Proceedings of the 2009 Winter Simulation Conference
M. D. Rossetti, R. R. Hill, B. Johansson, A. Dunkin, and R. G. Ingalls, eds.

A Petri Net model for Service Availability in Redundant Computing Systems

Felix Salfner

Department of Computer Science
Humboldt-Universität zu Berlin
Unter den Linden 6, 10099 Berlin, Germany

Katinka Wolter

Institute of Computer Science
Freie Universität Berlin
Takustr. 9, 14195 Berlin, Germany

ABSTRACT

In this paper we present and analyse a coloured stochastic Petri net model of a redundant fault-tolerant system. As our measure of interest we are interested in a dependability metric, i.e., service availability. Service availability is defined as the number of successfully completed jobs relative to the total number of arrived jobs. This paper is the first step towards a comprehensive comparison of redundancy and rejuvenation, i.e., the preventive restart of servers when studying service availability. The question we strive to answer in this paper is whether and to what degree additional redundant servers can increase service availability in all load scenarios. We find that the first redundant server improves service availability by almost 90% in a highly loaded system, while adding a second and third redundant server yields further but much lower improvement. Under low load the benefit of additional servers is not as pronounced.

1 INTRODUCTION

Traditionally, availability assessment has been concerned with system availability, i.e., with failure and repair of computing systems. Steady-state system availability is formally defined as the ratio of mean time to failure (MTTF) and the total time, i.e. the sum of MTTF and mean time to repair (MTTR). Methods for improving availability are typically based on redundancy in space or time. The most popular methods include reactive methods such as checkpointing (Elnozahy, Alvisi, Wang, and Johnson 2002) as well as the use of spare components (Sun, Han, and Levendel 2001), and proactive methods such as preventive maintenance (Garg, Puliafito, Telek, and Trivedi 1998) and in particular software rejuvenation (Huang, Kintala, Kolettis, and Fulton 1995), i.e., a restart even though the system has not yet failed. The optimal choice of parameters in reactive as well as proactive methods is often determined through the analysis of stochastic models. An overview is given in (Trivedi, Ciardo, Dasarathy, Grottke, Rindos, and Varshaw 2008) and the references therein.

With the shift in paradigm from a system's view to a services view, system availability has become less relevant. The attention now focuses on the availability of a service rather than the availability of the system hosting it. While in earlier work (Salfner and Wolter 2008b) we investigated improvements of service availability through failure prevention and software rejuvenation (Salfner and Wolter 2008a, Salfner and Wolter 2009), this paper focuses on the use of additional servers. Additional servers improve service availability by reducing the load on the entire system and by reducing the probability that the overall system is down. Ultimately, we want to compare whether the addition of a redundant component or rejuvenation is more effective to increase service availability. Our presumption is that the addition of the first redundant server outperforms rejuvenation but as more redundant servers are added the relative improvement becomes less. Rejuvenation, on the other hand, yields a constant improvement in service availability. We want to come to guidelines when it is advisable to use either software rejuvenation or to add redundancy. In this paper we take the first step in the necessary analysis. We will only analyse the model with added redundancy and leave the investigation of combined rejuvenation and redundancy as future work.

We determine service availability by modelling the system as a simple queueing system processing jobs/requests using a stochastic coloured Petri net (Knoke and Zimmermann 2006). More specifically, we model environments with a completely reliable queue and servers that are subject to single points of failure, i.e., the model incorporates failure and repair. Similar models have been investigated in performability analysis, using queueing systems with server failure (Goseva-Popstojanova

and Trivedi 2000), pointing to the impact of the considered metric. Queues with server vacation (Haverkort 1998) use a similar model than we do, but are typically not used to determine service availability. The purpose of this paper is to investigate potential improvement in service availability through the use of secondary servers. We will see that the benefit of using secondary servers depends on the utilisation of the system. Service availability in a highly loaded system can be improved greatly by adding redundancy while for low load the improvement is much less.

