HOMOGENEOUS FINITE-SOURCE RETRIAL QUEUES WITH SERVER SUBJECT TO BREAKDOWNS AND REPAIRS

B. Almási, J. Roszik, J. Sztrik
University of Debrecen, Debrecen, Hungary
{almasi|jsztrik}@math.klte.hu
jroszik@delfin.klte.hu

Abstract

This paper deals with a retrial queueing system with a finite number of homogeneous sources of calls and a single non-reliable server, which means that the server is subject to random breakdowns depending on whether it is busy or idle. The failure of the server may block or unblock the systems's operations and the service of the interrupted request may be resumed or the call can be transmitted to the orbit. All random variables involved in the model constructions are supposed to be exponentially distributed and independent of each other.

The novelty of the investigation is the variability of this non-reliablility of the server which makes the system rather complicated. The MOSEL tool was used to formulate and solve the problem and the main performance and reliability measures were derived and graphically displayed. Several numerical calculations were performed to show the effect of the non-reliability of the server on the mean reponse times of the calls.

Keywords: retrial queueing systems, finite number of sources, non-reliable server, performance tool, performance and reliability measures

1 Introduction

Retrial queues have been widely used to model many problems arising in telephone switching systems, telecommunication networks, computer networks and computer systems, etc. For a systematic account of the fundamental methods and results, furthermore an accessible classified bibliography on this topic the interested reader is referred to, for example [2], [7], [11], [12], and references therein.

Since in practice some components of the systems are subject to random breakdowns (see for example [24], [29],[34]) it is of basic importance to study reliability of retrial queues with server breakdowns and repairs because of limited ability of repairs and heavy influence of the

[1]Research is partially supported by German-Hungarian Bilateral Intergovernmental Scientific Cooperation, OMFB-DLR No. 21-2000, Hungarian Scientific Research Found OTKA T0-34280/2000 and FKFP grant 0191/2001.

breakdowns on the performance measures of the system. However, so far the repairable retrial queues are analyzed only by queueing theory. For related literature the reader is referred to the works [1], [6], [25], [35] where infinite-source non-reliable retrial queues were treated.

In many practical situations it is important to take into account the fact that the rate of generation of new primary calls decreases as the number of customers in the system increases. This can be done with the help of finite-source, or quasi-random input models. Queueing systems without retrials, that is systems with classical waiting lines and finite population have been review in detail by Takagi [32]. Since Kornyisev [23], which was the first paper devoted to finite-source retrial queues, there has been a rapid growth in the literature on this topic. A complete survey on related results can be found in Artalejo [2] for systems of type $M/G/1//K$ and $M/M/c//K$ in Kendall's notation. In addition, in the papers Falin and Artalejo [13], Falin [14] not only the outside observer's distributions of the systems in steady state, but also the stationary performance characteristics are considered on more detail. In particular, all main measures were expressed in terms of the server utilization. Arriving customer's distribution of the system state, busy period and waiting time processes (which is especially complex for retrial queues due to the overtaking) were investigated, too. Further recent results with finite-source of primary requests can be found in [4], [5], [10], [12], [15], [16], [22], [26], [30].

Retrial queues with quasi-random input are recent interest in modelling magnetic disk memory systems [28], cellular mobile networks [33], computer networks [17], and local-area networks with nonpersistent CSMA/CD protocols [26], with star topology [19, 27], with random access protocols [20], and with multiple-access protocols [21].

In this paper finite-source systems with the following assumptions are investigated. Consider a single server queueing system, where the primary calls are generated by K, $1 < K < \infty$ homogeneous sources. The server can be in three states: idle, busy and failed. If the server is idle, it can serve the calls of the sources. Each of the sources can be in three states: free, sending repeated calls and under service. If a source is free at time t it can generate a primary call during interval $(t, t + dt)$ with probability $\lambda dt + o(t)$. If the server is free at the time of arrival of a call then the call starts to be served immediately, the source moves into the under service state and the server moves into busy state. The service is finished during the interval $(t, t + dt)$ with probability $\mu dt + o(t)$ if the server is available. If the server is busy, then the source starts generation of a Poisson flow of repeated calls with rate ν until it finds the server free. After service the source becomes free, and it can generate a new primary call, and the server becomes idle so it can serve a new call. The server can fail during the interval $(t, t+dt)$ with probability $\delta dt + o(t)$ if it is idle, and with probability $\gamma dt + o(t)$ if it is busy. If $\delta = 0, \gamma > 0$ or $\delta = \gamma > 0$ *active or independent breakdowns* can be discussed, respectively. If the server fails in busy state, it either *continues servicing* the interrupted call after it has been repaired or the interrupted request *returns to the orbit*. The repair time is exponentially distributed with a finite mean $1/\tau$. If the server is failed two different cases can be treated. Namely, *blocked sources* case when all the operations are stopped, that is no new primary and repeated calls are generated. In the *unblocked (intelligent) sources* case only service is interrupted but all the other operations are continued (new and repeated calls can be generated). All the times involved in the model are assumed to be mutually independent of each other.
As it can be seen this systems is rather complicated since it involves two types of failures, continued or repeated service and blocked or unblocked operations during breakdowns.

