Journal of Mathematical Sciences, Vol. 105, No. 6, 2001

STOCHASTIC SIMULATION OF MARKOV-MODULATED FINITE-SOURCE QUEUEING SYSTEMS*

J. Sztrik (Debrecen, Hungary) **and O. Möller** (Trier, Germany) UDC 519.2

1. Introduction

1.1. Problem formulation. Let us consider a system with n parallel running machines and a server. Each of the machines operates for a random time and then breaks down. When this happens, a request is sent to the server. Thus, we have a queueing system of finite-source type originating from the so-called machine interference problem, which is why we use its terminology. However, this system has often been used for mathematical modeling of different problems encountered in computer and communication sciences (see [4, 6, 8, 9, 11]). For easier understanding, it can be seen in the following picture:

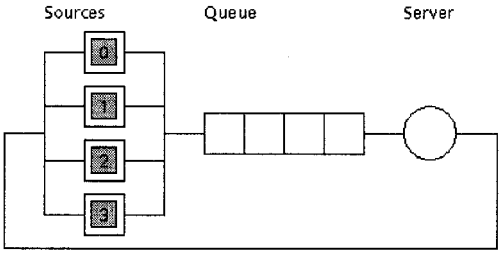

Fig. 1

Now we state the maximum number m of stopped machines for an operating system. If more than m machines have broken down at a certain point of time, then we consider the whole system as nonoperating or "down." Otherwise it is "up." We are interested in the main steady-state performance measures of the system, such as server utilization, machine utilization, mean waiting times or mean response times, the probability of an operating system or the mean system operating time. This is a special kind of level crossing problem.

Besides the processes in the queueing system, one or more Markov chains run in the background to provide the random environment of each machine and the server. The machines are stochastically heterogeneous, that is, each machine is characterized by its own operating and repair times. The jth machine runs through a fixed number A_j of phases until it breaks down. Each phase k_j is exponentially distributed with a parameter $\lambda_j(i, k_j)$ that is dependent of the state i of the machine's random environment and the phase, that is, it is hypoexponentially distributed for a given state of the governing process.

Similarly, the service process for machine j takes S_j phases, which are exponentially distributed with parameters $\mu_j(i, k_j)$. These are dependent on the state of the random environment of the server and the phase; thus, they are also hypoexponentially distributed.

In particular, the exponential, the Erlangian times without random environments can be obtained which were used to test the proposed procedure.

It should be noted that such systems with exponentially distributed running and repair times have been treated in [1, 10] by using asymptotic methods, and in [5], by applying a numerical approach. Since there is a huge amount of literature on performance evaluation of computer and communication systems, reliability aspects of complex systems, and, furthermore, solution methods of different queueing situations, the authors restrict their interest to [3, 4, 7], in which the interested reader can find the relevant materials. The need for such a special simulation tool is of practical

*Supported by the Hungarian Ministry of Education (grant No. FKFP 4/1999) and German–Hungarian Bilateral Intergovernmental Scientific Cooperation (grant No. 015-97).

Proceedings of the Seminar on Stability Problems for Stochastic Models, Nalęczow, Poland, 1999, Part I.

interest, since to the best knowledge of the authors such finite-source queueing systems evolving in random environments cannot be simulated by the existing tools, which are too general.

1.2. Random environments. Three different constellations of random environment types are possible in the simulation. They differ in the number of Markov chains used. The involved random environments are assumed to be independent of each other. In the following pictures, a system of four machines is considered. Every ellipse represents a single random environment.

