Proceedings of the 17th World Congress
The International Federation of Automatic Control
Seoul, Korea, July 6-11, 2008

A Simulation Study of a Reconfigurable Database

James M. Metzler * **N.Eva Wu** **

* *Air Force Research Laboratory, Information Directorate, Rome, NY*
** *Department of Electrical and Computer Engineering, Binghamton University*

Abstract: The effect of supervisory control on a redundant database unit representing a command and control (C2) system that supports air operations is investigated through simulation. Several supervisory control policies are considered. They authorize restoration and/or routing upon the failure of a server in the system. The performance of the modeled system under these policies is evaluated based on the measures of system mean-time-to-failure (MTTF), steady-state availability, expected response time, and overhead. The system is modeled as a discrete event system using a simulation tool. In addition, a system update process is implemented to ensure the currency of the information contained in the database unit.

Keywords: Discrete event systems; supervisory control; fault tolerant/reliable systems; reliability and safety analysis; fault tolerant control for networked systems.

1. INTRODUCTION

The focus of this work is on studying the effect of supervisory control (Cassandras and Lafortune [1999]) on a number of important measures that pertain to C2 system performance with a redundant architecture first proposed and studied in Wu et al. [2005] where a database is partitioned in a way that allows multiple servers to process customers in parallel with information backed-up throughout the system. The proposed architecture is shown in Fig. 1, where the data are partitioned into the sets A, B, and C. Customers entering the system are routed based on the type of information they require.

To enhance fault-tolerance in the face of crash and site failure, and improve the responsiveness to queries, supervisory control is applied to the partitioned database unit. The response time and availability can be potentially improved by strategically routing customers based on the state of the servers. Supervisory control introduces policies that allow the restoration of lost data and/or the routing of queries based on the state of the information in the system.

The objectives of this work are to qualitatively analyze the performance of the partitioned database unit under supervisory control and varying structural parameters, in terms of MTTF, availability, response time, and overhead, based on the results obtained through discrete event system (DES) simulation.

The paper is organized as follows. Section 2 describes the database system, its operating policies, and its model. Section 2 also summarizes the analytic results obtained in Wu et al. [2005], and points out limitations of the analytic study. Section 3 presents the simulation model, and comments on the results of cross-verification between the analytic and simulation models. Section 4 presents main results of database analysis via simulation, and highlights the benefits of the simulation study that extended the scope of the earlier analytic study by the authors in three areas: removal of the finite query population restriction, extension of the event life distributions beyond exponential, and introduction of the deterministic system update process. Section 5 concludes the paper.

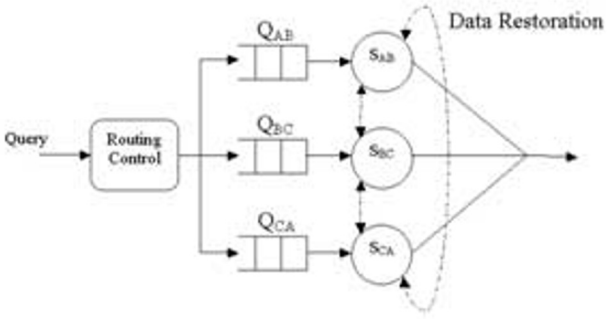

Fig. 1. Partitioned database unit.

2. BACKGROUND

The presentation in this section is drawn from Wu et al. [2005] to recapitulate aspects of modeling, control, and performance analysis of the database unit shown in Fig. 2 (Wu et al. [2005]) to identify the limitations of the analytical method employed there, and to briefly describe the extensions made in this paper.

2.1 System Model

The database unit to be studied is taken from Wu et al. [2005], which is intended to be representative of a C2 sup-

⋆ This work was supported in part by the U.S. Air Force Research Laboratory Contract F30602-20-C-0225, and in part by the U.S. Air Force Office of Scientific Research under Grant FA9550-06-0456.

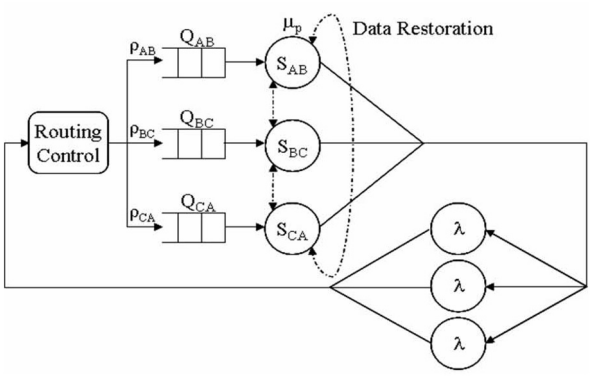

Fig. 2. Closed queuing network model.

porting system. A closed queuing network representation of the unit is shown in Fig. 2. The information contained within the system is partitioned into sets A, B, and C and placed on three servers that exist in parallel to answer three classes of queries A, B, and C, respectively. Server S_{AB} contains class A primary data and class B secondary data. Server S_{BC} contains class B primary data and class C secondary data. Server S_{CA} contains class C primary data and class A secondary data. When a server fails, both its primary and secondary data are lost. A server is "down" when either class of data are lost, and a system failure occurs when two servers are down concurrently.