Modelling service availability rather than system availability is not very common yet in reliability modelling. In (Wang and Trivedi 2005), the authors present a formalism to compute user-perceived service availability including user behaviour on the basis of stochastic reward nets (SRN) with the limitation that exponential distributions have to be used to model system behaviour such as failures. Our model is on one hand more realistic as it uses distributions that appear in real systems, e.g., the lognormal distribution matches the failure process better than the exponential distribution, on the other hand it includes less modelling detail.

The paper is organised as follows: We present our model for service availability in Section 2. Section 3 shows results from experiments investigating the effect of server replication under various load situations. Section 4 concludes the paper and provides an outlook to forthcoming investigations.

2 THE MODEL

We consider a system that processes long running scientific computations and that is subject to failure and repair. Our measure of interest is service availability, i.e., the number of successfully processed jobs relative to the total number of submitted jobs. Our system consists of a powerful primary server which can be augmented by up to three slower secondary servers to improve service availability.

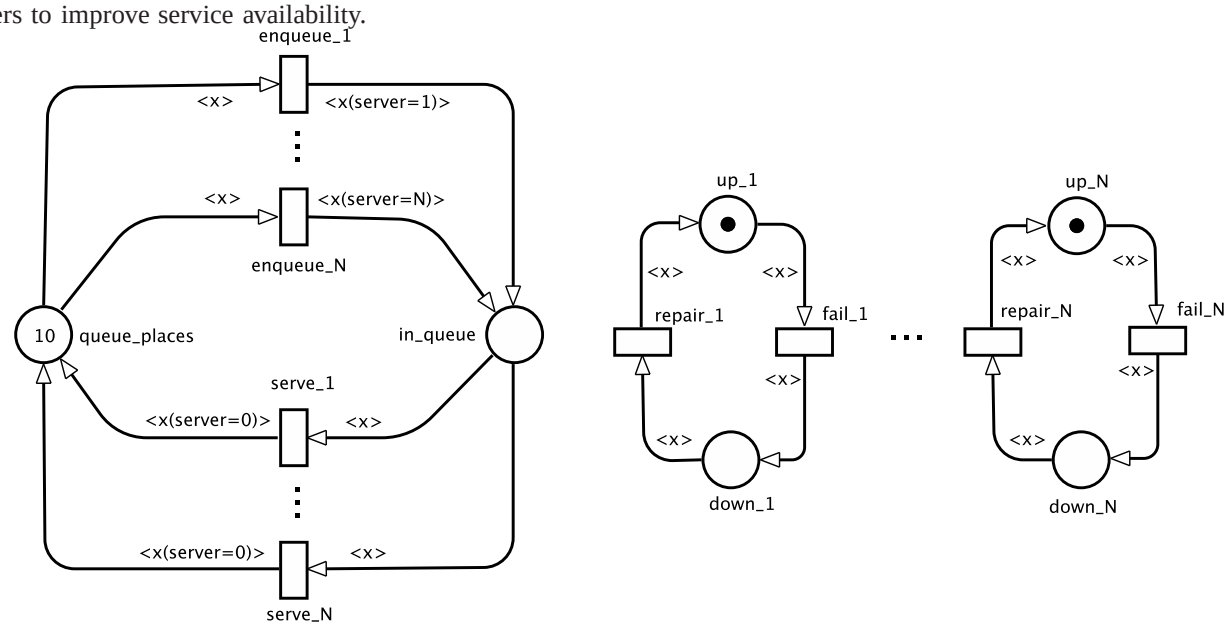

Figure 1: Model for N servers subject to failure and repair.

We model the considered system as a stochastic coloured Petri net shown in Figure 1. The model consists of two separate parts, a performance model, represented by a queueing system on the left and a dependability model, represented on the right. For analysis purposes the queue has finite capacity. We assume that jobs arrive to the system from outside. If there is a free place in the queue left and at least one server is operational the job enters the queue (transition *enqueue_i*). The enqueueing transition assigns a job/token to the respective server by adding its name as an attribute to the token that enters place *in_queue*. Transition *serve_i* selects a token that is assigned to server *i* and resets the attribute (x(server=0)) upon completion. Both, arrival and service transitions are only enabled as long as the corresponding server is operational.

