Annales Univ. Sci. Budapest., Sect. Comp. 32 (2010) 201-220

TOOL SUPPORTED PERFORMABILITY INVESTIGATIONS OF HETEROGENEOUS FINITE-SOURCE RETRIAL QUEUES

J. Sztrik (Debrecen, Hungary)

C.S. Kim (Wonju, Korea)

Abstract. This paper concerns with a retrial queueing system with a finite number of heterogeneous sources of calls serviced by a single unreliable 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 system'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 heterogeneous sources and the variability of this non-availability of the server which makes the system rather complicated. That is the reason why the MOSEL tool is used to formulate and solve the problem and the main stead-state performability measures are derived and graphically displayed. Several numerical calculations are performed to show the effect of the breakdown of the server on the mean response times of the calls and the mean number of requests staying at the service facility.

1. Introduction

Queueing system with repeated requests have wide practical use in designing telephone switching systems, telecommunication networks, computer networks and computer systems, call centers, etc. For a systematic account of the fundamental methods and results, applications, furthermore an accessible classified bibliography on this

topic the interested reader is referred to, for example [7], [9], [12], [16], [17], and references therein.

Since in practice some components of the systems are subject to random breakdowns (see for example [29], [34], [39]) 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 by queueing theory. For related literature the reader is referred to the works [6], [11], [9], [30], [40] 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 reviewed in detail by Takagi [37]. Since Kornyshev [28], 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 [7] 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 [18], Falin [19] 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 [9], [10], [15], [17], [20], [21], [27], [31], [35].

Retrial queues with quasi-random input are recent interest in modelling magnetic disk memory systems [33], cellular mobile networks [38], computer networks [22], and local-area networks with non-persistent CSMA/CD protocols [31], with star topology [24, 32], with random access protocols [25], and with multiple-access protocols [26].

This paper generalizes the results of [17] where homogeneous systems with reliable multiple servers were dealt with. Similarly, it is another extension of investigations for heterogeneous finite-source queueing systems without retrials but with server's breakdowns which were treated in [36]. Finally, it can be considered as the natural continuation of [3] in which reliable heterogeneous finite-source retrial systems were analyzed. It gives further investigations for retrial queues treated in [5] but because of page limitations some case studies were omitted.

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 $\vec{M}/\vec{M}/1//K$ retrial queueing model

In this paper single server finite-source queueing systems with the following assumptions are investigated. The primary calls are generated by K, $1 < K < \infty$ heterogeneous sources. The server can be in operational (up) or non-operational (down) states. If it is idle and up, it can serve the calls of the sources. Each of the sources can be in three states: generating a primary call (busy), sending repeated calls and under service. The i-th source can generate a primary call during interval $(t, t + dt)$ with probability $\lambda_i dt + o(t)$. 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 $\mu_i dt + o(t)$ if the server is available (up). If the server is busy, then the source starts generating of a Poisson flow of repeated calls with rate ν_i until it finds the server free. After service the source 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 *non-blocked (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 non-blocked operations during breakdowns.

Our objective is to give the main usual stationary performance and reliability (performability) 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 [13], 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).

2.1. The underlying Markov chain

Because of the exponentiality of the involved random variables the following process will be a Markov chain. The state of the system at time t can be described by the

process

$$X(t) = ((Y(t); \alpha_{C(t)}; \beta_1, \ldots, \beta_{N(t)}), t \geq 0),$$

where $Y(t) = 0$ if the server is up, $Y(t) = 1$ if the server is down, $C(t) = 0$ if the server is idle, $C(t) = 1$ if the server is busy, and $\alpha_{C(t)}$ is the index of the request under service at time t if the server is busy. Let $N(t)$ be the number of sources of repeated calls at time t, and because of the heterogeneity of the sources we need to identify their indices that are denoted by β_j, $j = 1, ..., N(t)$ if there is a customer in the orbit, otherwise the third component is 0.

Since its state space is finite the process $(X(t), t \geq 0)$ is ergodic for all values of the rate of generation of new primary calls, and from now on we assume that the system is in the steady state.

