*Annales Univ. Sci. Budapest., Sect. Comp. 22 (2003) 23-33*

# ANALYZING MARKOV–MODULATED FINITE SOURCE QUEUEING SYSTEMS

**B. Almási** (Debrecen, Hungary)
**G. Bolch** (Erlangen, Germany)
**J. Sztrik** (Debrecen, Hungary)

*Dedicated to Professor Karl-Heinz Indlekofer on his 60th birthday*

**Abstract.** This paper deals with a First-Come-First-Served (FCFS) queueing model to analyze the steady-state behaviour of heterogeneous finite-source queueing system with a single server. The request sources and the server are supposed to operate in random environments, thus allowing the arrival and sevice processes to be Markov-modulated ones. Each request of the sources is characterized by its own exponentially distributed source and service time with parameter depending on the state of the corresponding environment, that is the request generation and service rates are subject to random fluctuations. Our aim is to get the usual stationary performance measures of the system, such as utilizations, mean queue lengths, average response times.

In this paper we describe the mathematical model and introduce a software tool to produce analytical computational results for the investigated model. The MARKMOD software package was built on MOSEL and SPNP (see [6], [9]) and gives an easy way to use Markov-modulated queueing systems for modeling real life computer and communication systems. Finally, we show some numerical examples to illustrate the efficiency of the software tool.

Research is partially supported by German-Hungarian Bilateral Intergovernmental Scientific Cooperation, OMFB-DLR No. 21-2000, György Békési Scholarship BÖ 053/2001, and Hungarian Scientific Research Found OTKA TO-34280/2000 and FKFP grant 0191/2001.

## 1. Introduction

Performance modeling of recent computer and communication system development has become more complicated as the size and complexity of the system has increased, (see [5], [8]). Finite source queueing models are efficiently used for performance evaluation of computer systems (see [1], [5] and [10]). Realistic consideration of certain stochastic systems, however, often requires the introduction of a random environment, sometimes referred as to Markov-modulation, where system parameters are subjected to randomly occuring fluctuations or bursts. This situation may be attributed to certain changes in the physical environment such as personal changes and work load alterations. Gaver *et al.* [7] proposed an efficient computational approach for the analysis of a generalized structure involving finite state space birth-and-death processes in a Markovian environment.

This paper deals with a First-Come-First-Served (FCFS) queueing model to analyze the behaviour of heterogeneous finite-soruce system with a single server. The request sources and the server are supposed to operate in independent random environments, respectively, allowing the arrival and service processes to be Markov-modulated ones. Each request of the sources is characterized by its own exponentially distributed generation and service time with parameter depending on the state of the corresponding environment, that is the request generation and service rates are subject to random fluctuations. Our aim is to get the usual stationary performance measures of the system, such as utilizations, mean queue lengths, average repsonse times. The main problem is that the state space of the underlying continuous-time Markov-chain will be very large, so we have the state space explosion problem. In this paper we describe the queueing model of finite source heterogeneous Markov-modulated systems, and introduce a software tool which can be used efficiently to produce analytical computational results for the investigated model. The MARKMOD (MARKov-MODulated queueing models) software package was built on MOSEL (MOdeling Specification and Evaluation Language) and SPNP (Stochastic Petri Net Program), and it gives an appropriate tool for modeling real life computer and communication systems with Markov-modulated queueing systems.

## 2. The queueing model

Consider a finite-source queueing system wtih $N$ heterogeneous sources (requests) and a single server. The sources and the server operation is influenced by random environments. The server and the requests are collected into $M$ groups ($1 \leq M \leq N+1$). The members of a group may operate in a common random environment. The environmental changes are reflected in the values of the access and service rates that prevail at any point of time. The main objective is to adapt these parameters to respond to random changes effectively and thus maintain derived level of system performance.

The members of group $p$ are assumed to operate in a random environment governed by an ergodic Markov chain $(\xi_p(t),\ t \geq 0)$ with state space $(1, \ldots, r_p)$ and with transition density matrix

$$\left( a^{(p)}_{i_p j_p}, \quad i_p, j_p = 1, \ldots, r_p, \quad a^{(p)}_{i_p i_p} = \sum_{k \neq i_p} a^{(p)}_{i_p k} \right).$$

Whenever the environmental process $\xi_p(t)$ is in state $i_p$ the probability that source $c$ (a member of group $p$) generates a request in the time interval $(t, t+h)$ is $\lambda_c(i_p)h + o(h)$, $p = 1, \ldots, M$. Each request is transmitted to a server where the service immediately starts if it is idle, otherwise a queueing line is formed. The service discipline is First-Come-First-Served (FCFS). Assuming, that the server belongs to group 1 and the environmental process $\xi_1(t)$ is in state $i_1$ the probability that the service of the request originating from source $c$ is completed in time interval $(t, t+h)$ is $\mu_c(i_1)h + o(h)$.