The queue is considered to be absolutely reliable, i.e., no jobs are lost under any circumstances. However, if the queue is full or there is no server available (operational) the job is lost. If a server fails the jobs assigned to this server remain in the queue but cannot be processed until the server has been repaired. Therefore, corresponding tokens stay in place *in_queue* until the server is repaired.

The dependability model consists of a separate failure and repair circle for each server. The dependability model is an extended deterministic Petri net using general distributions for the firing times of both failures and repair. All servers have the same failure and repair characteristics. Both parts of the model are interconnected by global guard expressions. Table 1 lists the transitions, their parameters and the guard expressions.

Table 1: Specifications for transitions in Figure 1.

Name	Delay	Global Guard	Local Guard
enqueue_1	EXP((1+$\sum_{j=2}^{N}$ #up_j)*arrival_time)	#up_1>0	–
enqueue_2	EXP((1+$\sum_{j=1}^{N-1}$#up_j)*arrival_time)	#up_N>0	–
serve_1	EXP(service_time)	#up_1>0	x.server==1
serve_N	EXP(service_time*1.3^{N-1})	#up_N>0	x.server==N
repair_1	Uni(0.5,48.0)	–	–
fail_1	LogNorm(8.37,0.4)	–	–
repair_N	Uni(0.5,48.0)	–	–
fail_N	LogNorm(8.37,0.4)	–	–

Arrivals to all servers follow a Poisson process with mean time between arrivals defined by the variable *arrival_time*. However, since we are investigating the effect of adding extra servers without changing overall system requirements, the external arrival rate remains constant. From this follows that transition times for each individual *enqueue_i* transition are adapted such that the sum of all enqueue transitions equals *arrival_time*. More specifically, this property holds at every point in time, even for times when some servers are not operational.

3 EXPERIMENTS AND RESULTS

As we want to investigate the effects of different utilisation of the queue on overall service availability, we adjust the mean time between arrivals accordingly. We have assessed service availability by measuring the effective system service rate, i.e. the rate at which the system has served jobs including all periods of downtime, empty queues, and periods where less than the maximum number of servers are present. This has been achieved by keeping track of the number of firings of each of the *serve_i* transitions. Service availability is then computed as

$$A_s = \frac{\text{effective service rate}}{\text{arrival rate to the system}} = \text{effective service rate} * \mathit{arrival\_time} \tag{1}$$

Since the performance model is a coloured Petri net and the dependability model has non-exponential distributions[1] being linked via guard expressions we had to simulate the Petri net.

Table 2: Parameters and range of values varied in experiments.

Name	Range of values
arrival_time	$15h,\ldots,60h$
service_time	$15h$
number of servers N	$1,\ldots,4$

The two parameters varied in the experiments are the number of servers N and *arrival_time*, resulting in $\rho = \mathit{service\_time}/\mathit{arrival\_time} = 0.25,\ldots,1$. Service times are exponentially distributed with mean service time defined by the variable *service_time*. The mean service time is set to 15 hours for the first server. The $N-1$-th redundant server is slower than the primary one by a factor of 1.3^{N-1}.

[1] In forthcoming models, there will be more than one outgoing non-exponential transitions (see Section 4)

We assume a server fails on average after 4675 hours (approx. 195 days) of operation and the time to failure is lognormally distributed. This failure characteristic has been observed in a large real-world telecommunication system (Salfner 2008). The repair time is uniformly distributed and takes between half an hour (time for a simple restart) and two days.

Table 2 summarises the parameters that are varied in experiments together with their range of values.

