Yugoslav Journal of Operations Research
12 (2002), Number 2,

SIMULATION OF MACHINE INTERFERENCE IN RANDOMLY CHANGING ENVIRONMENTS

J. SZTRIK

Institute of Mathematics and Informatics, University of Debrecen, Debrecen
jsztrik@math.klte.hu

O. MÖLLER,

Department of Computer Science, University of Trier, Trier, Germany
moeller@stud.informatik.uni-trier.de

Abstract: The simulation tool lcpSim can be used to investigate special level crossing problems of queuing systems of type $HYPO_k$ / $HYPO_r$ / 1 / /n embedded in different Markovian environments (recently referred to as Markov modulated ones). Our observed system consists of n heterogeneous machines (requests) and a server that "repairs" the broken machines according to the most commonly used service disciplines, such as FIFO, LIFO, PPS, HOL, Preemptive Priorities (Resume, Repeat), Transfer, Polling, Nearest. We specify a maximum number of stopped machines for an operating system and our aim is to give the main steady-state performance measures of the system, such as, server utilization, machine utilization, mean waiting times, mean response times, the probability of an operating system and the mean operating time of the system. These values can be calculated by lcpSim level crossing problem Simulation package for different random environment types and service disciplines.

Keywords: Machine interference, finite-source queuing, random environments, simulation, reliability theory.

1. INTRODUCTION

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 got a queuing system of a finite-source type originating from the so-called machine interference problem, that 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, cf. Dshalalow [4], Takagi [9]. Furthermore, we state a 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 non-operating or "down". Otherwise it is "up". In addition to the processes in the queuing 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 $j-$th machine runs through a fixed number A_j of phases until it breaks down. Each phase k_j is exponentially distributed with a parameter $\lambda(i,k_j)$, which is dependent on 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. 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.* Furthermore, the probability of an operating system or the mean system operating time are also given, which is a special kind of level crossing problem for stochastic processes and is often difficult to obtain. It should be noted that such systems without random environments have been investigated by many authors under different assumptions on the distribution of the running and repair times and service rules, see for example, Takagi [9]. Systems embedded in Markovian environments with exponentially distributed running and repair times have been treated, among others, in Anisimov and Sztrik [1], Sztrik and Bunday [7, 8] by using asymptotic methods, and in Gaver et.al. [5] by applying a numerical approach.

Due to the large literature on performance evaluation of the computer and communication systems, reliability aspects of complex systems, furthermore solution methods of different queuing situations, see for example, Bolch et.al. [3], Dshalalow [4] and Haverkort [6], in which the reader can find the relevant materials.

Such a special simulation tool is of a practical interest, since to the best knowledge of the authors such finite-source queuing systems evolving in random environments cannot be simulated by existing, too general tools. The results are new, they are in the streamline of the first author's research. When dealing with such kind of systems the usual approach is to construct an appropriate continuous-time Markov chain. However, its state space will be extremely large, so the traditional methods cannot be used directly. In these cases, for example, different approximation or simulation techniques are applied.

In the present situation we prefer the latter one since for simpler systems we have numerical results for validation. To obtain more realistic mathematical models the failure and repair rates are assumed to vary in time. This can be implemented by the inclusion of random environments which govern the changes of the rates. Actually, it is supposed that these parameters depend on the state of the corresponding continuous-time Markov chain.

Three different constellations of random environment types are possible in the simulation. They differ in the number of Markov chains used.

1. One random environment: one for all sources and the server.
2. Two random environments: one for all sources and one for the server.
3. Several random environments: each source has its own and the server has its own.

2. THE SIMULATION TOOL

- **System Requirements:** The lcpSim simulation tool was developed for Linux and Windows operating systems. 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 [10] is essential to run the programs, as it provides the classes for the graphical interface.
- **Simulation model:** The simulation uses the event-oriented model. These events may be request arrivals, 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.

Of course, we understand that these are only the basics, but our aim is to introduce the tool only and not to give detailed programming tricks. The interested readers can feel free to ask for information and to use the tool. Because of the page limit, our intention is to show the main features of our software and to focus on its flexibility in several practical situations by using different environments, running and repair times as well as various service disciplines.

