Available online at www.sciencedirect.com

SCIENCE @ DIRECT®

Mathematical and Computer Modelling 42 (2005) 673–682

MATHEMATICAL AND COMPUTER MODELLING

www.elsevier.com/locate/mcm

Homogeneous Finite-Source Retrial Queues with Server Subject to Breakdowns and Repairs

B. ALMÁSI, J. ROSZIK AND J. SZTRIK
University of Debrecen, Debrecen, Hungary
<almasi><jroszik><jsztrik>@inf.unideb.hu

(Received December 2003; accepted February 2004)

Abstract—This paper deals with a single server retrial queue with a finite number of homogeneous sources of calls and a single nonreliable 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' 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 nonreliability 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 nonreliability of the server on the mean response times of the calls. © 2005 Elsevier Ltd. All rights reserved.

Keywords—Retrial queuing systems, Nonreliable server, Performance tools, 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, [1–5], and references therein.

Since in practice some components of the systems are subject to random breakdowns (see, for example, [6–8]), 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 breakdowns on the performance measures of the system. However, so far the repairable retrial queues are analyzed only in queuing theory. For related literature, the reader is referred to the works [9–12], where infinite-source nonreliable retrial queues were treated.

This research is partially supported by German-Hungarian Bilateral Intergovernmental Scientific Cooperation, OMFB-DLR No. 21-2000, Hungarian Scientific Research Fund OTKA T0-34280/2000 and FKFP Grant 0191/2001.

0895-7177/05/$ - see front matter © 2005 Elsevier Ltd. All rights reserved. Typeset by $\mathcal{A}\mathcal{M}\mathcal{S}$-TEX
doi:10.1016/j.mcm.2004.02.046

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. Queuing systems without retrials, that is systems with classical waiting lines and finite population have been reviewed in detail by Takagi [13]. Since Kornyshev [14], 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 [1] for systems of type $M/G/1//K$ and $M/M/c//K$ in Kendall's notation. In addition, in the papers [15,16], 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 customers 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 [17–24].

Retrial queues with quasi-random input are recent interest in modeling magnetic disk memory systems [25], cellular mobile networks [26], computer networks [27], and local-area networks with nonpersistent CSMA/CD protocols [23], with star topology [28,29], with random access protocols [30], and with multiple-access protocols [31].

In this paper, finite-source systems with the following assumptions are investigated. Consider a single server queuing 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 $\lambda\, dt + o(dt)$. 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 $\mu\, dt + o(td)$ 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 $\delta\, dt + o(dt)$ if it is idle, and with probability $\gamma\, dt + o(dt)$ 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 neither new primary calls nor 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 [32], is used to formulate and solve the problem. We show how this system can be modeled, 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 queuing systems without retrials but with server's breakdowns which were treated in [33]. Similarly, it generalizes the results of [5], 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 multicomponent 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 QUEUING 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 $S = \{0,1\} \times \{0,1\} \times \{0,1,\dots,K-1\}$. 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

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

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*
$$U_R = \sum_{r=0}^{1} \sum_{j=0}^{K-1} P(1,r,j);$$
- *availability of the server*
$$A_S = \sum_{r=0}^{1} \sum_{j=0}^{K-1} P(0,r,j) = 1 - U_R;$$
- *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*
$$M = E[N(t) + C(t)] = N + \sum_{q=0}^{1} \sum_{j=0}^{K-1} P(q,1,j);$$
- *the mean rate of generation of primary calls*
$$\bar{\lambda} = \begin{cases} \lambda E[K - C(t) - N(t); Y(t) = 0], & \text{for blocked case,} \\ \lambda E[K - C(t) - N(t)], & \text{for unblocked case;} \end{cases}$$
- *the mean response time*
$$E[T] = \frac{M}{\bar{\lambda}};$$
- *the mean waiting time*
$$E[W] = \frac{N}{\bar{\lambda}};$$
- *the blocking probability of a primary call*
$$B = \begin{cases} \dfrac{\lambda E[K - C(t) - N(t); Y(t) = 0; C(t) = 1]}{\bar{\lambda}}, & \text{for blocked case,} \\ \dfrac{\lambda E[K - C(t) - N(t); C(t) = 1]}{\bar{\lambda}}, & \text{for unblocked case.} \end{cases}$$

