Pre-Proceedings of

9th International Conference on Applied Mathematics

25-28 September, 2013
Baia Mare, Romania

Editor: Petrica Pop Sitar

Baia Mare
2013

EDITURA BIBLIOPHIL
ISBN 978-606-93094-8-3

Tool supported analysis of queueing systems with Future Internet applications

János Sztrik and Tamás Bérczes
Faculty of Informatics, University of Debrecen, Hungary
sztrik.janos@inf.unideb.hu, berczes.tamas@inf.unideb.hu

Abstract. **This paper deals with the role of performance modeling tools for investigation of queueing systems. Advantages and drawbacks are considered when modeling and analysis issues are treated. For illustration the authors introduce 4 major tool developer centers. Due to the growing importance in practical applications, for example in Future Internet engineering, the reliability analysis of finite-source retrial queueing system is presented by using the MOdeling, Specification and Evaluation Language (MOSEL) software. A sample numerical example is given demonstrating the effect of failure of server on the mean response time of sources. Finally, investigating this model benchmarks to compare the efficiency of MOSEL and PRobabIliStic Model Checker (PRISM) tools concerning execution time, model construction time, model checking time are carried out.**

1.1. Some recent modeling tools. In the following 4 major tool developer centers are introduced briefly.

Tools at Faculty of Informatics, University of Dortmund, Germany This traditionally famous center has developed several software packages which can be downloaded from the site: `http://ls4-www.informatik.uni-dortmund.de/tools.html`.
Parallel to their methodological work the center continuously developed and used tools for performance evaluation and performability. Their intention is to provide facilities for a model description close to the original system specification and hiding details of the analysis techniques. Such tools map the model specification automatically to an analyzable model.

Möbius Tool Möbius is a software tool for modeling the behavior of complex systems. It is one of the major research projects of the Performability Engineering Research Group (PERFORM) in the Coordinated Science Laboratory at the University of Illinois at Urbana-Champaign, USA. Many advanced modeling formalisms and innovative and powerful solution techniques have been integrated in the Möbius framework.
The package can be found at `http://www.mobius.uiuc.edu/`.

MOSEL Tool

Performance modeling tools usually have their own textual or graphical specification language which depends largely on the underlying modeling formalism. MOSEL (MOdeling,Specification and Evaluation Language) tool, developed at the University of Erlangen, Germany, is based on the following idea: Instead of creating another tool with all the components needed for system description, state space generation, stochastic process derivation, and numerical solution, it focuses on the formal system description part and exploit the power of various existing and well tested packages for the subsequent stages. In order to reuse existing tools for the system analysis, the environment is equipped with a set of translators which transform the MOSEL model specification into various tool-specific system descriptions. For further details see [**2**].
MOSEL-2 provides means by which many interesting performance or reliability measures and the graphical presentation of them can be specified straightforwardly. All information can be found at `http://www4.informatik.uni-erlangen.de/Projects/MOSEL/`.

PRISM Tool

The PRobabIliStic Model checker PRISM provides a high-level modeling language for describing systems that exhibit probabilistic behavior, with models based on continuous-time Markov chains (CTMCs) as well as discrete-time Markov chains (DTMCs) and Markov decision procedures (MDPs). For specifying system properties, PRISM uses the continuous stochastic logic (CSL) for CTMCs and probabilistic computation tree logic (PCTL) for DTMCs and MDPs, both logics being extensions of computation tree logic (CTL), a temporal logic that is used in various classical model checkers for specifying properties. For further details, see for example [**7**].
The package can be downloaded from `http://www.prismmodelchecker.org`.

