A TOOL FOR SIMULATION OF MARKOV MODULATED FINITE-SOURCE QUEUEING SYSTEMS *

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

Oliver Möller
University of Trier,
D-54286 Trier, Germany
Dept. IV, Subdept. of Computer Science
e-mail: moeller@stud.informatik.uni-trier.de

Abstract

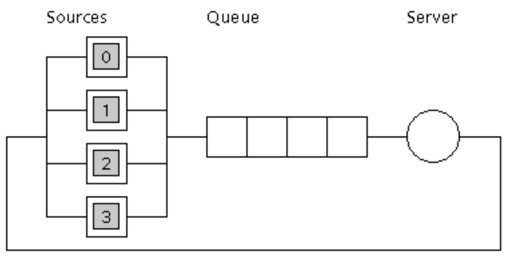
The simulation tool `lcpSim` can be used to investigate special level crossing problems of queueing systems of type $HYPO_k/HYPO_r/1//n$ embedded in different Markovian environments (recently referred to as Markov modulated ones). Our observed system consists of n heterogeneous machines (requests) and a server that "repairs" the broken machines according to the most commonly used service disciplnes, such as FIFO, LIFO, PPS, HOL, Preemptive Priorities (Resume, Repeat), Transfer, Polling.
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.

*Research is partially supported by Hungarian Scientific Research Fund under grant OTKA T 030685/99 and German-Hungarian Bilateral Intergovernmental Scientific Cooperation, OMFB-DLR No. 015-97.

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 got a queueing system of a finite-source type originated from the so-called machine interference problem, that is why we use its terminology. However, this system has often been used for mathematical modelling of different problems encounterred in computer and communication sciences, cf. Dshalalow [4], Harrison and Patel [6], Kouvatsos et.al. [8] , Mitrani [9], Takagi [11]. For the easier understanding it can be seen in the following picture:



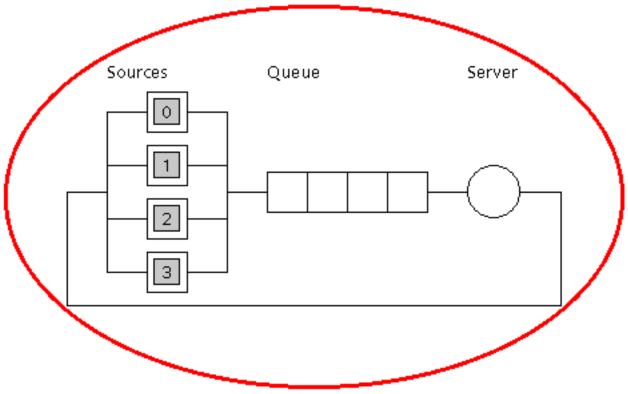
Now we state a maximum number m of stopped machines for an operating system. If more than m machines have broken down at a certain point of time, then we consider the whole system as non-operating or "down". Otherwise it is "up". We are interested in the main steady-state performance measures of the 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 characterised 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 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.

It should be noted that such systems with exponentially distributed running and repair times have been treated in Anisimov and Sztrik [1], Sztrik [10] by using asymptotic methods, and in Gaver et.al.[5] by applying a numerical approach. Due to the huge literature on performance evaluation of computer and communication systems, reliability aspects of complex systems, furthermore solution methods of different queueing situation, the authors restrict their interest to Bolch et.al. [3], Dshalalow [4] and Haverkort [7] in which the interested reader can find the relevant materials. The need for such a special simulation tool is of practical interest, since to the best knowledge of the authors such a finite-source queueing systems evolving in random environments cannot be simulated by the existing too general tools.

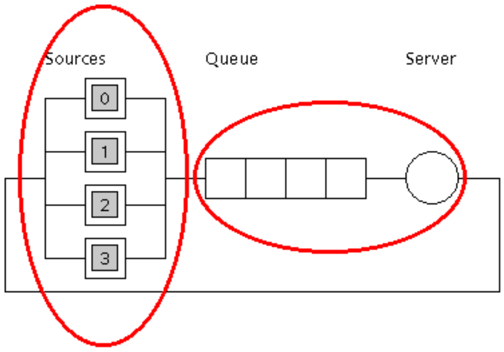
1.2 Random Environments

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

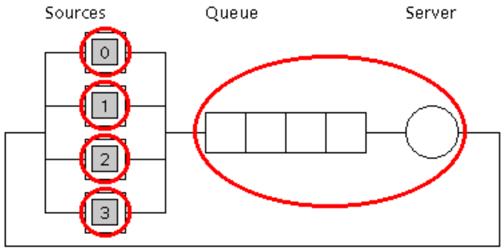
1. One random environment: one for all sources and the server



