Experience of Redundant System Simulation for Evaluation of Timing Characteristics

Alexander Pastsyak, Artem Ozhigin
Corporate Technology, Dependable Systems Group
Siemens LLC
St. Petersburg, Russia
artem.ozhigin@siemens.com, alexander.pastsyak@siemens.com

***Abstract* — Redundant units are introduced in many systems to assure continuous system operation in critical situations. When such unit operates in hot-reserved mode important characteristic is the time to bring it into normal conditions. This property is highly affected by the algorithm which manages unit state switches and its reliable evaluation can only be done when this logic is designed and implemented. That is why simulation of system behavior remains the only choice to evaluate timing characteristics and design decisions before system engineering. In this paper we present an experience of using discrete-time event based approach for estimation of performance properties of a synchronization manager – the component of a larger system which is responsible for the handling of overall system redundancy. Results of this work include performance measurements of different state switching algorithms and summary of the gained experience.**

Keywords — redundant system, descret-time event simulation, transition time distribution, switching latency.

I. INTRODUCTION

Redundant components are used in many safety-critical systems to enhance reliability of the whole system. Two important characteristics need to be considered during design of such system: time before failure (TBF) for every component and time to switch to redundant unit in case of failure. The TBF property is usually affected by hardware design and operating conditions and can be estimated during early development phase. On the other hand the second property is influenced by the logic which defines switch conditions and if there are restrictions applied to the timing characteristics they must be taken into account earlier than at implementation stage. Simulation is a good way to find proper design or architecture of the system.

In our work we present an experience of using discrete-time event based simulation approach for estimation of switching times of Synchronization Manager (SM) – component which is responsible for redundant operation of a plant control system. This system consists of two identical units and SM manages the switches between them in case of failures. The results of this work include estimation of switching times under several failure conditions. These estimations have been done with different algorithms for state switching and simulating instability of the communication channel. The summary of the gained experience is also presented.

II. RELATED WORK

Investigation of timing characteristics of redundant systems has been a topic of interest for a long time and practical results were received in many disciplines from avionics [6, 7] to development of dependable web applications [2] and analysis of fault-tolerant switching in communication networks [8]. For example, in [1] authors describe the way they used to simulate redundant firewall system during implementation of ISP cache server system. They are focused mainly on throughput characteristics and their dependencies on number of nodes. They utilize an approach of scaled-down testing using a set of hardware nodes simulating parts of the system and its environment (internet, intranet, traffic generation). In order to evaluate the impact of particular node performance on the overall system the handicap factor is used in experiments. Described setup allowed for evaluation of performance impact of different system parts showing its strengths and weaknesses.

A simulation-based approach to software performance modeling is presented in [6]. It is based on UML Profile for Schedulability, Performance and Time Specification and uses specially annotated use case, activity and deployment diagrams representing the software architecture to build process-oriented simulation models. Methodology is illustrated using an example of web-based video streaming application. Simulation results are represented as steady-state average delays.

In our work we focus on performance comparison of different algorithms for state switching of redundant nodes and analysis of system behavior in case of problems in communication network between two duplicating components.

III. THE SYSTEM UNDER STUDY

Industrial automation system contains a couple of identical units working in hot-reserve mode. It is required to have only one of the units performing operations at a time while the other unit is standing by. SM is a part of unit's software responsible for coordination of units to assure correct selection of active unit and appropriate role switch in case of failure. SM consumes information about unit's external environment and internal status, besides that redundant communication (Ethernet-based) is established between a pair of units for SMs to coordinate state switches (link status is diagnosed and used as input as well). In case of redundant link malfunction SM is capable of using Application Network for negotiations with

counterpart to some limited extents. Main decision mechanism of an SM is designed as a state machine. The entire set of states can be divided in two subsets: active states, where unit is performing its operations actively and inactive states, where unit is working as a hot reserve.

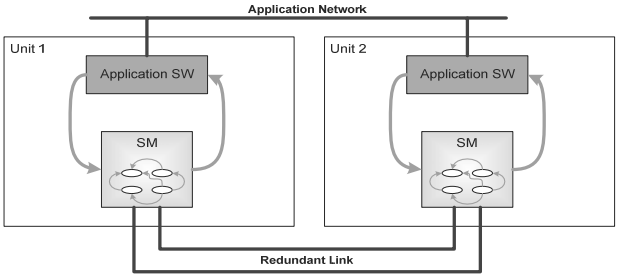

Figure 1. Structure of the redundant system

There are the following active states:

1) SINGLE – unit is active and considers the second unit as inept to become active (either due to a known failure of the second unit or absence of information about the second unit).

2) MASTER – unit is active and considers the second unit as capable to become active.

And the following inactive states:

1) SLAVE – unit is inactive but capable to take over the active state.

2) FAIL – unit is inactive and inept to become active due to detected failure(s).