Our objective is to give the main usual stationary performance and reliability measures of the system and to display the effect of different parameters on them. To achieve this goal a tool called MOSEL (Modeling, Specification and Evaluation Language) developed at the University of Erlangen, Germany, see [8], is used to formulate and solve the problem. We show how this system can be modelled, and how easily performance measures can graphically be represented using IGL (Intermediate Graphical Language). This model is another extension of investigations for homogeneous finite-source queueing systems without retrials but with server's breakdowns which were treated in [31]. Similarly, it generalizes the results of [12] where homogeneous systems with reliable multiple servers were analyzed.

The paper is organized as follows. In Section 2 the full description of the model by the help of the corresponding multi-component Markov chain is given. Then the main performance and reliability measures of the system are derived that can be obtained using MOSEL tool. In Section 3 several numerical examples are presented and some comments are made. Finally, the paper ends with a Conclusion.

2 The $M/M/1//K$ retrial queueing model with unreliable server

2.1 The underlying Markov chain

The system state at time t can be described with the process $X(t) = (Y(t); C(t); N(t))$, where $Y(t) = 0$ if the server is up, $Y(t) = 1$ if the server is failed, $C(t) = 0$ if the server is idle, $C(t) = 1$ if the server is busy, $N(t)$ is the number of sources of repeated calls at time t. Because of the exponentiality of the involved random variables this process is a Markov–chain with finite state space $S = \{0, 1\} \times \{0, 1\} \times \{0, 1, ..., K - 1\}$. Since the state space of the process $(X(t), t \geq 0)$ is finite, the process is ergodic for all values of the rate of generation of primary calls, and from now on we will assume that the system is in the steady state.

We define the stationary probabilities:

$$P(q; r; j) = \lim_{t\to\infty} P(Y(t) = q, C(t) = r, N(t) = j), \quad q = 0, 1, \quad r = 0, 1, \quad j = 0, ..., K-1.$$

Knowing these quantities the **main performance measures** can be obtained as follows:

- *Utilization of the server*

$$U_S = \sum_{j=0}^{K-1} P(0, 1, j)$$

- *Utilization of the repairman*

$$U_R = \sum_{q=0}^{1} \sum_{j=0}^{K-1} P(1, q, j)$$

- *Availability of the server*

$$A_S = \sum_{q=0}^{1} \sum_{j=0}^{K-1} P(0,q,j) = 1 - U_R$$

- *The mean number of sources of repeated calls*

$$N = E[N(t)] = \sum_{q=0}^{1} \sum_{r=0}^{1} \sum_{j=0}^{K-1} jP(q,r,j)$$

- *The mean number of calls staying in the orbit or in service*

$$M = E[C(t) + N(t)] = \sum_{q=0}^{1} \sum_{r=0}^{1} \sum_{j=0}^{K-1} (r+j)P(q,r,j)$$

- *The mean rate of generation of primary calls*

$$\overline{\lambda} = \begin{cases} \lambda E[K - C(t) - N(t); Y(t) = 0] & \text{for blocked case,} \\ \lambda E[K - C(t) - N(t)] & \text{for unblocked case.} \end{cases}$$

- *The mean response time*

$$E[T] = M/\overline{\lambda}$$

- *The mean waiting time*

$$E[W] = N/\overline{\lambda}$$

- *The blocking probability of a primary call*

$$B = \begin{cases} \frac{\lambda E[K - C(t) - N(t); Y(t)=0; C(t)=1]}{\overline{\lambda}} & \text{for blocked case,} \\ \\ \frac{\lambda E[K - C(t) - N(t); C(t)=1]}{\overline{\lambda}} & \text{for unblocked case.} \end{cases}$$