2. Two independent random environments: one for all sources and one the server



3. $Many$ independent random environments: one for each source and one for the server



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 the 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: $input =$

$\texttt{PREFERENCES} \cdot$

$\texttt{ENVIRONMENT} \cdot (\texttt{ONE} \cdot \mathit{one} + \texttt{TWO} \cdot \mathit{two} + \texttt{EACH} \cdot \mathit{each})$

The expression for one random environment: $one =$

$\texttt{MARKOV\_CHAIN} \cdot$

$\texttt{STATES} \cdot states_{int} \cdot$

$\texttt{START} \cdot start_{int} \cdot$
$\texttt{GENERATOR} \cdot ((generator^{(ij)}_{double})_{0 \le j < states_{int}})_{0 \le i < states_{int}} \cdot$
$\texttt{SOURCES} \cdot$
$\texttt{NUMBER} \cdot sources_{int} \cdot$
$\{\texttt{SOURCE} \cdot$
$\texttt{PHASES} \cdot phases^{(s)}_{int} \cdot$
$\texttt{RATES} \cdot ((rate^{(s)(ij)}_{double})_{0 \le j \le states_{int}})_{0 \le i < phases^{(s)}_{int}} \cdot$
$\texttt{WEIGHT} \cdot weight^{(s)}_{double}\ \}_{0 \le s < sources_{int}} \cdot$
$\texttt{SERVER} \cdot$
$\texttt{PHASES} \cdot phases_{int} \cdot$
$\texttt{RATES} \cdot (((rate^{(ijk)}_{double})_{0 \le k < states_{int}})_{0 \le j < phases_{int}})_{0 \le i < sources_{int}}$

The expression for two random environments: $two =$

$\texttt{SOURCES} \cdot$
$\texttt{MARKOV\_CHAIN} \cdot$
$\texttt{STATES} \cdot states^{(0)}_{int} \cdot$
$\texttt{START} \cdot start^{(0)}_{int} \cdot$
$\texttt{GENERATOR} \cdot ((generator^{(0)(ij)}_{double})_{0 \le j < states^{(0)}_{int}})_{0 \le i < states^{(0)}_{int}} \cdot$
$\texttt{NUMBER} \cdot sources_{int} \cdot$
$\{\texttt{SOURCE} \cdot$
$\texttt{PHASES} \cdot phases^{(s)}_{int} \cdot$
$\texttt{RATES} \cdot ((rate^{(s)(ij)}_{double})_{0 \le j < states^{(0)}_{int}})_{0 \le i < phases^{(s)}_{int}} \cdot$
$\texttt{WEIGHT} \cdot weight^{(s)}_{double}\ \}_{0 \le s < sources_{int}} \cdot$
$\texttt{SERVER} \cdot$
$\texttt{MARKOV\_CHAIN} \cdot$
$\texttt{STATES} \cdot states_{int} \cdot$
$\texttt{START} \cdot start_{int} \cdot$
$\texttt{GENERATOR} \cdot ((generator^{(ij)}_{double})_{0 \le j < states_{int}})_{0 \le i < states_{int}} \cdot$
$\texttt{PHASES} \cdot phases_{int} \cdot$
$\texttt{RATES} \cdot (((rate^{(ijk)}_{double})_{0 \le k < states_{int}})_{0 \le j < phases_{int}})_{0 \le i < sources_{int}}$

The expression for many random environments: $each =$

$\texttt{SOURCES} \cdot$
$\texttt{NUMBER} \cdot sources_{int} \cdot$
$\{\texttt{SOURCE} \cdot$
$\texttt{MARKOV\_CHAIN} \cdot$
$\texttt{STATES} \cdot states^{(s)}_{int} \cdot$
$\texttt{START} \cdot start^{(s)}_{int} \cdot$
$\texttt{GENERATOR} \cdot ((generator^{(s)(ij)}_{double})_{0 \le j < states^{(s)}_{int}})_{0 \le i < states^{(s)}_{int}} \cdot$
$\texttt{PHASES} \cdot phases^{(s)}_{int} \cdot$

