AN ADVANCED SIMULATION MODEL FOR DEPENDABLE DISTRIBUTED SYSTEMS

Ciprian Dobre, Florin Pop, Valentin Cristea
Department of Computer Science

University POLITEHNICA of Bucharest
Spl. Independentei 313, Bucharest, Romania
E-mails: {ciprian.dobre, florin.pop, valentin.cristea}@cs.pub.ro

Joanna Kolodziej
Department of Mathematics and Computer Science
University of Bielsko-Biala
ul. Willowa 2, 43-309 Bielsko-Biala, Poland
E-mail: jkolodziej@ath.bielsko.pl

KEYWORDS
Simulation model, dependability, large scale distributed systems

ABSTRACT

We present a simulation model designed for evaluation of dependability in distributed systems. The model is a modification of the MONARC simulation model by adding new capabilities for capturing the reliability, safety, availability, security, and maintainability requirements. It includes components for failures injection, and it provides evaluation mechanisms for different replication strategies, redundancy procedures, and security enforcement mechanisms. The model is implemented as an extension of the multi-threaded, process oriented simulator MONARC, which allows the realistic simulation of a wide-range of distributed system technologies, with respect to their specific components and characteristics. The experimental results show that the application of the discrete-event simulators in the design and development of the dependable distributed systems is appealing due to their efficiency and scalability

INTRODUCTION

Up until recently the research efforts in the area of large scale distributed systems (LSDS) mainly targeted the development of functional infrastructures. As the application domains of LSDS are extending, researchers have to deal with new requirements. Today LSDS are required to offer reliability, safety, availability, security (all attributes of dependability).

In this paper we present a simulation model designed to evaluate the dependability in LSDS. The proposed model extends the MONARC simulation model (Dobre et al. 2008a) with new capabilities for capturing reliability, safety, availability, security, and maintainability requirements. The simulation model includes the necessary components to inject failure events, and provides the mechanisms to evaluate different strategies for replication, redundancy procedures, and security enforcement mechanisms, as well. The paper extends the results presented in (Dobre et al. 2008b), introducing the simulation model designed for dependability of distributed systems. The results achieved in experimental analysis show that the application of discrete-event simulators in the design and development of distributed systems is appealing due to their efficiency and scalability.

The rest of this paper is structured as follows. Section 2 presents related work in the area of modeling distributed systems. In Section 3 we present the extended simulation model to simulate dependable LSDS. Sections 4 and 5 present details for the proposed fault tolerance and security simulation models. The experimental evaluation results are demonstrated in Section 6. The paper ends with final remarks and conclusions in Section 7.

2 RELATED WORK AND OUR APPROACH

Modeling and simulation are the effective tools for the development of the new algorithms and technologies. They enable the enhancement of large-scale distributed systems, where analytical validations are prohibited by the scale of the encountered problems. The use of discrete-event simulators in the design and development of LSDS is appealing due to their efficiency and scalability.

SimGrid (Casanova et al. 2008) is a simulation toolkit that provides core functionalities for the evaluation of scheduling algorithms in distributed applications in a heterogeneous, computational Grid environment. It aims at providing the right model and level of abstraction for studying Grid-based scheduling algorithms and generates correct and accurate simulation results. The Grid simulator toolkit developed for the investigation is the *GridSim* system introduced by Buya et al. in (Buyya and Murshed 2002). The main concept of the simulator is on the computational economy.

OptorSim (Venters et al. 2007) is a Data Grid simulator designed for evaluating optimization in data access technologies for Grid environments. It adopts a Grid structure based on a simplification of the architecture proposed by the EU DataGrid project. Another simulator package is *ChicagoSim* (Ranganathan and Foster 2002). It is dedicated for the implementation of