We have simulated the stochastic Petri net in Figure 1 using the stochastic coloured Petri net (SCPN) module of the software tool TimeNET (Zimmermann, Freiheit, German, and Hommel 2000, Knoke and Zimmermann 2006).

We have run the simulation for 27 years (model time), as the SCPN TimeNet module used unfortunately does not provide a confidence interval as stopping criterion. Hence, reasonable quality in the results can only be achieved by arbitrary, but very long simulations.

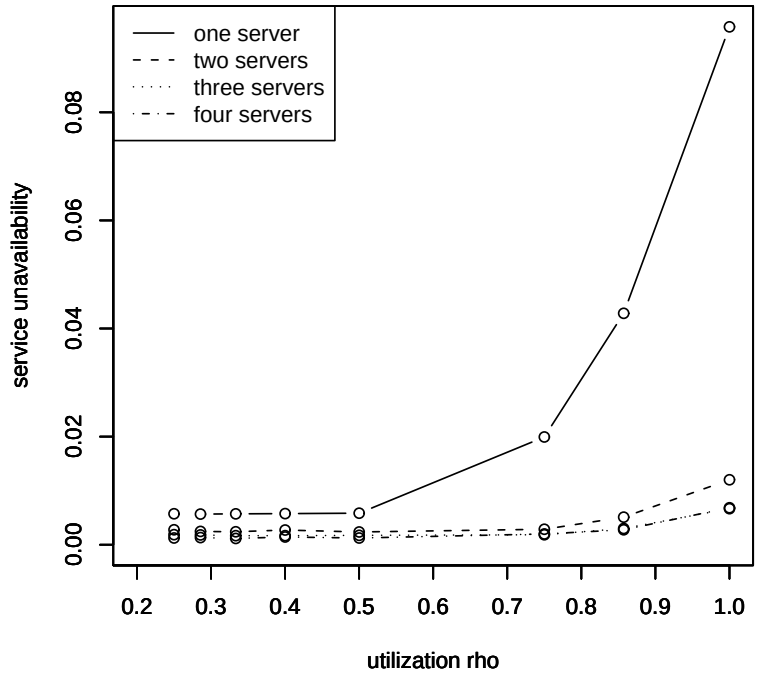

Figure 2: Utilisation versus service unavailability for different number of servers

Figure 2 shows the service unavailability, that is $1-A_s$, for different values of system utilisation in a configuration with only one server and up to three added redundant servers. While for the single server model service unavailability at low load is roughly 0.005 it rapidly increases with the load of the system and reaches 0.1 at full utilisation.

Adding a redundant server can decrease service unavailability significantly, especially with high utilisation of the system. The second and third added redundant server still yields an improvement in service availability of the system but at a rather low degree.

In contrast to Figure 2, which illustrates service unavailability as a function of utilisation, Figure 3 shows the effects of varying the number of servers. Each curve indicates one utilisation level of the system. Please note that utilisation is determined from a system with only one server. That is, e.g., for $\rho = 0.5$, we determine *arrival_time* to be $\frac{15h}{0.5} = 30h$. When adding more servers, *arrival_time* is kept at $30h$, and hence the actual system utilisation is lower due to reduced overall *service_time*. We chose this approach since we assumed that additional servers are introduced in order to improve service availability rather than to improve overall throughput. Again, the improvement at high utilisation shows very clearly, while at lower utilisation the system benefits much less from added redundancy.

Both, Figure 2 and Figure 3 show the improvement of service availability in absolute values. In contrast, Figure 4 illustrates relative improvement. For each utilisation the relative improvement by adding first one, then two and three redundant servers is shown. At high utilisation the addition of the first redundant server gives most relative improvement.