1.2. Finite-source retrial queues with a single server subject breakdowns and repairs. Due to the growing importance of using retrial queueing systems for modeling complex communication

systems in Future Internet applications we show an example for using MOSEL to analyze a retrial queueing system with the following assumptions. Consider a single server queueing system, where the primary calls are generated by K, $1 < K < \infty$ homogeneous sources. The server can be in three states: idle, busy and failed. If the server is idle, it can serve the calls of the sources. Each of the sources can be in three states: free, sending repeated calls and under service. If a source is free at time t it can generate a primary call during interval $(t, t + dt)$ with probability $\lambda dt + o(dt)$. If the server is free at the time of arrival of a call then the call starts to be served immediately, the source moves into the under service state and the server moves into busy state. The service is finished during the interval $(t, t + dt)$ with probability $\mu dt + o(dt)$ if the server is available. If the server is busy at the time of arrival of a call, then the source starts generation of a Poisson flow of repeated calls with rate ν until it finds the server free. After service the source becomes free, and it can generate a new primary call, and the server becomes idle so it can serve a new call. The server can fail during the interval $(t, t + dt)$ with probability $\delta dt + o(dt)$ if it is idle, and with probability $\gamma dt + o(dt)$ if it is busy. If $\delta = 0, \gamma > 0$ or $\delta = \gamma > 0$ *active or independent breakdowns* can be discussed, respectively. If the server fails in busy state, it either *continues servicing* the interrupted call after it has been repaired or the interrupted request *transmitted to the orbit*. The repair time is exponentially distributed with a finite mean $1/\tau$. If the server is failed two different cases can be treated. Namely, *blocked sources* case when all the operations are stopped, that is neither new primary calls nor repeated calls are generated. In the *unblocked (intelligent) sources* case only service is interrupted but all the other operations are continued (primary and repeated calls can be generated). All the times involved in the model are assumed to be mutually independent of each other. More information about retrial queueing systems can be read, for example in [**1**], [**4**], [**5, 6**].

For illustration our main objective is to demonstrate how the mean response time of calls depend on server's failure rates. To achieve this goal MOSEL is used to formulate and solve the problem.

The system state at time t can be described by the process
$X(t) = (Y(t); C(t); N(t))$, where $Y(t) = 0$ if the server is up, $Y(t) = 1$ if the server is failed, $C(t) = 0$ if the server is idle, $C(t) = 1$ if the server is busy, $N(t)$ is the number of sources of repeated calls at time t.

Because of the exponentiality of the involved random variables this process is a Markov chain with a finite state space. Since the state space of the process $(X(t), t \geq 0)$ is finite, the process is ergodic for all reasonable values of the rates involved in the model construction, hence from now on we will assume that the system is in the steady state.

We define the stationary probabilities:

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

$$q = 0, 1, \;\; r = 0, 1, \;\; j = 0, ..., K^*,$$

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

The advantage of using tools is that we do not have to deal with the numerical problems of finding the steady-state probabilities. We have to concentrate how to formulate the problem by the help of a describing language (usually Petri nets) and then the whole process is automatic. In our case, the states and transition rates of the continuous time finite state Markov chain is constructed and the stationary distribution is calculated by the help of a so-called solver. The problem arising in practical problems is the state space explosion which causes numerical problems due to the large number of unknowns. It should be noted that there is no closed form solution to this problem. However, as usual an algorithmic approach could be used to get the steady-state probabilities, see [**1**],[**8**] . This means an extra work in programming and displaying the results graphically which is in our case is an built-in module that is why we can omit it.

1.3. Numerical examples. We used the tool SPNP which was able to handle the model with up to 126 sources. In this case, on a computer containing a 1.1 GHz processor and 512 MB RAM, the running time was approximately 1 second. A more detailed investigation was published in [**3**] illustrating other effects as well.