If a given source has sent a request it stays idle and it can not generate another one. After having serviced each request immediately returns to its source and the whole procedure starts again. All random variables involved here and the random environments are supposed to be independent of each other.

The system state at time $t$ can be described by the process

$$X(t) = (\xi_1(t), \ldots, \xi_M(t); Z(t)),$$

where $\xi_j(t)$ denotes the states of the background processes $(j = 1, \ldots, M)$, and $Z(t)$ contains the indices of the job sources staying at the server (in the order of their generation), or $Z(t) = 0$, if the server is idle.

The steady state probabilities can be denoted by

$$P(b_1, \ldots, b_M; j_1, \ldots j_l) =$$

$$= \lim_{t \to \infty} P(\xi_1(t) = b_1, \ldots, \xi_M(t) = b_M; Z(t) = (j_1, \ldots, j_l)),$$

$l = 0, \ldots, N;\ j_i \in \{1, \ldots, N\},\ i = 1, \ldots, l;\ b_s = 1, \ldots, r_s,\ s = 1, \ldots, M.$

The next sections will discuss the MOSEL and the MARKMOD software tools which can be used efficiently to formulate the problem and to calculate the steady state probabilities. Once we have these probabilities the main system performance measures can be derived as follows.

Let us define

$$P(j_1, \ldots, j_l) = \sum_{i_1=1}^{r_1} \cdots \sum_{i_M=1}^{r_M} P(b_{i_1}, \ldots, b_{i_M}; j_1, \ldots, j_l),$$

that is $P(j_1, \ldots, j_l)$ denotes the probability, that the requests from sources $j_1, \ldots, j_l$ are at the service station.

(i) Average length of the server's queue

$$n_j = \sum_{l=1}^{N} \sum_{j_1, \ldots, j_l \in V_N^l} l P(j_1, \ldots, l_l),$$

where $V_N^l$ denotes the set of all $(j_1, \ldots, j_l)$ indices (as defined above).

(ii) Utilization of the server

$$U_s = \sum_{l=1}^{N} \sum_{j_1, \ldots, j_l \in V_N^l} P(j_1, \ldots, j_l).$$

(iii) Utilization of the source $i$

$$U_i = \sum_{l=1}^{N} \sum_{j_1, \ldots, j_l \in V_N^l} P(j_1, \ldots, j_l) \prod_{k=1}^{l} (1 - \delta(i, j_k)),$$

where $\delta(a, b) = 1$, if $a = b$, and 0 otherwise.

(iv) Probability of staying at the service facility (for the source $i$)

$$Q_i = \sum_{l=1}^{N} \sum_{j_1,\ldots,j_l \in V_N^l} \sum_{k=1}^{l} \delta(i, j_k) P(j_1, \ldots, j_l).$$

(v) Throughput at the source $i$ (assuming, that the parameters of the source $i$ are modulated by the background process $b$)

$$\gamma_i = \sum_{i_1=1}^{r_1} \cdots \sum_{i_b=1}^{r_b} \cdots \sum_{i_M=1}^{r_M} \sum_{l=1}^{N} \sum_{j_1,\ldots,j_l \in V_N^l}$$

$$\lambda_i(i_b) P(b_{i_1}, \ldots, b_{i_M}; j_1, \ldots, j_l) \prod_{k=1}^{l} (1 - \delta(i, j_k)).$$

(vi) Average response time (for the source $i$)

$$T_i = \frac{Q_i}{\gamma_i}.$$

Similar models were studied earlier by different authors (see [7], [11], [12]), but usually simulation tools were used to calculate numerical results. In the followings we shortly introduce a software tool for numerical investigations.

## 3. MOSEL – The language environment of the implementation

This section gives a short description of MOSEL - the most important basics of our software tool. The language and compiler MOSEL (MOdeling Specification and Evaluation Language) was developed at the University of Erlangen. The MOSEL system uses a macro-like language (see [9] and [5]) tuned especially to describe stochastic Petri nets. The stochastic Petri nets are widely used in the world of stochastic modeling, so MOSEL can become a popular tool of this area. We give a short overview on the structure of a MOSEL program, which is useful for studying the MOSEL implementation of the different models discussed later in this paper.

