Homogeneous Finite-Source Retrial Queues with Search of Customers from the Orbit*

Patrick Wüchner[1], János Sztrik[2], and Hermann de Meer[1]

[1] Faculty of Informatics and Mathematics,
University of Passau, Innstraße 43,
94032 Passau, Germany,
patrick.wuechner@uni-passau.de
[2] Faculty of Informatics, University of Debrecen,
Egyetem tér 1. Po.Box 12, 4010 Debrecen, Hungary,
jsztrik@inf.unideb.hu

Abstract. We consider a retrial queueing system with a finite number of homogeneous sources of calls and a single server. Each source generates a request after an exponentially distributed time. An arriving customer finding the server idle enters into service immediately; otherwise the customer enters into an orbit. The service times are supposed to be exponentially distributed random variables. An orbiting customer competes for service, the inter-retrial times are exponentially distributed. Upon completion of a service, with a certain probability the server searches for an orbiting customer. Assuming the search time to be negligible, the source, service, and retrial times to be independent random variables, we perform the steady-state analysis of the model computing various steady-state performance measures and illustrative numerical examples are presented.
The novelty of the investigation is the introduction of orbital search by the server for customers in finite-source retrial queues. The MOSEL-2 tool is used to formulate and solve the problem.

Keywords: retrial queueing systems, finite number of sources, orbital search, performance tool, performance measures

1 Introduction

Retrial queues have been widely used to model many problems arising in telephone switching systems, telecommunication networks, computer networks, 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, [4, 13, 12], and the references therein.

* This research is partially supported by the German-Hungarian Intergovernmental Scientific Cooperation, HAS-DFG, 436 UNG 113/180/0-1, by the Hungarian Scientific Research Fund, OTKA K60698/2006, by the Network of Excellence EuroFGI – IST 028022, and by EPSRC, GR/S69009/01.

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 (aka quasi-random input) models. Queueing systems without retrials, i.e., systems with classical waiting lines and finite population, have been reviewed in detail by Takagi [21]. Since Kornyisev [18], 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 [4] for systems of type $M/G/1//K$ and $M/M/c//K$ in Kendall's notation [7]. In addition, in the papers of Falin and Artalejo [15, 14], not only the outside observer's distributions of the systems in steady state, but also the stationary performance characteristics are considered in 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 among customer, were investigated, too. Further recent results with finite-source of primary requests can be found in [1–3, 10, 12, 16, 9, 20].

Retrial queues with quasi-random or finite-source input are recent interest in modeling magnetic disk memory systems, cellular mobile networks, computer networks, and local-area networks with nonpersistent CSMA/CD protocols. See, e.g., [4, 17, 18].

In the retrial setup, each service is preceded and followed by the server's idle time because of the ignorance of the status of the server(s) and orbital customers by each other. We are interested in designing finite-source retrial queues that reduce the server's idle time and achieve this by the introduction of search of orbital customers immediately after a service completion. For this, with search we associate a probability. Search for orbital customers was introduced in [19] where the authors examined classical queue with search for customers immediately on termination of a service. Recently, retrial queueing systems with an infinite source of customers where the server(s) search for customers after service have been investigated in [5, 8, 11]. The following application examples for retrial queues are mentioned in [11].

Consider an inventory where customers requiring the item arrive for purchase. In case the item is out of stock at a demand epoch, such customers, after recording their name in a register, proceed to the orbit which might be their own residence. On replenishment, the management scans through the register and select persons according to some priority, to provide the item.

Yet another example is the repair service with search for customers. Here, a job shop keeps a register of customers who are forced to leave the system since they encountered a busy server at the time of arrival. On completing a service, the server decides to have the next service by picking up an unsatisfied orbital customer with probability p_j, where j is the number of customers on the orbit. Simultaneously, the customers in the orbit try to reach the server themselves and there is a competition between primary and orbital customers and the search mechanism.

In this paper, finite-source systems with the following assumptions are investigated. Consider a single-server queueing system, where the primary calls (customers, requests, or jobs) are generated by K, $1 < K < \infty$, homogeneous sources. The sources' generation times are assumed to be exponentially distributed random variables with rate λ, that is, each source 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 service is finished during the interval $(t, t + dt)$ with probability $\mu dt + o(dt)$. If the server is busy, then the source starts the generation of a Poisson flow of repeated calls with rate ν until it finds the server free. After service, the call returns to the source which can then generate a new primary call again, and the server becomes idle. Upon completion of a service, with probability $p, 0 \leq p \leq 1$, the server takes a customer, if any, from the orbit for service. This behavior is called "orbital search". The time needed to search for a customer is assumed to be insignificant. With probability $1 - p$, the server will become idle until an orbital customer or a new arrival captures the server. Note that when $p = 1$ the model reduces to the classical $M/M/1//K$ queue and when $p = 0$ it becomes $M/M/1//K$ with retrials. All the times involved in the model are assumed to be mutually independent of each other.