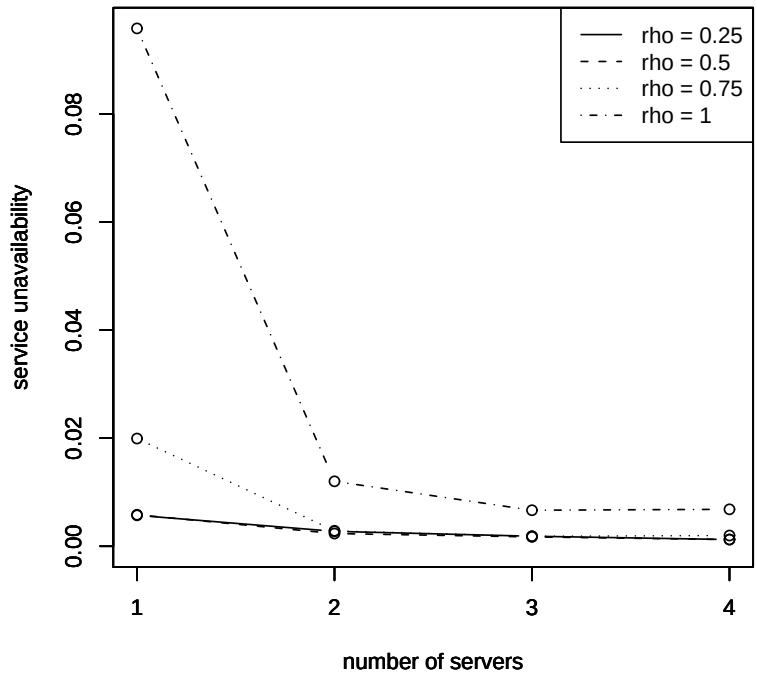

Figure 3: Number of servers versus service unavailability for different utilisation.

Adding two redundant servers reduces service unavailability still further, but at much lower degree. The same holds if three redundant servers are added. At low utilisation the relative improvement is much less, but saturation is not achieved with three added redundant servers. It should be noted, however, that while the relative improvement at low utilisation is much less than at high utilisation, of course, the (absolute) service unavailability is much lower. It is therefore more difficult to reduce service unavailability and more redundant servers are needed to do so.

We can conclude from our analysis that for high utilisation one redundant server should be added if possible. This will greatly improve service availability of the system. The positive effect certainly is supported by the fact that an additional server reduces the load on the system. For high utilisation adding more than one redundant server certainly has a positive effect but at much lower degree. It is worth mentioning, that at high load three added servers are not enough to obtain a highly available service. Already in earlier work we noticed the tremendous effect of load on service availability. The primary aim must be, therefore, to reduce load on the system.

For low utilisation added redundancy can achieve very high availability but this only comes at the cost of several redundant servers. Here, added redundancy has no relevant effect on the load but directly improves availability.

4 CONCLUSIONS AND OUTLOOK

We have presented a stochastic Petri net model for service availability improvement through added redundancy. The model consists of two parts, a queueing model, represented by a coloured stochastic Petri net, and a dependability model, represented by an extended deterministic Petri net. We have simulated the Petri net model for different levels of utilisation and have computed service unavailability in the different model configurations. From earlier work we know that high utilisation has a detrimental effect on service availability. Here, again, we find that at high utilisation the addition of at least one redundant server is very beneficial. At low utilisation added redundancy achieves less relative improvement in service availability although the absolute service unavailability is much lower. This work is only the first step of a comprehensive analysis of various mechanisms that improve service availability. Future analysis will incorporate server rejuvenation, i.e., the preventive