The MOSEL programs consist of four parts: the declarations, the node definitions, the transitions rules and the results. The MOSEL source code begins with the declarations, where we can define the most important types

and constants. In the following example we define the number of background processes to 2, and then we define the number of states of the background processes to 3:

```
/*======= Constants for the background processes =====*/
#define NBG 2
#define NBGST1 3
#define NBGST2 3
```

The second part of a MOSEL program is the node part. In this part we define the nodes for the system (i.e. the place of tokens in the stochastic Petri Net terminology). We can use the constants and enum types defined in the declaration part. We can also give initial values for the nodes (i.e. the number of tokens), as you can see in the following examples:

```
/*======= Node Definition - background processes =======*/
<1..NBG>  NODE  bgp<#1>[NBGST<#1>]=0;
```

In this example we used a shortcut, which forms a cycle form 1 to NBG.

The transitions of the Petri Net are defined in the transition rule part. This is the most important part of the MOSEL program, which describes the system's behavior using FROM ... TO style rules. These rules give an easy implementation of the state-space transitions. Detailed description can be found in [4] for further details. Here is a small example of the rule part:

```
/*=========== The rule part begins ===========*/
/*Example - changing the state of the first background process.*/
<1..NBGST1> IF bgp1 !=0 FROM bgp1 (<#1>) TOE W bgmat1_<#1>_0;
```

At the end of the MOSEL program the result part can be found. This section calculates the output results. The results are specified by equations, giving the name of the "output variable" on the left side, and the formula on the right side. On the right side we can use the word PROB to refer to the steady-state probability of the given state. The form **RESULT>>** will print out the output variable (otherwise it will not appear in the output file).

```
/*=========== The results part begins ===========*/
   RESULT if (bgp1 == 0) p_up+=PROB;
```

## 4. The MARKMOD Software Tool

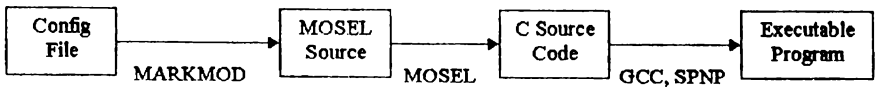


*Figure 1.*

The software tool introduced here can produce numerical results for the described model. The system consists of three parts: MARKMOD, MOSEL, SPNP (see Figure 1) and can be used in any unix-like environment.

The program MARKMOD implements the above mentioned Markov-modulated queueing model. The input of MARKMOD is a very simple and well commented parameter file (text file), from which the program generates a MOSEL source file. The generated MOSEL source is very complicated and not so easy to understand, needs deep knowledge of the language. This MOSEL source file is the input for the MOSEL system (i.e. "MOSEL compiler"). The output of the MOSEL system is a C program, which uses the SPNP library system to solve the Markov chains for the model. SPNP is a powerful mathematical library (see [6] and [13]), written in C and containing well-tuned Markov chain solvers. Using SPNP we can generate an executable program from the MOSEL output with a C compiler. This executable program will produce numerical results for the investigated model.

The three systems work together in the backround. The only neccessary user interaction is to edit the configuration file (which contains only the input parameters) and to issue the MARKMOD command. Then we will get the numerical results automatically.

## 5. Numerical example - Terminal system with server breakdown

In this case we investigate a client-server system with server's breakdowns. We consider a queueing system consisting of $n$ clients connected with a server. Client $i$ has exponentially distributed request generation and processing times with rate $\lambda_i$ and $\mu_i$, respectively. Let us assume that the server is subject to random breakdowns stopping the server's service and the client's work, too. The failure-free operation times and repair times are exponentially distributed

random variables with parameters $\alpha$ and $\beta$. We investigate here how the server's repair parameter ($\beta$) influences the utilizations and the response times. The detailed model description can be found in [1] and [2].

To implement this system in MARKMOD we collect the clients and the server into one group influenced by a background process. The background process has two states: 0 means that the server is operational, 1 means that the system is down. If the server is down (i.e. the background process is in state 1) then the request generation and service intensities are very small (0.001), thus approximating the stop of the work at the server and at the clients.

**Input parameters**

| $n = 4$ | $\alpha = 0.25$ | $\beta = 0.1 - 0.5$ |
|---|---|---|

| $i$ | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| $\lambda_i(0)$ | 0.500 | 0.400 | 0.300 | 0.200 |
| $\mu_i(0)$ | 0.900 | 0.800 | 0.600 | 0.500 |
| $\lambda_i(1)$ | 0.001 | 0.001 | 0.001 | 0.001 |
| $\mu_i(1)$ | 0.001 | 0.001 | 0.001 | 0.001 |

We made experiment series in this case to illustrate how the server's repair parameter (i.e. the server repair efficiency) influences the utilizations and response times (see Figure 2). It can be seen, that the utilizations are linearly increasing (for the server and for the clienst, too), and the response times are roughly exponentially decreasing with the server's repair parameter. It can also be seen, that the server's utilization is the most sensible one to the parameter variation.

Also the correct impelentation of the model can be verified by this example, because the performance measures are the same as it was in case 3 of [1]. It is also an interesting result, that the response time decreases exponentially with the server's repair parameter, but it increases linearly with the server's breakdown parameter (see [3]).