2.2 The MOSEL implementation

We used the software tool MOSEL (Modeling, Specification and Evaluation Language) to formulate the model and to calculate the main performance measures. In this section we show the base MOSEL program and the explanation of its main parts without the technical details of programming. It doesn't contains the picture section, which is needed to generate various graphical representations of the measures. The figures in the next section are automatically generated by the tool with the corresponding picture part. In the MOSEL program we used the following terminology: The server and the sources are referred to as a CPU and terminals, respectively.

```
/*retrialnr-cpu.msl begins */ /* Non-reliable terminal-system with
retrying jobs for NT terminals */
/*----------------------------------------- Definitions
-----------*/ #define  NT   3     // change it to 4, 5, 6, 7, ...

/*================== No changes required below
=====================*/ <NT> #define NT1 <#+1> VAR double prgen;
VAR double prretr; VAR double prrun; VAR double cpubreak_idle; VAR
double cpubreak_busy; VAR double cpurepair;

enum cpu_states {cpu_busy, cpu_idle}; enum cpu_updown {cpu_up,
cpu_down}; enum terminal_states {term_busy, term_retrying,
term_waiting};

/*----------------------------------------- Node definitions
--------*/ <1..NT>  NODE terminal#[terminal_states] = term_busy;
         NODE cpu_state[cpu_states]      = cpu_idle;
         NODE cpu[cpu_updown]            = cpu_up;

/*---------------------------------------------- Transitions
--------*/ <1..NT> IF cpu==cpu_up FROM cpu_idle,
terminal#[term_busy]
           TO cpu_busy, terminal#[term_waiting]
           W prgen;
<1..NT> IF cpu_state==cpu_busy AND cpu==cpu_up FROM
terminal#[term_busy]
           TO terminal#[term_retrying]
           W prgen;
<1..NT> IF cpu==cpu_up FROM cpu_idle, terminal#[term_retrying]
           TO cpu_busy, terminal#[term_waiting]
           W prretr;
<1..NT> IF cpu==cpu_up FROM cpu_busy, terminal#[term_waiting]
           TO cpu_idle, terminal#[term_busy]
           W prrun;
IF cpu_state==cpu_idle FROM cpu_up TO cpu_down W cpubreak_idle; IF
cpu_state==cpu_busy FROM cpu_up TO cpu_down W cpubreak_busy; FROM
cpu_down TO cpu_up W cpurepair;

/*-------------------------------------------------- Results
        RESULT>> if(cpu==cpu_up AND cpu_state==cpu_busy)
                                                     cpuutil += PROB;
<1..NT> RESULT>> if(cpu==cpu_up AND terminal#==term_busy)
                                                   termutil# += PROB;
<1..NT> RESULT>> if(terminal#!=term_busy) termwaiting# += PROB;
<1..NT> RESULT>> if(cpu==cpu_up AND terminal#==term_retrying)
                                                     retravg += PROB;
        RESULT>> if(cpu == cpu_up) goodcpu += PROB;
<1..NT> RESULT>> resptime# = termwaiting# / (prgen * termutil#);

/* retrialnr-cpu.msl ended */
```

In the **declaration part** we define the number of terminals (NT), this is the only program code line, that must be modified when modeling larger systems. The terminals have three states: busy (primary call generation), retrying (repeated call generation) and waiting (job service at the CPU). The CPU has two states: idle and busy, and it can be up or failed in both states. We define the three parameters for the terminals: $prgen$ denotes the rate of primary call generation, $prretr$ references to the rate of repeated call generation and $prrun$ denotes the service rate. The $cpubreak\_idle$, $cpubreak\_busy$ and $cpurepair$ variables denote the failure rate in the two CPU states and the repair rate.

The **node part** defines the nodes of the system: Our queueing network contains $NT + 2$ nodes: the two nodes for the CPU (which is idle and up at the starting time) and NT number of terminals (they are busy when the system starts).

The **transition part** describes how the system works. The first transition rule defines the successful primary call generation: the CPU moves from the idle state to busy and the terminal from busy to waiting. The second rule shows an unsuccessful primary call generation: if the CPU is busy when the call is generated then the terminal moves to state retrying. The third rule handles the case of a successful repeated call generation: the CPU moves from the idle state to busy and the terminal from retrying to waiting. The fourth rule describes the request service at the CPU. The fifth and sixth rules describe the CPU fail in idle and busy state. The last rule shows the CPU repair.

Finally the **result part** calculates the output performance measures.

3 Numerical examples

In this section we consider some sample numerical results to illustrate the influence of the non–reliable server on the mean response time $E[T]$. The results in the reliable case were validated by the Pascal program given in [12], too.

3.1 Validation of results

	non–rel. retrial(cont.)	non–rel. retrial(orbit)	non–rel. FIFO
Number of sources:	3	3	3
Request's generation rate:	0.1	0.1	0.1
Service rate:	1	1	1
Retrial rate:	1e+25	1e+25	-
Server's failure rate:	0.01	0.01	0.01
Server's repair rate:	0.05	0.05	0.05
Utilization of the server:	0.22327965614	0.22327965229	0.22327964521
Utilization of the sources:	0.74426549207	0.74426549375	0.74426549684
Mean response time:	1.43606563309	1.43606560277	1.43606554705

Table 1: Validations

	NT	λ	μ	ν	δ	γ	τ
Figure 1	6	x axis	4	0.4	0.05	0.05	0.1
Figure 2	6	5	10	x axis	0.05	0.05	0.1
Figure 3	6	0.1	x axis	0.4	0.05	0.05	0.1
Figure 4	6	x axis	4	0.4	0.005(0.05)	0.05	0.1
Figure 5	6	5	10	x axis	0.005(0.05)	0.05	0.1
Figure 6	6	0.1	x axis	0.4	0.005(0.05)	0.05	0.1

Table 2: Input parameters

3.2 Input parameters

As it can be seen in the first 3 cases the independent breakdowns are treated, then the state dependent and independent ones are considered. In each case different comparisons are made according to the breakdowns *(dependent, independent)*, service continuation *(continued)* and system operations *(blocked, unblocked)*.

3.3 Comments

- In Table 1 the results were tested by the help of non-reliable FIFO case, since if the retrial rate in the repeated calls model tends to infinity, the measures should approach the corresponding ones in FIFO discipline. The derived results are the same up to the 7-th decimal digit.