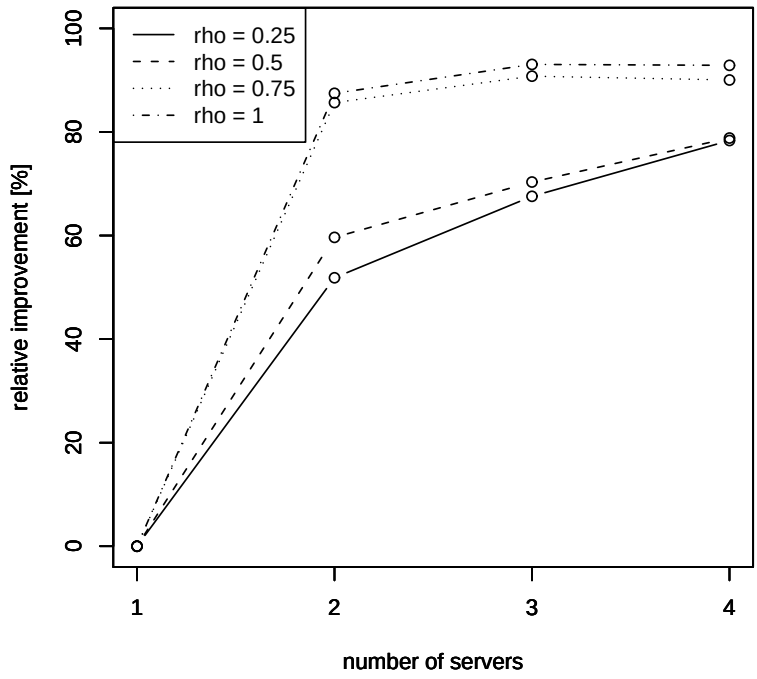

Figure 4: Relative service unavailability improvement

restart of servers. Figure 5 shows an extension of the model used in this paper. The performance model (left part) remains the same while the dependability model is extended (guard expressions need to be adjusted accordingly). The objective of this work is to evaluate effectiveness of server replication, server rejuvenation, and a combination of both. Our hypothesis is that, starting from a single server, replication is the most effective way to improve service availability. However, as we have seen in this study, the effect of adding servers "fades" with an increasing number of servers. We therefore expect that at some point, rejuvenating the servers might be more effective than adding another server. In summary, we strive to analyse various system configurations, as shown in Figure 6.

This work combines two approaches for increasing service availability and aims at providing guidelines for the most effective composition of both methods. In this paper we have used just one, the addition of redundancy, and have quantified its impact on service availability.

REFERENCES

Elnozahy, E. N., L. Alvisi, Y.-M. Wang, and D. B. Johnson. 2002. A Survey of Rollback-Recovery Protocols in Message-Passing Systems. *ACM Computing Surveys* 34 (3): 375–408.

Garg, S., A. Puliafito, M. Telek, and K. Trivedi. 1998. Analysis of preventive maintenance in transactions based software systems. *IEEE Trans. Comput.* 47 (1): 96–107.

Goseva-Popstojanova, K., and K. Trivedi. 2000. *Performance Evaluation - Origins and Directions*, Chapter Stochastic Modeling Formalisms for Dependability, Performance and Performability, 385–404. Lecture Notes in Computer Science. Springer Verlag.

Haverkort, B. R. 1998. *Performance of Computer Communication Systems: A Model-Based Approach*. Chichester, UK: John Wiley & Sons.

Huang, Y., C. Kintala, N. Kolettis, and N. D. Fulton. 1995, June. Software Rejuvenation: Analysis, Module and Applications. In *Proc. 25th Symposium on Fault Tolerant Computing*, 381–390. Pasadena, CA: IEEE.