1. One random environment: one for all sources and the server (see Fig. 2)

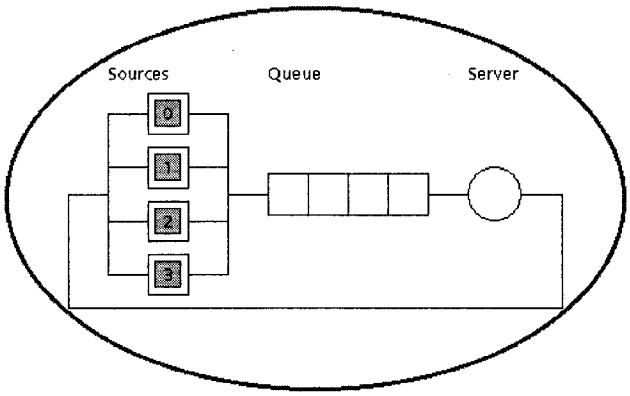

Fig. 2

2. Two independent random environments: one for all sources and one for the server (see Fig. 3)

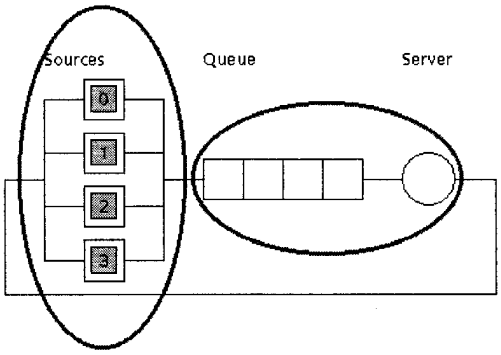

Fig. 3

3. Many independent random environments: one for each source and one for the server (see Fig. 4)

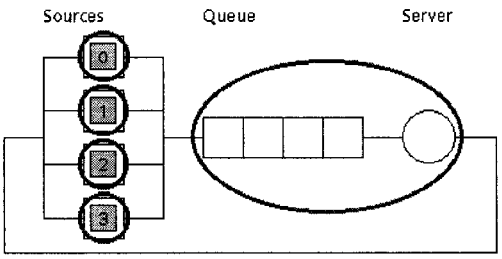

Fig. 4

2. The Simulation Tool

2.1. General remarks.

• System Requirements:

The simulation tool lcpSim was developed for the operating systems Linux and Windows. The source code is written in $C++$ and may be compiled with the GNU project $C++$ Compiler $g++$.

The Qt library of Troll Tech AS [12] is essential to run the programs, as it provides the classes for the graphical interface.

• Simulation model:

The simulation uses an event-oriented model. These events may be request productions, the start of a request service by a server or similar incidents. Whenever a new future event is produced, it is registered in the Future Event Set (FES). All these events are handled in their temporary order.

2.2. Input format. The input format for lcpSim is now explained in detail. For this purpose, the syntax is specified by declaring the expression *input*.

Main expression: $input =$

PREFERENCES·

ENVIRONMENT · (ONE · one + TWO · two + EACH · $each$)

The expression for one random environment: $one =$

MARKOV_CHAIN·

STATES · $states_{int}$·

START · $start_{int}$·

GENERATOR · $((generator^{(ij)}_{double})_{0\le j\le states_{int}})_{0\le i<states_{int}}$·

SOURCES·

NUMBER · $sources_{int}$·

{SOURCE·

PHASES · $phases^{(s)}_{int}$·

RATES · $((rate^{(s)(ij)}_{double})_{0\le j\le states_{int}})_{0\le i<phases^{(s)}_{int}}$·

WEIGHT · $weight^{(s)}_{double}\}_{0\le s<sources_{int}}$·

SERVER·

PHASES · $phases_{int}$·

RATE · $(((rate^{ijk}_{double})_{0\le k<states_{int}})_{0\le j<phases_{int}})_{0<i<sources_{int}}$

The expression for two random environments: $two =$

SOURCES·

MARKOV_CHAIN·

STATES · $states^{(0)}_{int}$·

START · $start^{(0)}_{int}$·

GENERATOR · $((generator^{(0)(ij)}_{double})_{0\le j<states^{(0)}_{int}})_{0\le i<states^{(0)}_{int}}$·

NUMBER · $sources_{int}$·

{SOURCE·

PHASES · $phases^{(s)}_{int}$·

RATES · $((rate^{(s)(ij)}_{double})_{0\le j<states^{(0)}_{int}})_{0\le i<states^{(0)}_{int}}$·

WEIGHT · $weight^{(s)}_{double}\}_{0\le s<sources_{int}}$·

SERVER·

MARKOV_CHAIN·

STATES · $states_{int}$·

START · $start_{int}$·

GENERATOR · $((generator^{(ij)}_{double})_{0\le j<states_{int}})_{0<i<states_{int}}$·

PHASES · $phases_{int}$·

RATES · $(((rate^{(ijk)}_{double})_{0\le k<states_{int}})_{0\le j<phases_{int}})_{0\le i<sources_{int}}$