3) CONNECTING – unit is inactive since its role is being negotiated with the counterpart.

SMs states and related transitions are shown in Figure 2. Transitions are triggered by the state vector which includes the following parameters (all of the parameters are boolean values): APP – state of the main application, RLS – state of the redundant link between two SMs, RSM – state of the remote SM (this byte equals 1 if the remote SM in states MASTER or SINGLE and 0 otherwise) and RL – state of the reserved link.

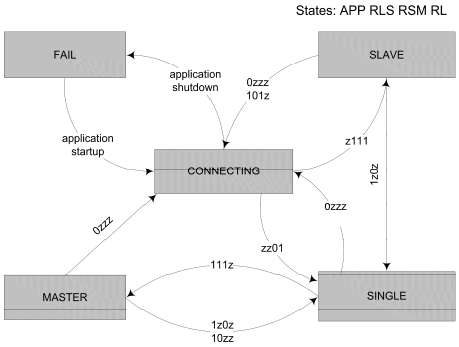

Figure 2. States and transitions of SM. Vector parameters are: APP- application state, RLS – redundant link state, RSM – state of the remote SM, RL – state of the reserved link, 'z' means 'any value'.

According to current state SM controls the main service part of units software. For a pair of SMs there were identified the sate combinations which are valid and invalid in terms of operation consistency. E.g. cases when one SM is active (MASTER) and another is inactive (SLAVE) is valid; while any case when both SMs are active is invalid. Detailed list of possible state combinations is presented in Table 1.

TABLE I. STATE COMBINATIONS FOR TWO SMS

	FAIL	CONN.	M.	SLAVE	SINGLE
FAIL	T	T	T	T	V
CONN.	T	T	T	T	V
MASTER	T	T	F	V	F
SLAVE	T	T	V	T	T
SINGLE	T	T	F	T	F

V – Valid state combination, F – Forbidden state combination, T – acceptable as transitional only.

SM analyzes input information every 300 ms and makes the decision either to change its state or not. Every decided state transition is negotiated with the counterpart through redundant Ethernet link to assure consistent behavior of the units. When this link is unavailable the negotiation is handled over the application network link. During normal operation one of the units stands in the MASTER state while another one in the SLAVE state. This work is particularly focused on timing characteristics of two scenarios:

1) Detected failure of active unit (MASTER). This failure is causing active unit to switch to inactive mode (FAIL) and its counterpart (SLAVE) to become active (SINGLE). Transitions happening in this situatioin are shown in Figure 3a (This scenario is called MasterFailure).

2) Persistent failure of communication channel between units. This failure is causing active unit (MASTER) to switch to (SINGLE) and its counterpart (SLAVE) to switch to negotiation mode (CONNECTING) in order to be ready for re-synchronization in case of link recovery, see transitions in Figure 3b. (This scenario is called LinkFailure).

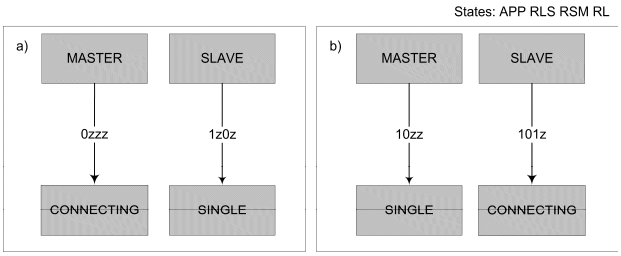

Figure 3. Transitions, which occure during failure conditions. a) MasterFailure, b) LinkFailure

For every situation two different time characteristics where analyzed:

1) Distribution of the transition time in case of ideal communication.

2) System stability in case of lost packets. System behavior havily relies on network communication between two SMs and estimation of the behavior variations in case of problems

in communication is very important. Two particular questions are the subject of our study:

a) Can the system enter invalid state combination? (When there is no any MASTER or SINGLE unit, which means that system can not perform its functions)

b) How is the distribution of transition time affected by channel instability?

Another topic of this study is the investigation of the differences between timeout-based and event-based behavior for state switches. Here the timeout-based algorithm – is the default algorithm where decision about state transition is made every 300 ms, while the event-based algorithm – SM algorithm where the decision about transition is made exactly at the time when the change in external environment is detected. We were particularly interested in difference between these two algorithms for the time transition in case of ideal communication.

IV. Simulation Approach

Simulation models can be classified into three subsets with respect to how they treat system state as a function of time [4].

- *Continuous time* – system state changes continuously with time, differential equations are usually used to describe such models.
- *Discrete time* – system state is observed only at selected moments in time which are usually equally separated.
- *Continuous time-discrete event* – system behavior is defined by the sequence of event times which need not be equally separated. System state may vary continuously between events but is observed only at event times.