- In Figures 1–3 we can see the mean response time $E[T]$ for the reliable and the non–reliable retrial system with continuous, non-continuous service after repair, with blocked and unblocked operations during service failure when the primary request generation rate, retrial rate and service rate increase. In these cases, the server's failure rate is independent of the state of the server (busy or idle). Figure 1 demonstrates a surprising phenomenon of retrial queues having a maximum of $E[T]$ which was noticed in [13], too. The difference between continuous, non-continuous service, moreover blocked, unblocked (intelligent) systems's operations is clearly shown. However, if the retrial or service rate increases the continuous and non-continuous service result in the same measure, as it was expected, see Figure 2, 3.

- In Figures 4–6 the mean response time $E[T]$ is displayed with continuous service after repair but the server's failure rate depends on its state. The system operation is either blocked or unblocked. In Figure 4 we can see that the curves of independent failure with blocked operations and dependent failures with unblocked operations intersect each other. In each case the difference between the independent and dependent failures is clearly demonstrated.

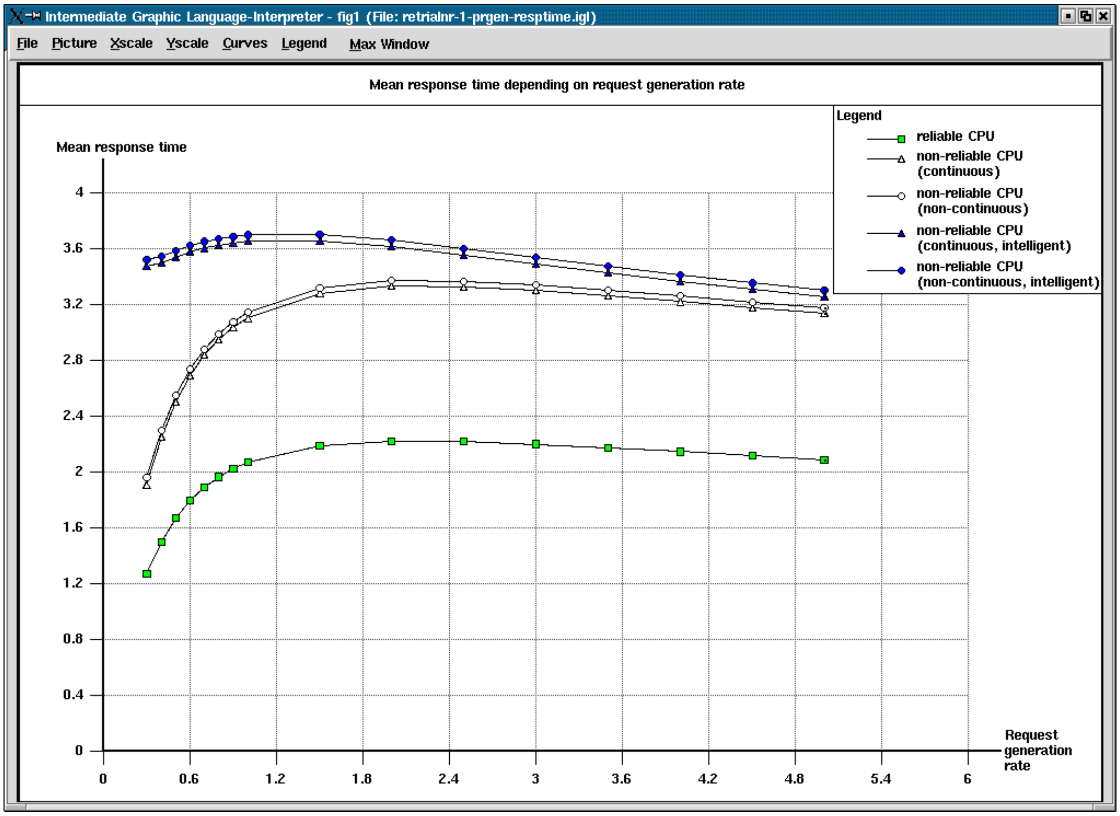

Figure 1: $E[T]$ versus primary request generation rate

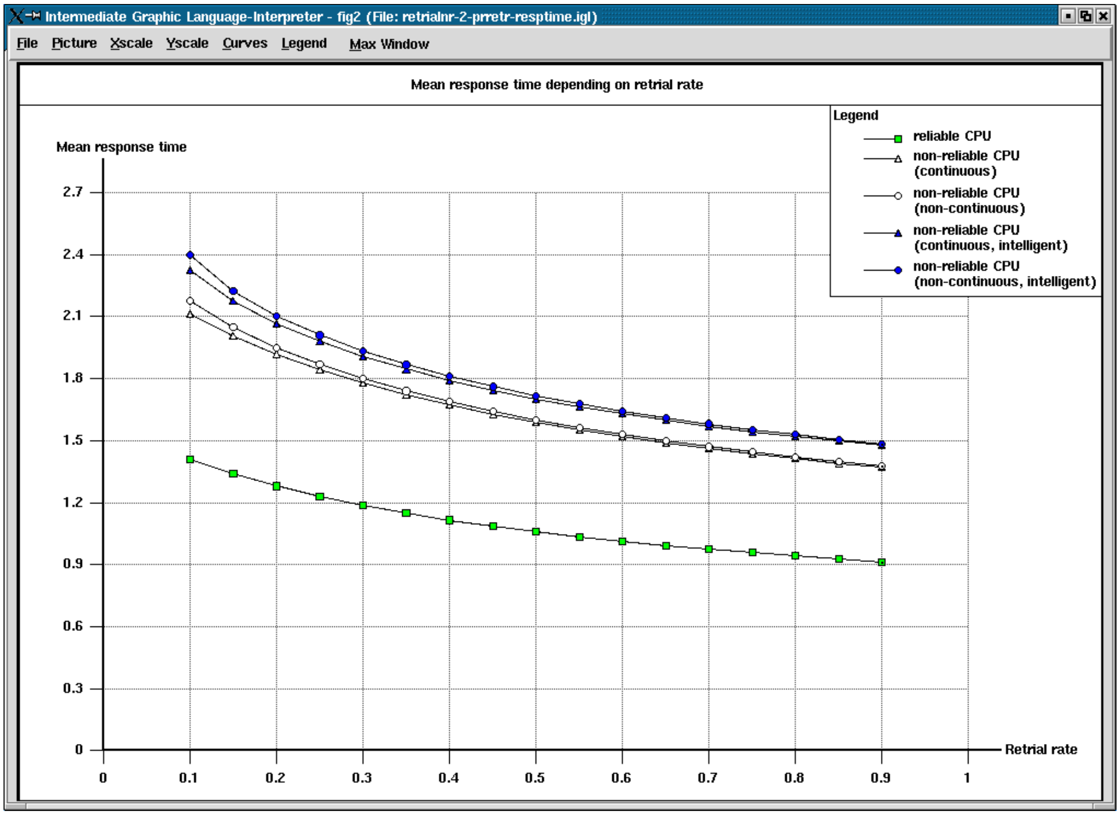