2.2. The MOSEL Implementation

Because of the fact that the state space of the describing Markov chain is very large, (especially in the heterogeneous model we would like to investigate later), it is difficult to calculate the system measures in the traditional way of solving the system of steady-state equations. To simplify the procedure and to make our study more usable in practice, we used the software tool MOSEL to formulate the model and to calculate the main performance measures. By the help of MOSEL, we can use various performance tools (like SPNP, Stochastic Petri Net Package) to get these measures.

In this section, we show the base MOSEL program and the explanation of its main parts without the technical details of programming. This program belongs to the case of continued service after server's repair and request's generation is blocked during the server repairing. It does not contain 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-hom-cpu-cont.msl begins */
/*--------------------------------------- Definitions -----------*/
#define NT 3
/*================= No changes required below ==================*/
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};
/*--------------------------------------- Node Definitions ------*/
NODE busy_terminals[NT] = NT;
NODE retrying_terminals[NT] = 0;
NODE waiting_terminals[1] = 0;
NODE cpu_state[cpu_states] = cpu_idle;
NODE cpu[cpu_updown] = cpu_up;
/*-------------------------------------------- Transitions ------*/
IF cpu == cpu_up FROM cpu_idle, busy_terminals
   TO cpu_busy, waiting_terminals W prgen*busy_terminals;
IF cpu == cpu_up AND cpu_state == cpu_busy FROM busy_terminals
   TO retrying_terminals W prgen*busy_terminals;
IF cpu == cpu_up FROM cpu_idle, retrying_terminals
   TO cpu_busy, waiting_terminals W prretr*retrying_terminals;
IF cpu == cpu_up FROM cpu_busy, waiting_terminals
   TO cpu_idle, busy_terminals 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;
RESULT>> if(cpu==cpu_up) goodcpu += PROB;
RESULT if(cpu==cpu_up) busyterm +=(PROB * busy_terminals);
```

```
RESULT>> termutil = busyterm / NT;
RESULT>> if(cpu == cpu_up) retravg += (PROB*retrying_terminals);
RESULT if(waiting_terminals > 0) waitall += (PROB*waiting_terminals);
RESULT if(retrying_terminals > 0) retrall += (PROB*retrying_terminals);
RESULT>> resptime = (retrall + waitall) / NT / (prgen * termutil);
RESULT>> overallutil = cpuutil + busyterm;
/* retrialnr-hom-cpu-cont.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 queuing network contains five nodes: one node for the number of busy, retrying, and waiting terminals, respectively, and two nodes for the CPU. The CPU is idle and up and all the terminals are busy at the starting time.

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 requested output performance measures.

Table 1. Validations in the reliable case.

	Nonrel. Retrial (cont.)	Nonrel. Retrial (orbit)	Reliable in [5]
Number of sources	5	5	5
Request's generation rate	0.2	0.2	0.2
Service rate	1	1	1
Retrial rate	0.3	0.3	0.3
Utilization of the server	0.5394868123	0.5394867440	0.5394867746
Mean response time	4.2680691205	4.2680667075	4.2680677918

Table 2. Validations in the nonreliable case.

	Nonrel. Retrial (cont.)	Nonrel. Retrial (orbit)	Nonrel. 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	0.2232796561	0.2232796553	0.2232796452
Mean response time	1.4360656331	1.4360656261	1.4360655471

3. NUMERICAL EXAMPLES

In this section, we consider some sample numerical results to illustrate graphically the influence of the nonreliable server on the mean response time $E[T]$. As it can be seen in the first three 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*).

We used the tool SPNP which was able to handle the model with up to 126 sources. In this case, on a computer containing a 950 MHz processor and 512 MB RAM, the running time was approximately one second.