Note that finite-source models are often more complicated than models with an infinite number of sources. This is because in infinite-source models, the overall arrival rate is independent from the number of jobs in the system. Especially for a system with only a few sources, the infinite-source model is only a very coarse approximation in general. In this paper, we also focus on a single server to preserve conciseness. However, the presented model can be extended easily to a model with multiple servers.

Our objective is to give the main usual stationary performance measures of the system and to display the effect of different parameters on them. To achieve this goal, a tool called MOSEL-2 (Modeling, Specification and Evaluation Language, version 2) developed at the University of Erlangen and University of Passau, Germany, see [6, 22], 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 the Intermediate Graphic Language (IGL, see [6]).

The paper is organized as follows. In Section 2, the full description of the model is given by the help of the corresponding generalized stochastic Petri net and underlying continuous-time two-dimensional Markov chain. Then, the main performance measures of the system are derived that can be obtained conveniently using the MOSEL-2 tool. In Section 3, several numerical results are presented and some comments are made. Finally, the paper ends with a conclusion and directions for future work.

2 The $M/M/1//K$ Retrial Queueing Model with Search of Customers from the Orbit

Starting from a high-level description of our model in the form of a generalized stochastic Petri net (GSPN), in this section, the structure of the underlying Markov chain will be derived.

2.1 The High-Level Petri Net Model

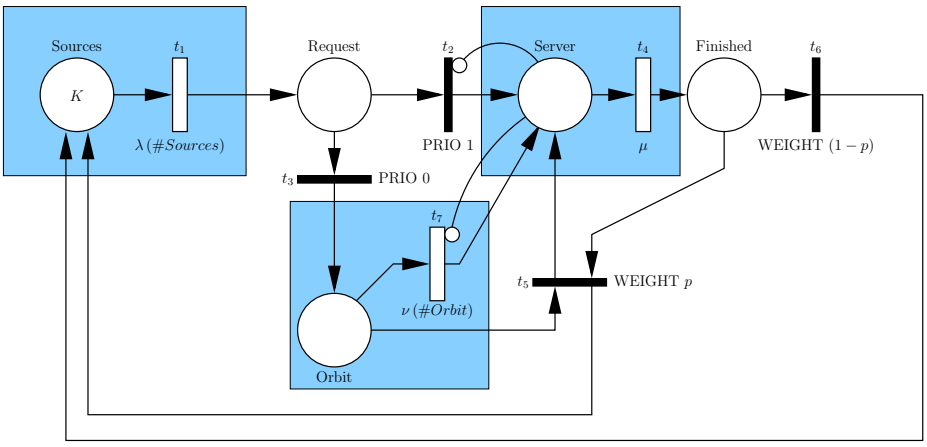

Fig. 1. Petri net model of finite-source retrial queue with orbital search.

The graphical representation of a possible Petri net modeling the scenario described in Section 1 is given in Figure 1. For the theoretical and formal background on GSPNs and stochastic reward nets, we refer the interested reader to [7].

The 5-tuple (#Sources, #Request, #Orbit, #Server, #Finished) describes all possible markings (i.e., states) of the given GSPN. For instance, the initial marking is given by $m_0 = (K, 0, 0, 0, 0)$.

For K=4, the extended reachability graph of the Petri net is given in Figure 2. There, the boxes with rounded corners labeled with $m_i, 0 < i < 7$, are tangible markings, and the dark gray boxes labeled $m_{vi}, 0 < i < 7$, are vanishing markings. The probability for the Petri net residing in one of the vanishing markings m_{vi} is almost surely zero, because at least one immediate transition is enabled in these markings. Thus, we can abstract from the vanishing markings by reducing the extended reachability graph. The resulting reachability graph shown in Fig. 3 can then be easily mapped to a continuous-time Markov chain.

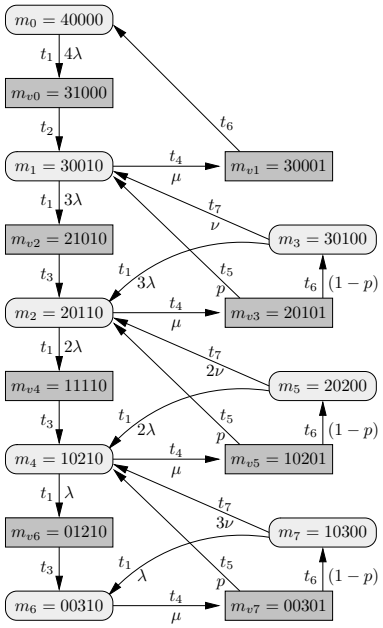

Fig. 2. Extended reachability graph of Petri net model with K=4.

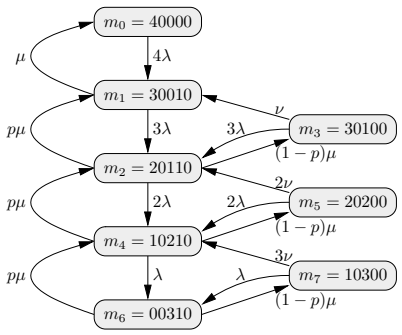

Fig. 3. Reduced reachability graph of Petri net model with K=4.