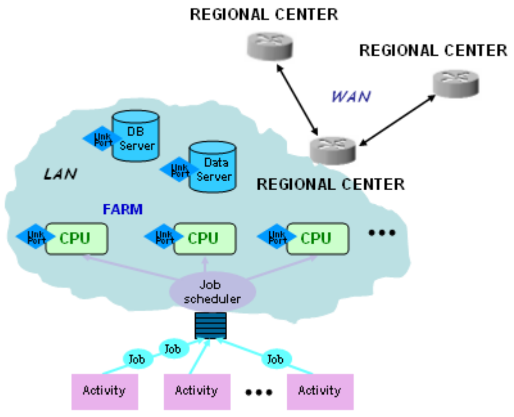
Proceedings 25th European Conference on Modelling and Simulation ©ECMS Tadeusz Burczynski, Joanna Kolodziej Aleksander Byrski, Marco Carvalho (Editors)
ISBN: 978-0-9564944-2-9 / ISBN: 978-0-9564944-3-6 (CD)

the scheduling strategies in conjunction with the data location.

These simulators do not present general solutions to modeling dependability technologies for LSDS. They provide basic simulation models for evaluating LSDS. We propose a simulation model that provides the means to evaluate a wide-range of solutions for dependability in LSDS. Security in particular has never been properly handled by any of these projects. The only simulator that is able to evaluate LSDS security aspects is G3S (Grid Security Services Simulator) (Naqvi and Riguidel 2005). Similar to its model, we support all the mechanisms made available in G3S. Our proposed simulation includes capabilities for modeling security aspects, from patterns of attack to intrusion detection, authentication or privacy enforcement solutions. It also includes the mechanisms to evaluate security in a more general context, modeling more realistically distributed systems, with their specific characteristics. The model is able to describe actual distributed system technologies, and provides the mechanisms to describe concurrent network traffic, to evaluate different strategies in data replication, and to analyze job scheduling procedures. MONARC offers ample customization possibilities, thus enabling users to integrate different proprietary solutions. For example, unlike G3S, the MONARC's model can incorporate custom security solutions designed by the user for particular scenarios.

3 MONARC'S ARCHITECTURE

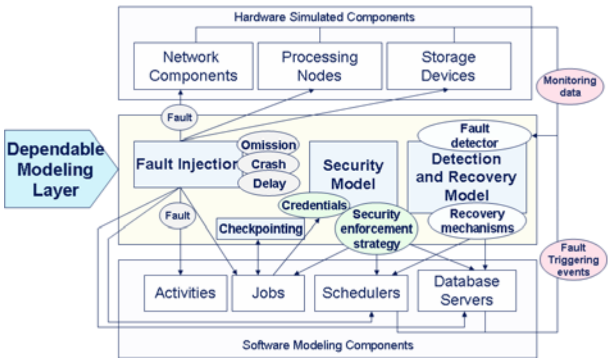
MONARC is based on a process oriented approach for discrete event simulation, which is suited to describe concurrent running programs, network traffic as well as all the stochastic arrival patterns specific for such type of simulations (Legrand et al. 2003). The simulator is able to handle experiments involving thousands of processing nodes executing a large number of concurrent jobs, and thousands of concurrent network transfers.



Figures 1: The Regional Center model.

MONARC uses a simulation model that abstracts components found in LSDS infrastructures. The model includes components for simulating processing units, databases, and network traffic. Such components can be grouped in regional centers (Figure 1). A regional center is used to abstract a group of resources that are under the control of a single organization. Several regional centers can be linked to simulate cooperation with other resources, similar to how clusters are used in larger Grids. The simulation model also include components to model the behavior of the applications and their interaction with users. Such components are used to generate data processing.

One of the strengths of MONARC is that it can be easily extended. This is made possible by its layered structure. The first two layers contain the core of the simulator (called the "simulation engine") and models for the basic components of a distributed system (processing units, jobs, databases, networks, job schedulers etc). The particular components can be different types of jobs, job schedulers with specific scheduling algorithms or database servers that support data replication. Using this structure it is possible to build a wide range of models, from the very centralized to the distributed system models, with an almost arbitrary level of complexity (multiple regional centers, each with different hardware configuration and possibly different sets of replicated data).