The queues preceding S_{AB}, S_{BC}, and S_{CA} are named Q_{AB}, Q_{BC}, and Q_{CA}, respectively. They are of sufficient size that no queries are lost or blocked and operate on a first come, first serve (FCFS) basis.

The delay elements, each labeled λ indicating an average delay $1/\lambda$, are representative of the response times incurred at other nodes of the C2 supporting system which are not modeled here. The three elements imply that there are only three customers in the system at any given time, a limitation of the Markov model in Wu et al. [2005] that is to be removed in this study. Upon completion of processing at a server, a customer returns to one of its delay elements, and after a period of time, re-enters the system. Each time a customer enters the system it is equally likely to require information of class A, B, or C. Therefore, under normal operating conditions, the routing probabilities ρ_{AB}, ρ_{BC}, and ρ_{CA}, where $\rho_{AB} + \rho_{BC} + \rho_{CA} = 1$, are given the same values.

The model is built with the premise that event lifetime distributions have been established for all the processes involved. The delay process, or equivalently, query generation, has an exponential distribution $\exp(\lambda) \equiv 1 - e^{-\lambda t}$, where λ is the rate and $1/\lambda$ is the mean. The same is true for the process of service completion ($\exp(\mu_p)$), the process of server failure ($exp(\nu)$), the process of data restoration ($exp(\gamma)$), and the process of unit overhaul ($exp(\omega)$), when the entire unit is repaired due to system failure. All processes are independent. Note that all rates and therefore means are relative and carry the units time^{-1} and time, respectively.

u_1	u_2	S_{AB}	ρ_{AB}	ρ_{BC}	ρ_{CA}
0	1	2	0	1/2	1/2
1	0	2 (1)	1/3 (1/3)	1/3 (1/3)	1/3 (1/3)
1	1	2 (1)	0 (1/6)	2/3 (1/6)	1/3 (2/3)

Table 1. Examples of routing probabilities.

2.2 Control Policies

To maximize the efficiency of the database unit under server failures, two supervisory control inputs are introduced based on the state information of the system. These control actions alter the transition rates of the system when data loss occurs in a server for the purpose of improving performance. The necessary state information is the current state of the servers. Define server state S_{AB}, S_{BC}, $S_{CA} \in$ 0,1,2 where "2" $\equiv$ both the primary data and the secondary data are lost in a server, "1" $\equiv$ the primary data have been restored but the secondary data have not yet been restored, and "0" $\equiv$ the primary data and secondary data in the server are both intact. A server is failed, or in the down state, when either class of data are lost, and is up when both the primary and secondary data are intact.

Two supervisory control inputs, u_1 and u_2, govern restoration and routing, respectively. The control input u_1 allows an intact server to halt its current process and restore lost data in a failed server, and input u_2 adjusts the routing probability of customers based on the state of the servers. Because of the symmetry of the model, the control inputs and policies may be sufficiently described by the case of only one failed server S_{AB}, where the remaining two servers must be intact for the system to be up. The control inputs may be summarized as follows.

$$u_1 = \begin{cases} 0,\ S_{AB} = 2, S_{BC} \text{ serves}, \\ \quad S_{CA} \text{ serves (no restoration)} \\ 1, \begin{cases} S_{AB} = 2, S_{BC} \text{ serves}, \\ \quad S_{CA} \text{ restores class A data} \\ S_{AB} = 1, S_{CA} \text{ serves}, \\ \quad S_{BC} \text{ restores class B data} \end{cases} \end{cases}, \tag{1}$$

$$u_2 = \begin{cases} 0,\ S_{AB} = 2, \\ \quad \rho_{AB} = \rho_{BC} = \rho_{CA} = 1/3 \\ 1, \begin{cases} S_{AB} = 2, \rho_{AB}(2, u_1), \\ \quad \rho_{BC}(2, u_1), \rho_{CA}(2, u_1) \\ S_{AB} = 1, \rho_{AB}(1, u_1), \\ \quad \rho_{BC}(1, u_1), \rho_{CA}(1, u_1) \end{cases} \end{cases}, \tag{2}$$

Recall the routing probabilities ρ_{AB}, ρ_{BC}, and ρ_{CA}. Under supervisory control, these probabilities are dependent not only on the routing control input u_2 and the state of the servers, but also on the restoration control input u_1. Table 1 shows three sets of routing probabilities.