2.1. Screenshots

The following picture is a screenshot of **lcpSim**. 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.

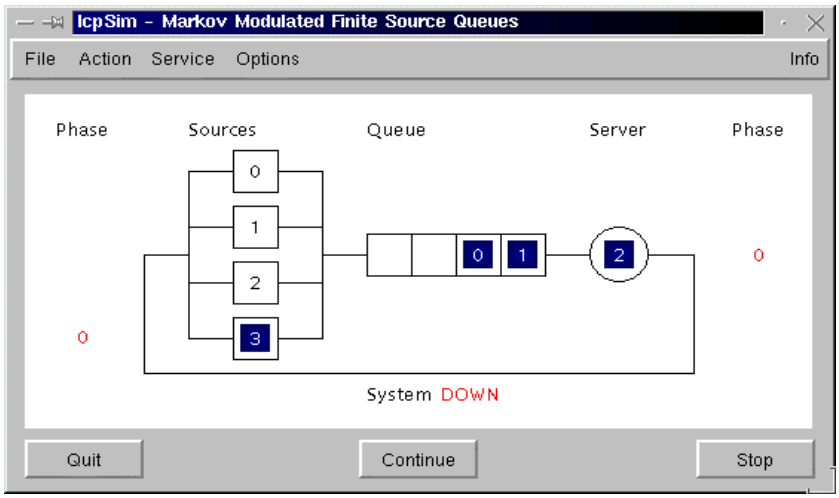

Figure 1: System layout and snapshot

2.2. Test Results

One random environment was used for all components with one state. This model was used for testing the simulation by comparing the simulated outputs with the numerical results treated and obtained in Almási and Sztrik [2].

2.2.1. Input File

```
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
```

2.2.2. Simulation Outputs

Here are the output results for the system above applying FIFO, PPS and Polling disciplines. The results were collected from 3 output files, respectively.

	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 :	4.1543	4.97915	4.05398
Server Utilization :	0.90351	0.90712	0902993
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

2.2.3. Computational Results

The corresponding numerical results are

	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
```

It has been realized that the simulation results are generally exact up to the 3rd digit, so we think that the tool operates well.

2.3. More Complex Systems

This is a very simple illustration of a system with random environments for every single component. The two machines 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 machine gets a weight providing its priority. The service depends on the index of machine, the current service phase and the state of the server's Markov chain. For larger systems the input file has the same structure.

2.3.1. Input File

```
PREFERENCES
  ENVIRONMENT   EACH            // MARKOV CHAINS FOR EVERY SINGLE
                                // 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
```

2.3.2. 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  : 0
         Simulation Duration  : 30000
         Round Interval       : 1

         Machine   0 - Utilization        : 0.591184
                     - Mean waiting time  : 0.763134
                     - Mean response time :  2.93532
         Machine   1 - Utilization        :  0.71804
                     - Mean waiting time  : 0.578126
                     - Mean response time :  4.61385
```

```
Server      - Utilization          : 0.548961
Mean number of stopped machines    : 0.690545
System      - P (up)               : 0.451015
            - P (down)             : 0.548985
            - Mean up time         :  3.11761
            - Mean down time       :  3.79483

30000 End of simulation
```

2.4. 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,* see Figure 1, and several submenus, see Figure 2.