2.2 The Underlying Markov Chain

The system state of our model at time t can be described by the two-dimensional stochastic process $X(t) = (C(t), N(t))$, where $C(t) = 0$ if the server is idle, $C(t) = 1$ if the server is busy, and $N(t)$ is the number customers in the orbit at time t. The Petri net's marking at time t can then be given in the form $m(t) = (K - (C(t) + N(t)), 0, N(t), C(t), 0)$. Because of the exponentiality of all involved random variables, $X(t)$ constitutes a continuous-time Markov chain with finite state space $S = \{0, 1\} \times \{0, 1, ..., K - 1\}$. Note, that the maximum number of customers in the orbit is $K - 1$, because customers can only enter the orbit if there is one customer at the server. Thus, the Markov chain comprises $2K$ states altogether.

Since the state space of the process $(X(t), t \geq 0)$ is both finite and irreducible, i.e., all states are pairwise reachable from each other via a finite number of transitions, the process is ergodic for all values of the generation rate of primary calls λ. From now on, the system will be assumed to be in the steady state, i.e., $t \to \infty$.

2.3 The Main Performance Measures

Based on the underlying Markov chain, we define the stationary probabilities as follows:

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

The stationary probabilities can be derived solving the well known equation (see [7] for details)

$$\boldsymbol{\pi}\mathbf{Q} = \mathbf{0}\,, \quad \boldsymbol{\pi}\mathbf{1} = 1\,, \tag{1}$$

where $\boldsymbol{\pi} = [P(r,j)]$ is the stationary probability vector containing the stationary probabilities ordered according to the markings' order in the following way:

$$\begin{aligned}\boldsymbol{\pi} &= (\pi_0, \pi_1, \ldots, \pi_{2K-1}) \\ &= (P(0,0), P(1,0), P(1,1), P(0,1), P(1,2), P(0,2), \ldots, P(1,K-1), P(0,K-1))\,.\end{aligned} \tag{2}$$

Following the order given by Eq. (2), the infinitesimal generator matrix $\mathbf{Q}$ of the underlying Markov chain for $K = 4$ can be built directly by inspecting Fig. 3:

$$\mathbf{Q} = \begin{pmatrix} -4\lambda & 4\lambda & 0 & 0 & 0 & 0 & 0 & 0 \\ \mu & -\mu-3\lambda & 3\lambda & 0 & 0 & 0 & 0 & 0 \\ 0 & p\mu & -\mu-2\lambda & (1-p)\mu & 2\lambda & 0 & 0 & 0 \\ 0 & \nu & 3\lambda & -\nu-3\lambda & 0 & 0 & 0 & 0 \\ 0 & 0 & p\mu & 0 & -\mu-\lambda & (1-p)\mu & \lambda & 0 \\ 0 & 0 & 2\nu & 0 & 2\lambda & -2\nu-2\lambda & 0 & 0 \\ 0 & 0 & 0 & 0 & p\mu & 0 & -\mu & (1-p)\mu \\ 0 & 0 & 0 & 0 & 3\nu & 0 & \lambda & -3\nu-\lambda \end{pmatrix}. \tag{3}$$

The derivation of the stationary probabilities $P(r,j)$ by solving Eq. (1) can be done in several ways, e.g., by exploiting the special structure of the underlying Markov chain arising from Fig. 3. This structure also inheres in the block-tridiagonal form of the generator matrix $\mathbf{Q}$ and can be seen in Eq. (3). However, in this paper, we focus on an engineering point of view. Therefore, in Section 2.4 it is described, how the software tool MOSEL-2 can be used to obtain the stationary probabilities and other performance measures of interest in a more convenient way. It is also shown in Section 3.2 that the MOSEL-2 model is sufficiently scalable for wide ranges of the number of sources and the number of servers.

Knowing the stationary probabilities the main performance measures of the finite-source retrial queue can be obtained as follows:

- *Utilization of the server:* $U_S = \sum_{j=0}^{K-1} P(1,j)\,.$
- *Mean number of customers in the orbit:* $N = \sum_{r=0}^{1}\sum_{j=0}^{K-1} jP(r,j)\,.$
- *Mean number of calls in the orbit or in service:* $M = \sum_{r=0}^{1}\sum_{j=0}^{K-1}(r+j)P(r,j)\,.$
- *Mean generation rate of primary calls:* $\overline{\lambda} = \lambda(K-M)\,.$
- *Mean response time:* $E[T] = M/\overline{\lambda}\,.$
- *Mean waiting time:* $E[W] = N/\overline{\lambda}\,.$
- *Blocking probability of a primary call:* $B = \frac{\lambda E[K-C(t)-N(t),C(t)=1]}{\overline{\lambda}}\,.$

2.4 The MOSEL-2 Model Implementation

To formulate and evaluate the GSPN model shown in Fig. 1, we used the Modeling, Specification and Evaluation Language (MOSEL-2). The MOSEL-2 evaluation environment is capable of solving model configurations up to approximately 65000 sources. The MOSEL-2 model is presented in Listing 1.1. There, however, we refrain from stating the lines of code ("Picture Part") used for generating the graphical results presented in Section 3 due to space limitations. We refer the interested reader to [6] and [22] for more details.