The composition of u_1 and u_2 gives rise to four different control policies. The case of (u_1 and u_2) = (0, 0) corresponds to the case of a single point failure, and is therefore not considered in the performance analysis. The control policies in the other three cases are named

$$\begin{array}{l} \text{Policy 1: } (u_1, u_2) = (0, 1) \text{ when a server is down,} \\ \text{Policy 2: } (u_1, u_2) = (1, 0) \text{ when a server is down,} \\ \text{Policy 3: } (u_1, u_2) = (1, 1) \text{ when a server is down.} \end{array} \tag{3}$$

Note that policy 2 does not permit routing, whereas policy 1 does not permit restoring. A special consideration with

the case $u_1 = 0$ is the rerouting of the customers who have arrived at a server before the server fails to the delay elements.

2.3 Analytic Results

The above system was first studied in Wu et al. [2005] where the database unit was modeled as a closed Markov queuing network. The performance of the system under supervisory control was evaluated based on the performance measures of mean time to failure (MTTF) of the system, steady-state availability, expected response time, and service overhead.

System failure is defined as the loss of a second server before the restoration of a first failed server is completed. MTTF is a measure of the life of the system. The MTTF was found to significantly improve under policies 2 and 3, apparently attributed to the introduction of restoration. Availability, a measure of the percentage of time the system is available to serve customers (i.e., not in a system failure state), also improved under these policies.

The expected response time, defined as the length of time a query spends in the upper portion of the system shown in Fig. 2, also benefited from policies 2 and 3 for a sufficiently high restoration rate γ. However, at low values of γ, the system profited from not having to devote a majority of its resources to restoring failed servers but simply servicing customers with its two intact nodes under policy 1. Policy 3 showed slightly better performance than policy 2; this advantage is expected to improve with the addition of customers to the system.

Overhead is defined as the cost incurred by the system for self-preservation. It is calculated as the ratio of time invested in restoring the system to its overall busy time, and does not include the overhaul process. As the failure rate ν increased, policies 2 and 3 became expensive, and surpassed the overhead associated with policy 1.

2.4 Limitations

The Markov model of the database unit presented in Wu et al. [2005] suffered many limitations. The complexity of the model was restricted by the need for a manageable number of states and the exponential event lifetime distributions. A linear increase in either the number of customers or the number of servers allowed in the system causes an exponential growth in the number of states, whereas any non-exponential event lifetime distribution destroys the memoryless properties essential to a Markov model, although many of the processes under consideration are most adequately described by non-exponential distributions.

A majority of database units require a periodic update to the information contained within the system to maintain the currency of the data. As seen in Fig. 2, Wu et al. [2005] omitted the updating process to avoid the explosion of the size of the state space due to the additional class of customers and the non-Poisson nature of the update requests.

Modeling the database unit by means of a simulation tool enables us to remove the limit on the number of customers, diversify the event lifetime distributions, and include the update process.

3. SIMULATION MODEL

3.1 Discrete Event System Simulation

Modeling of systems in which the state variable changes only at a discrete set of points in time is known as discrete event system (DES) simulation (Banks et al. [2001]). Simulation implies solving for the system variables through numerical rather than analytic methods. Observations of the variables collected throughout the history of the model are stored and processed to evaluate system performance measures. A major component in a discrete event system simulation is the future event list which contains the notices for all future events scheduled to occur. For each event that occurs, beginning with the first event of the simulation, durations are either computed or drawn from a statistical distribution, and the end-event is added to the future event list. The advantage of this method is that every time instance need not be evaluated, allowing the simulation to omit time intervals where the state of the system does not change.

The simulation package used in this study is Arena® Professional Edition (Rockwell [2003]). Arena® utilizes an object-based design for graphical model development (Banks et al. [2001]). Objects called modules are used to model system logic and physical components such as servers and queues. In addition, Arena® provides methods for statistical distributions, failure modeling, statistics collection, and process analysis. Arena® allows any number of independent replications to be run for a simulation, with the replication terminating upon a user defined condition. System as well as user defined statistics are collected for each replication and evaluated for the entire simulation.

