The Impact of Retrials on the Performance of Self-Organizing Systems‡

P. Wüchner[1], J. Sztrik[2], H. de Meer[1]

[1] Chair of Computer Networks and Communications,
University of Passau, Innstr. 43, 94023 Passau, Germany
[2] Department of Informatics Systems and Networks, Faculty of Informatics,
University of Debrecen, Debrecen, P.O. Box 10, 4010, Hungary

ABSTRACT

This article describes the application of the theory of retrial queues in capturing certain aspects of self-organizing behavior that arises, for example, in Peer-to-Peer networks. It can be shown that retrials have a fair effect on the performance of such self-organizing systems, and thus, should be taken into account adequately during the design and evaluation of these systems. Moreover, it can be shown that there is a notable difference between finite-source and infinite-source retrial queueing models. The main goal of this paper is to show the practical applicability of retrial queues and some of their varieties.

1 INTRODUCTION

Consider a service facility and customers that try to get service from the server. Typically, it is assumed that customers line up in a queue if the server is busy and stay in this queue until they get served. However, particularly in telecommunication systems, there are scenarios in which the customers leave the service area if the server is busy and return after some while to *retry* to get serviced.

In this article, we focus on the phenomenon of such retrials and their influence on non-functional properties – especially on the performance – of any system, particularly complex and self-organizing systems. For the modeling and evaluation of retrials, *retrial queues* have been introduced by queueing theorists (see, e.g., [1, 2, 3]). Moreover, there are several extension to and varieties of retrial queues that make them a very flexible tool for the modeling and evaluation of various systems that comprise some sort of retrials.

Before presenting the retrial queueing model in more detail, we show its applicability to model and evaluate the behavior of self-organizing systems. As an example, we discuss the performance of a Peer-to-Peer (P2P) file-sharing system. We choose a P2P file-sharing scenario, because P2P traffic still plays a very important role in today's Internet. This statement was confirmed by recent traffic measurements (see, e.g., [4]) and it also can be observed that the impact of P2P traffic still pushes forward the development of deep-packet-inspection technologies and traffic shapers suitable for recognizing and controlling such delicate traffic. However, P2P can be used not only for file sharing, but due to the scalability and decentralization, the concept of P2P can be exploited for a far wider range of application areas, e.g., for providing anonymity overlays [5], load balancing [6], and distributed storage [7]. Thus, it remains important to understand and possibly exploit the self-organizing behavior of P2P systems. In fact, P2P systems are widely accepted to comprise self-organizing behavior (see, e.g., [8, 9]). Moreover, approaches, too numerous to count, have been proposed to artificially introduce even more self-organizing and bio-inspired concepts into P2P technology for improving the resource lookup, routing, and resilience. Thus, it seems reasonable to investigate the behavior of such complex systems with respect to their both unintended and purposefully introduced self-organizing behavior. To this end, the construction of abstract models and the evaluation of these models make a fair contribution to the understanding of the system behavior.

The abstract mathematical models presented in this article focus on the modeling and evaluation of systems that comprise retrying behavior. It is shown that retrials may have a considerable effect on the performance of such systems, and thus, system modelers, developers, and engineers should be aware of this behavior and of its negative impact on system performance.

For those readers familiar with retrial queueing theory, we can already state more precisely: we are looking at retrial queues with finite population size. Additionally, we extend the model to allow for the direct extraction of jobs from the orbit by the server after service periods with some probability p. This model can then be used conveniently to study the performance differences between classical FIFO and retrial queues. Such models are also known as *"retrial queueing models with search for customers in the orbit"* [10, 11], but they have not been

‡ This work 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, and by the Network of Excellence EuroFGI – IST 028022.

studied before in the finite-population context. Moreover, this article focuses on the practical applicability of this theory and not on the mathematical concepts the theory is based upon. In particular, for the model evaluation we rely on the convenient performance evaluation tool MOSEL-2 [12].