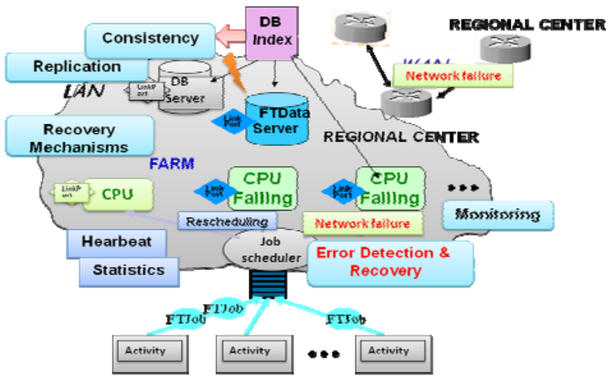
Figures 2: The dependable simulation model and its components.

In this paper we present an extended MONARC simulation model that is able to evaluate dependability aspects for LSDS. The model is designed to evaluate fault detection, fault recovery, or security solutions in a unitary and aggregated way. Figure 2 presents the components of the dependable modeling layer. The extensions to the simulation model were introduced for modeling faults, failure detection, fault recovery, and security aspects within a modeled distributed system.

4 FAULT TOLERANCE MODEL

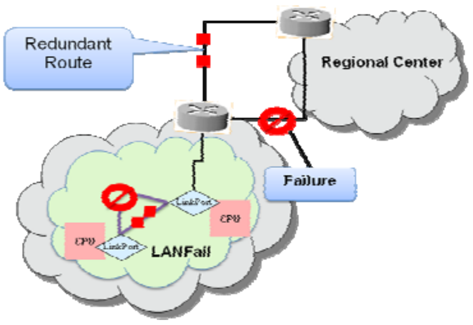
We first extended the simulation model (see Figure 3) and added the mechanisms to evaluate fault tolerance in LSDS. The model includes components necessary to inject and recover from faults in the processing, networking as well as database layers.

The fault tolerance model also allows the simulation of hybrid systems, in which failing components can coexist with traditional components of the MONARC's original model. The mechanisms for fault tolerance are added as extensions to MONARC's simulated components, and in simulation experiments both types of components, with and without the fault tolerance extensions, can coexist.



Figures 3: The extended fault tolerance model.

Figure 4 presents the modeling of fault tolerancein in communication layer. In this case various failures can occur within the network links, and routers. In addition, the model can simulate redundancy and recovery protocols based.



Figures 4: A fault tolerant network model.

At the hardware layer different distributed components can be modeled as failing: the processing unit, the network connectivity as well as the storage devices. In software we consider faults occurring in a middleware component (a faulty scheduler, a database server returning incorrect results, etc.) or within higher-level distributed application (jobs could fail to return correct results).

The model includes faults such as crash faults, omission faults, time faults, as well as Byzantine faults. All modeled components have mechanisms to simulate injection of faults (or the modeling of their failure). The mechanisms are configurable via metrics for calculating processor availability. The first mechanism is based on the MTTF (mean time to failure) parameter. The fault injection uses the MTTF together with a mathematical probability distribution (such as binomial, Poisson, Gaussian, standard uniform, etc)., such as at random intervals a component can experience faulty behavior (failures). The second error injection mechanism uses random occurrence of fault events. This is useful in modeling Byzantine failures. For such errors the simulation model allows resuming the normal behavior of the faulty component.

The fault injection mechanisms are used together with fault detection and recovery mechanisms. For that the model includes a monitoring component. The component is responsible with the management of events related to the triggering of faults. In the event of a failure each component can take global actions (such as update of the service catalogue if the experiment requires it). It also updates the states of the distributed system, and informs other components of the event.

The scheduler also implements a fault-tolerant mechanism. Whenever a new job is submitted the scheduler also produces a special simulation event that triggers when a timeout occurs. The timeout depends on the user's specifications and is used as a signal if the job fails to return results in due time. In this case the scheduler is interrupted either when the job finishes or when the timeout event occurs. The same mechanisms are implemented within the network simulation model. In this case a job is informed if a transfer failed to finish in a specified amount of time (possible due to network congestion) and can take appropriate measures (such as canceling the transfer or saving the state).

The simulation model also includes mechanisms model check-pointing or logging of the system's state. Such mechanisms are implemented using MONARC's simulation events. The model is able to simulate both static and dynamic check-pointing strategies.