Since the behavior of our system is fully defined by external asynchronous events the latter approach is the most suitable one to build the model. Among available tools we have chosen discrete-time event based simulator JavaSim [9]. The choice in favor of this library was determined by its simplicity, wide provided functionality and extensibility. Analysis of simulation results includes processing of high volume of statistical information for system timing properties. To simplify this process we extended the simulator with JAIDA [10] library to provide an interface to Java Analysis Studio [11] which is a powerful tool for data analysis.

Three main processes were defined during simulation: two identical processes simulated SM behavior and one process was a control process responsible for managing the external environment, analysis of SM states, simulation of failures and gathering of statistical information. In addition to the above entities separate auxiliary processes were defined to simulate the external application software and redundant link. All processes were implemented as Java classes which are extending SimulationEntity class from JavaSim library.

The whole simulation was handled in a loop which included the following steps:

- *Put both SMs in FAIL state.* Before startup one of the SMs is delayed for some randomly generated timeout.
- *Wait while one of the SMs come into MASTER state and another one into SLAVE state* (This is an expected behavior during normal operation). During operation SMs send a lot of messages through a communication link. For simulation of network delays all these messages were delayed according to the Gaussian distribution law.
- *Simulate failure condition.* To introduce stochastic behavior failures were simulated at arbitarary time with uniform distribution law.
- *Wait while both of the SMs come into expected states, depending on the failure type. (See Figure 3).*
- *Save time for transition, reset the processes and repeat the loop*

The loop was repeated 5000 times. The simulation was done on 2.2 Ghz Core2Duo PC running under Windows XP operating system.

V. Simulation Results

For both failure conditions (MasterFail and LinkFail) we made five different experiments:

1) Experiment with ideal communication between SMs with timeout-based algorithm for state switches.

2) Experiments for analysis of system behavior with timeout-based algorithm for state switches in case of network problems. To simulate communication instability we introduced possibility to completely lose network packet. Three experiments have been done with such probability equals to 1%, 2% and 3% appropriately.

3) Experiment with ideal communication between SMs with event-based algorithm for state switches.

The comparison of mean switching times as well as it's RMS for all of the above experiments is presented in Table 2. It can be concluded that: 1) Communication problems affect system behavior only in MasterFail situation (RMS for transition time increases in almost 2 times), while for LinkFail the effect of such instability is less important (RM increases only at 25 %); 2) Event based algorithm gives significant advance in transition time and RMS in both cases. Figure 4 shows the difference in transition time distribution for both failures in case of problems in communication between two SMs. It can be seen that in the case of LinkFail communication instability only leads to slightly broader distribution of transition time without any significant changes in its shape.

However for MasterFail such instability brings completely new behavior in state transition which leads to cases with longer transition time (several times longer). It immediately leads to significant increase of RMS characteristic (indicated on Table 2) and overall system behavior in this case gets less predictable. Also communication instability in MasterFail situation can bring the whole system to the forbidden state (See the bin near 3000ms in Figure 4.) In this situation both units

reside in SLAVE state and none of them is able to perform main system functions. Probability of such situation is estimated as 0.5%. Our model was able to capture this system behavior because it implemented all the switching logic.

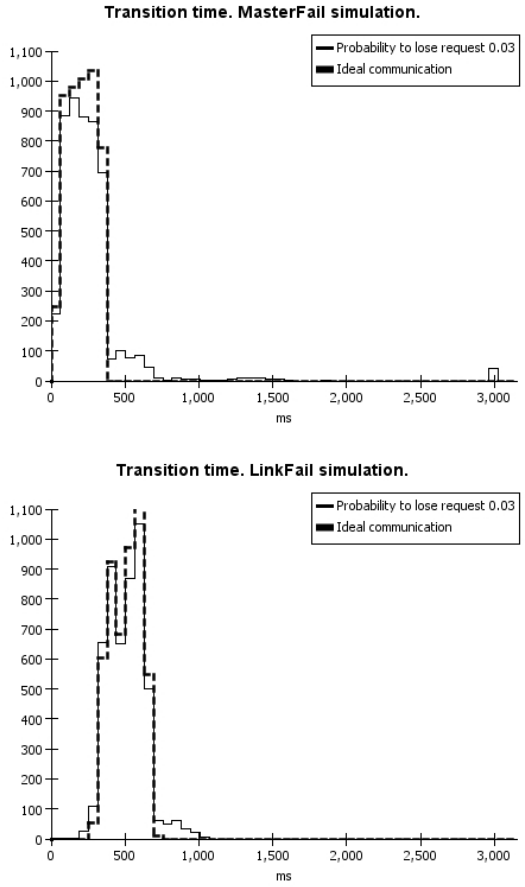

Figure 4. Distribution of transition time in case of ideal communication and lost packets