The remainder of this article is organized as follows. After motivating the occurrence of retrials in a self-organizing P2P system in Section 2, a more theoretical background on retrials and retrial queues is introduced in Section 3 that can be skipped by readers already familiar with retrial queues. In Section 4, a novel system model based on stochastic Petri nets is presented. This model is used in Section 5 to discuss different setups and varieties of retrial queues and their influence on the numerical evaluation results. Finally, a conclusion is presented in Section 6.

2 RETRIALS ARISING IN EDONKEY/EMULE NETWORKS

Intuitively, it can be observed that scenarios in which information polling is more dominant than information pushing seem to be more prone to the effects of retrials. Due to the locality of interactions and the simplicity of local rules, information polling – and thus, the occurrence of retrials – is also common in many self-organizing systems. To present a practical example, in this paper, we discuss retrials in the eDonkey network (ed2k, see [13]), which is currently one of the most popular P2P file-sharing protocols that is capable of organizing multi-source downloads. Moreover, several open-source client implementations, like the very popular eMule client, are available for the eDonkey network.

Retrials can be observed in eDonkey/eMule networks at least in two scenarios. Let us call the first class of retrials "*retrials after service"*. These retrials are caused by either data pollution (see [14]) or by CRC checksum failures. Data pollution occurs when decoy files with false or corrupted content are fed into the network by malicious users. CRC checksum failures are induced by network errors. In both cases, the downloader has to retry to get the data by issuing another download.

The second class of retrials we refer to as "*retrials before service"*. These retrials are induced by the eDonkey/eMule protocol. Here, we concentrate on these *retrials before service* and give now a more detailed explanation for their occurrence. Our assumptions are based on both the eMule protocol specification presented in [15] and the current eMule source code (version 0.48a) available at `SourceForge.net`. However, due to space limitations, we will refrain from presenting details that are only of secondary interest to our investigation.

Let us call peers that offer (parts of) files to the network *source peers*. Accordingly, we call remote peers that are interested in these files *requesting peers*. The upload capacity of source peers is limited. Hence, each source peer maintains a waiting list of requesting peers. The rank of each requesting peer within the waiting list is calculated by several factors including the waiting time and a credit system. As soon as a new upload slot is available, the first requesting peer in the waiting list will be informed by the source peer and the upload will be started. However, the requesting peer cannot be informed directly by the source peer if the requesting peer is located behind a firewall or NAT gateway. We will now concentrate on such firewalled requesting peers.

In contrast to the eDonkey protocol, eMule tries to avoid callback requests in such cases. Callback requests are sent by the source peer to the firewalled requesting peer via an eDonkey/ed2k super peer (server). Instead, if no other ways of contacting the firewalled requesting peer exist (e.g., via the Kademlia network), the source peer has to wait for a new request (retrial) by the firewalled requesting peer. Thus, in this case, the service is no longer processed in a FIFO manner, but can be more adequately described by a retrial queue.

3 RETRIAL QUEUES

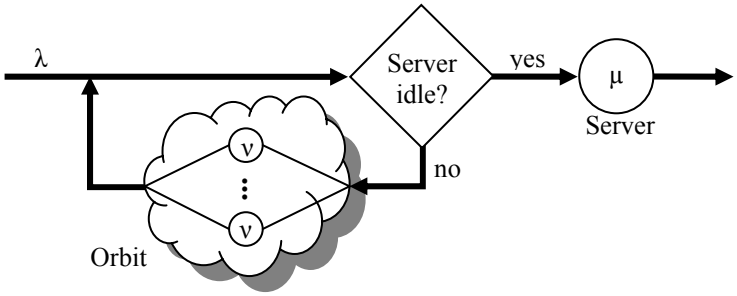

Fig. 1 Basic retrial queue.

Figure 1 shows a basic retrial queueing model in which *jobs* (workload, customers, service requests, or tokens) arrive at the retrial queue with exponentially distributed inter-arrival times (i.e., this constitutes a Poisson arrival process) with parameter λ (jobs per time unit) which is also known as the *arrival rate*. Note that in the model all times are measured in abstract time units. In this paper, to keep it concise, we restrict ourselves to the investigation of exponentially distributed inter-event times. In principle, however, the investigation can be quite easily extended to phase-type distributions or, more intricately, to even more general stochastic processes (see, e.g., [16]).