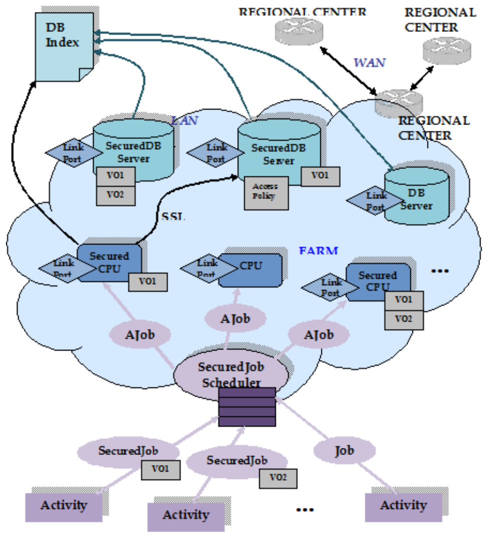
The simulation model also includes mechanisms for the evaluation of replication and redundancy mechanisms. Replication provides mechanisms to use multiple instances of the same system or subsystems and choose the result based on quorum. The simulation model allows the simulation of DAG distributed activities. This is useful in modeling job replication, when the same job can be executed on multiple processing units, and another job receives the outputs and selects the correct results. The possibility to model replication mechanism was demonstrated in (Eremia, *et al*, 2010). Redundancy results were demonstrated in experiments presented in (Dobre et al. 2008b).

5 SECURITY MODEL FOR LARGE SCALE DISTRIBUTED SYSTEMS

LSDS are vulnerable to security threats because they rely on distributed access control mechanisms necessary to access remotely wide-spread resources that are under different administrative domains. The MONARC's simulation model includes components for the analysis of security-dependent experiments (Figure 5). It is capable to simulate security solutions used in real-world distributed environments, such as GSI, PKI, SSL, cryptographic solutions, etc. In addition, the model includes various simulated security attacks. It allows the addition of detection mechanisms for such attacks, by providing simulation mechanisms for message encryption or authentication and authorization.

The model considers the general case of security, as a mean to ensure that systems remain safe and reliable to errors, threats or malicious changes. The model includes solutions for *data privacy, data integrity* and *system availability*.

The security model allows the specification of security policies. A security policy describes which actions are allowed and which are prohibited. Entities to which these actions apply include users, services, information, machinery, etc. Once the security policy is established, the necessary security enforcement mechanisms are considered. The model includes various security mechanisms (Johnston, 2004): *confidentiality* (it includes mechanisms to ensure that an authenticated entity can access only the information that has been authorized to*)*, *authentication (*the models includes mechanisms to identify entities involved in a communication or collaboration*)*, a*uthorization (*the model guarantees that once the entity has been authenticated, its options will be restricted / limited to those operations that it is authorized to perform*)*, and a*udit (*the models includes the mechanisms to guarantee the non-repudiation of origin and content of a message).



Figures 5: The security simulation model.

The security model includes a secured job that carries authentication tokens or certificates, and is able to request data based on specific rights. The user can specify the use of X.509 certificate, together with a PKI infrastructure for example, or can easily add new means of authentication. In particular for Grid systems an additional important concept considered by the security model is the Virtual Organization (VO). In a VO different organizations (commercial companies, universities, etc.) collaborate to share resources and work together to solve common problems. Each organization within a VO is managed independently and has its own security solutions such as Kerberos or PKI infrastructure (Public Key Infrastructure). To define VOs the model uses security policies shared between regional centers. The model includes mechanisms to evaluate various authentication solutions. Such authentication mechanisms are applied to the scheduler, processing unit, and to jobs requesting data from the database servers. For example, the job scheduler includes restrictions to where to execute specific jobs, based on the VO to which they belong. The processing units are capable to verify if a particular job is allowed to be executed. The access control verification can be implemented based on various schemas (RBAC, MAC, DAC, etc).