The running time of the experiment series (9 experiments!) was less than 1 minute, which confirms, that the software tool is really appropriate for such problems.

## 6. Conclusion

In this paper a Markov-modulated finite source non-homogeneous queueing

**Performance measures (utilizations, response times)**

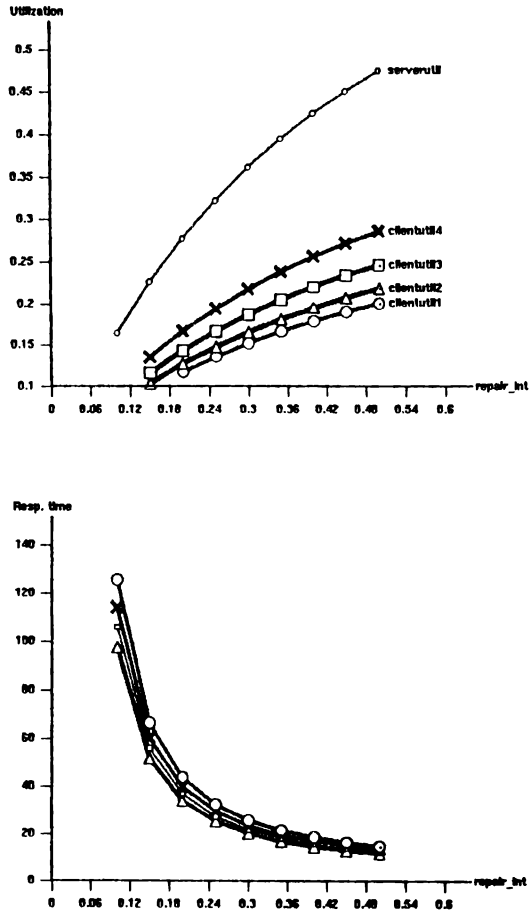


*Figure 2.*

model has been treated. Also a software tool is introduced (based on MOSEL and SPNP) which can be used to calculate numerical results for the model. Furthermore, a numerical example illustrates the problem in question and confirms, that the tool is useful for performance analysis of such systems.

## References

[1] **Almási B.,** A queueing model for a non-homogeneous terminal system subject to breakdowns, *Computers and Mathematics with Applications,* **25** (4) (1993), 105-111.

[2] **Almási B.,** Response time for the finite heterogeneous nonreliable queueing systems, *Computers and Mathematics with Applications,* **31** (11) (1996), 55-59.

[3] **Almási B., Bolch G. and Sztrik J.,** Performability modeling a client-server communication system with randomly changing parameters using MOSEL, *Proceedings of PMCCS 5,* University Erlangen-Nürnberg, Erlangen, 2001, 37-41.

[4] **Bolch G. and Herold H.,** *MOSEL MOdeling Specification and Evaluation Language*, Technical Report, University Erlangen-Nürnberg, 1999.

[5] **Begain K., Bolch G. and Herold H.,** *Practical performance modeling,* Kluwer, 2001.

[6] **Ciardo G. and Muppala J.K.,** *Manual for the SPNP package, Version 3.1,* Duke University, Durham, NC, USA, 1991.

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

[8] **Haverkort B.,** *Performance of computer communication systems,* John Wiley & Sons, 1998.

[9] **Herold H.,** *MOSEL A universal language of modeling computer, communication and manufacturing systems,* Phd dissertation, University Erlangen, 2000.

[10] **Kameda H.,** A finite-source queue with different customers, *J. ACM,* **29** (1982), 478-491.

[11] **Sztrik J. and Kouvatsos D.D.,** Asymptotic analysis of a heterogeneous multiprocessor system in a randomly changing environment, *IEEE Transactions on Software Engineering,* **17** (1991), 1069-1075.

[12] **Sztrik J. and Moeller O.,** A tool for simulation of Markov-modulated finite-source queueing systems, *Proc. of Messung Modellirung und Berwertung (MMB99),* Trier, Germany, 1999, 109-114.

[13] **Trivedi K.S. and Ciardo G.,** *A decomposition approach for stochastic reward net models,* Duke University, Durham, NC, USA, 1991.

**B. Almási**
Institute of Informatics
University of Debrecen
H-4010 Debrecen, Pf. 12
Hungary
almasi@math.klte.hu

**G. Bolch**
Dept. of Computer Scince IV
University of Erlangen-Nürnberg
Martenstr. 1
D-91058 Erlangen, Germany
bolch@informatik.uni-erlangen.hu

**J. Sztrik**
Institute of Informatics
University of Debrecen
H-4010 Debrecen, Pf. 12
Hungary
jsztrik@math.klte.hu