If an arriving job finds the *server* idle, the job enters the service and is processed with a service rate of μ. After being serviced, the job leaves the retrial queue. On the other hand, if an arriving job finds the server busy, the job is not enqueued into a first-in first-out (FIFO) waiting queue, as it is done in classical queueing systems, but it enters a so-called *orbit* instead. All jobs that are currently staying in the orbit are retrying (hence the name) to get service by the server. The inter-retrial times are again assumed to be exponentially distributed with rate ν.

The basic model shown in Figure 1 describes an open retrial queueing system. Here, the number of jobs in the orbit might be arbitrarily large and the arrival rate does not depend on the number of jobs that are already located within the retrial queue. Such queueing models are also called “infinite-source queueing models” or “queueing models with infinite population size” and, although having an infinite state space, can be evaluated quite easily by exploiting the structure of the underlying Markov process. For details, the interested reader is referred to [3] and the references therein.

However, as it turns out, infinite queueing systems are not always a good model of finite real systems. Later in this article, the difference between infinite-source and finite-source (see, e.g., [17]) models is made visible by results of numerical analysis.

4 SYSTEM MODEL

We first restrict or view to a finite-source model in which the number of jobs that potentially arrive at the retrial queue (also known as the population size) is finite. In the current context of the given eMule scenario, the main motivation for a finite-source model is based on the fact that the number of firewalled peers, requesting a file from the same source peer is rather limited.

In addition to the limitation of the population size, we extend the model in such a way that the server extracts jobs from the orbit after service periods with some probability p. This model with *“orbital search”* can then be used to study the performance differences between classical FIFO and retrial queues. In this section, the modified model is presented more formally and in more detail by employing a stochastic Petri net model.