The model adds the possibility to include secure data transport protocols. For example, the SSL protocol offers the possibility to encrypt the messages being exchanged between entities in a simulation experiment. In addition the model implements handshake mechanisms for protocols supporting authentication capabilities. The user can easily add and evaluate new protocols and mechanisms. The model includes mechanisms for data encryption, keys and certificate management, etc. In addition, it includes mechanisms for traffic filtering by specifying exclusion rules based on various metrics (ports, addresses, protocols, etc) and corresponding actions (reject for example).

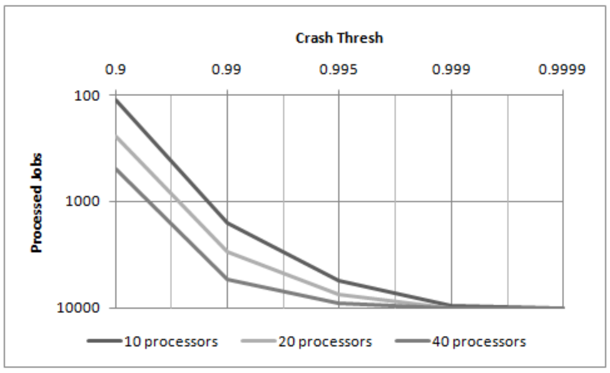
The security model also enables the protection of message content sent throughout the network against attacks such as interception (eavesdropping), and thus keeping its confidentiality, by encrypting its content. It also ensures secure data transfers by using protocols to allow the authentication of the parties involved in the communication. This ensures both the integrity of messages transmitted, and their protection against attacks such as man in the middle.

The implementation also includes an exclusion rule based traffic filtering of all components of a virtual organization. This mechanism can be used to prevent attacks such as DoS. In case of many connections coming from the same address, for example, the filtering policy can specify that that particular address is banned for a certain period of time (or permanently).

6. EXPERIMENTAL RESULTS

To analyze the validity and performance of the dependability simulation model we conducted several simulation experiments.

We first evaluated fault tolerance. The first experiment analyzed how the number of processing units relates to the reliability in processing a batch of tasks. The objective was to guarantee that a given number of tasks can be processed, without considering delays caused by failed processing units. If no processing unit is working at a given moment, the experiment fails.
In the experiment a number of jobs are sent for processing. The job scheduler is responsible with finding a suitable processing unit for each of these jobs.



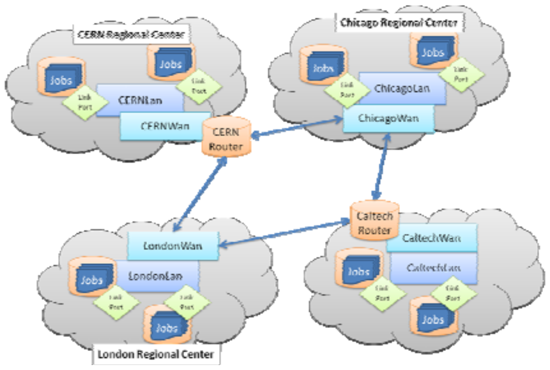
Figures 6: Results obtained for batch of tasks.

The results in Figure 6 were obtained for different cases (10, 20, and 40 processors) and 10,000 jobs sent for execution. The *CrashThresh* parameter shows the probability of the processing units to experiment permanent failures. In these experiments, for particular cases (Figure 6), the job scheduler gets into a state where there are no more processing units to execute jobs. In this case the scheduler is no longer able to mask failure and, therefore, the user sees a lower number of processing jobs successfully executed (the vertical axis).

Table 1: Results for transient failures.

Jobs	CPUs	Transient Thresh	Avg. Failed CPUs	Processed
10000	10	0.5	7	4931
10000	10	0.6	3	10000
10000	10	0.7	1	10000

We continued with experiments where 10 processing units experience transient failures. In these experiments we varied the probability of processors to experience failures (the *Transient Thresh* parameter).

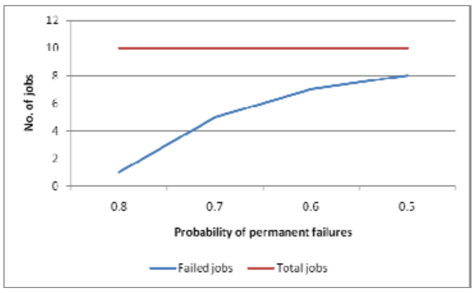


Figures 7: The simulation scenario used with the Network Failure experiments.