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

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

SERVER $\cdot$

MARKOV_CHAIN $\cdot$

STATES $\cdot$ $states_{int}$ $\cdot$

START $\cdot$ $start_{int}$ $\cdot$

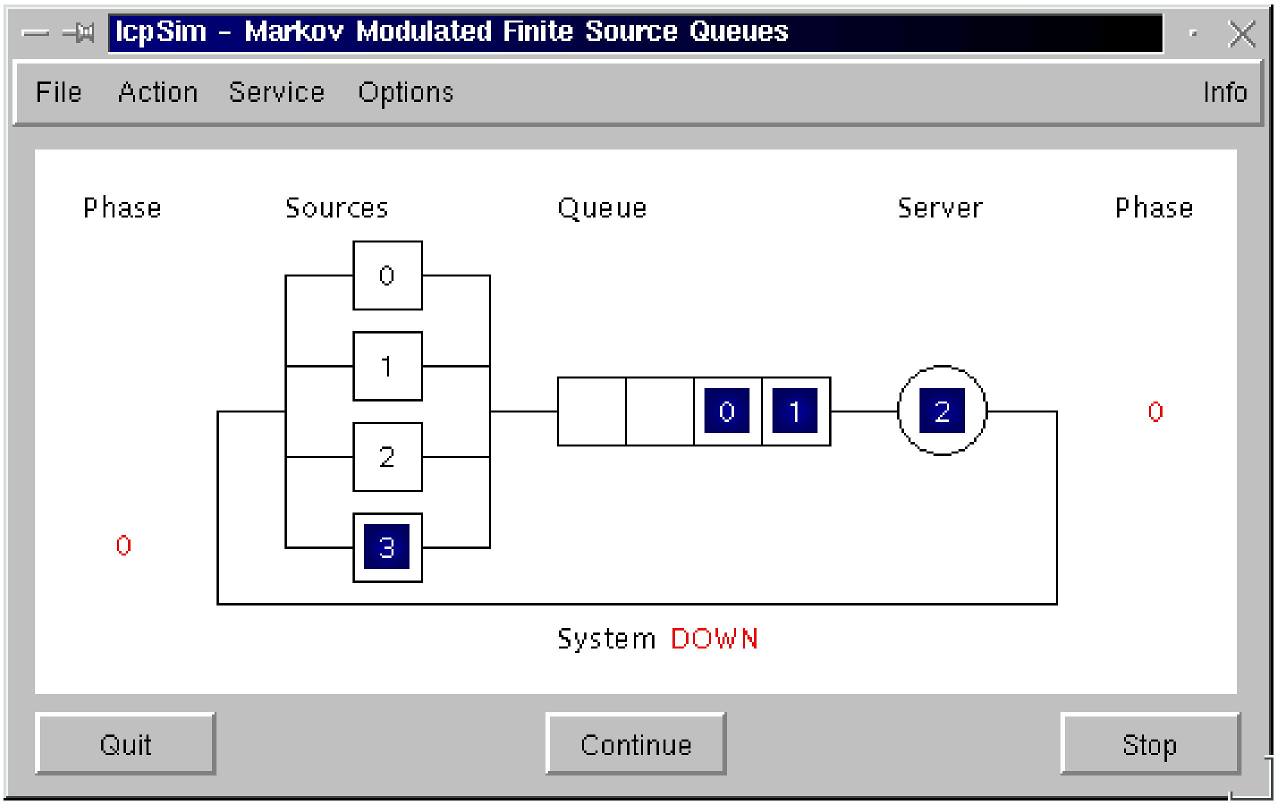
GENERATOR $\cdot$ $((generator^{(ij)}_{double})_{0\le j<states_{int}})_{0\le i<states_{int}}$ $\cdot$

PHASES $\cdot$ $phases_{int}$ $\cdot$

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

2.3 Screenshot

The following picture is a screenshot of `lcpSim`:



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 Almasi and Sztrik [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 system above applying FIFO, PPS and Polling disciplines. The reults were collected from 3 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 :   4.1543   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

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 it can be seen the simulation results are generally exact up to the 3-rd 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 ther server's Markov chain.