Listing 1.1. MOSEL-2 model of Fig. 1.

```
1  /*** CONSTANTS AND PARAMETERS ***************************************************/
2  CONST      K      := 3;                                          // population size
3  CONST      mu     := 1;                                          // service rate
4  PARAMETER lambda := 0.0001, 0.001, 0.01 .. 0.1 STEP 0.01;       // request gen. rate
5  PARAMETER nu     := 0.001, 0.0025, 0.005;                       // retrial rate
6  PARAMETER p      := 0.00000001, 0.00001, 0.01;                  // search probability
7
8  /*** NODES **********************************************************************/
9  NODE Sources[K]  := K;                                          // sources
10 NODE Request[1]  := 0;                                          // generated call
11 NODE Server[1]   := 0;                                          // the server
12 NODE Orbit[K]    := 0;                                          // retrying customers
13 NODE Finished[1] := 0;                                          // serviced customer
14
15 /*** RULES ***********************************************************************/
16 FROM Sources   TO Request  RATE Sources*lambda;                  // primary requests
17 FROM Request   TO Server   PRIO 1;                              // to server if idle
18 FROM Request   TO Orbit    PRIO 0;                              // to orbit  if busy
19 FROM Orbit     TO Server   RATE Orbit*nu;                       // retrials
20 FROM Server    TO Finished RATE mu;                             // service
21 FROM Finished  TO Sources  WEIGHT 1-p;                          // without polling
22 FROM Finished, Orbit TO Server, Sources WEIGHT p;               // with polling
23
24 /*** RESULTS *********************************************************************/
25 PRINT util_server := UTIL(Server);                              // server utilization
26 PRINT mean_lambda := MEAN(Sources)*lambda;                      // mean arrival rate
27 PRINT mean_T      := (K-MEAN(Sources))/mean_lambda;             // mean response time
28 PRINT mean_orbit  := MEAN(Orbit);                              // mean cust. in orbit
```

Please note the compactness of the MOSEL-2 model implementation. Furthermore, the given model can be easily extended to homogeneous multiple-server finite-source retrial queues with orbital search by modifying lines 11 and 20 in Listing 1.1.

3 Numerical Results

In this section, we present sample numerical results to illustrate the influence of the orbital-search probability p on the mean response time $E[T]$.

3.1 Model Validation

The validation of the proposed model can be done in several ways.

First, if $p = 1$ the model should behave similar to a finite-source FIFO queue. It can be seen in Fig. 3 that if $p = 1$, the Markov chain reduces to a birth-death

Table 1. Comparison of proposed model with [1].

K	λ	μ	ν	$E[T]_{[1]}$	$E[T]$
3	0.5	1	3	2.091	2.091
3	0.5	1	9	1.904	1.904
3	0.5	1	25	1.839	1.839
3	0.2	0.6	4	2.819	2.819
3	0.2	3	4	0.412	0.412
3	300	1	10	3.006	3.006

Table 2. Comparison of proposed model with [2].

K	λ	μ	ν	$E[T]_{[2]}$	$E[T]$
5	0.2	1	0.3	4.268	4.268
6	0.6	4	0.4	1.796	1.796
6	1.5	4	0.4	2.188	2.188
6	3.0	4	0.4	2.201	2.201
6	5	10	0.1	1.408	1.408
6	5	10	0.4	1.115	1.115
6	5	10	0.9	0.913	0.913
6	0.1	0.1	0.4	53.360	53.360
6	0.1	0.4	0.4	10.052	10.052
6	0.1	0.9	0.4	3.529	3.529

process with level-dependent birth rates. The same birth-death process arises in finite-source FIFO queues.

Secondly, if p is set close to zero, the MOSEL-2 results of the model we proposed can be compared to the results for the homogeneous case given in [1], where the search for customers in the orbit was not examined. The comparison is given in Table 1 and shows that the mean response time $E[T]$ of our proposed model perfectly matches the mean response time $E[T]_{[1]}$ of the model given in [1] when we restrict the investigations to the homogeneous case without the search for customers.

The MOSEL-2 results of our model can also be compared to the results given in [2]. Again, we choose $p \approx 0$. Furthermore, we assume the server to be reliable. The comparison is given in Table 2 and also shows a perfect match.

