K. (2004) (presented by J. Sztrik)

[IC7] **Roszik J., Sztrik J.** Performance analysis of finite-source retrial queues with non-reliable heterogenous servers, *XXIV International Seminar on Stability Problems for Stochastic Models*, Jurmala, Latvia (2004) (presented by J. Sztrik)

[IC8] **Hyytiä E., Lassila P., Penttinen A., Roszik J.** Traffic load in dense wireless multihop network, *The 2nd ACM International Workshop on Performance Evaluation of Wireless Ad Hoc, Sensor, and Ubiquitous Networks (PE-WASUN '05)*, Montreal, Quebec, Canada (2005) (presented by E. Hyytiä)

Hungarian

[HC1] **Roszik J.** Nem-megbízható kiszolgálót és forrásokat tartalmazó visszatéréses sorbanállási rendszerek teljesítményelemzése, *XXVI. Operációkutatási Konferencia*, Győr, Hungary (2004) (presented by J. Roszik)

[HC2] **Sztrik J., Roszik J.** Véges forrású visszatéréses sorbanállási rendszerek különböző nem-megbízható kiszolgálókkal és kiszolgálási elvekkel, *XXVI. Operációkutatási Konferencia*, Győr, Hungary (2004) (presented by J. Sztrik)

[HC3] **Roszik J.** Visszatéréses sorbanállási rendszerek a telekommunikációs hálózatok modellezésében, *Informatika a Felsőoktatásban 2005*, Debrecen, Hungary (2005) (presented by J. Roszik)

Retrial Queues and their Application in Performance Modelling of Communication Networks

Értekezés a doktori (PhD) fokozat megszerzése érdekében a
matematika- és számítástudományok tudományágban.

Írta: Roszik János okleveles programtervező matematikus

Készült a Debreceni Egyetem Matematika- és Számítástudományok doktori iskola
Informatikai rendszerek és hálózatok programja
keretében

Témavezető: Dr. Sztrik János

A doktori szigorlati bizottság:

elnök:	Dr.	.....
tagok:	Dr.	.....
	Dr.	.....

A doktori szigorlat időpontja 200... ....

Az értekezés bírálói:

	Dr.	.....
	Dr.	.....
	Dr.	.....

A bírálóbizottság :

elnök:	Dr.	.....
tagok:	Dr.	.....
	Dr.	.....
	Dr.	.....
	Dr.	.....

Az értekezés védésének időpontja: 200... ....