2.5.1 Input File

```
PREFERENCES
  ENVIRONMENT   EACH           // MARKOV CHAINS FOR EVERY SINGLE COM-
                               // PONENT, 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.

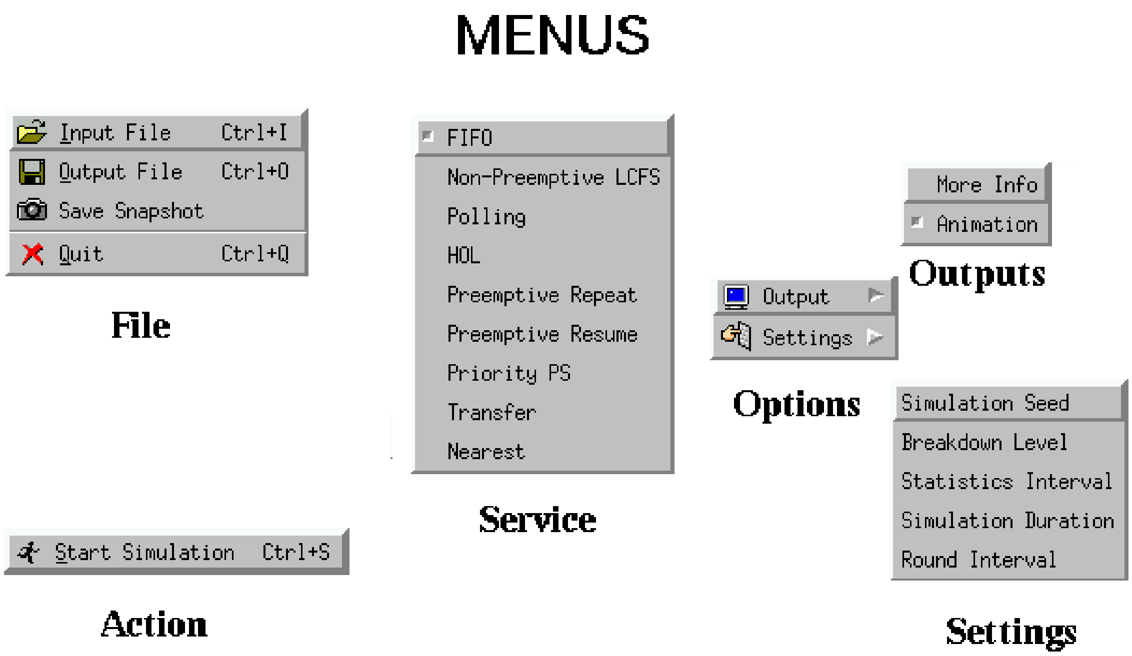
```
lcpSim Simulation Output
       Input File          : /home/info04/janos/lcpSim/input/
                             Example System
       Environment Type    : Each
       Source Number       : 2
       Service             : FIFO
       Simulation Seed     : 12345
       Breakdown Level     : 0
       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 :  2.94876
      Machine   1 - Utilization        : 0.715878
                  - Mean waiting time  : 0.643219
                  - Mean response time :  4.63115
      Server      - Utilization        : 0.553878
      Mean number of stopped machines  : 0.700086
      System      - P (up)             : 0.446122
                  - P (down)           : 0.553878
                  - Mean up time       :   3.1067
                  - Mean down time     :  3.85978

20000 Statistical values

      Machine   0 - Utilization        :  0.58584
                  - Mean waiting time  : 0.774939
                  - Mean response time :  2.94688
      Machine   1 - Utilization        : 0.716545
                  - Mean waiting time  : 0.591144
                  - Mean response time :  4.60266
      Server      - Utilization        : 0.552066
      Mean number of stopped machines  : 0.697367
      System      - P (up)             : 0.447773
                  - P (down)           : 0.552227
                  - Mean up time       :  3.08065
                  - Mean down time     :  3.79929