In this study, the effect of supervisory control is evaluated for MTTF, system availability, expected response time, and overhead, as in Wu et al. [2005]. MTTF is evaluated using the method of independent replications, whereas system availability, expected response time, and overhead are evaluated using the method of regenerative simulations (Law and Kelton [2000]). All calculated performance measures are obtained from simulation with 100 replications unless otherwise noted. The Process Analyzer is a tool in Arena® that allows a series of simulations (scenarios) with varying system parameters (controls) to be run automatically in succession and displays the chosen system outputs (responses). This feature proved extremely useful in the evaluation of multiple performance measures as a function of varying system parameters.

3.2 Model Verification

The model shown in Fig. 2 and described in Section 2.1 was simulated via Arena® under the supervisory control policies presented in Section 2.2. The results from each modeling method were compared for verification purposes. The MTTF and availability corresponded between the two simulation methods, as did the system overhead. However, the expected response time calculated from simulation was significantly lower due to the fact that the simulation

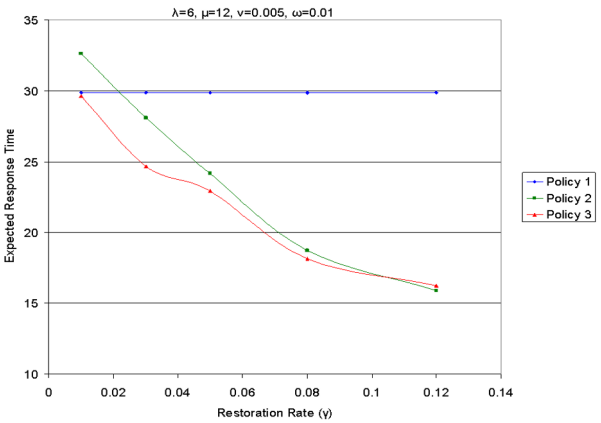

Fig. 3. Expected response time of open queuing network versus restoration rate.

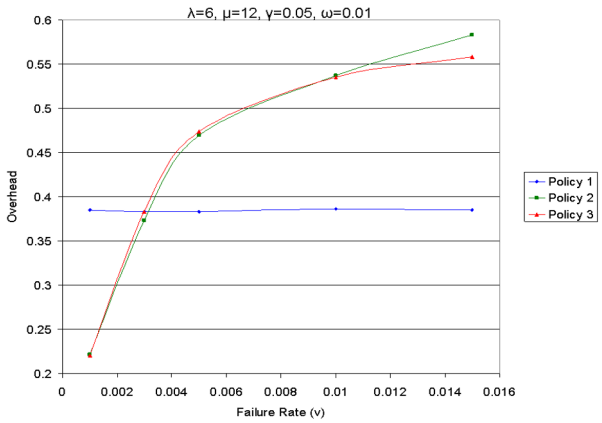

Fig. 4. Overhead of open queuing network versus failure rate.

calculation is not a steady-state measure. Response time statistics are only able to be collected for customers that enter and exit the system. Customers that are trapped outside a failed system do not contribute to the calculated response time. Therefore, only customers who are in a server when the system fails will suffer the delay of the overhaul process. The limited number of customers makes this delay insignificant, resulting in lower response times for the simulation model. This deficiency no longer exists when an open queuing system is introduced in the next section.

With a simulation model constructed and verified in Arena®, the limitations on the number of customers, the event lifetime distributions, and the update process discussed previously may now be removed, as presented in the following section.

4. ANALYSIS VIA SIMULATION

4.1 Open Queuing Network

A fixed number of customers severely limits our ability to fully observe the behavior of the system, but it is necessary to model the database analytically. Simulation modeling removes this restriction, and more realistic measures of system performance are provided.

The system is modeled in Arena® as the open queuing network shown in Fig. 1 where customers enter the system with an exponentially distributed inter-arrival time exp(λ). The customers are removed from the system upon service completion.

The MTTF and availability of the open queuing network are statistically indistinguishable from that obtained in the closed simulation model. The expected response time however does increase significantly now that customers are freely allowed to enter and accumulate in the system. The benefits of routing control are apparent, as shown in Fig. 3. Policy 3 realizes a lower response time with customers routed strategically by supervisory control input u_2. However, as failed servers are restored at a higher rate, the advantage of policy 3 decreases. Routing control becomes less beneficial because queue lengths do not grow as large at failed servers.

Overhead is less sensitive to the type of queuing network. The values shown in Fig. 4 correspond to those obtained analytically in the closed queuing network. The overhead of policy 1 is unaffected by an increase in the rate of failure because the system is never required to restore itself. Restoration is beneficial at low failure rates, however, beyond some threshold, it becomes expensive to the system.