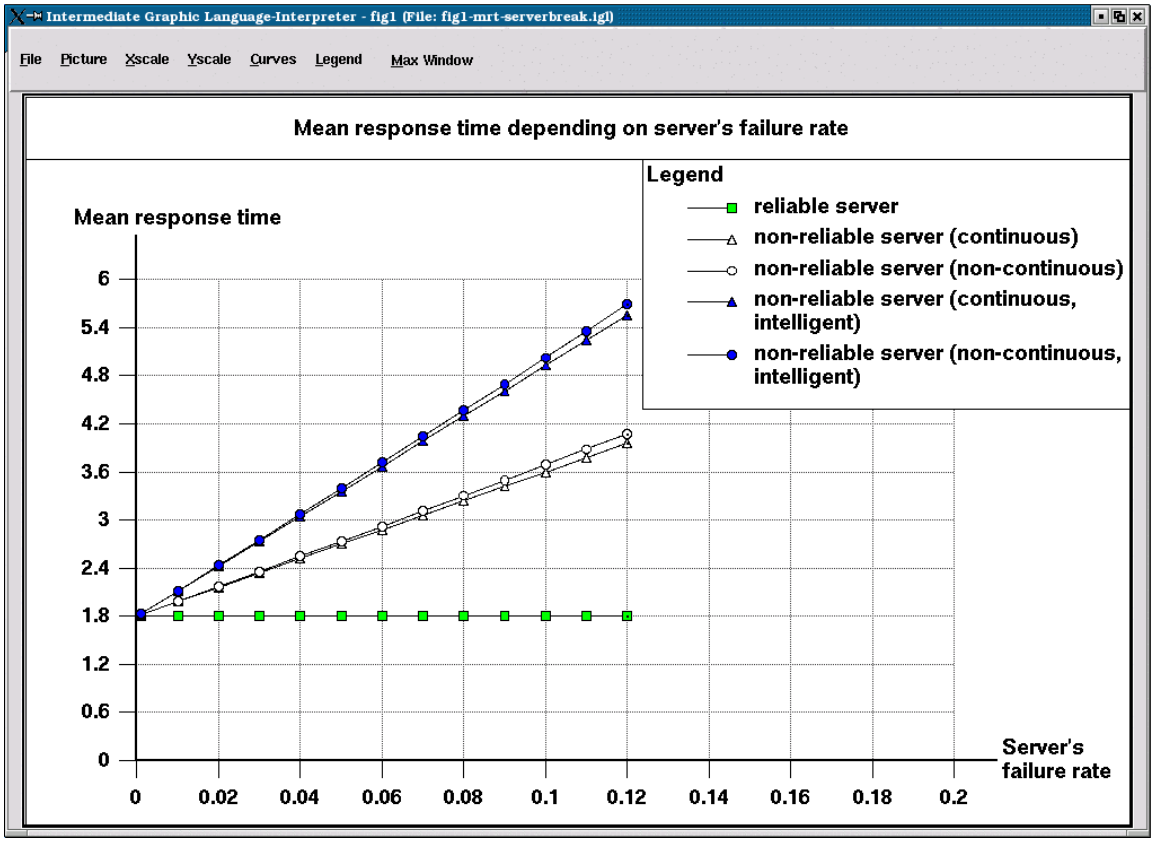

FIGURE 1 $E[T]$ versus server's failure rate

In Figure 1 we can see the mean response time for the reliable and the non-reliable retrial system when the server's failure rate increases. The input parameters are collected in Table 1.

	K	λ	μ	ν	δ, γ	τ
Figure 1	6	0.8	4	0.5	x axis	0.1

TABLE 1 Input system parameters

Comments

In Figure 1, we can see that in the case when the request returns to the orbit at the breakdown of the server, the sources will have always longer response times. Although the difference is not considerable it increase as the failure rate increase. The almost linear increase in $E[T]$ can be explained as follows. In the blocked (non-intelligent) case the failure of the server blocks all the operations and the response time is the sum of the down time of the server, the service and repeated call generation time of the request (which does not change during the failure) thus the failure has a linear effect on this measure. In the intelligent case the difference is only that the sources send repeated calls during the server is unavailable, so this is not an additional time.

Benchmarks In the following we give some comparisons between the MOSEL and PRISM tools showing how the number of terminals (NT) influences the execution time and total execution time.

The following preliminary conclusions can be drawn from benchmark:

- The execution times of the MOSEL system almost stay constant independently of NT;