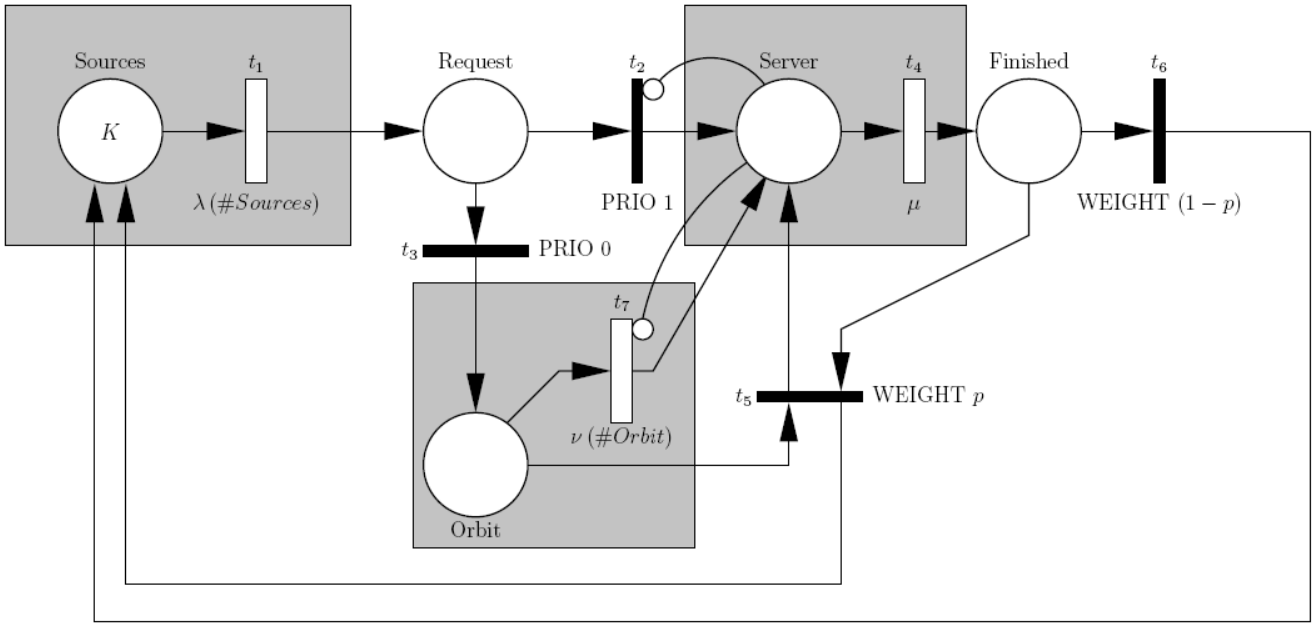

Fig. 2 Stochastic Petri net of retrial queue with search for jobs in orbit.

A *stochastic Petri net* (more precisely: *stochastic reward net*; see [3]) that models the sketched behavior is shown in Fig. 2. The model can be described as follows. Initially, there are K tokens in place *Sources*. Each of these active sources models one requesting peer and is generating a primary service request with the rate of λ via the timed transition t_1. If the *Server* is idle, the generated service request immediately enters the server via immediate transition t_2 with high priority. Only in the case of an already busy server, a primary request instead enters the *Orbit* using immediate transition t_3 with low priority. Thus, only one token might occupy the server at the same time and the maximum number of tokens in the orbit is *(K-1)*. The job in service is processed during an exponentially distributed service time with rate μ modeled by the timed transition t_4. As soon as a job finishes its service, it returns to the sources immediately. With a probability of p, one job is transferred from the orbit directly to the server at the same time via t_5. With a probability of *(1-p)*, however, no job gets removed from the orbit and only the recently processed job returns to the sources immediately using transition t_6. In parallel, all jobs in the orbit are retrying to get service from the server with a rate of ν modeled by t_7.

For the steady-state evaluation of this model, the stochastic Petri net given in Figure 2 can be transferred into a continuous-time Markov chain (see [3]) with finite state space. This Markov chain has $S_{Server} \cdot S_{Orbit} = 2 \cdot (K-1+1) = 2K$ states, where $S_{Server} = 2$ is the number of states of the server (i.e., idle or busy) and S_{Orbit} is the number of states of the orbit (i.e., *0 ... K-1* jobs). Using numerical analysis, the steady state probabilities can then be obtained by the help of which all other performance measures of interest can be calculated.

5 NUMERICAL RESULTS

For increasing the modeler's convenience, the generation of the underlying Markov chain and the solution of the system of steady-state global-balance equations can be automated. Fortunately, several software tools are available that allow to derive the steady-state (or transient) performance measures directly from a high-level (e.g., Petri net) model. Here, we use the tool MOSEL-2 (see [12] for details). The results we present here aim in two directions. First, in Scenario 1, we show that the parameter p indeed has a notable influence on the performance of the system. That is, the abstraction from retrials by using classical FIFO queues has a considerable impact on the numerical results of the investigation. Secondly, in Scenario 2, we discuss the difference between finite-source and infinite-source retrial queues and show that infinite-source models are unsuitable for the modeling of systems with a finite number of customers. The parameter sets of both scenarios are summarized in Table 1.

Table 1 Model parameters of Scenario 1 and Scenario 2.

Parameter	Symbol	Scenario 1	Scenario 2
Number of sources	K	3	1 … 50,000
Request generation rate	λ	10 … 100	10^{-8} … 10
Retrial rate	ν	10	10
Service rate	μ	1	1
Search probability	p	0.001 … 0.999	0.5

5.1 SCENARIO 1

The first scenario examines the impact of the search probability p. If p is close to one, the model mimics a classical FIFO queue with finite population size. If p is close to zero, the model is a classical finite-source retrial queue. Figure 3 shows the mean response time (in abstract time units) of the model depending on the parameters λ (arrival rate) and p (probability of orbital search). It can be seen that the parameter p does not only have an influence on the values of the mean response time but also on the shape of the graph. The maximum arising in the mean response time of finite-source retrial queues has been noticed by other researchers already (see, e.g., [17, 18]). However, to the best of our knowledge, no thorough explanation has been given for it, yet. We are currently working on this issue. Our approach is based on quasi-birth-death processes (see [3]) that allow finding an explicit multi-variate closed-form solution for the mean response time. However, this mathematical discussion of the maximum is not the main focus of this paper. Therefore, and due to the mathematical complexity of the discussion and space limitations, we refrain from presenting the details of this approach here.