The obtained results (see Table 1) show a bottleneck for the number of jobs that are successfully executed. In this case the job scheduler considers that CPUs fail if they don't answer for one heartbeat and they are repaired if one positive answer is received.

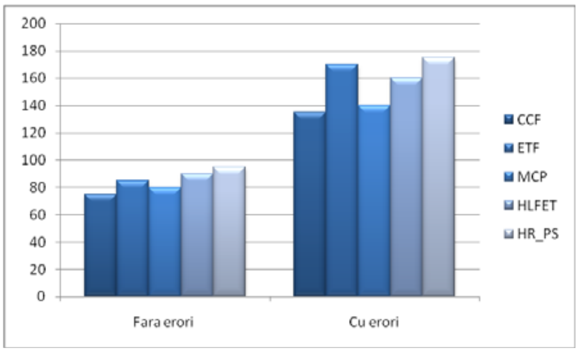
These experiments reveal the importance of taking repairing actions in case of faulty resources. If no permanent faults occur, and transient faults occur in a reasonable range, a task still finishes, independently of the batch size. This is because the processing units are repaired faster than they break.

Because tasks conserve the work they've done when stopped, the time is proportional with the average percentage of failed processors if all tasks are completed. In this case efficiency is computed by dividing the ideal completion time to the actual completion time if failures occur. By default jobs are not reset when rescheduled, resulting in efficiency values proportional to the average number of working processors. If jobs are reset when rescheduled, efficiency is much more correlated to MTBF. If a processor can never finish by itself a job, no jobs will be completed, resulting null efficiency.



Figures 8: Results obtained for different probabilities of links to experience permanent failures.

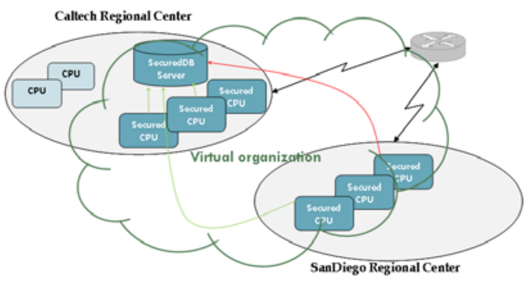
Another set of experiments was further designed to evaluate the relation between redundant network links and link reliability. The goal is to send a number of packets, without considering delays. Because of its resilience to missed packets, TCP was chosen as the transport protocol.



Figures 9: Comparison between execution time (s) for scheduling algorithms with and without errors.

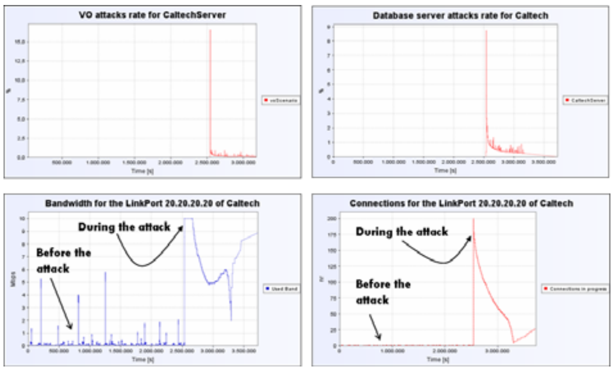
Figure 7 shows the experiment's network topology. *Cern LAN* sends packets to *Caltech LAN*. Packets are routed by *Cern Router* through the two possible paths towards *Caltech Router* in respect to network load. Figure 8 shows the results obtained for the case when the network links can experience permanent failures. We considered a number of 10 jobs that are sending messages. We then varied the probability of a link to experience permanent failures. The vertical axis shows the number of jobs that were able to complete their tasks of transferring the data.

We also evaluated various fault-tolerant scheduling algorithms for DAGs. The experiments considered the case of several complex DAG dependent tasks that were submitted for execution, and the cases when faults occur or not. The results are presented in Figure 9. Differences between the submitted jobs and the finalized ones represent the number of jobs that were successfully rescheduled (when faults occurred).