3.2 Further Results and Comments

Retrial-Queue Behavior vs. FIFO-Queue Behavior First, we discuss the results shown in Fig. 4. The parameters used for producing Fig. 4 are shown in Table 3. It can be seen that the mean response time of the retrial queue has a maximum. The location and the amplitude of the maximum depend on both parameters ν and p. Particularly, for lower values of the search probability p, the maximum becomes less dominant. Fig. 5 shows the behavior of the model for higher values of p and a wider range of λ. This behavior was already noticed in other papers dealing with finite-source retrial queues (e.g., [15, 1, 2]) but we are not aware of any thorough explanation for it.

Intuitively, the arising phenomenon can be discussed as follows. Fig. 4 seems to show a superposition of two effects. Let λ_{peek} be the generation rate inducing the maximum of the mean response time. For $\lambda < \lambda_{\text{peek}}$ the utilization of the server rises relatively fast (see Figure 6), and thus, the probability that a new primary request has to enter the orbit is also growing. The time spent by requests in the orbit leads to an increasing response time.

Table 3. Model parameters used to create Fig. 4.

Parameter	Symbol	Value/Range
Number of sources	K	3
Service rate	μ	1
Generation rate	λ	$0.0001 \ldots 1$
Retrial rate	ν	0.001, 0.005
Search probability	p	1E-8$\ldots$0.99

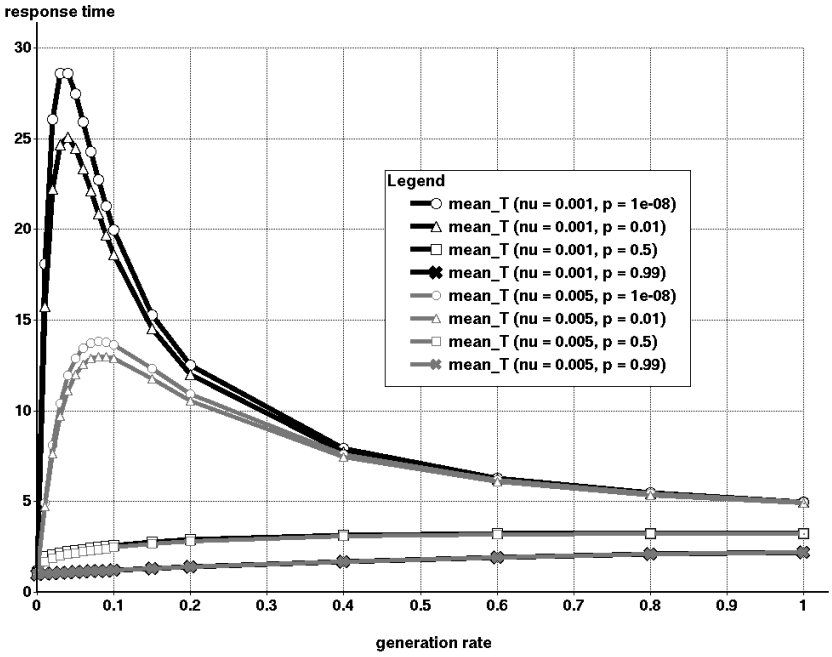

Fig. 4. MOSEL-2 results: Mean response time for $0 < p < 1$ and $0 < \lambda \leq 1$.

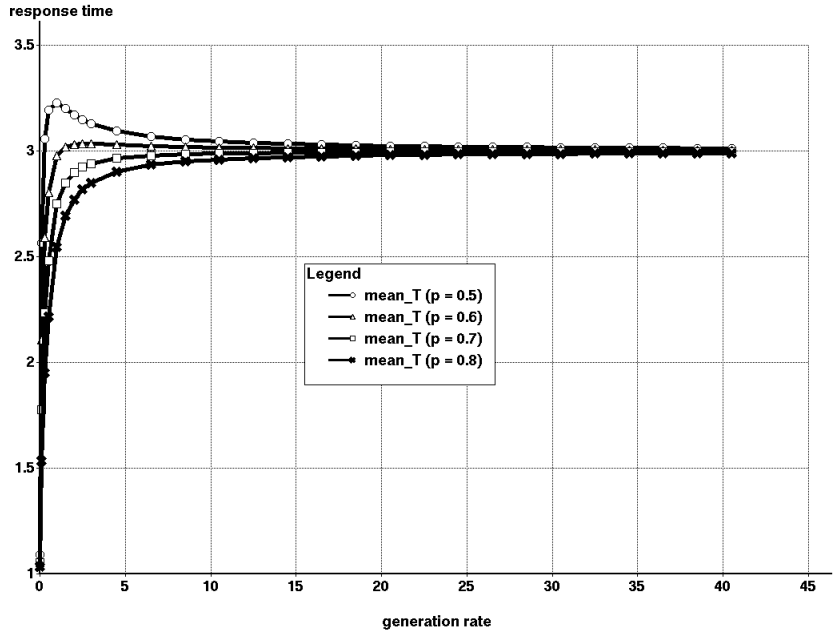

Fig. 5. MOSEL-2 results: Mean response time for $0.5 \leq p < 1$ and $0 < \lambda < 41$.