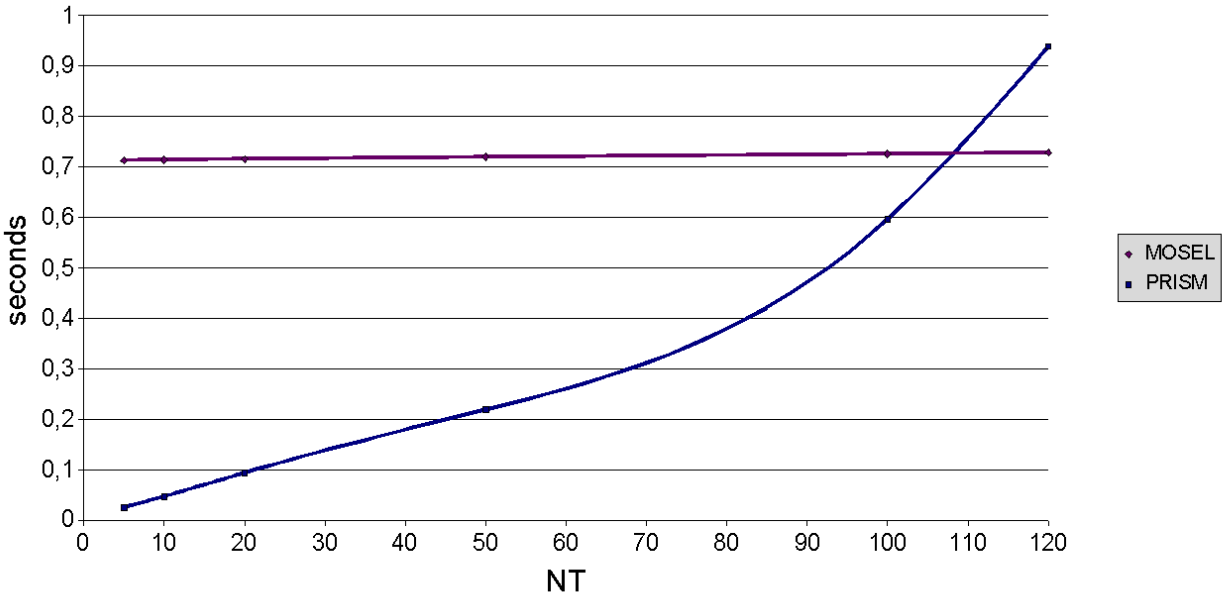

FIGURE 2 Execution Time

- The execution times of the PRISM system increase rapidly with the increase of *NT*.
- The model construction time in PRISM dominates the execution time rather than the model checking time.

While MOSEL is thus more efficient for smaller models, with PRISM also larger models can be analyzed. Furthermore, once a PRISM model is constructed, it can be arbitrarily often model checked with different parameter values. For such scenarios, the model checking time is more relevant than the model construction time.

Acknowledgment: This research was partially granted by the Hungarian Science and Technology Foundation, Hungarian- French Bilateral Cooperation under grant TeT 10-1-2011-0741, FR- 25/2010. The publication was also supported by the TÁMOP-4.2.2.C-11/1/KONV-2012- 0001 project. The project has been supported by the European Union, co- financed by the European Social Fund.

Bibliography

[1] Artalejo, J.R. and Gomez-Corral, A., *Retrial Queueing Systems*, Springer, 2008

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

[3] Bérczes, T. and Guta, G. and Kusper, G. and Schreiner, W. and Sztrik, J., Evaluating a probabilistic model checker for modeling and analyzing retrial queueing systems, *Annales Mathematicae et Informaticae*, Vol. 37, 51-75 (2010)

[4] Bérczes, T. and Orosz, P. and Moyal, P. and Limnios, N. and Georgiadis, S. and Sztrik, J., Tool supported modeling of sensor communication networks by using finite-source priority retrial queues, *Carpathian Journal of Electronic and Computer Engineering*, Vol. 5, 13-18 (2012)

[5] Do, V.T. , An efficient solution to a retrial queue for the performability evaluation of DHCP , *Computers & OR*, Vol. 37, 1191-1198 (2010)

[6] Do, V.T. , A new computational algorithm for retrial queues to cellular mobile , *Computers & Industrial Engineering*, Vol. 59, 865-872 (2010)

[7] Kwiatkowska, M. and Norman, G. and Parker D., PRISM: Probabilistic Model Checking for Performance and Reliability Analysis, *ACM Sigmetrics Performance Evaluation*, Vol. 36, 40-45 (2009)

[8] Lakatos, L. and Szeidl, L. and Telek, M., *Introduction to Queueing Systems with Telecommunication Applications*, Springer, 2012