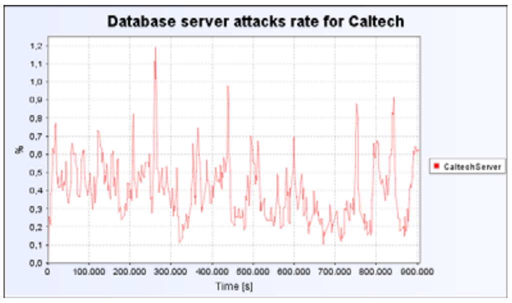
Figures 10: Experiment evaluating the security simulation model.

An experiment designed to evaluate the security model considering the case of two regional centers is presented in Figure 10. The experiment involves sharing several processing units and a database server within a virtual organization. The purpose is to demonstrate the functionality of an access policy within the secured database server. The experiment uses two types of jobs: one requests the creation of a database and writes data in it; and the other one connects to the server and requests the data matching a specific pattern.



Figures 11: Results obtained for the security experiment.

We associated a security policy resembling the UNIX file access policies to the database server belonging to the VO. We considered that members of the VO have read and write rights over the database server. A *get* operation is ignored and the operation is considered an implicit attack on the database server. The experiment consisted in the insertion of many jobs of the types previously presented. The results (Figure 11) demonstrate that during an attack the throughput increases, in contrast with the initial conditions of the experiments. Also, the number of received connections increases during an attack. The results demonstrate the validity of the proposed security model, as they are well mapped with the analytical results expected from the experiment. We also conducted a number of other experiments, trying to evaluate the components proposed within the security model, ranging from securing communication to imposing access control at virtual organization level.



Figures 12: The percent of attacks recognized on the database side from the total number of requests.

By extending the security model, we were able to concurrently simulate both ordinary jobs, as well as ones that tried different operations on the database without having sufficient rights. We logged and compared how many attacks were randomly generated (reads without the read right, etc.) versus how many attacks did the database server successfully recognized (Figure 12).

In all these cases not only the security solutions designed and included in the proposed security model correctly handled possible attacks, but also the performance of the distributed simulated environment (throughput in the network or processing capability of the simulated processing units) was not affected beyond rendering the environment to be used anymore.

7. CONCLUSIONS

As society increasingly becomes dependent of distributed systems (Grid, P2P, network-based), it is becoming more and more imperative to engineer solutions to achieve reasonable levels of dependability for such systems. Simulation plays an important part in building and evaluating dependable distributed systems. In this paper we presented a simulation model designed to evaluate the dependability in distributed systems. The model extends the MONARC simulation model with new capabilities for capturing reliability, safety, availability, security, and maintainability requirements. The model extends the multithreaded, process oriented simulator MONARC. It includes the necessary components to inject various failure events, and provides the mechanisms to evaluate different strategies for replication, redundancy procedures, as well as security enforcement mechanisms. The results obtained in presented simulation experiments probe that the use of discrete-event simulators, such as MONARC, in the design and development of dependable distributed systems is appealing due to their efficiency and scalability.

ACKNOWLEDGEMENT

The research presented in this paper is supported by national project “DEPSYS – Models and Techniques for ensuring reliability, safety, availability and security of Large Scale Distributes Systems”, Project “CNCSIS-IDEI” ID: 1710, by the national project: "TRANSYS – Models and Techniques for Traffic Optimizing in Urban Environments", Project CNCSIS-PN-II-RU-PD ID: 238, and by the Sectorial Operational Programme Human Resources Development 2007-2013 of the Romanian Ministry of Labour, Family and Social Protection through the Financial Agreement POSDRU/89/1.5/S/62557.

REFERENCES

Buyya, R.; and M. Murshed. 2002. “GridSim: A Toolkit for the Modeling and Simulation of Distributed Resource Management and Scheduling for Grid Computing”. *Journal of Concurrency and Computation: Practice and Experience* (CCPE), Vol. 14, 1175–1220.

Casanova, H.; A. Legrand; and M. Quinson. 2008 “SimGrid: a Generic Framework for Large-Scale Distributed Experimentations”. In *Proc. of the 10th IEEE Int. Conference on Computer Modelling and Simulation* (UKSIM/EUROSIM'08), Cambridge, UK, 126-131.