30000 Statistical values

      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
```

2.6 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`.



- 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 can not 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.
  `Non-Preemptive LCFS`: Last Come - First Served: the request that has arrived at the latest point of time is served first without preemptioning the one beeing 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 a 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 a 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 currenly served request is newly adapted to the new situation. In the case that 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.

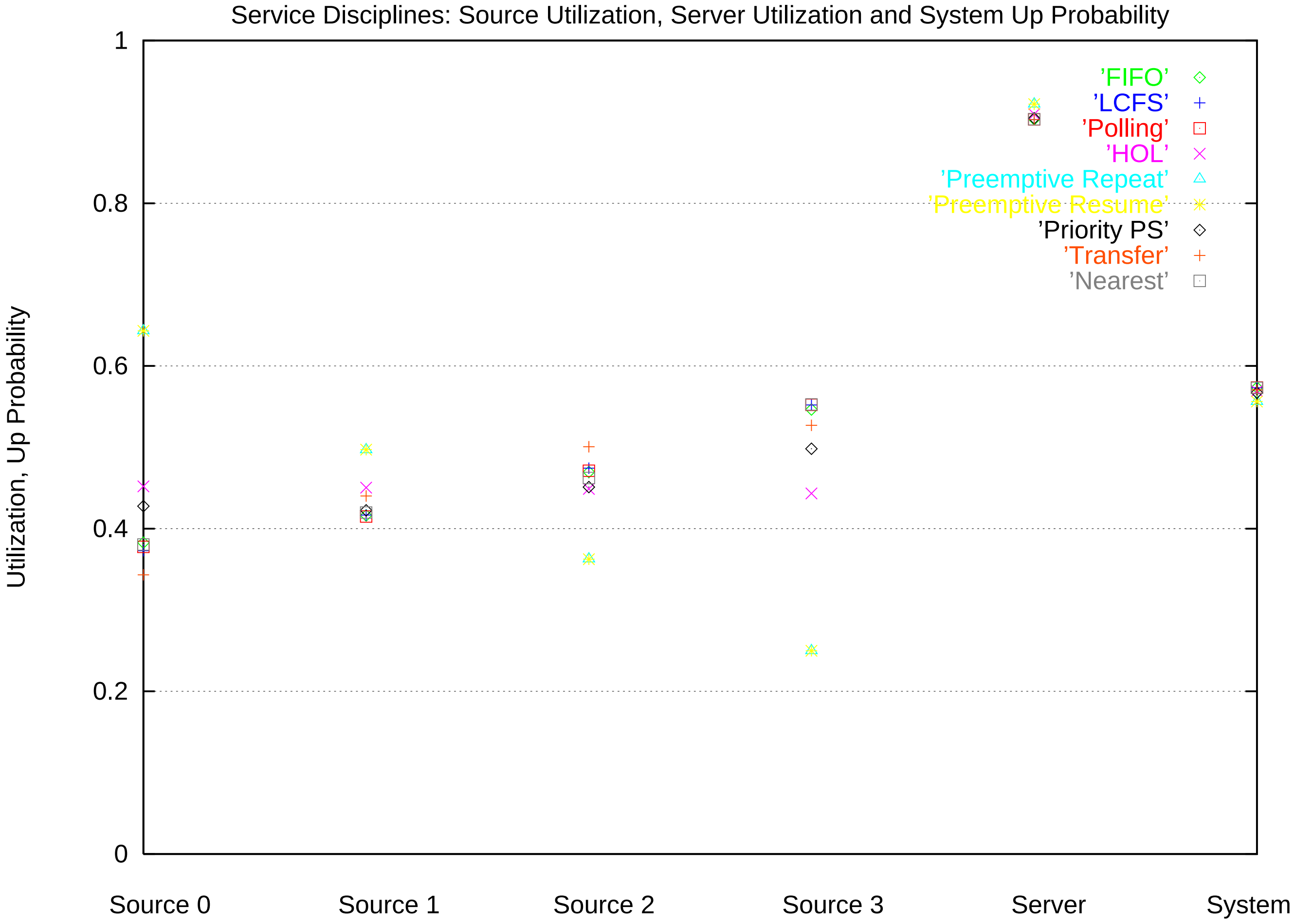
- The `Options` menu:
  In this menu additional input parameters are controlled.
  `More Info`: Increase or decrease the amount of output information that is produced during the simulation process.
  `Animation`: Activates or deactivates the graphical representation of the current simulation state.
  `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 which were set. The same information is included as 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 the possible incorrect datas.

2.7 Service Discipline Effects

The service discipline influences all statistical values the simulation produces. Here the same system as in the test 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 are nearly the same, but the source utilizations (probability for a working machine) may change very much.



References

[1] **Anisimov V.V., Sztrik J.** Asymptotic analysis of some controlled finite-source queueing systems, *Acta Cybernetica* 9, pp. 27-38, 1989.

[2] **Almasi B., Sztrik J.** A queueing model for a non-homogeneous terminal system subject to breakdowns, *Computers and Mathematics with Applications* 25, pp. 105-111, 1993.

[3] **Bolch G., Greiner S., de Meer H., Trivedi K.S.** *Queueing 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 Queueing, Models and Applications in Science and Engineering,* CRC Press, Boca Raton, 1997.

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

[6] **Harrison P., Patel N.M** *Performance Modelling of Communication Networks and Computer Architectures,* Addison-Wesley Publ. Comp., 1993.

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

[8] **Kouvatsos D.D., Fretwell R., Sztrik J.** Bounds on the effects of correlation in a stable MMPP/MMPP/1/N queue: An asymptotic approach, *Proceedings of Second Workshop on Performance Modelling and Evaluation of ATM Networks,* Chapman and Halls, London, pp. 261-281, 1995.

[9] **Mitrani I.** *Modelling of Computer and Communication Systems,* Cambridge University Press, Cambridge, 1987.

[10] **Sztrik J.** Modelling of a Multiprocessor System in a Randomly Changing Environment, *Performance Evaluation* 17, pp. 1-11, 1993.

[11] **Takagi H.** *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