The results in the reliable case (with very low failure rate and very high repair rate) were validated by the (a little modified) Pascal program for the reliable case given in [5, pp. 272–274]. See Table 1 for some test results. The nonreliable case was tested with the nonreliable FIFO model, see Table 2.

3.1. Comments

- In Table 2, the results were tested by the help of nonreliable FIFO case, since if the retrial rate in the repeated calls model tends to infinity, the measures should approach the

Table 3. Input parameters.

	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

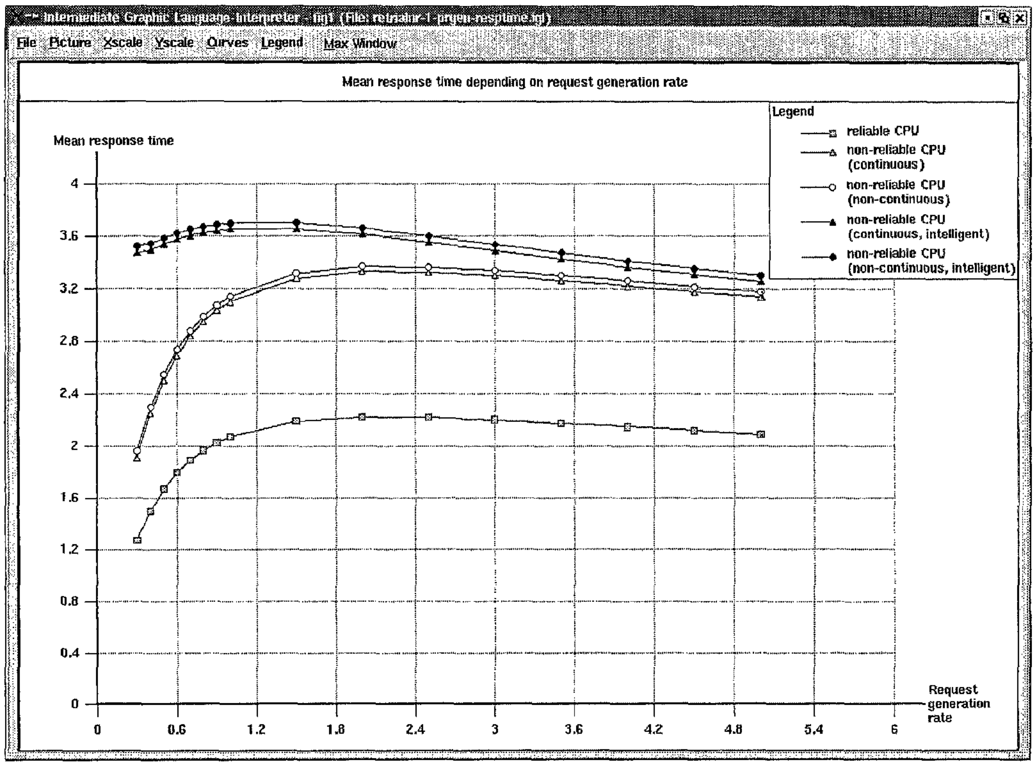

Figure 1. $E[T]$ versus primary request generation rate.