However, for $\lambda > \lambda_{\text{peek}}$, the mean arrival rate $\overline{\lambda}$ decreases because an increasing amount of requests is staying in the orbit, and thus, the mean number of active sources given by $(K - M)$ is decreasing (see Figure 7). However, each request returning to the sources will generate a new request faster with increasing λ. This behavior decreases the probability for the server being idle, and thus, the throughput rises. With the number of jobs M in the retrial queueing system getting closer to its maximum K and therefore getting more and more constant (see Figure 7), the rising throughput leads to a decreasing response time (Little's law, [7]).

For now, we can also state that the discussed phenomenon is becoming less dominant with increasing search probability p, and thus, with increasing FIFO-like behavior. This is again caused by the rising server utilization and rising throughput due to the more active search for customers from the orbit by the server that comes with increasing p.

Still, this discussion needs to be made more solid, and therefore, we are still working on finding close but manageable approximative closed-form formulas for the mean response time to be able to verify our assumptions and to analyze the phenomenon and its parameter dependencies in more detail by using curve sketching.

Disappearing Impact of Search Probability Figure 8 shows that the impact of the search probability p decreases with rising generation rate λ. Similarly,

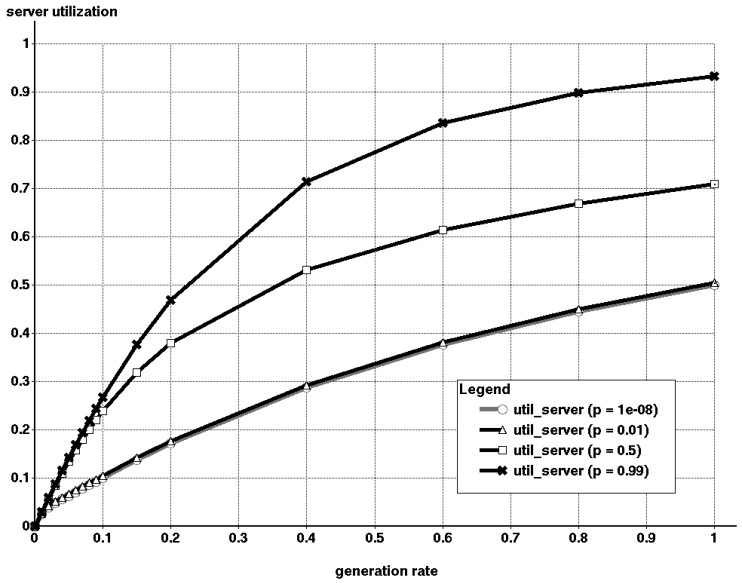

Fig. 6. MOSEL-2 results: Server utilization for $\nu = 0.001$.

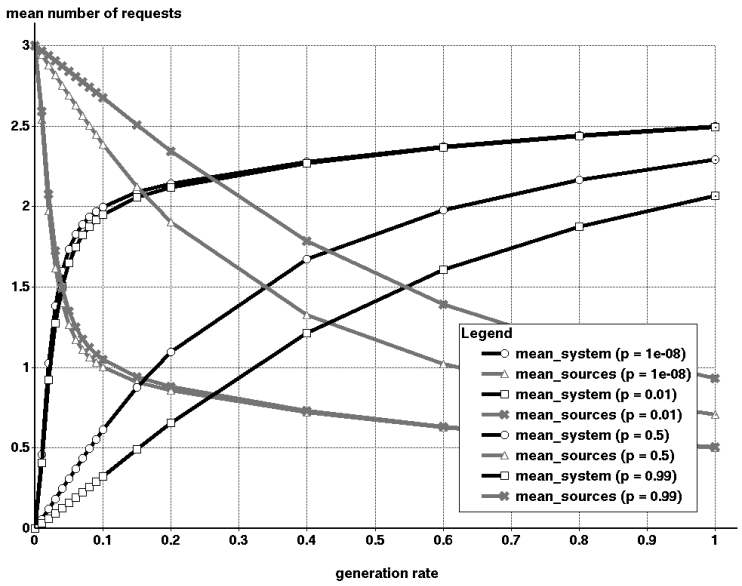

Fig. 7. MOSEL-2 results: Mean number of jobs staying in the orbit or service (M, black curves labeled "mean_system") and mean number of active sources ($K - M$, gray curves labeled "mean_sources") for $\nu = 0.001$.

for a rising retrial rate ν, the impact of the search probability p on the mean response time decreases as well. This is because with increasing λ or ν the server's utilization rises, and thus, the probability for the server being idle while requests are waiting in the orbit shrinks. Thus, with high values for λ or ν, the server is refilled quickly after service—even without the server actively searching for customers—by frequent generations of primary calls or aggressive retrials done by orbiting customers.