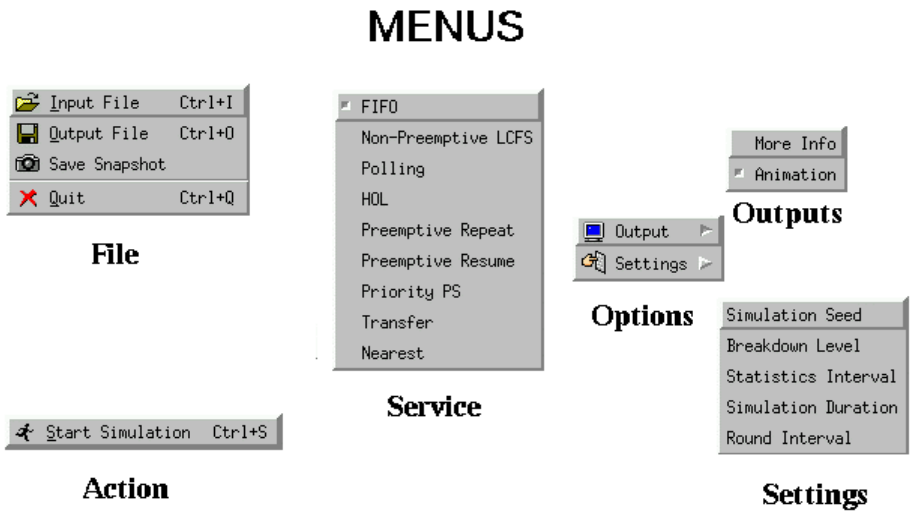

Figure 2: Menu system

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 to 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 starts the following procedures. 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 by pressing these buttons. Only the *Quit* button is shown all the time.

The **Service** menu:

Here the service disciplines for the server are specified. The default discipline is FIFO.
FIFO: First In - First Out: the request that has arrived earlier is served earlier, too.
Non-Preemptive LCFS: Last Come - First Served: the request that has arrived at the latest point of time is served first without preempting the one being under service.
Polling: Go round and round.
Non-Preemptive HOL: Head Of Line: smaller index - higher priority, non-preemptive priority.
Preemptive Priority (Repeat): If a request with higher priority arrives, then the current service is interrupted. The service of the preempted request must begin from the start later on.
Preemptive Priority (Resume): If a request with higher priority arrives, then the current service is interrupted. The service of the preempted request 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 that 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: Increases or decreases 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: Sets the seed of the random number generator for the next simulation run.
Breakdown Level: Sets the maximum number of stopped machines for an operating system (0: all machines must run).
Statistics Interval: Sets the time interval for simulation interruptions to show some statistical values.

Simulation Duration: Sets the duration of a simulation run.
Round Interval: Sets the time interval for one round with Priority PS service discipline.

The **Info** menu

System Info: Shows information about the current system and the parameters which were set. The same information is included as header of each output file.
Program Info: Shows some information about the program **lcpSim**.

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

3. CONCLUSION

Main features of a simulation tool have been introduced whic was developed for the investigation of the machine interference problem evolving in random environment(s). Steady-state performance measures of the system have been obtained under several service disciplines, assuming hypoexponentially distributed running and service times. Some sample input and output files have been shown to make easier the understanding of this user friendly and easy-to-use package.

Acknowledgement: Research is partially supported by Hungarian Scientific Research Fund under grants OTKA-T30685/99, T034280/2000, German-Hungarian Bilateral Intergovernmental Scientific Cooperation, No. 21-2000 and Hungarian Ministry of Education FKFP 0191/2001

REFERENCES

[1] Anisimov, V.V., and Sztrik, J., "Asymptotic analysis of some controlled finite-source queuing systems", *Acta Cybernetica*, 9 (1989) 27-38.
[2] Almasi, B., and Sztrik, J., "A queuing model for a non-homogeneous terminal system subject to breakdowns", *Computers and Mathematics with Applications,* 25 (1993) 105-111.
[3] Bolch, G., Greiner, S., de Meer, H., and Trivedi, K.S., *Queuing Networks and Markov Chains: Modeling and Performance Evaluation with Computer Science Applications*, John Wiley and Sons, New York, 1998.
[4] Dshalalow, J.H., *Frontiers in Queuing, Models and Applications in Science and Engineering*, CRC Press, Boca Raton, 1997.
[5] 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.
[6] Haverkort, B.R., *Performance of Computer Communication Systems: A Model-Based Approach*, John Wiley and Sons, New York, 1998.
[7] Sztrik, J., and Bunday, B.D., "Machine interference problem with a random environment", *Eur. Journ. Oper. Res.*, 65 (1993) 259-269.
[8] Sztrik, J., and Bunday B.D., "Asymptotic analysis of the heterogeneous machine interference problem with random environments", *Applied Mathematical Modelling,* 17 (1993) 105-110.
[9] Takagi, H., *Queuing Analysis, A Foundation of Performance Evaluation, Vol. 2.*, Finite Systems, North-Holland, Amsterdam, 1993.
[10] Troll Tech AS, *The Qt Toolkit*, http://www.troll.no/dl