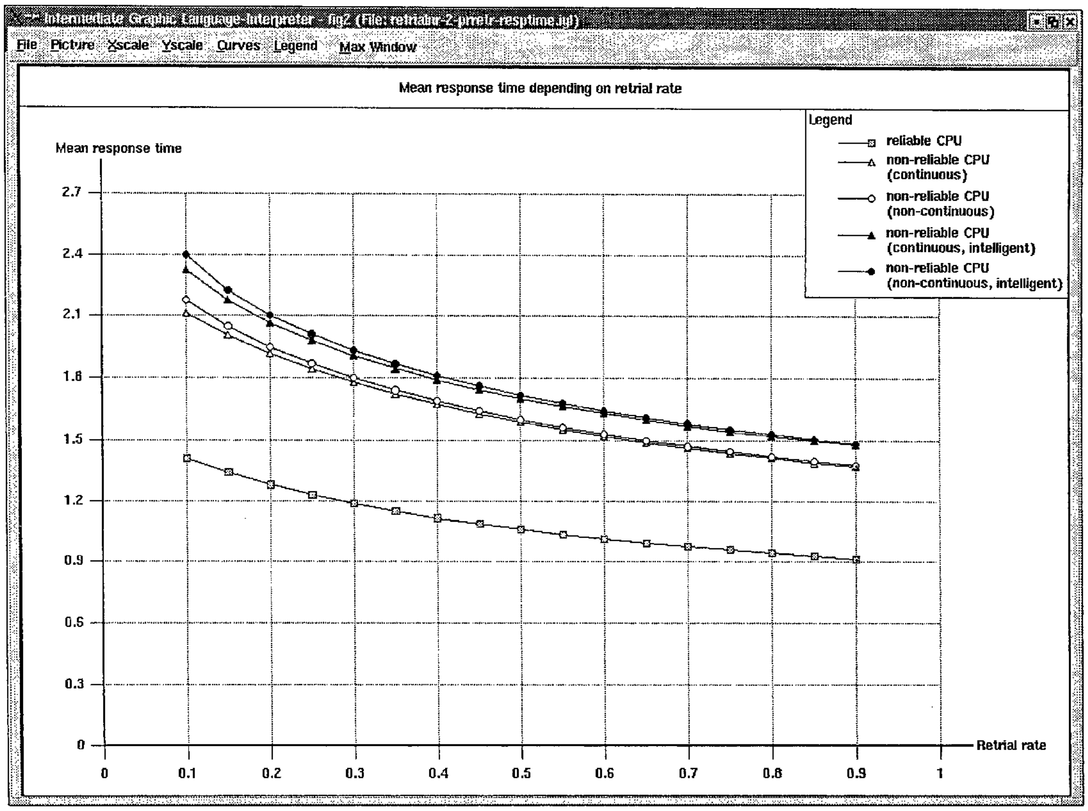

Figure 2. $E[T]$ versus retrial rate.