We define the stationary probabilities

$$P(q; 0; 0) = \lim_{t\to\infty} P(Y(t) = q; C(t) = 0; N(t) = 0), \qquad q = 0, 1,$$

$$P(q; j; 0) = \lim_{t\to\infty} P(Y(t) = q; \alpha_1 = j; N(t) = 0),$$

$$q = 0, 1, \qquad j = 1, \ldots, K,$$

$$P(q; 0; i_1, \ldots, i_k) = \lim_{t\to\infty} P(Y(t) = q; C(t) = 0; \beta_1 = i_1, \ldots, \beta_k = i_k),$$

$$q = 0, 1, \qquad k = 1, \ldots, K^*,$$

$$P(q; j; i_1, \ldots, i_k) = \lim_{t\to\infty} P(Y(t) = q; \alpha_1 = j; \beta_1 = i_1, \ldots, \beta_k = i_k),$$

$$q = 0, 1, \qquad k = 1, \ldots, K - 1.$$

where

$$K^* = \begin{cases} K - 1 & \text{for blocked case,} \\ K & \text{for non-blocked case.} \end{cases}$$

The traditional way is to derive the related Kolmogorov equations for these probabilities and using the norming condition somehow we have to solve the set of equations. Usually it is not so easy, but in our case these two steps are performed by the help of the tool demonstrated in the next subsection.

Once we have obtained these limiting probabilities the **main system performance measures** can be derived in the following way.

1. The server utilization with respect to source j

$$U_j = \text{P (the server is up and busy with source } j \text{)}$$

that is, we have to summarize all the probabilities where the first component is 0 and the second component is j. Formally,

$$U_j = \sum_{k=0}^{K-1} \sum_{i_1,\ldots,i_k \neq j} P(0; j; i_1, \ldots, i_k).$$

Hence the *server utilization*

$$U_S = E[Y(\infty) = 0;\ C(\infty) = 1] = \sum_{j=1}^{K} U_j.$$

2. Utilization of source i

$$U^{(i)} = \text{P (source } i \text{ generates a new primary call)}.$$

It should be mentioned that in the blocked case the server have to be up, but in the non-blocked case the request generation is independent of the server's state.

3. Utilization of the repairman

$$U_R = E[Y(\infty)] = \sum_{j=0}^{K} \sum_{k=0}^{K^*} \sum_{i_1,\ldots,i_k \neq j} P(1; j; i_1, \ldots, i_k).$$

4. Availability of the server

$$A_S = 1 - U_R.$$

Let us denote by $P_O^{(i)}$ the steady state probability that request i is staying in the orbit. It is easy to see that

$$P_O^{(i)} = \sum_{q=0}^{1} \sum_{j=0, j\neq i}^{K} \sum_{k=1}^{K^*} \sum_{i \epsilon (i_1,\ldots,i_k)} P(q; j; i_1, \ldots, i_k).$$

Similarly, it can easily be seen, that the steady-state probability $P_S^{(i)}$ that request i is at the server is given by

$$P_S^{(i)} = \sum_{q=0}^{1} \sum_{k=0}^{K^*} \sum_{i \neq i_1,\ldots,i_k)} P(q; i; i_1, ..., i_k).$$

Hence, the probability $P^{(i)}$ that request i is at the service facility can be obtained by

$$P^{(i)} = P_S^{(i)} + P_O^{(i)}.$$

5. Mean response time of source i

The derivation of the following formulae are based on [1, 37]. Let us denote by $E[T_i]$ the mean response time of customer i, and by γ_i the *throughput* of request i, that is, the mean number of times that request i is served per unit time. These are related by

$$\gamma_i = \frac{1}{E[T_i] + E[S_i]} = \lambda_i U^{(i)} = \mu_i U_i, \qquad i = 1, \ldots, K, \tag{2.1}$$

where $E[S_i]$ denotes the mean sojourn time of request i in the source. Since the server is subject to random breakdowns which may stop the operations of the sources, it is clear that $E[S_i] = E[D_i] + 1/\lambda_i \geq 1/\lambda_i$, where $E[D_i]$ denotes the mean delay time due to the failure of the server.

Hence, with the aid of (2.1) for $U^{(i)}$ we get

$$U^{(i)} = \frac{1/\lambda_i}{E[T_i] + E[S_i]} = \frac{\mu_i U_i}{\lambda_i} \leq 1 - P^{(i)}, \qquad i = 1, \ldots, K,$$

and for $P^{(i)}$ we have

$$P^{(i)} = \frac{E[T_i]}{E[T_i] + E[S_i]} = \gamma_i E[T_i] = \lambda_i U^{(i)} E[T_i], \qquad i = 1, \ldots, K, \tag{2.2}$$

which represents **Little's theorem** for request i at the service facility.