Figure 2: $E[T]$ versus retrial rate

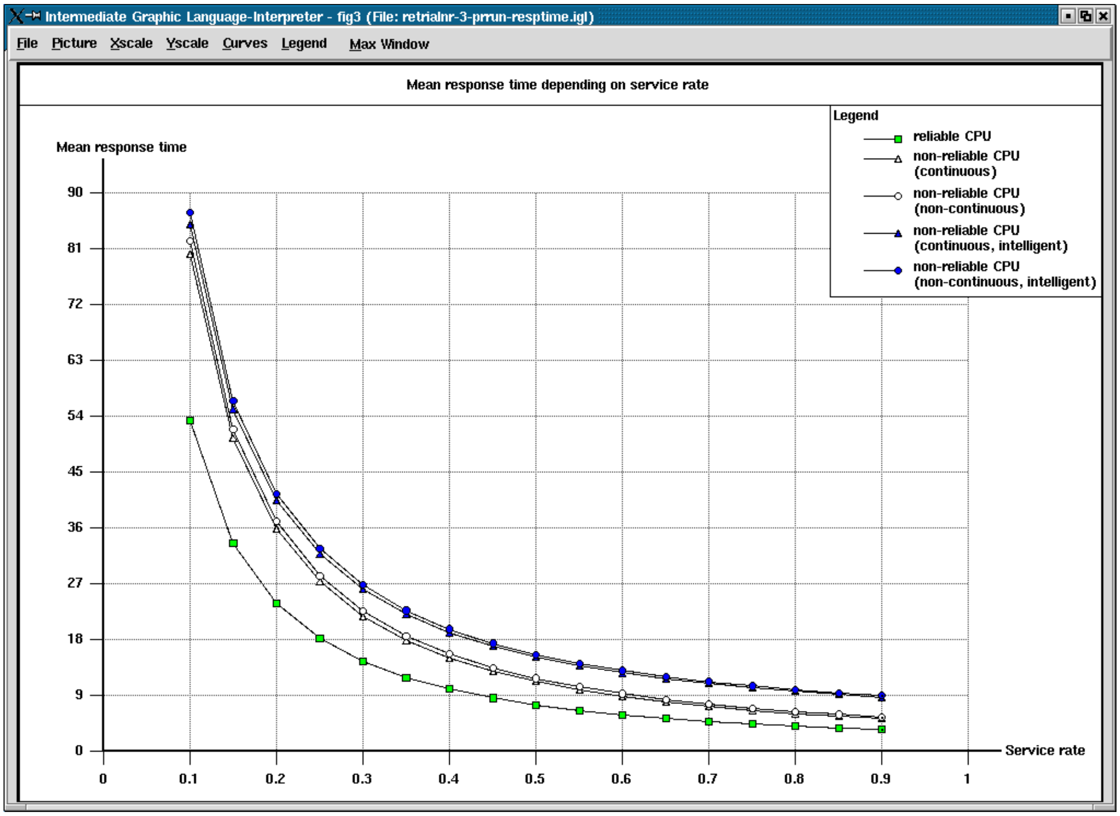

Figure 3: $E[T]$ versus service rate

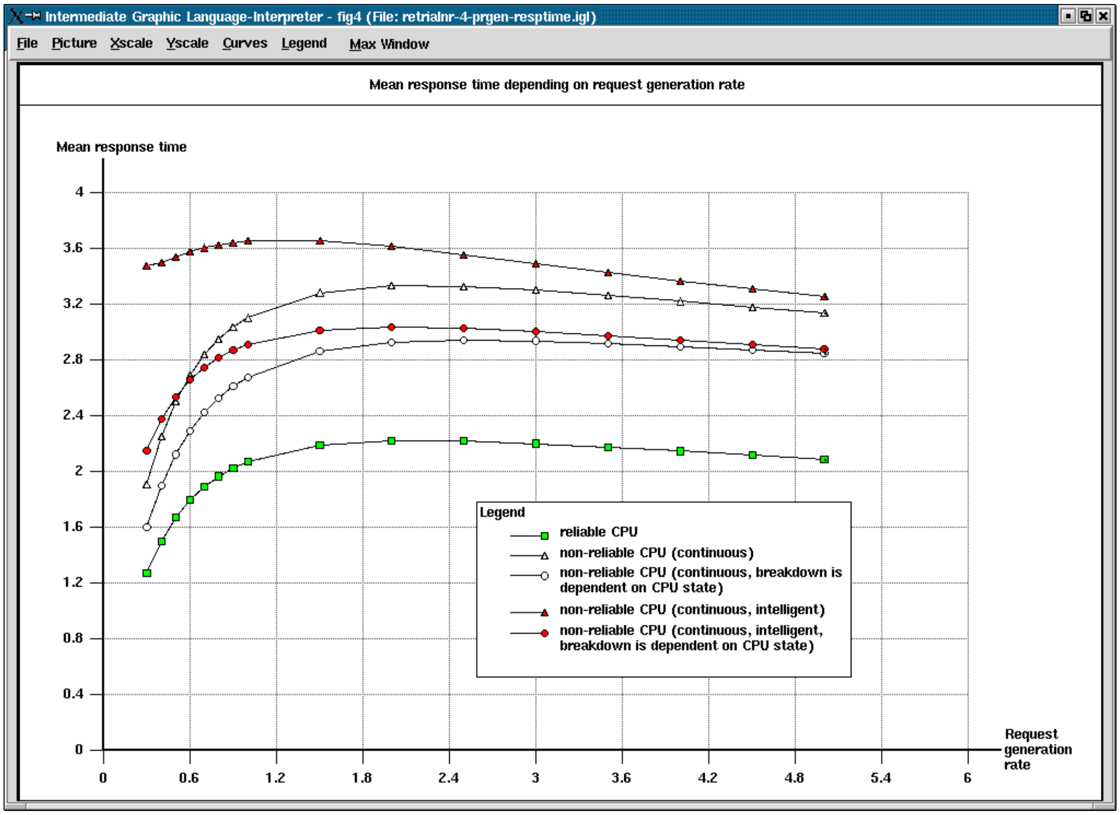

Figure 4: $E[T]$ versus primary request generation rate

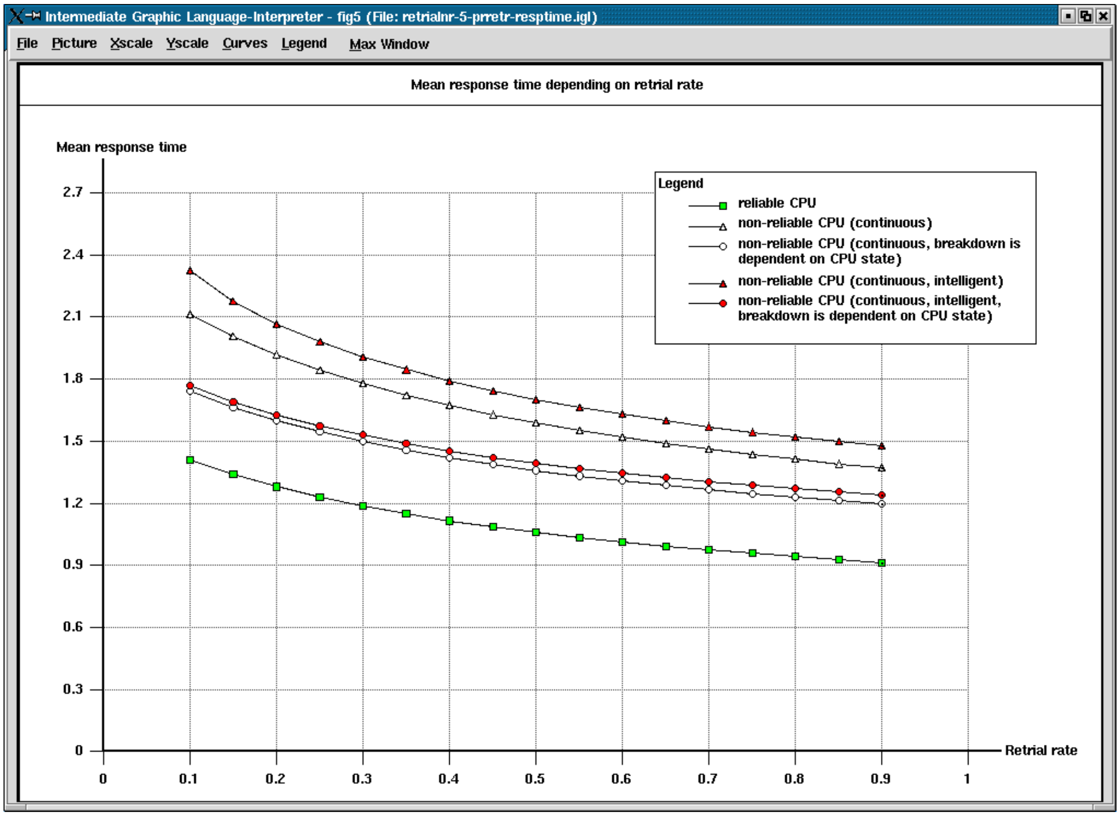

Figure 5: $E[T]$ versus retrial rate

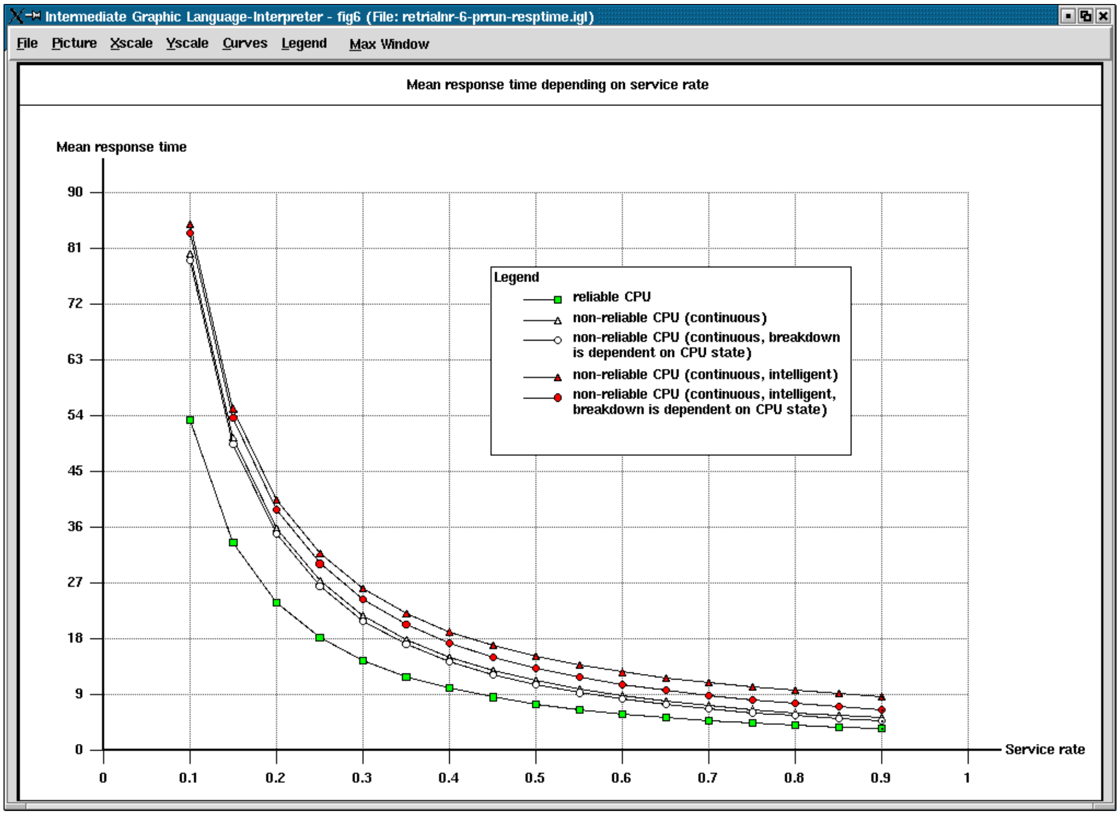

Figure 6: $E[T]$ versus service rate

4 Conclusions

In this paper a finite-source homogeneous retrial queueing system with non-reliable server is studied. The novelty of the investigation is this non–reliablility of the server which makes the system rather complicated. A tool MOSEL was used to formulate and solve the problem and the main performance measures were derived and graphically displayed . Several numerical calculations were performed to show the effect of the non-reliability of the server on the mean response times of the calls.

References

[1] **Artalejo J.R.** New results in retrial queueing systems with breakdown of the servers, *Statistica Neerlandica* 48(1994), 23–36.

[2] **Artalejo J.R.** Retrial queues with a finite number of sources, *J. Korean Math. Soc.* 35(1998) 503-525.

[3] **Artalejo J.R.** Accessible bibliography on retrial queues, *Math. Comput. Modeling* 30(1999) 1-6.

[4] **Artalejo, J.R. and Gomez-Corral, A.** Information theoretic analysis for queueing systems with quasi-random input, *Math. Comput. Modeling* 22(1995) 65-76.