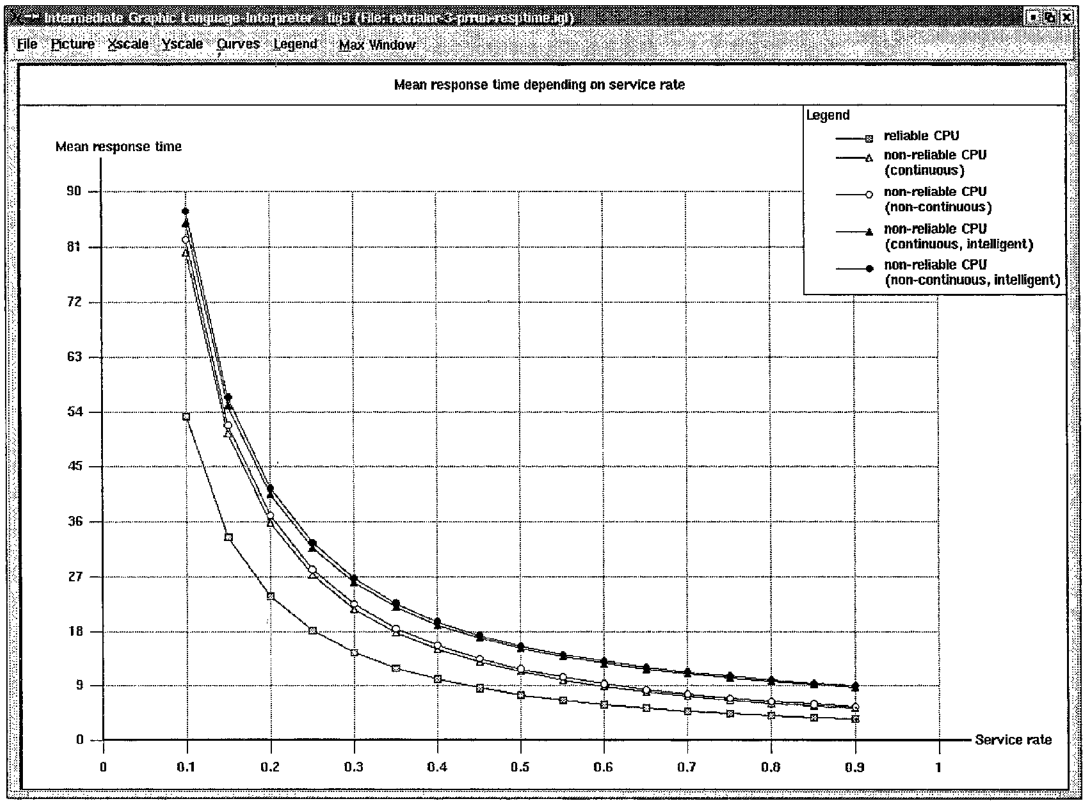

Figure 3. $E[T]$ versus service rate.

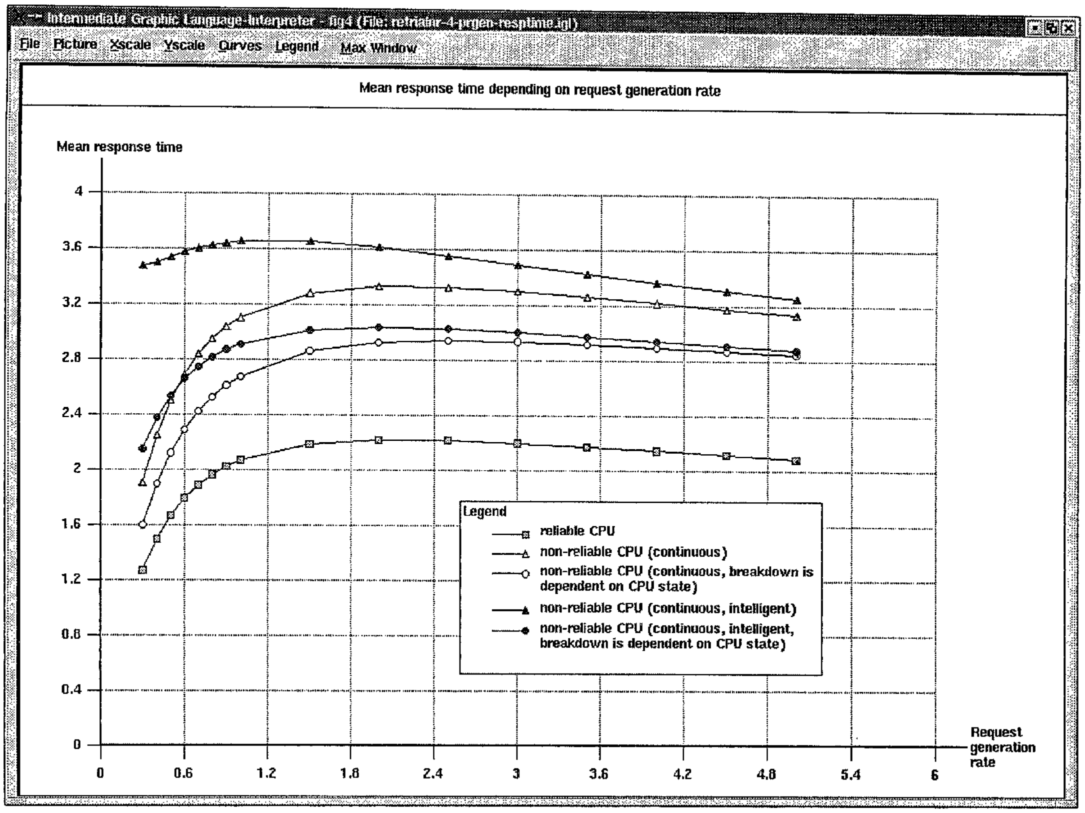

Figure 4. $E[T]$ versus primary request generation rate.