Our numerical investigation shows that there is a notable difference between the performance of classical FIFO and retrial queueing systems. Thus, retrials have to be addressed when evaluating real systems they appear in. Approximating retrial queues by classical FIFO queues is illegitimate in a wide range of application scenarios.

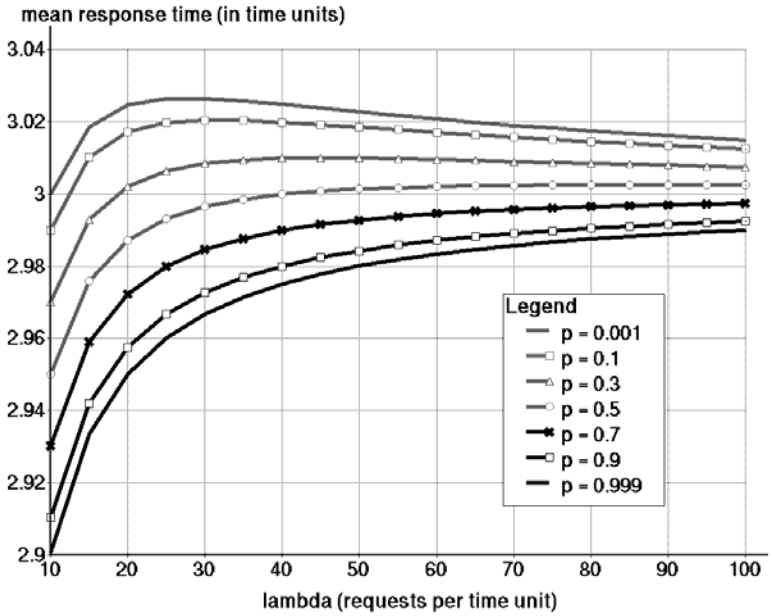

Fig. 3 Mean response time in Scenario 1.

5.2 SCENARIO 2

The second scenario is used to compare finite-source (closed) to infinite-source (open) models. In [19, 20] it is shown that using the arrival rate λK in infinite-source queues is a good approximation of finite-source queues with K sources and arrival rate λ in cases where λ is close to zero. Our results show the difference between the suggested approximation λK and the effective lambda $\lambda^* = \lambda \cdot \bar{k}_{Sources}$, where $\bar{k}_{Sources}$ is the mean number of active sources.

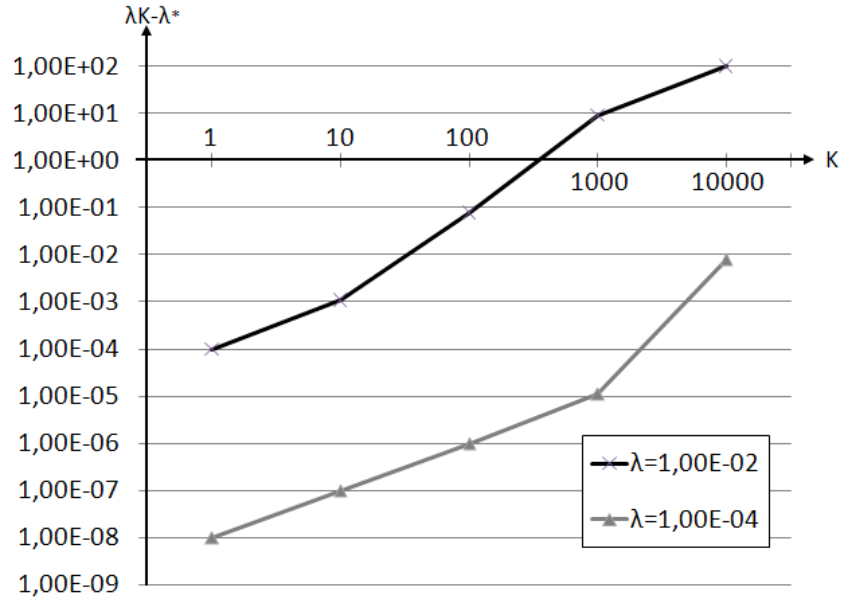

Fig. 4 Difference between λ^* and λK for different values of λ and K.

