6^{th} *International Conference on Applied Informatics*
Eger, Hungary, January 27–31, 2004.

Stochastic Simulation of Markov-Modulated Finite-Source Queues in Java Environment

Márk Kósa

Institute of Informatics, University of Debrecen
e-mail: mkosa@inf.unideb.hu

Abstract

The simulation tool lcpSim can be used to investigate special level crossing problems of queueing systems of type $HYPO_k/HYPO_r/\mathbf{1}//n$ embedded in different Markovian environments. 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.

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` (`l`evel `c`rossing `p`roblem `Sim`ulation) for several types of operating and service times of the machines and for different random environments. The simulation uses the Law–Carson algorithm to provide confidence intervals for the main performance measures of the investigated system.

We developed the Java language version of the `lcpSim` program last year. Besides the main application this version also contains an input generator (`inputGen`), both with user-friendly graphical interfaces. The effects of each service discipline on the performance measures of the given systems can easily be observed on the diagrams generated from the output files of the `lcpSim` program in terms of the increasing number of machines.

1. Introduction

The following problem has been considered for a long time and it is of practical importance. 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 queueing system of a finite source type originated from the so-called machine interference problem, that is why we use its terminology.

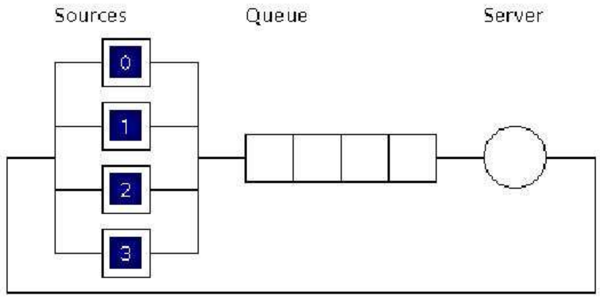


Figure 1: A system with four sources.

1.1. Hypoexponential distributions

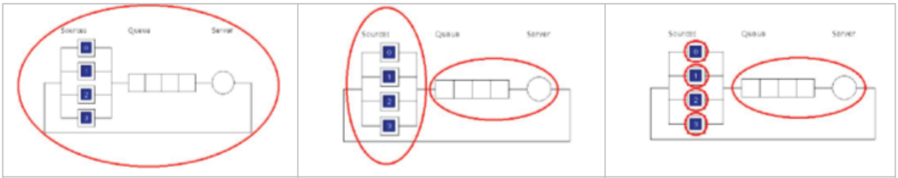
We specify a maximum number of stopped machines for an operating system and our aim is to give the main steady-state performance measures of the system, such as server utilization, machine utilization, mean waiting times, mean response times, the probability of an operating system and the mean operating time of the system.

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 machines 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 kj exponentially distributed with a parameter $\lambda_j(i, k_j)$ that is dependent on 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.

1.2. Random environments

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



(a) (b) (c)

Figure 2: Random environments in the system.
(a) One random environment for all sources and the server.
(b) Two independent random environments:
one for all sources and one for the server.
(c) Many independent random environments:
one for each source and one for the server.

2. The `lcpSim` simulation tool

The simulation tool **`lcpSim`** can be used to investigate special level crossing problems of queueing systems of type $HYPO_k/HYPO_r/\mathbf{1}//n$ 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.

2.1. The graphical user interface

The `lcpSim` tool is an application with graphical user interface (GUI). The main components of its control window are:

- menu bar with `File`, `Action`, `Service`, `Options` and `Info` menus,
- graphical representation of current simulation state,
- buttons for controlling the simulation.

The `File` menu provides loading a previously saved simulation, saving the current simulation, creating a screenshot of the simulation and exiting the program.

The `Action` menu starts the simulation, and in case of animated simulations puts `Continue` and `Stop` buttons on the desktop which control running of the program. The `Continue` button steps over the phases of the simulation and the `Stop` button stops the current simulation.

In the `Service` menu we can choose one of the following disciplines for the simulation: FIFO, non preemptive LCFS, HoL, preemptive repeat, preemptive resume, priority PS, transfer, nearest, polling.