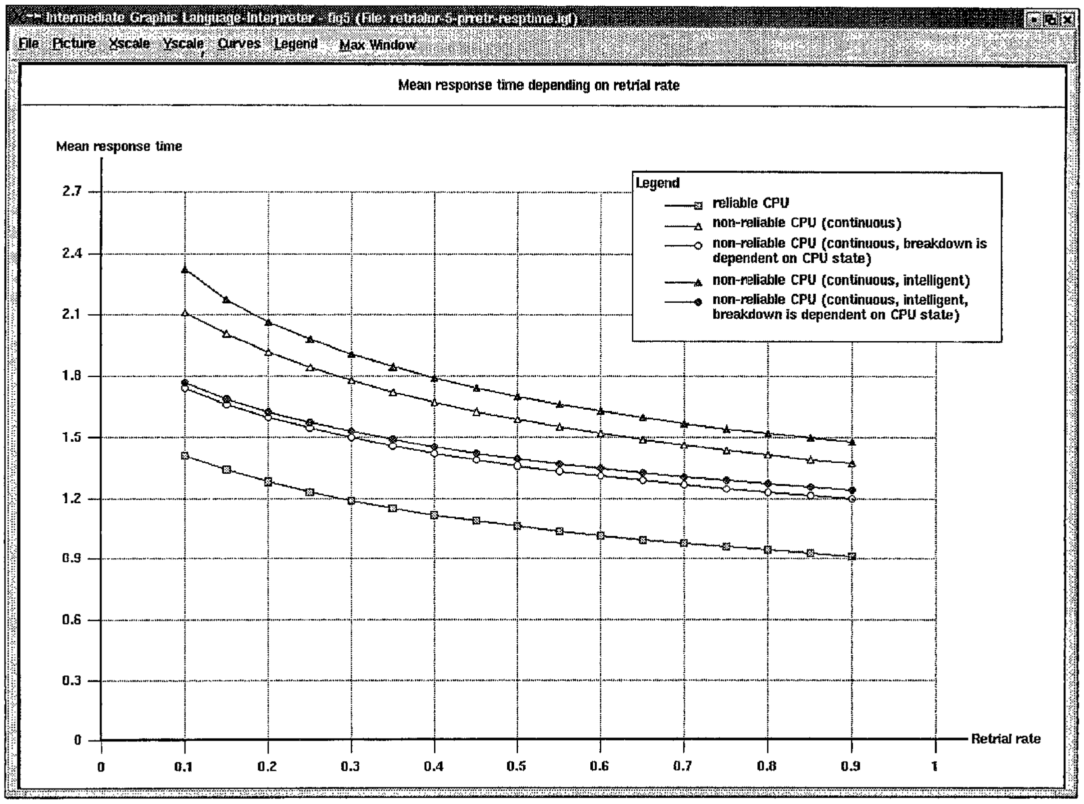

Figure 5. $E[T]$ versus retrial rate.