The expression for many random environments: $each =$

SOURCES·

NUMBER · $sources_{int}$·

{SOURCE·

MARKOV_CHAIN·

STATES · $states_{int}^{(s)}$·
START · $start_{int}^{(s)}$·
GENERATOR · $((generator_{double}^{(s)(ij)})_{0\le j<states_{int}^{(s)}})_{0\le j<states_{int}^{(s)}}$·
PHASES · $phases_{int}^{(s)}$·
RATES · $((rate_{double}^{(s)(ij)})_{0\le j<states_{int}^{(s)}})_{0\le i<phases_{int}^{(s)}}$·
WEIGHT · $weight_{double}^{(s)}\}_{0\le s<sources_{int}}$·
SERVER·
MARKOV_CHAIN·
STATES · $states_{int}$·
START · $start_{int}$·
GENERATOR · $((generator_{double}^{(ij)})_{0\le j<states_{int}})_{0\le i<states_{int}}$·
PHASES · $phases_{int}$·
RATES · $(((rate_{double}^{(ijk)})_{0\le k<states_{int}})_{0\le j<phases_{int}})_{0\le i<sources_{int}}$

2.3. Screenshot. The following picture is a screenshot of lcpSim:

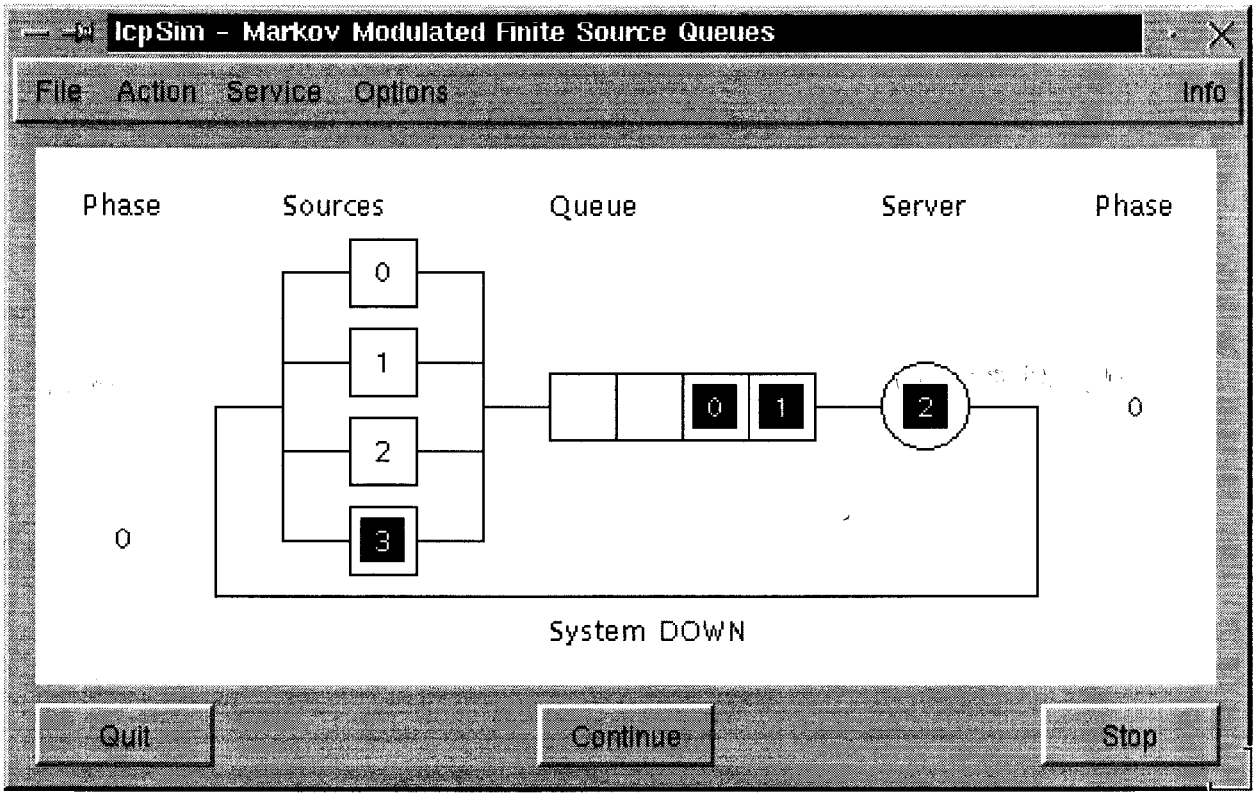

Fig. 5

It is a system with four machines. One of them is currently working, the others have broken down. Machine 2 is being repaired at the moment, while machines 0 and 1 are still waiting to be served.