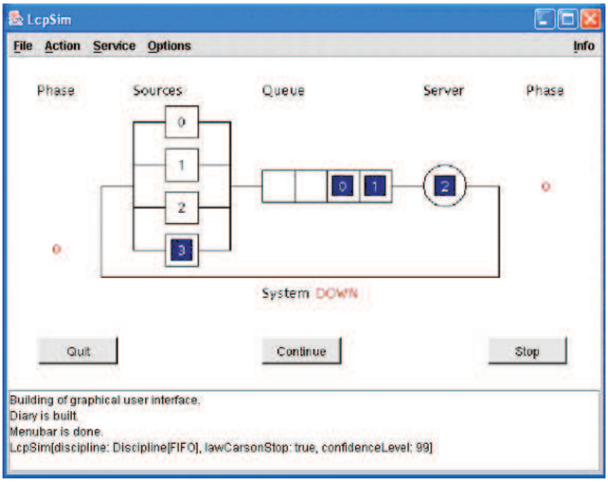


Figure 3: The GUI of `lcpSim` simulation tool. The simulated system consists of four machines, where the machine 3 is working, all of the others are breaking down. Machine 2 is just repaired, machine 0 and machine 1 are waiting to be repaired.

The `Options` menu provides controlling the output information (`Output`) and setting the system attributes (`Settings`). At the end of the simulation the statistical data are written to the standard output or into a given file.

The `Info` menu shows some information about the current parameters of the system and about the authors and programmers of the `lcpSim` tool.

2.2. System attributes

We can specify the parameters of simulation in the `Settings` submenu of the `Options` menu. The following attributes can be set:

- *Law–Carson Stop:* the simulation stops directly when all confidence intervals are found
- *Confidence Level:* confidence level of 90%, 95% and 99%
- *Interval Width:* the maximum width of a valid confidence interval in percentage of the mean value
- *Simulation Seed:* the seed of the random number generator
- *Breakdown Level:* the maximum number of stopped machines
- *Statistics Interval:* the time interval for simulation interruptions to show some statistical values

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

2.3. `InputGen`: the input file generator

Initially we need an input file which we can arbitrarily modify later. A part of the system is the `InputGen` program which we can be used to create such initial input files. This is a easy-to-use user friendly software which filters presumably faulty data from the data entered by the user.

The example below shows how to code a job in the input file:

```
SOURCE
    MARKOV\_CHAIN
        STATES      2
        START       1
        GENERATOR          -1.1    1.1
                            1.5   -1.5
    PARALLEL        1
    SELECTION                1       1
    COX             3
                             1       1
                             1       1
                           0.9     0.5
                           0.6     0.5
                           0.8     0.3
                           0.9     0.2
    WEIGHT          1.2
```

This machine has its own random environment with two phases, the initial phase is phase 1. As now there is only one branching-Erlang-system, this one will be chosen with a probability of 1 in both Markov-chain phases.

The machine works through 3 phases, until it stops. As the Markov chain in the random environment has two different states, there are 6 possible combinations of the machine's phases and the Markov chain's phases. We give the probabilities of job generation for each of these combinations. We only use the weight values for the priority PS service discipline.

2.4. Output

The statistical data will be written to a predefined output file (or to the standard output, if no output file is defined) at the end of the simulation. Data can be represented graphically, see Figure 5.

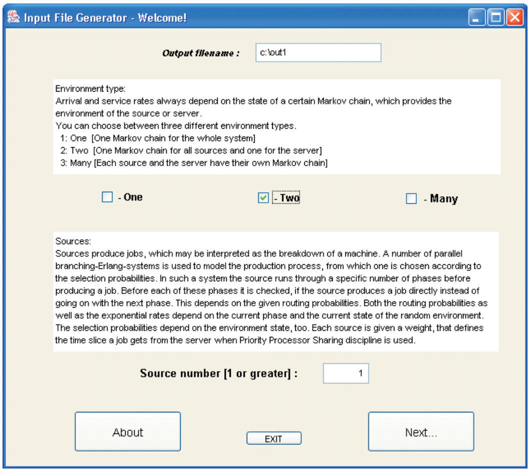


Figure 4: A screenshot of the `InputGen` program.

Here is an output example for the system represented by the screenshot above. All parameters are listed at the beginning of the output.

```
lcpSim Simulation Output
       Input File          : input/Default System
       Environment Type    : One
       Source Number       : 4
       Service             : FIFO
       Simulation Seed     : 12345
       Breakdown Level     : 2
       Statistics Interval : 0
       Simulation Duration : 1e+07
       Round Interval      : 1

     0 Start of simulation
 1e+07 Statistical values
       Machine    0 - Utilization        : 0.382663
                    - Mean waiting time  : 2.11572
                    - Mean response time : 3.22638
       Machine    1 - Utilization        : 0.41621
                    - Mean waiting time  : 2.07636
                    - Mean response time : 3.50503
       Machine    2 - Utilization        : 0.469739
                    - Mean waiting time  : 2.10474
                    - Mean response time : 3.77156
       Machine    3 - Utilization        : 0.546373
```

```
                 - Mean waiting time  : 2.15286
                 - Mean response time : 4.15306
        Server   - Utilization        : 0.903292
        Mean number of stopped machines  : 2.18501
        System   - P (up)             : 0.573387
                 - P (down)           : 0.426613
                 - Mean up time       : 3.05368
                 - Mean down time     : 2.27201
 1e+07 End of simulation
```