TABLE II. COMPARISON OF MEAN TIMES

	MasterFail		LinkFail	
	Mean Time (ms)	**RMS (ms)**	**Mean Time (ms)**	**RMS (ms)**
Timeout algorithm. Ideal Communication	207	91	506	101
Timeout algorithm. Prob. 1%	222	143	505	109
Timeout algorithm. Prob. 2%	238	175	506	118
Timeout algorithm. Prob. 3%	245	174	510	126
Event algorithm. Ideal communication.	70	14	372	38

Figure 5 shows the difference in timeout-based and event-based algorithms. It can be seen that distribution of a transition time in the case of event-based algorithm is much narrower in both cases. Such difference in the behavior can be explained by the rather large default timeout for timeout-based algorithm – 300ms, while other system times are much smaller 50 – 100ms. However simple decrease of this timeout looks rather dangerous and switching to the event-based algorithm is the most promising way for improvement of system behavior.

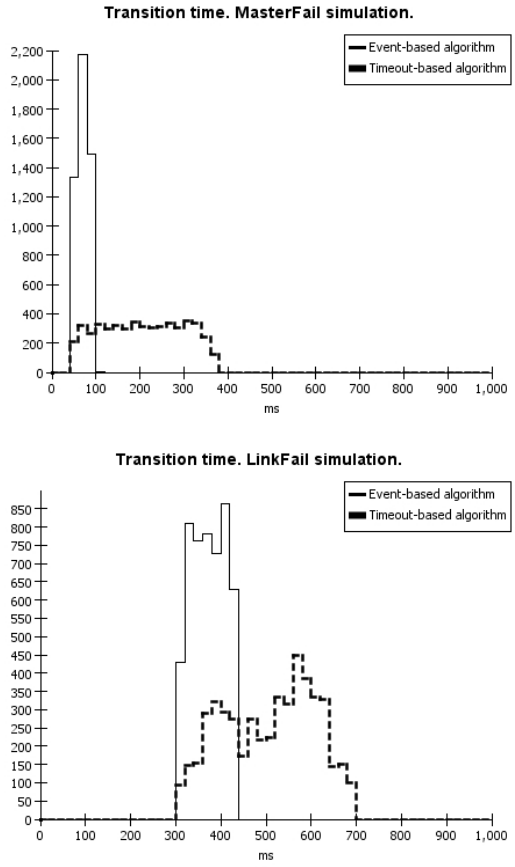

Figure 5. Difference in transition time distribution in case of timeout-based and event-based DL

VI. CONCLUSION

Experience gained during this project can be summarized as follows:

- Usage of simulation techniques allows estimation of relative timing characteristics of a redundant system in different conditions
- The behavior of a real redundant system is rather complicated and JavaSim library allowed to reflect this complexity in the model
- Minor changes in the algorithm responsible for state transitions may lead to significant variations in time distribution and overall system behavior. Such changes in real application need to be reflected in the model and one of the approaches here – incorporate appropriate program blocks directly.

VII. FUTURE WORK

The presented study shows time and performance characteristics of the redundant system under different conditions. However exact quantitative evaluation of timing characteristics requires validation of the model on adequacy with real system that remains a subject for future work. Another interesting topic is the further investigation of the differences in behavior of timeout-based and event-based algorithms under different failure conditions and especially the data flows which are generated through communication link.

REFERENCES

[1] S. Balsamo, M. Marzolla, "A Simulation-Based Approach to Software Performance Modelling", Proceedings of ESEC/FSE 2003

[2] Y. Chen, R. Mateer, "Performance Simulation of a Dependable Distributied System", SIMULATION, Vol. 77, No. 5-6, 230-237, 2001

[3] O-J. Dahl, B. Myhrhaug, K. Nygaard, "SIMULA Common Base Language", Norwegian Computing Centre, 1970.

[4] I. Mitrani, "Simulation Techniques for Discrete Event Systems", Cambridge University Press, Cambridge, 1982

[5] G. D. Parrington et al, "The Design and Implementation of Arjuna", Broadcast Project Technical Report, October 1994.

[6] R.Riter, "Modeling and Testing a Critical Fault-Tolerant Multi-Process System", Proceedings of the Twenty-Fifth International Symposium on Fault-Tolerant Computing, 1995

[7] M. Strickland, D. Palumbo., "Fault tolerant system performance modeling", NASA Technical Report, 1988

[8] F. Safaei, A. Khonsari, M. Fathy, N. Alzeidi, M.Ould-Khaoua, "Performance Modeling of Fault-Tolerant Circuit-Switched Communication Networks", Parallel Computing in Electrical Engineering, PAR ELEC ,2006.

[9] JavaSim User's Guide. Department of Computing Science, Computing Laboratory, The University, Newcastle upon Tyne, 1999.

[10] JAIDA http://java.freehep.org/jaida/.

[11] Java Analysis Studio http://jas.freehep.org/jas3/.