By the help of (2.2) we can express the mean response time $E[T_i]$ for request i as

$$E[T_i] = \frac{P^{(i)}}{\lambda_i U^{(i)}} = \frac{P^{(i)}}{\mu_i U_i}, \qquad i = 1, ..., K.$$

6. Mean waiting time of source i

The mean waiting time $E[W_i]$ of request i is due to the time spent in the orbit (irrespective of whether the server is up or down), and the delay time because of the server's failure. It is easy to see that $E[W_i]$ is given by

$$E[W_i] = E[T_i] - 1/\mu_i = \frac{P^{(i)} - U_i}{\mu_i U_i}, \qquad i = 1, \ldots, K.$$

7. Mean number of sources of repeated calls

$$N = E[N(\infty)] = \sum_{i=1}^{K} P_O^{(i)}.$$

8. Mean number of calls staying at the service facility

$$M = E[C(\infty) + N(\infty)] = \sum_{i=1}^{K} P^{(i)} = \sum_{i=1}^{K} (P_S^{(i)} + P_O^{(i)}) = \sum_{i=1}^{K} P_S^{(i)} + \sum_{i=1}^{K} P_O^{(i)}.$$

9. Mean rate of generation of primary calls

$$\overline{\lambda} = \sum_{i=1}^{K} \gamma_i = \sum_{i=1}^{K} \lambda_i U^{(i)} = \sum_{i=1}^{K} \mu_i U_i.$$

10. Blocking probability of primary call i

$$B_i = \begin{cases} \dfrac{\lambda_i \sum\limits_{j=1, j\neq i}^{K} \sum\limits_{k=0}^{K-1} \sum\limits_{i \neq i_1, \ldots, i_k} P(0; j; i_1, \ldots, i_k)}{\overline{\lambda}} & \text{for blocked case,} \\ \dfrac{\lambda_i \sum\limits_{j=1, j\neq i}^{K} \sum\limits_{k=0}^{K-1} \sum\limits_{i \neq i_1, \ldots, i_k} (P(0; j; i_1, \ldots, i_k) + P(1; j; i_1, \ldots, i_k))}{\overline{\lambda}} & \text{for non-blocked case.} \end{cases}$$

Hence *blocking probability of primary calls*

$$B = \sum_{i=1}^{K} B_i$$

which is the fraction of primary calls which were blocked (i.e. met the server busy).

It is easy to see that in the case of non-blocked operations (intelligent sources) with non-reliable server, $U^{(i)} = 1 - P^{(i)}, i = 1, ..., K$, and we get the same formulae derived in [3] that is, most performance measures can be expressed in the terms of the corresponding utilizations U_i as it was stated in [18].

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 a part of the base MOSEL program and explanations 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-het-cpu-cont.msl begins */
/*---------------------------------------- Definitions ---*/
#define  NT   3     // change it to 4, 5, 6, 7, ...

/*================== No changes required below =============*/
<1..NT> VAR double prgen#;
<1..NT> VAR double prretr#;
<1..NT> 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-het-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 heterogeneous terminals with shortcuts: $prgen$ 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 $E[T_i]$ and the mean number of request M staying at the service facility (in the orbit and at the server). In the cases when the server's failure rate is very small and the repair rate is large the non-reliable model should be very close to the reliable system. The results in the homogeneous case were validated by the Pascal program given in [17]. For the heterogeneous case the calculations were checked by the numerical results of [3]. In the case of homogeneous sources but with server's breakdowns the program was tested by the examples of [4].

3.1. Validation of results

In Table 1 some test results are collected when the server's failure rate is quite small and the requests' retrial rates are quite large. Hopefully the corresponding performance measures should be very close to each other in the case of continued, restarted (abbreviated by orbit) service after repair and FIFO disciplines. As we can see in the Table 1, the results confirm our expectation.

	non–rel. retrial(cont.)	non–rel. retrial(orbit)	non–rel. FIFO
Number of sources:	3	3	3
Request's generation rate:	0.2, 0.3, 0.5	0.2, 0.3, 0.5	0.2, 0.3, 0.5
Service rate:	1, 1.2, 1.1	1, 1.2, 1.1	1, 1.2, 1.1
Retrial rate:	1e+20	1e+20	-
Server's failure rate:	0.002	0.002	0.002
Server's repair rate:	0.04	0.04	0.04
Utilization of the server:	0.578593008176	0.578593460071	0.578595143583
Mean response time			
Source 1:	1.61016598407	1.61027143737	1.6109393482
Source 2:	1.41365083148	1.41357129589	1.41287007613
Source 3:	1.35362123345	1.35362137877	1.35372999206

Table 1. Validations

3.2. Further examples

In this part some additional sample results are displayed showing the effect of different parameters on mean response times of calls and the mean number of requests staying at the service facility. The input parameters are collected in Table 2. In each cases independent breakdowns were treated. In homogeneous cases the parameters are the arithmetic means of the corresponding values.