2.4. Test results. 2.4.1. *Input file.* Here is an input example for the system above. One random environment was used for all components with one state. It was used for testing the simulation to numerical results treated and obtained in [2].

PREFERENCES		
SEED	12345	// SEED OF RANDOM NUMBER GENERATOR
STATISTICS	0.0	// INTERVAL OF STATISTICAL OUTPUT
DURATION	990000.0	// DURATION OF SIMULATION
MAXIMUM_DOWN	2	// MAXIMUM NUMBER OF STOPPED MACHINES FOR OPERATING SYSTEM
ENVIRONMENT	ONE	// ONE MARKOV CHAIN FOR ALL COMPONENTS

MARKOV_CHAIN		// MARKOV CHAIN FOR THE WHOLE SYSTEM
STATES	1	// NUMBER OF DIFFERENT STATES
START	0	// STATE AT START OF SIMULATION
GENERATOR	0	// GENERATOR MATRIX OF MARKOV CHAIN

SOURCES		
NUMBER	4	// NUMBER OF DIFFERENT SOURCES
SOURCE		
PHASES	1	// ARRIVAL PHASES OF SOURCE 0
RATES	0.5	// ARRIVAL RATES OF SOURCE 0
SOURCE		
PHASES	1	// ARRIVAL PHASES OF SOURCE 1
RATES	0.4	// ARRIVAL RATES OF SOURCE 1
SOURCE		
PHASES	1	// ARRIVAL PHASES OF SOURCE 2
RATES	0.3	// ARRIVAL RATES OF SOURCE 2
SOURCE		
PHASES	1	// ARRIVAL PHASES OF SOURCE 3
RATES	0.2	// ARRIVAL RATES OF SOURCE 3

SERVER		
PHASES	1	// SERVICE PHASES OF SERVER
RATES	0.9	
	0.7	
	0.6	
	0.5	// SERVICE RATES OF SERVER FOR SOURCES

2.4.2. *Simulation outputs.* Here are the output results for the above system applying the FIFO, PPS, and Polling disciplines. The results were collected from three output files.

	FIFO	PPS	Polling
Machine 0			
- Utilization	0.38126	0.43099	0.37759
- Mean waiting time	2.11547	1.53210	2.18945
- Mean response time	3.23125	2.63986	3.30003
Machine 1			
- Utilization	0.41511	0.42659	0.41491
- Mean waiting time	2.08879	1.94006	2.09878
- Mean response time	3.52168	3.36345	3.52739

Machine 2			
- Utilization	0.46913	0.45460	0.47123
- Mean waiting time	2.10896	2.34364	2.07313
- Mean response time	3.77447	4.00310	3.74011
Machine 3			
- Utilization	0.54697	0.50095	0.55228
- Mean waiting time	2.15858	2.98768	2.05381
- Mean response time	4.15430	4.97915	4.05398
Server Utilization	0.90351	0.90712	0.902993
Mean number			
of stopped machines	2.18751	2.18685	2.18398
System P (up)	0.57217	0.57373	0.57349
P (down)	0.42782	0.42627	0.42651
- Mean up time	3.04898	2.95635	3.07433
- Mean down time	2.27981	2.19665	2.28635
990000 End of simulation			

2.4.3. *Computational results.*

	FIFO	PPS	Polling
Machine 0			
- Utilization	0.3825	0.4293	0.3772
- Mean response time	3.2290	2.6584	3.3018
Machine 1			
- Utilization	0.4163	0.4234	0.4147
- Mean response time	3.5058	3.4047	3.5290
Machine 2			
- Utilization	0.4692	0.4518	0.4713
- Mean response time	3.7716	4.0450	3.7383
Machine 3			
- Utilization	0.5462	0.5001	0.5521
- Mean response time	4.1548	4.9985	4.0562
Server Utilization	0.9034	0.9064	0.9030
Mean number			
of stopped machines	2.1859	2.1954	2.1847

2.4.4. *Comments.* As can be seen, the simulation results are generally exact up to the 3rd digit, so we think that the tool operates well.

2.5. More complex systems. This is a description of a system with random environments for every single component. The two sources use Markov chains with different state numbers, and they run through different numbers of phases before producing a request. For Priority PS service discipline, each source gets a weight, providing its priority. The service depends on the source number, the current service phase, and the state of the server's Markov chain.