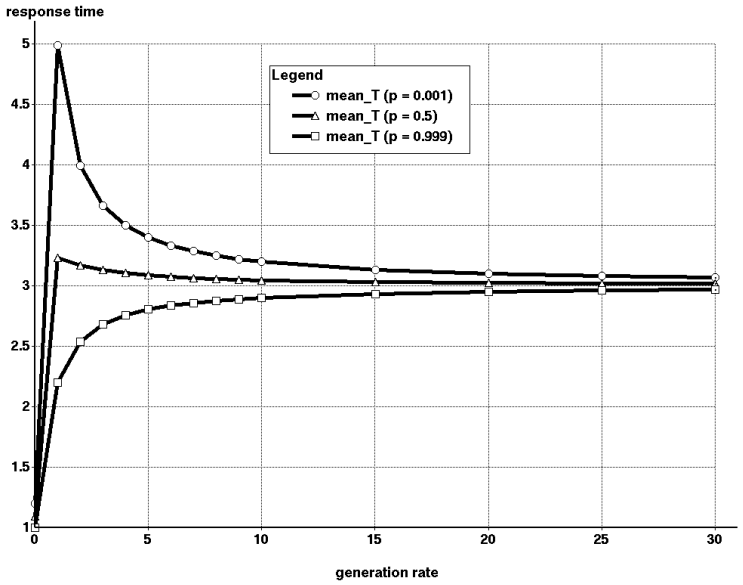

Fig. 8. MOSEL-2 results: Mean response time over generation rate λ.

Some Words on the Model Scalability Using MOSEL-2 running on a PC with Linux operating system, 2 GHz CPU, and 2 GB of RAM, the results of the 136 values needed to create Fig. 4 are calculated in only 14/8/6 seconds (real/user/system). However, in the single-server case with $K = 3$, the underlying Markov chain for each setting only has six states.

Still, from our previous experiences, we know that MOSEL-2 is capable of analyzing models with about one million states. However, due to restrictions of the underlying solvers, the maximum number of tokens in one place is approximately 65000. Considering both limitations, Table 4 shows some sample configurations of the proposed model that MOSEL-2 is capable of analyzing. The given configurations are sufficiently complex for a wide range of application areas and, thus, we are currently refraining from optimizing the evaluation method by exploiting the structure of the underlying Markov chain. In Table 4, the number of states

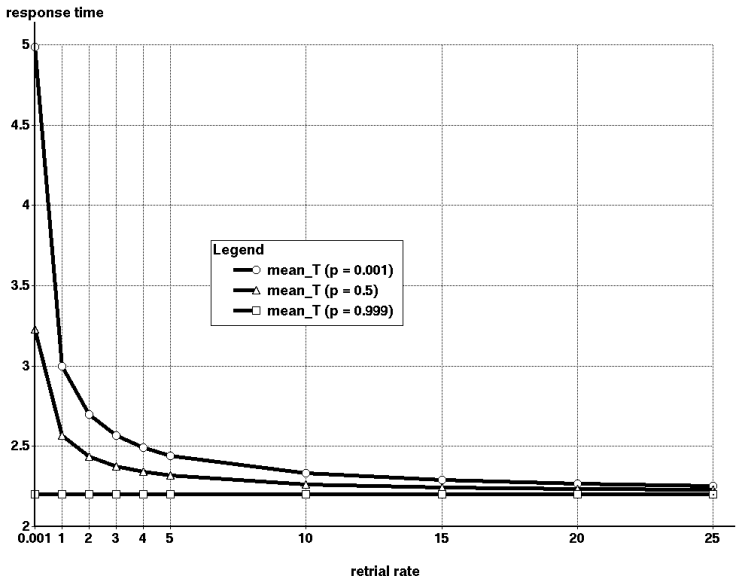

Fig. 9. MOSEL-2 results: Mean response time over retrial rate ν.

$|S|$ is calculated by

$$|S| = (K - c + 1)(c + 1)\,,$$

where K is the number of sources, and c is the number of servers. In the failed case, the underlying third-party solver aborts calculation unfortunately without giving a meaningful error message. The higher the number of states, the more time is spend for memory swapping. This, explains the growing discrepancy between the elapsed real time and user CPU time.

4 Conclusions

In this paper, a single server finite-source retrial queueing system with search for customers from the orbit is studied. The novelty of the investigation is the orbital search of the server in the context of finite-source retrial queues. Starting from an high-level stochastic Petri net, the underlying continuous-time Markov chain is derived. The tool MOSEL-2 is used to solve the system of steady-state equations automatically for a wide range of parameter sets and the main performance measures were derived and graphically displayed conveniently in this way. The effect of the server's search probability on the mean response times is shown and discussed.

As a next step, we plan to find (approximative) closed-form formulas for a closer investigation of the maximum of the mean response time arising in many retrial queues. For this, the exploitation of the special structure of the infinitesimal generator matrix seems to be promising. Our first approaches to an

Table 4. Model configurations MOSEL-2 can cope with.