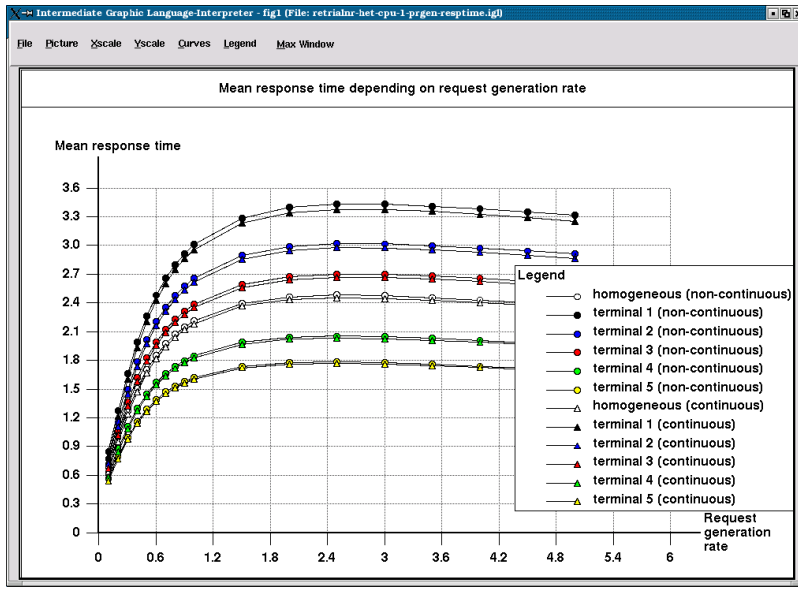

Figure 1. $E[T]$ versus primary request generation rate

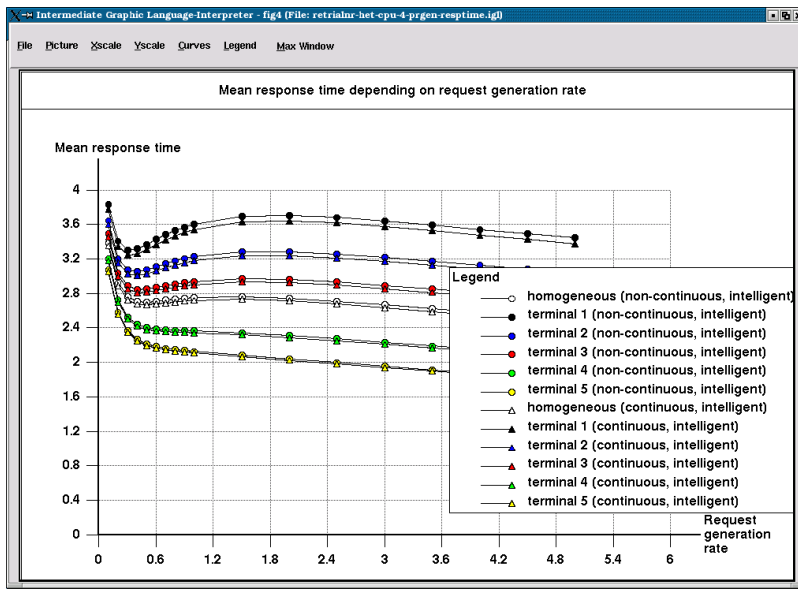

Figure 2. $E[T]$ versus primary request generation rate

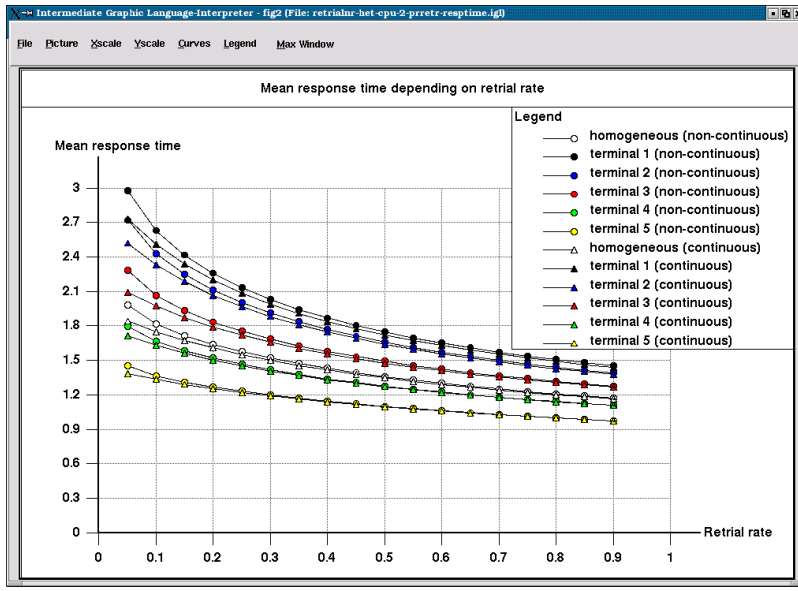

Figure 3. $E[T]$ versus retrial rate

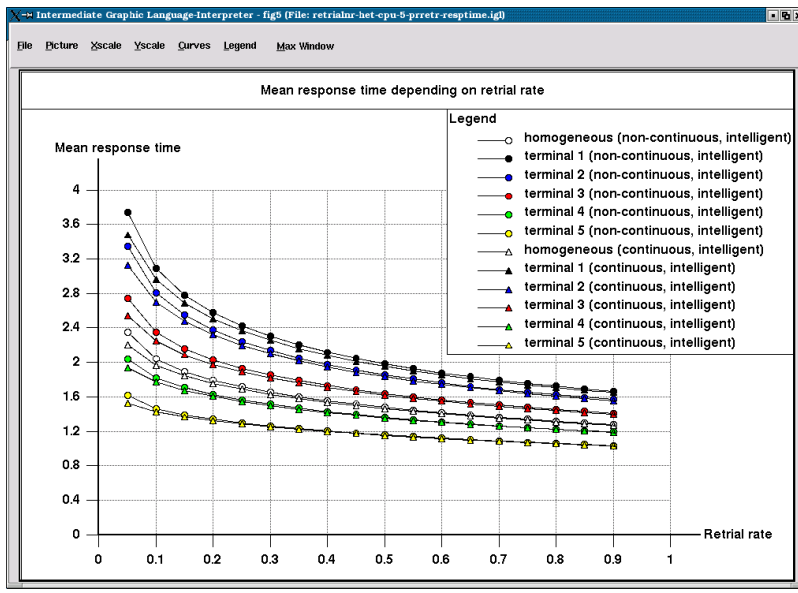

Figure 4. $E[T]$ versus retrial rate

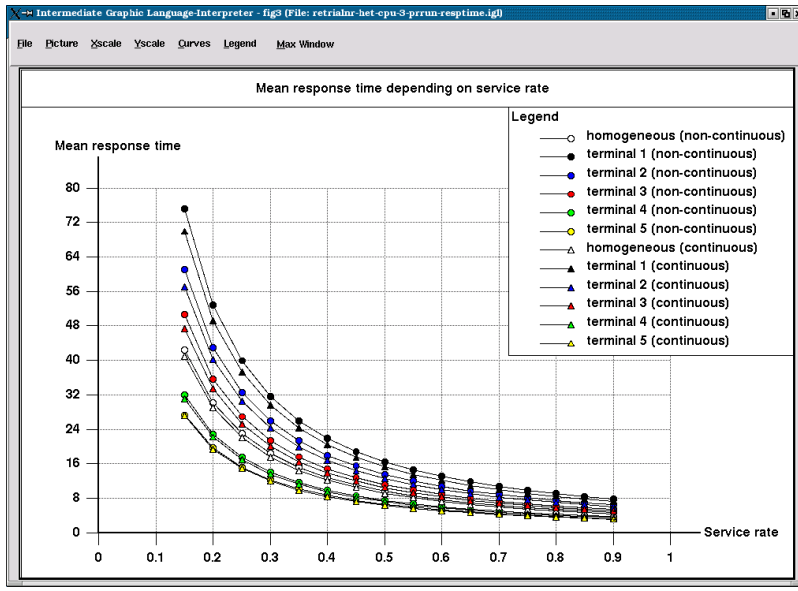

Figure 5. $E[T]$ versus service rate

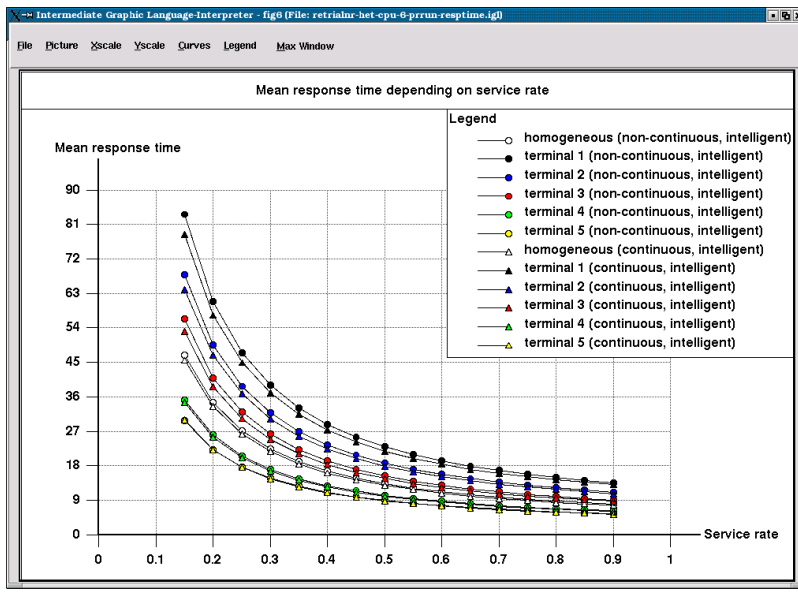

Figure 6. $E[T]$ versus service rate

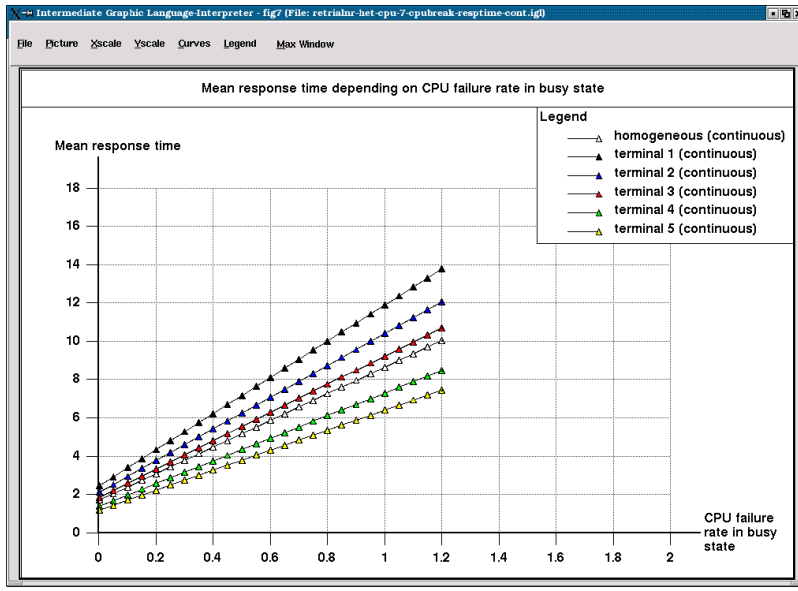

Figure 7. $E[T]$ versus CPU failure rate in busy state

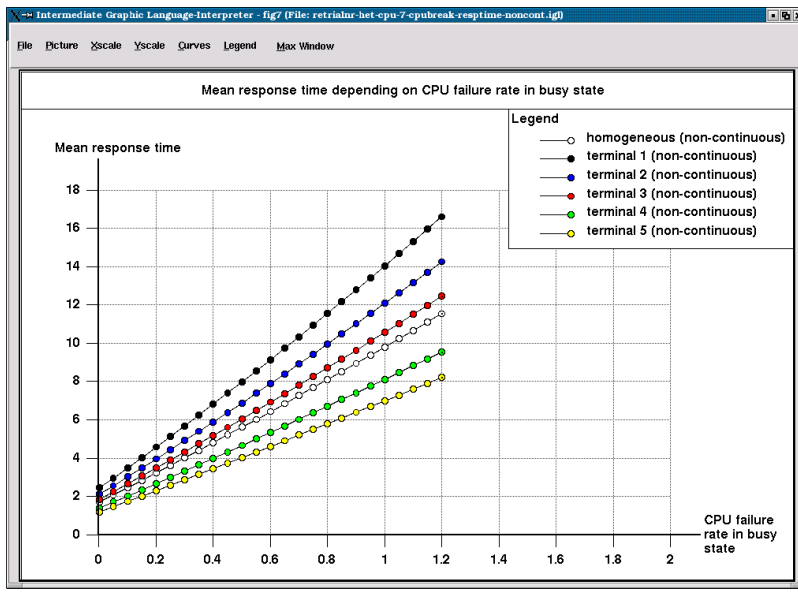

Figure 8. $E[T]$ versus CPU failure rate in busy state

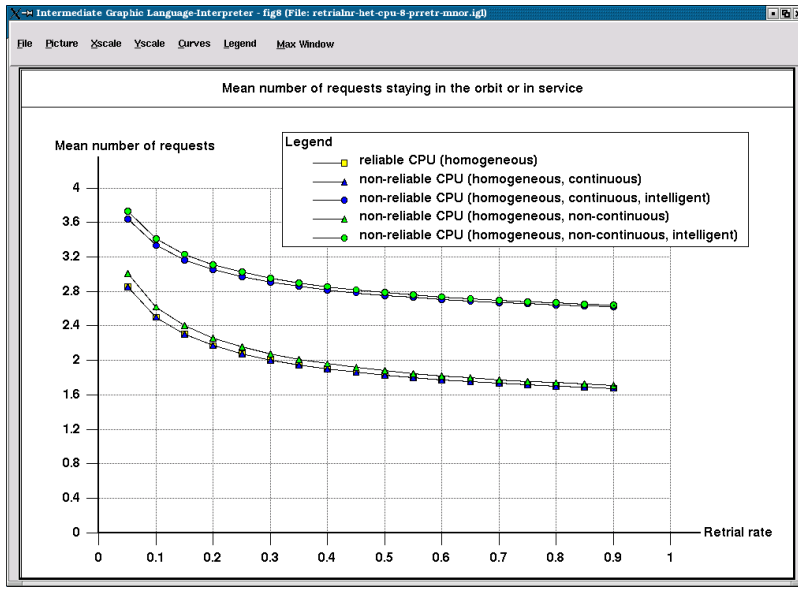

Figure 9. M versus retrial rate

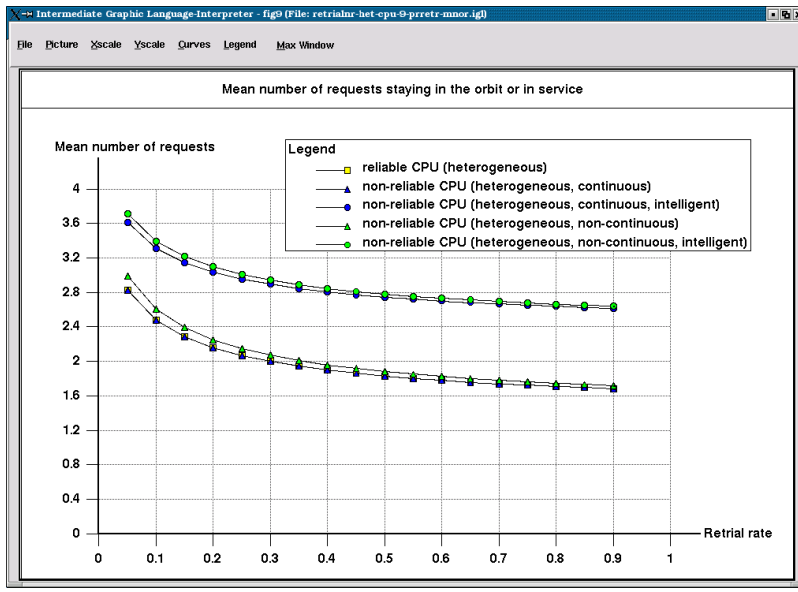

Figure 10. M versus retrial rate