2.5.1. *Input file.*

PREFERENCES ENVIRONMENT	EACH	// MARKOV CHAINS FOR EVERY SINGLE // COMPONENT, PROVIDING THE ENVIRONMENT

```
SOURCES
  NUMBER                2                  // NUMBER OF DIFFERENT SOURCES

SOURCE                                     // SOURCE 0
  MARKOV_CHAIN                             // MARKOV CHAIN 0 FOR SOURCE 0
  STATES                3                  // NUMBER OF DIFFERENT STATES (0, 1, 2)
  START                 1                  // STATE AT START OF SIMULATION
  GENERATOR             −0.8  0.2  0.6
                         0.1 −0.4  0.3
                         0.3  0.2 −0.5      // GENERATOR MATRIX OF MARKOV CHAIN
                                            // ROW i: PHASE TRANSITION RATES IN
                                            // STATE i; i = 0, 1, 2
  PHASES                2                  // NUMBER OF ARRIVAL PHASES (0, 1)
  RATES                  0.1  0.8  1.2
                         0.2  0.9  1.5      // ROW i: RATES IN PHASE i FOR
                                            // STATE 0, 1, 2; i = 0, 1
  WEIGHT                2.0                // WEIGHT FOR PRIORITY PS SERVICE

SOURCE                                     // SOURCE 1
  MARKOV_CHAIN                             // MARKOV CHAIN 1 FOR SOURCE 1
    STATES              2                  // NUMBER OF DIFFERENT STATES (0, 1)
    START               0                  // STATE AT START OF SIMULATION
    GENERATOR           −0.3  0.3
                         0.7 −0.7          // GENERATOR MATRIX OF MARKOV CHAIN
                                            // ROW i: PHASE TRANSITION RATES IN
                                            // STATE i; i = 0, 1
  PHASES                4                  // NUMBER OF ARRIVAL PHASES (0, 1, 2, 3)
  RATES                  0.4  0.6
                         0.1  0.2
                         0.9  1.1
                         2.0  2.2          // ROW i: ARRIVAL RATES IN PHASE i
                                            // FOR STATE 0, 1; i = 0, 1, 2, 3
  WEIGHT                0.5                // WEIGHT FOR PRIORITY PS SERVICE
```

```
SERVER                                     // SERVER
  MARKOV_CHAIN                             // MARKOV CHAIN 2 FOR SERVER
    STATES              3                  // NUMBER OF DIFFERENT STATES (0, 1, 2)
    START               2                  // STATE AT START OF SIMULATION
    GENERATOR           −0.5  0.2  0.3
                         0.8 −1.6  0.8
                         0.4  0.9 −1.3     // GENERATOR MATRIX OF MARKOV CHAIN
                                            // ROW i: PHASE TRANSITION RATES IN
                                            // STATE i; i = 0, 1, 2
  PHASES                2                  // NUMBER OF SERVICE PHASES (0, 1)
  RATES                  0.9  1.0  1.1
                         0.8  0.9  1.0
                         0.7  0.9  1.2
                         0.6  1.0  0.5     // ROW i IN BLOCK j: SERVICE RATES
                                            // FOR SOURCE j IN PHASE i FOR
                                            // STATE 0, 1, 2; i = 0, 1; j = 0, 1
```

2.5.2. *Output example.* Here is an output example for the system mentioned above for FIFO service. All parameters are listed at the beginning of the output.

lcpSim Simulation Output	
Input File	/home/info04/janos/lcpSim/input/Example System
Environment Type	Each
Source Number	2
Service	FIFO
Simulation Seed	12345
Breakdown Level	0
Statistics Interval	10000
Simulation Duration	30000
Round Interval	1

0 Start of simulation		
10000 Statistical values		
Machine 0	- Utilization	0.583014
	- Mean waiting time	0.756139
	- Mean response time	2.948760
Machine 1	- Utilization	0.715878
	- Mean waiting time	0.643219
	- Mean response time	4.631150
Server	- Utilization	0.553878
Mean number of stopped machines		0.700086
System	- P (up)	0.446122
	- P (down)	0.553878
	- Mean up time	3.106700
	- Mean down time	3.859780

20000 Statistical values		
Machine 0	- Utilization	0.585840
	- Mean waiting time	0.774939
	- Mean response time	2.946880
Machine 1	- Utilization	0.716545
	- Mean waiting time	0.591144
	- Mean response time	4.602660
Server	- Utilization	0.552066
Mean number of stopped machines		0.697367
System	- P (up)	0.447773
	- P (down)	0.552227
	- Mean up time	3.080650
	- Mean down time	3.799290