Dobre, C.; C. Stratan; and V. Cristea. 2008a. “Realistic simulation of large scale distributed systems using monitoring”. In *Proc. of the 7th International Symposium on Parallel and Distributed Computing* (ISPDC 2008), Krakow, Poland, 434-438.

Dobre, C.; F. Pop; and V. Cristea. 2008b. “A Simulation Framework for Dependable Distributed Systems”, In *Proc. of the First International Workshop on Simulation and Modelling in Emergent Computational Systems* (SMECS-2008), Portland, USA, 181-187.

Eremia, B.; C. Dobre; F. Pop; A. Costan; and V. Cristea. 2010. “Simulation model and instrument to evaluate replication techniques”, In *Proc. of the 3PGCIC 2010, International Conference on, P2P, Paralel, Grid, Cloud and Internet Computing*, Fukuoka, Japonia, 541-547.

Johnston, S. 2004. “Modeling security concerns in service-oriented architectures”. Accessed 07.02. 2011, From: http://www.ibm.com/developerworks/ rational/library/4860.html.

Legrand, I.C.; H. Newman; C. Dobre; and C. Stratan. 2003. “MONARC Simulation Framework”. In *Proc. of the Int. Workshop on Advanced Computing and Analysis Techniques in Physics Research*, Tsukuba, Japan, 133-138.

Naqvi, S.; and M. Riguidel. 2005. “Grid Security Services Simulator G3S) – A Simulation Tool for the Design and Analysis of Grid Security Solutions”. In *Proc. of the First International Conference on e-Science and Grid Computing*, Melbourne, Australia, 421-428.

Ranganathan, K.; and I. Foster. 2002. “Decoupling Computation and Data Scheduling in Distributed Data-Intensive Applications”. In *Proc. of the Int. Symposium of High Performance Distributed Computing*, Edinburgh, Scotland, 352-356.

Venters, W.; T. Cornford; M. Lancaster; Y. Zheng; and A. Kyrikidou. 2007. “Studying the usability of Grids, ethongraphic research of the UK particle physics community”. In *Proc. of the UK e-Science All Hands Conference*, Nottingham, UK, 683-685.

AUTHOR BIOGRAPHIES

Dr. Ciprian DOBRE received his PhD in Computer Science at the University POLITEHNICA of Bucharest in 2008. His main research interests are Modeling and Simulation, Grid Computing, Monitoring and Control of Distributed Systems, Advanced Networking Architectures, Parallel and Distributed Algorithms. His research activities were awarded with the Innovations in Networking Award for Experimental Applications in 2008 by the Corporation for Education Network Initiatives (CENIC).

Dr. Florin POP is a lecturer in the Computer Science Department of the University Politehnica of Bucharest. His research interests are oriented to: scheduling in Grid environments (his PhD research), distributed system, parallel computation, communication protocols and numerical methods. He received his PhD in Computer Science in 2008 with Magna cum laudae distinction. He is member of RoGrid consortium and participates in research projects from Romania and abroad.

Prof. Dr. Eng. Valentin CRISTEA is a professor of the Computer Science and Engineering Department of the University Politehnica of Bucharest (UPB). He teaches courses on Distributed Systems and Algorithms. As a PhD supervisor he directs thesis on Grids and Distributed Computing. Valentin Cristea is Director of the National Center for Information Technology of UPB and leads the laboratories of Collaborative High Performance Computing and eBusiness.

Dr. Joanna KOLODZIEJ graduated in Mathematics from the Jagiellonian University in Cracow in 1992, where she also obtained the PhD in Computer Science in 2004. She has served and is currently serving as PC Co-Chair, General Co-Chair and IPC member of several international conferences and workshops including PPSN 2010, ECMS 2011, CISIS 2011, 3PGCIC 2011, CISSE 2006, CEC 2008, IACS 2008-2009, ICAART 2009-2010. Dr Kolodziej is Managing Editor of IJSSC Journal and serves as a EB member and guest editor of several peer-reviewed international journals.