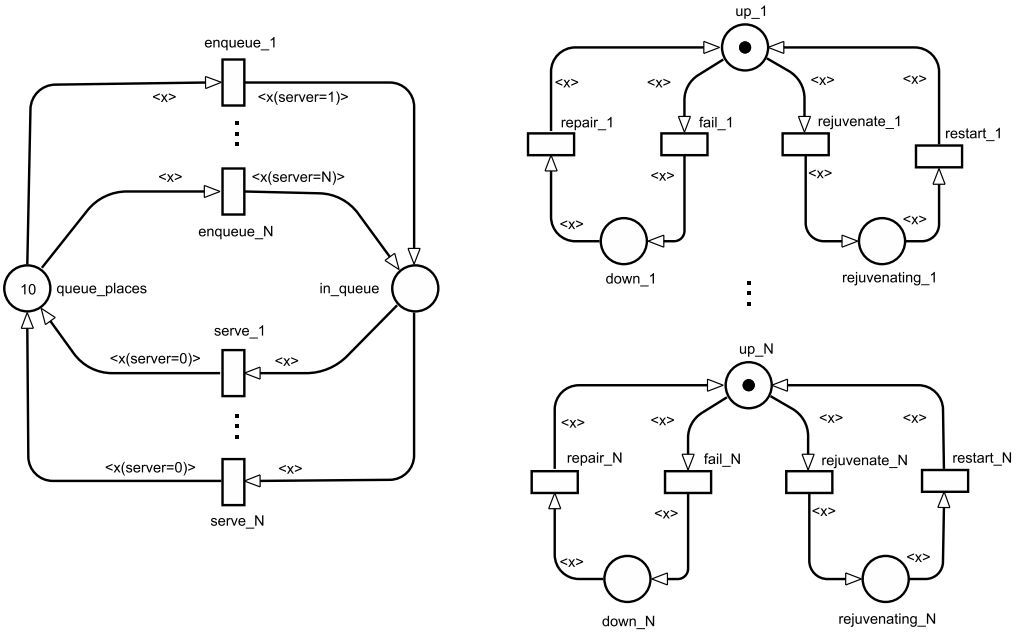

Figure 5: Future work: Model for N servers with rejuvenation.

Knoke, M., and A. Zimmermann. 2006. Distributed Simulation of Colored Stochastic Petri Nets With TimeNET 4.0. In *QEST '06: Proceedings of the 3rd international conference on the Quantitative Evaluation of Systems*, 117–118. Washington, DC, USA: IEEE Computer Society.

Salfner, F. 2008. *Event-based failure prediction: An extended hidden markov model approach*. Berlin, Germany: dissertation.de - Verlag im Internet GmbH. Available at `http://www.rok.informatik.hu-berlin.de/Members/salfner/publications/salfner08event-based.pdf`.

Salfner, F., and K. Wolter. 2008a. A queuing model for service availability of systems with rejuvenation. In *IEEE Proceedings of the Workshop on Software Aging and Rejuvenation (WoSAR), in conjunction with 19th International Symposium on Software Reliability Engineering (ISSRE)*. Seattle / Redmond, WA.

Salfner, F., and K. Wolter. 2008b. Service Availability of Systems with Failure Prevention. In *To appear in: IEEE Proceedings of International Workshop on Dependable and Secure Services Computing (DSSC)*. Jiaosi, Yilan, Taiwan.

Salfner, F., and K. Wolter. 2009. Analysis of service availability for time-triggered rejuvenation policies. submitted for publication.

Sun, H., J. Han, and H. Levendel. 2001. A Generic Availability Model for Clustered Computing Systems. In *Proc. Pcific Rim Symp. on Dependable Computing*, 241–248.

Trivedi, K., G. Ciardo, B. Dasarathy, M. Grottke, A. Rindos, and B. Varshaw. 2008. Achieving and Assuring High Availability. In *Parallel and Distributed Processing, 2008. IPDPS 2008. IEEE International Symposium on*, 1–7.

Wang, D., and K. Trivedi. 2005. Modeling user-perceived service availability. In *Service Availability*, Volume 3694 of *Lecture Notes in Computer Science (LNCS)*, 107. Springer.

Zimmermann, A., J. Freiheit, R. German, and G. Hommel. 2000. *Petri net modelling and performability evaluation with timenet 3.0*, Volume 1786 of *LNCS*, 188–202. Springer.