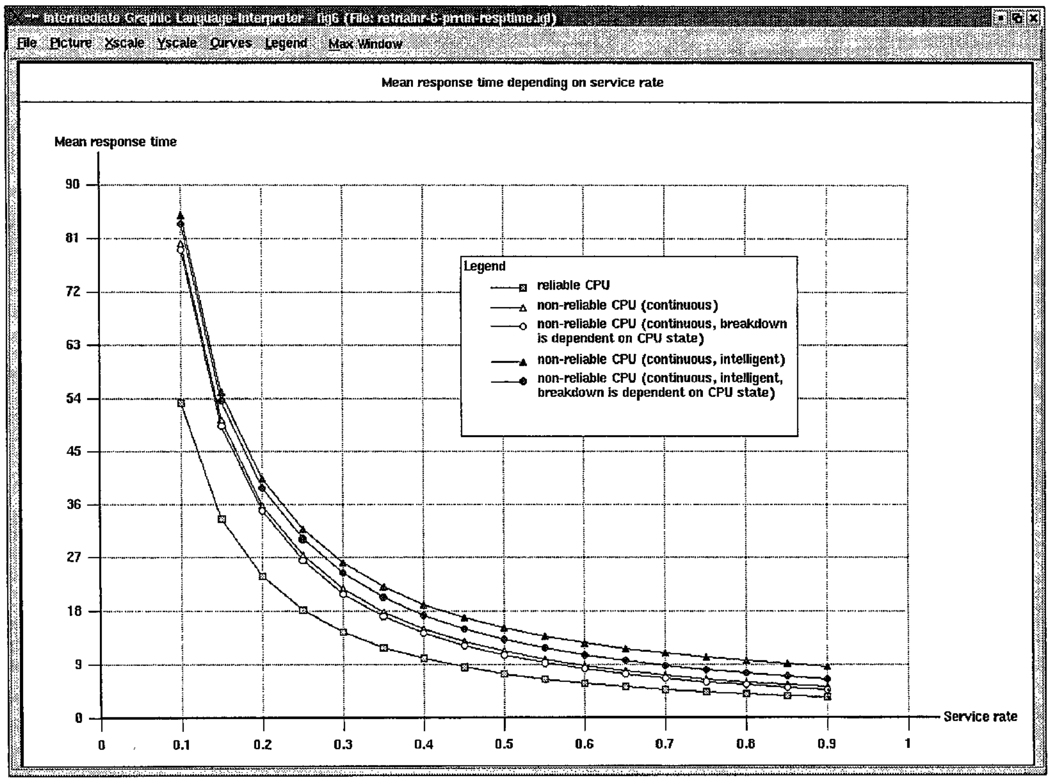

Figure 6. $E[T]$ versus service rate.

corresponding ones in FIFO discipline. The derived results are the same up to the sixth decimal digit.

- In Figures 1–3, we can see the mean response time $E[T]$ for the reliable and the nonreliable retrial system with continuous, noncontinuous 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. Figure 1 demonstrates a surprising phenomenon of retrial queues having a maximum of $E[T]$ which was noticed in [15], too. The difference between continuous, noncontinuous service, moreover blocked, unblocked systems' operations is clearly shown. However, if the retrial or service rate increases the continuous and noncontinuous service result in the same measure, as it was expected.
- 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.

4. CONCLUSIONS

In this paper, a finite-source homogeneous retrial queuing system with nonreliable server is studied. The novelty of the investigation is this nonreliability of the server which makes the system rather complicated. The 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 nonreliability of the server on the mean response times of the calls.