In Figures 1 - 6 the mean response time $E[T_i]$ is displayed in continuous and non-continuous service after repair in the case of blocked, non-blocked (intelligent sources) operations, as the function of primary request generation, retrial and service rates, respectively.
In Figures 7 - 8 also $E[T_i]$ is pictured in the case of blocked operations with continuous and non-continuous service after repair as the function of the server's failure rate in busy state, respectively.
Finally, in Figures 9 - 10 the mean number of request M staying at the service facility can be seen in reliable and non-reliable case with homogeneous, heterogeneous sources under blocked, non-blocked operations combined with continuous and non-continuous service after repair as the function of the retrial rate of repeated calls.

3.3. Comments

- In Table 1 we can see that there is a slight difference between continuous, restarted service and the FIFO disciplines which can be expected.
- In Figures 1-2 we can see the mean response time $E[T_i]$ for the non-reliable system with continuous, non-continuous service after repair, with blocked and non-blocked operations during service failure when the primary request generation rate increases. Figure 1 demonstrates a surprising phenomenon of retrial queues having a maximum of $E[T_i]$ which was noticed for homogeneous sources treated in [18], too. The difference between continuous, non-continuous service, moreover blocked, non-blocked (intelligent) systems's operations is clearly shown. The non-continuous case always have longer response times. Similarly, the intelligent sources suffers from longer times, too. However, their curves are very interesting, since for some sources at the beginning they decrease then increase, finally decrease again. This shows that the systems is very complex and at different parameter setup we can see different effects.
- In Figures 3, 4, and similarly in Figures 5, 6 the effect of retrial rate, service rate is demonstrated on $E[T_i]$, respectively. Again the non-continuous case always have longer response times, and the intelligent sources suffers from longer times, too. However, the difference between the continuous and non-continuous case decreases as the corresponding rate increase. Furthermore, in each cases $E[T_i]$ decreases as we expected.
- In Figures 7, 8 the effect of server's failure is displayed in the blocked operations case. Again the non-continuous case have always longer response times, which is clear. Moreover, for reliable busy server we have the same results which was expected, too. The almost linear increase in $E[T_i]$ in each cases can be explained as follows. Since the failure of the server blocks all the operations and the response time is the sum of the down time of the server, service and repeated call generation time of the request (which do not change during the failure) thus the failure has a linear effect on this measure.