2.5. Confidence interval example

In Figure 6 you can see the development of a confidence interval during a simulation. In this example we examined the mean response time for a given machine. The confidence level is 99%, the maximum width of the confidence interval is 3% of the mean value. As the simulation time is passing by, the mean values are closing up and the confidence interval is becoming smaller and smaller.

2.6. Service disciplines effects

The service discipline influences all statistical values the simulation produces. Here the same system as in the input example is examined for all disciplines that may be chosen in the simulation. We see that in this example the server utilization and the probability of a working system (system up probability) are nearly the same, but the source utilization (probability for a working machine) may change very much.

2.7. Summary

The claim for such a special simulation program has great practical significance, because as far as we know, the current (too general) programs cannot simulate such finite source queueing systems in random environments.

References

[1] Anisimov V. V., Sztrik J.: Asymptotic analysis of some controlled finite-source queueing systems. *Acta Cybernetica* 9, pp. 27–38, 1989.Andrews, G.R.: Concurrent Programming, Principles Practice, Benjamin/Cummings, 1991.

[2] Bolch G., Greiner S., de Meer H., Trivedi K. S.: *Queueing Networks and Markov Chains: Modelling and Performance Evaluation with Computer Science Applications*, John Wiley and Sons, New York, 1998.

[3] Haverkort B. R.: *Performance of Computer Communication Systems: A Model-Based Approach*, John Wiley and Sons, New York, 1998.

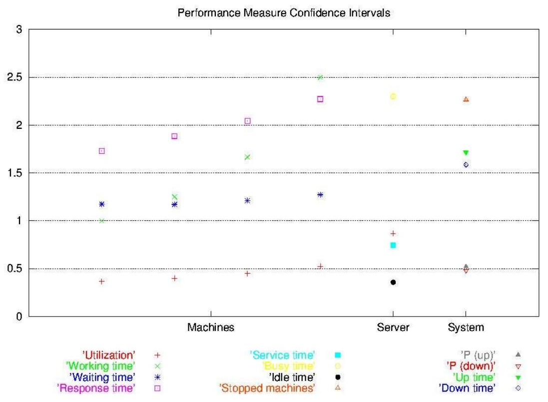


Figure 5: Graphical representation of system's performance measures.

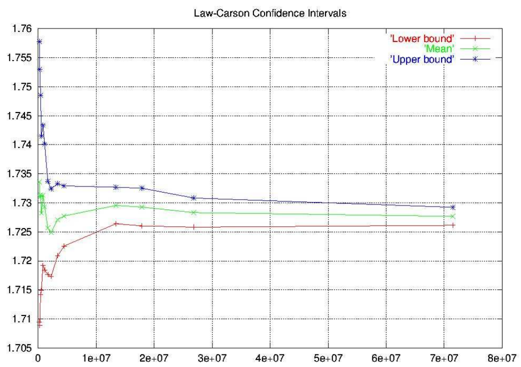


Figure 6: The simulation stops when the difference of the mean and the two bound values decreases under the percent value given as the confidence interval.

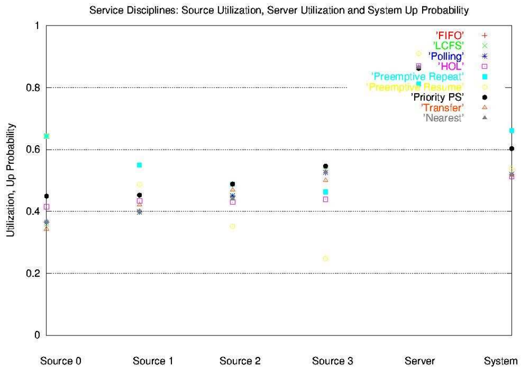


Figure 7: Source utilization, server utilization and probability of working system with four sources.

Postal address

Márk Kósa
Institute of Informatics
University of Debrecen
H-4010 Debrecen
P.O.Box 12
Hungary