REFERENCES

1. J.R. Artalejo, Retrial queues with a finite number of sources, *J. Korean Math. Soc.* **35**, 503–525, (1998).
2. J.R. Artalejo, Accessible bibliography on retrial queues, *Mathl. Comput. Modelling* **30** (3/4), 1–6, (1999).
3. J. Barcelo, L. Escudero and J. Artalejo, Editors, *Proceedings of the 1^{st} International Workshop on Retrial Queues (WRQ'98), Volume 7*, Madrid, Top, (1998).
4. G.I. Falin, A survey of retrial queues, *Queueing Systems* **7**, 127–168, (1990).
5. G.I. Falin and J.G.C. Templeton, *Retrial Queues*, Chapman and Hall, London, (1997).
6. I.N. Kovalenko, N.Yu. Kuznetsov and P.A. Pegg, *Mathematical Theory of Reliability of Time Dependent Systems with Practical Applications*, John Wiley and Sons, Chichester, (1997).
7. N. Ravichandran, *Stochastic Methods in Reliability Theory*, John Wiley and Sons, New York, (1990).
8. K.S. Trivedi, *Probability and Statistics with Reliability, Queueing and Computer Science Applications*, Prentice Hall, Englewood Cliffs, NJ, (1982).
9. J.R. Artalejo, New results in retrial queueing systems with breakdown of the servers, *Statistica Neerlandica* **48**, 23–36, (1994).
10. A. Aissani and J.R. Artalejo, On the single server retrial queue subject to breakdowns, *Queueing Systems Theory and Applications* **30**, 309–321, (1998).
11. V.G. Kulkarni and B.D. Choi, Retrial queues with server subject to breakdowns and repairs, *Queueing Systems Theory and Applications* **7**, 191–208, (1990).
12. J. Wang, J. Cao and Q. Li, Reliability analysis of the retrial queue with server breakdowns and repairs, *Queueing Systems Theory and Applications* **38**, 363–380, (2001).
13. H. Takagi, *Queueing Analysis, A Foundation of Performance Evaluation, Volume 2, Finite Systems*, North-Holland, Amsterdam, (1993).
14. Y.N. Kornyshev, Design of a fully accessible switching system with repeated calls, *Telecommunications* **23**, 46–52, (1969).
15. G.I. Falin and J.R. Artalejo, A finite source retrial queue, *European Journal of Operational Research* **108**, 409–424, (1998).
16. G.I. Falin, A multiserver retrial queue with a finite number of sources of primary calls, *Mathl. Comput. Modelling* **30** (3/4), 33–49, (1999).
17. J.R. Artalejo and A. Gomez-Corral, Information theoretic analysis for queueing systems with quasi-random input, *Mathl. Comput. Modelling* **22** (3), 65–76, (1995).
18. J.R. Artalejo, V. Rajagopalan and R. Sivasamy, On finite Markovian queues with repeated attempts, *Investigacion Operativa* **9**, 83–94, (2000).
19. V.I. Dragieva, Single-line queue with finite source and repeated calls, *Problems of Information Transmission* **30**, 283–289, (1994).
20. G.I. Falin and A. Gomez Corral, On a bivariate Markov process arising in the theory of single-server retrial queues, *Statistica Neerlandica* **54**, 67–78, (2000).
21. A. Gomez-Corral, Analysis of a single-server retrial queue with quasi-random input and nonpreemptive priority, *Computers Math. Applic.* **43** (6/7), 767–782, (2002).
22. A.G. Kok, Algorithmic methods for single server systems with repeated attempts, *Statistica Neerlandica* **38**, 23–32, (1984).
23. H. Li and T. Yang, A single server retrial queue with server vacations and a finite number of input sources, *European Journal of Operational Research* **85**, 149–160, (1995).
24. S.N. Stepanov, The analysis of the model with finite number of sources and taking into account the subscriber behaviour, *Automation and Remote Control* **55**, 100–113, (1994).
25. H. Ohmura and T. Takahashi, An analysis of repeated call model with a finite number of sources, *Electronics and Communications in Japan* **68**, 112–121, (1985).
26. P. Tran-Gia and M. Mandjes, Modeling of customer retrial phenomenon in cellular mobile networks, *IEEE Journal of Selected Areas in Communications* **15**, 1406–1414, (1997).
27. D.J. Houck and W.S. Lai, Traffic modelling and analysis of hybrid fibercoax systems, *Computer Networks and ISDN Systems* **30**, 821–834, (1998).
28. G.K. Janssens, The quasi-random input queueing system with repeated attempts as a model for collision-avoidance star local area network, *IEEE Transactions on Communications* **45**, 360–364, (1997).
29. M.K. Mehmet-Ali, J.F. Hayes and A.K. Elhakeem, Traffic analysis of a local area network with star topology, *IEEE Transactions on Communications* **36**, 703–712, (1988).
30. A.I. Kalmychkov and G.A. Medvedev, Probability characteristics of Markov local-area networks with random-access protocols, *Automatic Control and Computer Science* **24**, 38–45, (1990).
31. I.I. Khomichkov, Study of models of local networks with multiple-access protocols, *Automation and Remote Control* **54**, 1801–1811, (1993).
32. K. Begain, G. Bolch and H. Herold, *Practical Performance Modeling, Application of the MOSEL Language*, Kluwer Academic, Boston, MA, (2001).
33. J. Sztrik and T. Gál, A recursive solution of a queueing model for a multi-terminal system subject to breakdowns, *Performance Evaluation* **11**, 1–7, (1990).