[5] **Artalejo J.R., Rajagopalan V. and Sivasamy R.** On finite Markovian queues with repeated attemps, *Investigacion Operativa* 9(2000) 83-94.

[6] **Aissani A. and Artalejo J. R.** On the single server retrial queue subject to breakdowns, *Queueing Systems Theory Applications* 30(1998), 309–321.

[7] **Barcelo J., Escudero L. and Artalejo J. editors** *Proceedings of the 1st International workshop on retrial queues (WRQ'98)*, Madrid, Top 7(1998).

[8] **Begain K., Bolch G. and Herold H.** *Practical performance modeling, application of the MOSEL language,* Kluwer Academic Publisher, Boston, 2001.

[9] **Daigle J.N.** *Queueing theory for telecommunications,* Addison-Wesley, Nwe York, 1992.

[10] **Dragieva V.I.** Single-line queue with finite source and repeated calls, *Problems of Information Transmission* 30(1994) 283-289.

[11] **Falin G.I.** A survey of retrial queues, *Queueing Systems* 7(1990) 127-168.

[12] **Falin G.I. and Templeton J.G.C.** *Retrial queues,* Chapman and Hall, London, 1997.

[13] **Falin G.I. and Artalejo J.R.** A finite source retrial queue, *European Journal of Operational Research* 108(1998) 409-424.

[14] **Falin G.I.** A multiserver retrial queue with a finite number of sources of primary calls, *Mathematical and Computer Modelling* 30(1999) 33-49.

[15] **Falin G.I. and Gomez Corral A.** On a bivariate Markov process arising in the theory of single-server retrial queues, *Statistica Neerlandica* 54(2000) 67-78.

[16] **Gomez Corral A.** Analysis of a single-server retrial queue with quasi-random input and nonpreemptive priority, *Computers and Mathematics with Applications* 43(2002) 767-782.

[17] **Houck D.J. and Lai W.S.** Traffic modelling and analysis of hybrid fibercoax systems, *Computer Networks and ISDN Systems* 30(1998) 821-834.

[18] **Jain R.** *The art of computer systems performance analysis,* John Wiley and Sons, New York, 1991.

[19] **Janssens G.K.** The quasi-random input queueing system with repeated attempts as a model for collision-avoidance star local area network, *IEEE Transactions on Communications* 45(1997) 360-364.

[20] **Kalmychkov A.I. and Medvedev G.A.** Probability characteristics of Markov local-area networks with random-access protocols, *Automatic Control and Computer Science* 24(1990) 38-45.

[21] **Khomichkov I.I.** Study of models of local networks with multiple-access protocols, *Automation and Remote Control* 54(1993) 1801-1811.

[22] **Kok A.G.** Algorithmic methods for single server systems with repeated attemps, *Statistica Neerlandica* 38(1984) 23-32.

[23] **Kornyshev Y.N.** Design of a fully accessible switching system with repeated calls, *Telecommunications* 23(1969) 46-52.

[24] **Kovalenko I.N., Kuznetsov N.Yu. and Pegg P.A.** *Mathematical theory of reliability of time dependent systems with practical applications,* John Wiley and Sons, Chichester, 1997.

[25] **Kulkarni V. G. and Choi Bong Dae** Retrial queues with server subject to breakdowns and repairs, *Queueing Systems Theory and Applications* 7(1990), 191–208.

[26] **Li Hui and Yang Tao** A single server retrial queue with server vacations and a finite number of input sources, *European Journal of Operational Research* 85(1995) 149-160.

[27] **Mehmet-Ali M.K., Hayes J.F. and Elhakeem A.K.** Traffic analysis of a local area network with star topology, *IEEE Transactions on Communications* 36(1988) 703-712.

[28] **Ohmura H. and Takahashi Y.** An analysis of repeated call model with a finite number of sources, *Electronics and Communications in Japan* 68(1985) 112-121.

[29] **Ravichandran N.** *Stochastic methods in reliabilty theory,* John wiley and Sons, New York, 1990.

[30] **Stepanov S.N.** The analysis of the model with finite number of sources and taking into account the subscriber behaviour, *Automation and Remote Control* 55(1994) 100-113.

[31] **Sztrik J. and Gál T.** A recursive solution of a queueing model for a multi-terminal system subject to breakdowns, *Performance Evaluation* 11(1990) 1–7.

[32] **Takagi H.** *Queueing Analysis, A Foundation of Performance Evaluation, Vol. 2., Finite Systems*, North-Holland, Amsterdam, 1993.

[33] **Tran-Gia P. and Mangjes M.** Modeling of customer retrial phenomenon in cellural mobile networks, *IEEE Journal of Selected Areas in Communications* 15(1997) 1406-1414.

[34] **Trivedi K. S.** *Probability and statistics with reliability, queueing and computer science applications*, Prentice-Hall, Englewood Cliffs, 1982.

[35] **Wang Jinting, Cao Jinhua and Li Quanlin** Reliability analysis of the retrial queue with server breakdowns and repairs, *Queueing Systems Theory and Applications* 38(2001), 363–380.