#Sources	#Servers	#States	Eval. Time per Parameter Set		
K	c	$\|S\|$	real	user	system
3	1	6	1”	0”	1”
65000	1	130000	28”	26”	1”
65000	2	194997	54”	52”	2”
65000	5	389976	2’16”	2’5”	5”
65000	10	714901	9’11”	4’6”	17”
10000	25	259376	22”	18”	2”
10000	50	507501	1’25”	46”	5”
10000	75	754376	11’25”	1’23”	25”
10000	93	931352	23’19”	2’29”	49”
10000	94	941165		*failed*	
1000	500	251001	27”	20”	2”
10000	10000	10001	1”	1”	1”
65000	65000	65001	4”	3”	1”

approximative closed-form formulas were not successful because the approximations did not show the maximum any more. In the meantime, however, we found an approach to get closed-form results for the steady-state probabilities of the underlying Markov chain. From these closed-form results a closed-form formula of the mean response time can be derived. However, this is work in progress and thus, cannot yet be presented in all technical details.

Then, the investigation of (homogeneous and heterogeneous) multiple-server finite-source retrial queueing systems with search for customers from the orbit is projected. Using MOSEL-2, we can also easily extend the models to allow for phase-type distributed service, retrial, and search times. Furthermore, we plan to investigate open and closed networks of retrial queues. To cope with the arising state-space explosion, we again plan to exploit the structure of the infinitesimal generator matrix.

References

1. B. Almasi, G. Bolch, and J. Sztrik. Heterogeneous finite-source retrial queues. *Journal of Mathematical Sciences*, 121(5):2590–2596, jun 2004.
2. B. Almasi, J. Roszik, and J. Sztrik. Homogeneous finite-source retrial queues with server subject to breakdowns and repairs. *Mathematical and Computer Modelling*, 42:673–682, 2005.
3. J. Artalejo and A. Gomez-Corral. Information theoretic analysis for queueing systems with quasi-random input. *Mathematical and Computer Modelling*, 22:65–76, August 1995.
4. J. R. Artalejo. Retrial queues with a finite number of sources. *J. Korean Math. Soc.*, 35:503–525, 1998.
5. J. R. Artalejo, V. C. Joshua, and A. Krishnamoorthy. An M/G/1 retrial queue with orbital search by the server. In J. R. Artalejo and A. Krishnamoorthy, ed-

itors, *Advances in Stochastic Modelling*, pages 41–54. Notable Publications Inc., NJ, 2002.
6. K. Begain, G. Bolch, and H. Herold. *Practical Performance Modeling – Application of the MOSEL Language*. Kluwer Academic Publishers, 2001.
7. G. Bolch, S. Greiner, H. de Meer, and K. S. Trivedi. *Queueing Networks and Markov Chains*. John Wiley & Sons, New York, 2 edition, 2006.
8. S. R. Chakravarthy, A. Krishnamoorthy, and V. C. Joshua. Analysis of a multi-server retrial queue with search of customers from the orbit. *Perform. Eval.*, 63(8):776–798, 2006.
9. A. G. de Kok. Algorithmic methods for single server systems with repeated attempts. *Statistica Neerlandica*, 38:23–32, 1984.
10. V. I. Dragieva. Single-line queue with finite source and repeated calls. *Problems of Information Transmission*, 30(3):283–289, July–September 1994.
11. A. N. Dudin, A. Krishnamoorthy, V. C. Joshua, and G. V. Tsarenkov. Analysis of the BMAP/G/1 retrial system with search of customers from the orbit. *European Journal of Operational Research*, 157(1):169–179, 2004.
12. G. Falin and J. Templeton. *Retrial Queues*. Chapman & Hall, 1997.
13. G. I. Falin. A survey of retrial queues. *Queueing Syst. Theory Appl.*, 7(2):127–167, 1990.
14. G. I. Falin. A multiserver retrial queue with a finite number of sources of primary calls. *Mathematical and Computer Modelling*, 30:33–49, 1999.
15. G. I. Falin and J. R. Artalejo. A finite source retrial queue. *European Journal of Operational Research*, 108:409–424, 1998.
16. A. Gomez-Corral. Analysis of a single-server retrial queue with quasi-random input and nonpreemptive priority. *Computers and Mathematics with Applications*, 43:767–782(16), March 2002.
17. 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.
18. Y. N. Kornyshev. Design of a fully accessible switching system with repeated calls. *Telecommunications*, 23:46–52, 1969.
19. M. F. Neuts and M. F. Ramalhoto. A service model in which the server is required to search for customers. *Journal of Applied Probability*, 21(1):157–166, mar 1984.
20. 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(4):100–113, 1994.
21. H. Takagi. *Queueing Analysis: A Foundation of Performance Evaluation*, volume 2, Finite Systems. North-Holland, 1993.
22. P. Wuechner, H. De Meer, J. Barner, and G. Bolch. A brief introduction to MOSEL-2. In R. German and A. Heindl, editors, *Proc. of MMB 2006 Conference*, pages 473–476. GI/ITG/MMB, University of Erlangen, VDE Verlag, 2006.