30000 Statistical values		
Machine 0	- Utilization	0.591184
	- Mean waiting time	0.763134
	- Mean response time	2.935320
Machine 1	- Utilization	0.718040
	- Mean waiting time	0.578126
	- Mean response time	4.613850
Server	- Utilization	0.548961
Mean number of stopped machines		0.690545
System	- P (up)	0.451015
	- P (down)	0.548985
	- Mean up time	3.117610
	- Mean down time	3.794830
30000 End of simulation		

2.6. Operating instructions. After lcpSim is started, the input file Default System is loaded from the input directory. It may be replaced by a different valid input file to save time.

The window is arranged in three parts: a menu bar, a graphical representation of the current simulation state, and finally buttons to control the simulation.

There are five menus with the titles File, Action, Service, Options, and Info.

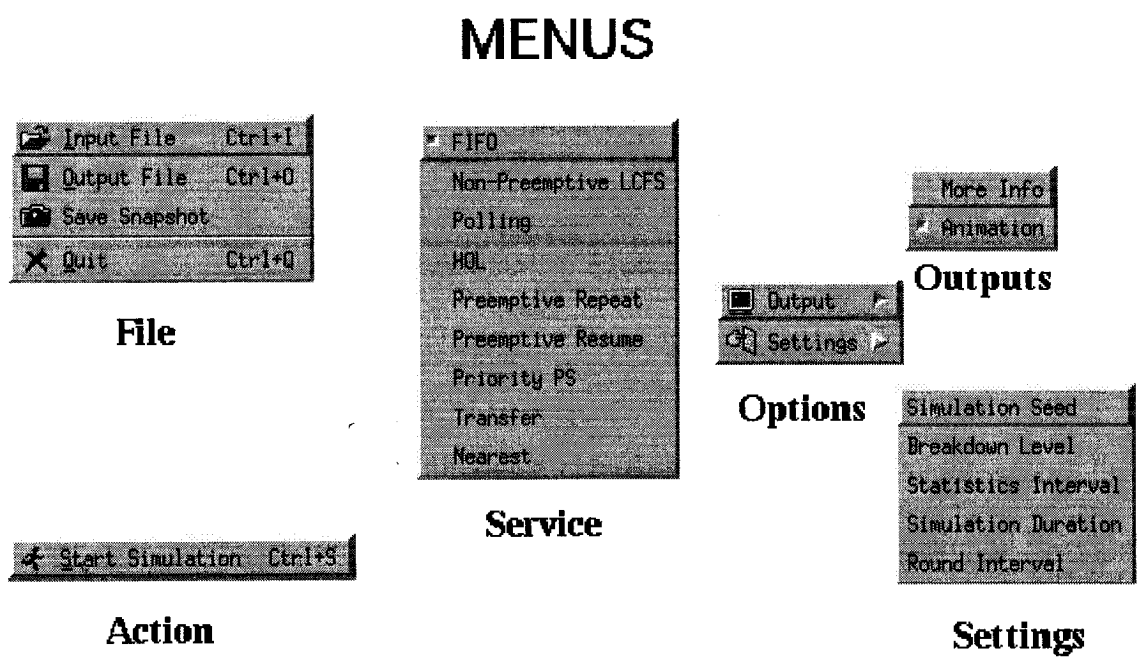

Fig. 6

• The File menu:

Input File: Read new input data. If you choose the file Standard Input Device, then the input is read from the standard input device instead of a file.

Output File: Specify the output destination. If Standard Output Device is chosen, the simulation output is written to the standard output device instead of a file.

Save Snapshot: Save a screenshot of the graphical representation of the system state. The output file will be in the BMP format.

• The Action menu:

Start Simulation: Its activation causes the following proceedings: at first, some menu points are deactivated, so that the input data cannot be changed during the simulation. Then the simulation is initialized, especially the statistical values and the future event set. For example, the production of new requests by the given sources is arranged. Additionally, the buttons Continue and Stop are shown. The simulation process can be interrupted in periodical intervals. At these moments, the simulation can be continued or stopped with these buttons. Only the Quit button is shown all the time.

• The Service menu:

Here the service discipline for the server is specified. The default discipline is FIFO.