For our comparison, we first evaluate the effective arrival rate λ^* of new primary requests for several parameter settings in the finite-source model. In Figure 4, for very small values of K and λ, the difference between λ^* and λK is very small. This is because in both the finite-source and infinite-source case, the orbit and the server are mainly idle. Thus, in the finite-source case, the probability that all K jobs are at place *Sources* is very high and, thus, λ^* is close to λK. However, as the server's utilization rises, the number of sources generating requests is decreasing, because more and more jobs will be in the orbit. Thus, if the orbit is used frequently, the approximation λK is not useful anymore, and the effective arrival rate λ^* should be used instead. Note, however, that the values for λ^* would only be available if the model was already evaluated in a finite-source context.

Finally, the values for the mean response time of the retrial queue with search for customers in the orbit is evaluated in the finite-source setting with arrival rate λ (per active source) and in the infinite-source setting with overall arrival rate λ^*. To preserve the graph's readability, the number of sources in the finite-source setting is now kept constant at $K=100$. The corresponding results are shown in Figure 5. It can be seen that for low arrival rates, and thus, low utilizations of the server, the results match. However, for higher arrival rates, i.e., when the server's utilization is approaching 1, even the usage of the effective arrival rate λ^* for approximating a finite-source model by an infinite-source model gives inadequate results.

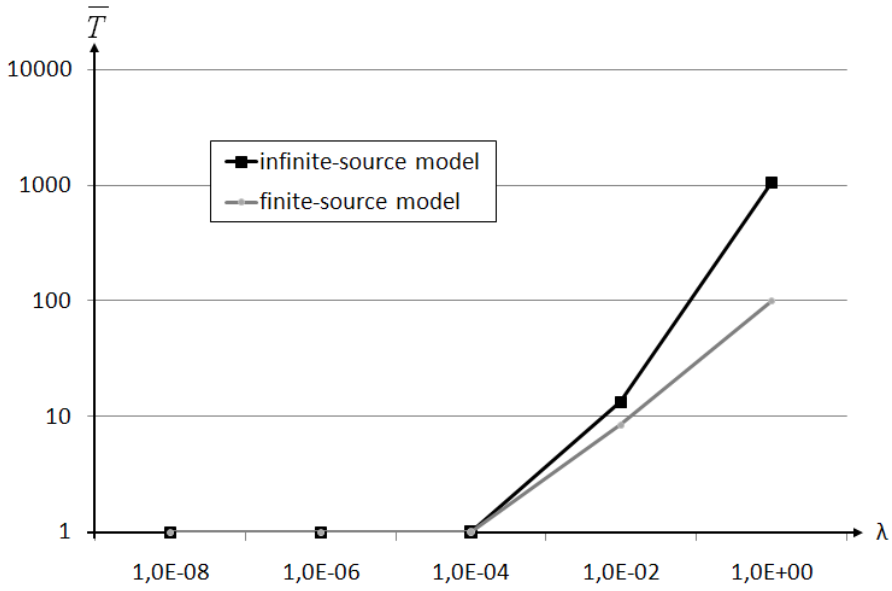

Fig. 5 Mean response time for finite-source and infinite-source model.

6. CONCLUSION

Retrial queues are capable of modeling certain aspects of the behavior of self-organizing systems. The retrials issued by firewalled peers in self-organizing P2P networks are not the only example where retrial queues can and should be applied. This paper deals with two modifications of the classical retrial queueing model that have a considerable effect on the behavior and performance measures of retrial queueing systems. The first modification is the introduction of the search for customers by the server after service periods; the second one is the limitation of the number of sources.