AUTHOR BIOGRAPHIES

FELIX SALFNER is a Senior Researcher at Humboldt-Universität zu Berlin. He received his Diploma degree in Computer Engineering in 2002 from Technical University Berlin and the Ph.D. in Computer Engineering in 2008 from Humboldt-Universität zu Berlin. In 2008, he has been a visiting scholar at International Computer Science Institute (ICSI) Berkeley and at SAP Labs, LLC, Palo Alto, California. His research focus is on failure prediction, proactive fault management, stochastic modelling, and machine learning. His Ph.D. thesis on event-based failure prediction has been published as a book and he

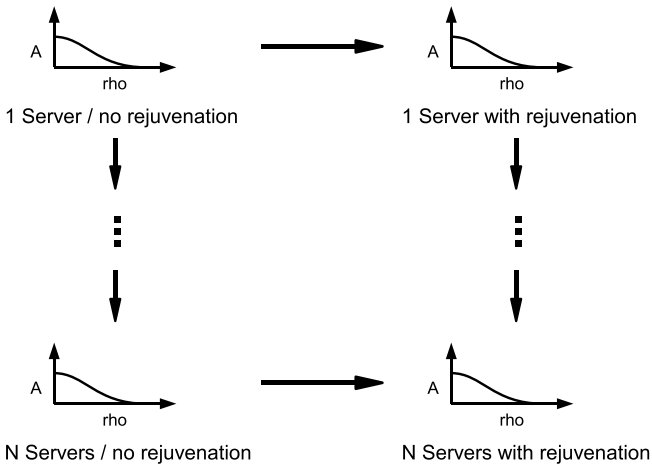

Figure 6: Future work: investigated configurations.

is author of about 25 publications including three chapters on dependability metrics and a comprehensive survey on online failure prediction methods.

KATINKA WOLTER is a Senior Researcher at Freie Universität Berlin. She received her Diploma degree in 1995 and her Ph.D. in Computer Science in 1999, both from the Technical University in Berlin, Germany. From 2002 until 2009 she held a position equivalent to that of Assistant Professor in the Computer Architecture and Communication group at Humboldt-University Berlin, where she received her habilitation degree in 2008. She leads a research group working on dependability of service-oriented systems and Quality-of-Service in wireless networks. Her interests are in performance, reliability and resilience evaluation of computer and communication systems, networks and applications.

Coloured Petri Nets is a language for the modelling and validation of concurrent and distributed systems and other systems in which concurrency plays a major role. The book introduces the constructs of the CPN modelling language and presents its analysis methods, and provides a comprehensive road map to the practical use of CP-nets. Furthermore, this book presents some selected industrial case studies illus-trating the practical use of CPN modelling and validation for design, specication, simulation, and verication in a variety of application domains. This book is aimed at use both in universi Petri Net is a model for complex discrete event systems built using graphical programming language for concurrent systems.Â Places in a Petri net may also contain a discrete number of marks called tokens. Purpose of the Petri Nets. It is to provide a variety of online services also for the international Petri Nets community. mailing lists. bibliographies. tool databases. newsletters. and also Addresses. Behavioral properties. â€¢ A Petri net is said to be k-Ââ€bounded or simply bounded if the number of tokens in each place does not exceed a nite number k for any marking reachable from M0. â€¢ The Petri net for vending machine is 1-Ââ€ bounded. â€¢ A 1-Ââ€bounded Petri net is also safe. Liveness. â€¢ A Petri net with ini4al marking M0 is live if, no maler what marking has been reached from M0, it is possible to ul4mately re any transi4on by progressing through some further ring sequence.Â Subclasses of Petri Nets (1). â€¢ Ordinary PNs. â€“ all arc weights are 1's â€“ same modeling power as general PN, more convenient for. analysis but less ecient. â€¢ State machine. Availability has long been a critical issue for online computer systems whose failure can halt business processes [2]. The need for high availability (HA) and disaster recovery (DR) in IT environment is more importance than other sectors of enterprises. Traditional high availability and disaster recovery solutions require a great deal of duplicate hardware and software.Â We construct a Petri net model for virtualized local DR and evaluate through both analytical and SHARPE tool simulation. 2. Proposed Virtualized Local Disaster Recovery.Â Each service under high availability DR solution needs at least two sites: primary site and disaster recovery site. The active physical server at primary site as well as DR site contains two or more VMs.