FIFO: First In - First Out: the request that has arrived earlier is served earlier, too.

Nonpreemptive LCFS: Last Come - First Served: the request that has arrived at the latest point of time is served first without preempting the one that is under service.

Polling: Go round and round.

Nonpreemptive HOL: Head Of Line: smaller index - higher priority, nonpreemptive priority.

Preemptive Priority (Repeat): If a request with higher priority arrives, then the current service is interrupted. If a request that was served at this moment, its service must begin from the start again later.

Preemptive Priority (Resume): If a request with higher priority arrives, then the current service is interrupted. If a request that was served at this moment, its service is resumed at this point later.

Priority PS: An approximation of a Processor Sharing discipline with priorities. In theory, each request in the queue gets an infinitely small time interval in each service round. In the simulation, a certain time (round interval) is specified and each request in the queue gets a slice of this time according to its weight, which specifies its priority. If a new request arrives, the time slice of the currently served request is newly adapted to the new situation. In the case where the new time slice is already over, the request service is interrupted and the next request is served.
Transfer: Go forward and backwards.
Nearest: Serves the nearest request in the queue from the current position.
• The Options menu:
In this menu, additional input parameters are controlled.
More Info: Increase or decrease the amount of output information that is produced during the simulation process.
Animation: Activates or deactivates the graphical representation of the current simulation state.
Simulation Seed: Set the seed of the random-number generator for the next simulation run.
Breakdown Level: Set the maximum number of stopped machines for an operating system (0: all machines must run).
Statistics Interval: Set the time interval for simulation interruptions to show some statistical values.
Simulation Duration: Set the duration of a simulation run.
Round Interval: Set the time interval for one round with Priority PS service discipline.
• The Info menu:
System Info: Show information about the current system and the parameters that were set. The same information is included as the header of each output file.
Program Info: Show some information about the program lcpSim.

Of course, at the beginning we have to make an Input file, which later can be modified. There is a program inputGEn to do this job. It is an easy-to-use, user-friendly software, correcting possible incorrect data.

REFERENCES

1. V. V. Anisimov and J. Sztrik, "Asymptotic analysis of some controlled finite-source queueing systems," *Acta Cybern.*, **9**, 27–38 (1989).
2. B. Almasi and J. Sztrik, "A queueing model for a nonhomogeneous terminal system subject to breakdowns," *Comp. Math. Appl.*, **25**, 105–111 (1993).
3. G. Bolch, S. Greiner, H. de Meer, and K. S. Trivedi, *Queueing Networks and Markov Chains: Modeling and Performance Evaluation with Computer Science Applications*, Wiley, New York (1998).
4. J. H. Dshalalow, *Frontiers in Queueing, Models, and Applications in Science and Engineering*, CRC Press, Boca Raton (1997).
5. D. P. Gaver, P. A. Jacobs, and G. Latouche, "Finite birth-and-death models in randomly changing environments," *Adv. Appl. Probab.*, **16**, 715–731 (1984).
6. P. Harrison and N. M. Patel, *Performance Modeling of Communication Networks and Computer Architectures*, Addison-Wesley, New York (1993).
7. B. R. Haverkort, *Performance of Computer Communication Systems: A Model-Based Approach*, Wiley, New York (1998).
8. D. D. Kouvatsos, R. Fretwell, and J. Sztrik, "Bounds on the effects of correlation in a stable MMPP/MMPP/1/N queue. An asymptotic approach," in: *Proceedings of Second Workshop on Performance Modeling and Evaluation of ATM Networks*, Chapman and Hall, London (1995), pp. 261–281.
9. I. Mitrani, *Modeling of Computer and Communication Systems*, Cambridge University Press, Cambridge (1987).
10. J. Sztrik, "Modeling of a multiprocessor system in a randomly changing environment," *Perform. Eval.*, **17**, 1–11 (1993).
11. H. Takagi, *Queueing Analysis, A Foundation of Performance Evaluation, Vol. 2: Finite Systems*, North-Holland, Amsterdam (1993).
12. Troll Tech AS, *The Qt Toolkit*, http://www.troll.no/dl.

Institute of Mathematics and Informatics,
Lajos Kossuth University,
Debrecen,
Hungary
e-mail: jsztrik@math.klte.hu

Department IV, Subdepartment of Computer Science,
University of Trier,
D-54286 Trier,
Germany
e-mail: moeller@stud.informatik.uni-trier.de