A paper including a more mathematical approach and further results for finite-source retrial queues with orbital search is currently in preparation. Also the thorough mathematical discussion of the maximum of the mean response time in finite-source retrial queues is one of our short-term goals. A medium-term goal is to combine several retrial queues into a network of queues. To preserve the scalability of the evaluation, the structure of the underlying Markov chain will be investigated and possibly exploited.

REFERENCES

1 G. I. Falin and J. Templeton, "Retrial Queues", Chapman & Hall, 1997.

2 G. I. Falin, "A survey of retrial queues", Queueing Syst. Theory Appl., 7(2):127-167, 1990.

3 G. Bolch, S. Greiner, H. de Meer, and K. S. Trivedi, "Queueing Networks and Markov Chains", 2nd Edition, John Wiley & Sons, March 2006.

4 Cisco Systems Inc., "Global IP Traffic Forecast and Methodology, 2006-2011", White Paper, August 2007. Online Available: http://www.cisco.com/application/pdf/en/us/guest/netsol/ns537/c654/cdccont_0900aecd806a81aa.pdf (last visited: 11/2007).

5 O. Landsiedel, A. Pimenidis, K. Wehrle, H. Niedermayer, G. Carle, "Dynamic Multipath Onion Routing in Anonymous Peer-To-Peer Overlay Networks", IEEE Global Communication Conference (GlobeCom), Washington D.C., Nov. 2007.

6 S. Surana, B. Godfrey, K. Lakshminarayanan, R. Karp, and I. Stoica, "Load balancing in dynamic structured peer-to-peer systems", Perform. Eval., 63(3):217-240, Mar. 2006.

7 C. Blake and R. Rodrigues, "High availability, scalable storage, dynamic peer networks: pick two", Proceedings of the 9th Conference on Hot Topics in Operating Systems - Volume 9, Lihue, Hawaii. USENIX Association, Berkeley, CA, May 2003.

8 H. de Meer and C. Koppen, "Self-Organization in Peer-to-Peer Systems", in Peer-to-Peer Systems and Applications, R. Steinmetz, K. Wehrle (Eds.), Lecture Notes in Computer Science LNCS 3485, Part V, pp. 247-266, Springer Heidelberg, 2005.

9 J. Ledlie, J. Taylor, L. Serban, and M. Seltzer, "Self-Organization in Peer-to-Peer Systems", in 10th EW SIGOPS, Sep. 2002.

10 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, March 1984.

11 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.

12 P. Wüchner, H. de Meer, J. Barner, and G. Bolch, "A Brief Introduction to MOSEL-2", Proc. of MMB 2006 Conference, Nuremberg, Germany, March 27-29, pp. 473-476, VDE-Verlag, 2006, Mar. 2006.

13 S. B. Handurukande, A. Kermarrec, F. Le Fessant, L. Massoulié, and S. Patarin, "Peer sharing behaviour in the eDonkey network, and implications for the design of server-less file sharing systems", SIGOPS Oper. Syst. Rev., 40(4):359-371, Oct. 2006

14 U. Lee, M. Choi, J. Cho, M. Y. Sanadidi, M. Gerla, and S. R. Maeng, "Understanding Pollution Dynamics in P2P File Sharing," The 5th International Workshop on Peer-to-Peer Systems (IPTPS'06), Santa Babara, USA, Feb. 2006.

15 Y. Kulbak and D. Bickson, "The eMule Protocol Specification", Technical report, TR-2005-03, the Hebrew University of Jerusalem, Jan. 2005.

16 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.

17 G. I. Falin and J. R. Artalejo. "A finite source retrial queue", European Journal of Operational Research, 108:409-424, 1998.

18 B. Almasi, G. Bolch, and J. Sztrik. "Heterogeneous finite-source retrial queues", Journal of Mathematical Sciences, 121(5):2590-2596, Jun. 2004.

19 D. Gross and C. M. Harris, "Fundamentals of Queueing Theory", 2nd ed., John Wiley & Sons, New York, 1985.

20 N. K. Jaiswal, "Priority Queues", Academic Press, New York, 1968.