- In Figures 9, 10 the effect of the retrial rate on the mean number of request M staying at the service facility is pictured. The curves confirm our expectation with respect to the service after repair and blocked operations during the failure. M decreases as the retrial rate increases, which is clear. It is also worth pointing out that the values for the reliable case and non-reliable blocked case coincide. However, it is not so surprising since during the failure the number of requests remain the same.

	NT	λ	μ	ν	δ	γ	τ
Fig. 1,2	5	x axis	4.1,4.3,4.5,4.7,4.9	0.35,0.4,0.45,0.6,0.7	0.05	0.05	0.1
Fig. 3,4	5	2.5,3,4,6.5,9	6,7,8,13,16	x axis	0.05	0.05	0.1
Fig. 5,6	5	0.04,0.06,0.1,0.14,0.16	x axis	0.2,0.25,0.3,0.55,0.7	0.05	0.05	0.1
Fig. 7,8	5	0.6,0.7,0.8,0.9,1	4.1,4.3,4.5,4.7,4.9	0.35,0.4,0.45,0.6,0.7	0.05	x axis	0.1
Fig. 9	5	0.1	0.5	x axis	0.05	0.05	0.1
Fig. 10	5	0.06,0.08,0.1,0.12,0.14	0.3,0.4,0.5,0.6,0.7	x axis	0.05	0.05	0.1

Table 2. Input parameters

4. Conclusions

In this paper a heterogeneous finite-source homogeneous retrial queueing system with non-reliable server is studied. The novelty of the investigation is this non–reliability of the server and the heterogeneity of sources 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 and the mean number of requests staying at the service facility.

Acknowledgment

This research has been supported by Sangji University Research Grant 2009. The work is also supported by the TÁMOP 4.2.1./B-09/1/KONV-2010-0007 project. The project is implemented through the New Hungary Development Plan, co-financed by the European Social Fund and the European Regional Development Fund.

References

[1] **Almási B.,** Response time for finite-source heterogeneous nonreliable queueing systems, *Computers and Mathematics with Applications* , **31** (1996), 55-59.

[2] **Almási B., Bolch G. and Sztrik J.,** Performability Modeling of Non-homogeneous Terminal Systems Using MOSEL, *5th International Workshop on Performnability Modeling of Computer and Communication Systems, Erlangen, Germany, 2001*, 37-41.

[3] **Almási B., Bolch G. and Sztrik J.** Heterogeneous finite-source retrial queues, *Journal of Mathematical Sciences*, **121** (2004), 2590-2596.

[4] **Almási B., Roszik J. and Sztrik J.,** Homogeneous finite-source retrial queues with server subject to breakdowns and repairs, *Mathematical and Computer Modeling*, **42** (2005), 673-682.

[5] **Almási B., Roszik J. and Sztrik J.,** Heterogeneous Finite-Source Retrial Queues with Server Subject to Breakdowns and Repairs, *Journal of Mathematical Sciences*, **132** (2006), 677-685.

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

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

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

[9] **Artalejo J.R. and Gomez-Corral A.,** *Retrial Queueing Systems, A Computational Approach,* Springer Verlag, Berlin, 2008.

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

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

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

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

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

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

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

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

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

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

[20] **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.

[21] **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.

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

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

[24] **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.

[25] **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.

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

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

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

[29] **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.

[30] **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.

[31] **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.

[32] **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.

[33] **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.

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

[35] **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.

[36] **Sztrik J. and Pósafalvi A.,** On the heterogeneous machine interference with limited server's availability, *European Journal of Operational Research*, **28** (1987), 321-328.

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

[38] **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.

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

[40] **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.

(Received September 2, 2010)

J. Sztrik
Faculty of Informatics
University of Debrecen
H-4010 Debrecen, P.O.B. 12
Hungary
jsztrik@inf.unideb.hu

C.S. Kim
Sangji University
Wonju, Korea