4.2 Generalized Distributions

Simulation of the database unit permits the removal of the limitation to exponentially distributed event lifetimes. The exponential distribution has a constant failure rate and therefore is unfit for many event lifetimes. For example, a component with a failure process described by an exponential distribution has a constant failure rate and is therefore probabilistically always as good as new, regardless of its age (Trivedi [1982]), while in reality, most components are more likely to fail as they age. For the distributions described below, parameters such as the shape parameter α and the scaling parameter β, are chosen to provide a mean equivalent to that of the exponential distribution previously used.

The arrival process lifetime remains exponentially distributed (exp(λ)). Often systems undergo certain "busy" periods, but for the purposes of this study, customers will arrive at a constant rate. The gamma distribution is often used to represent the time required to complete a task (Kelton et al. [2004]) and is therefore used to describe the process of service completion (gamma($1/\alpha\beta = \mu$)). A component lifetime is better described by a distribution that reflects the age of the component. The Weibull distribution has a rate that varies with time, and is used to describe the failure process (Weibull($\alpha, 1/\beta = \nu$)). For $\alpha > 1$, the probability of failure increases with age. Triangular distributions with mode m are used for the restoration process (tria($1/m = \gamma$)) and the overhaul process (tria($1/m = \omega$)) because these event lifetimes are relatively deterministic. The time required to restore a known amount of data should not vary significantly.

With the failure event lifetime now dependent on time, the MTTF and availability of the system, as shown in Table 2, is expected to decrease. This is true for the policies

	MTTF		Availability	
Policy	Expo.	Gen.	Expo.	Gen.
1	169	172	.61	.62
2	317	244	.71	.69
3	317	265	.71	.69

Table 2. Comparison of exponential (Expo.) and generalized (Gen.) distributions.

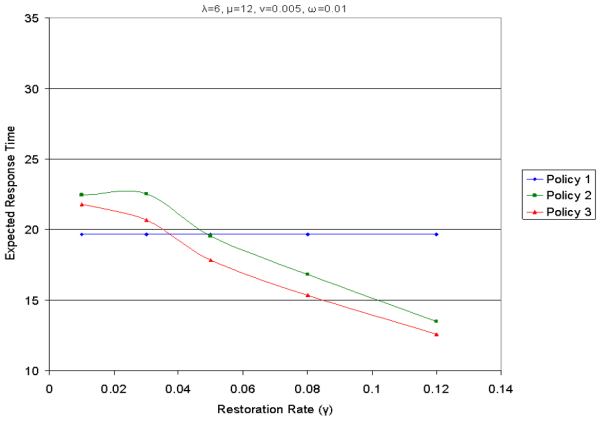

Fig. 5. Expected response time versus restoration rate using generalized distributions.

involving restoration; however it is not the case for policy 1, which is unaffected. For policies allowing restoration, the MTTF is dependent on a failed server recovering before another failure occurs. A time dependent failure rate causes failures to occur more closely (assuming the lifetimes begin concurrently), increasing the likelihood of overlapping failures, and reducing the MTTF. Policy 1 is unaffected because the failure of a second server always results in a system failure. The MTTF obtained for policy 1 under each type of distribution is statistically equal because the mean values of both distributions are equal.

Expected response time decreases under the generalized event lifetime distributions, as shown in Fig. 5, however the values follow the same trend as those shown in Fig. 3.

Overhead is shown in Fig. 6 for the generalized distributions. Comparison with Fig. 4 shows a decrease over that observed from the use of exponential distributions. Using the Weibull distribution, servers are more likely to fail as they age, resulting in less time devoted to restoration in the early stages of their lifetime. As the system ages, more simultaneous failures are likely to occur. When failures occur close together, the time a server spends restoring another server decreases because the overhaul process takes over to restore the system.

4.3 System Update Process

In order to keep the information stored in the database unit current and useful, the system must be periodically updated. While an overhaul of the system will update the data, system failure should occur infrequently, resulting in the need for a system update at a steady interval in the form of an update entity.

Update entities arrive at a deterministic rate, with the inter-arrival time being the acceptable age of the information in the system. An update entity is sent to each server and becomes the first in-line at the queue. Both the primary data and secondary data for each server are contained on the update entity. It follows that the time to process an update entity is described by twice the restoration process distribution. A server is unavailable while processing an update entity. An update entity arriving at a failed server will restore that server, a significant benefit to policy 1, as well as the remaining policies, under which servers no longer have to restore a failed server that is processing an update entity. It is important to note that an update to a server does not reset the lifetime of the component.

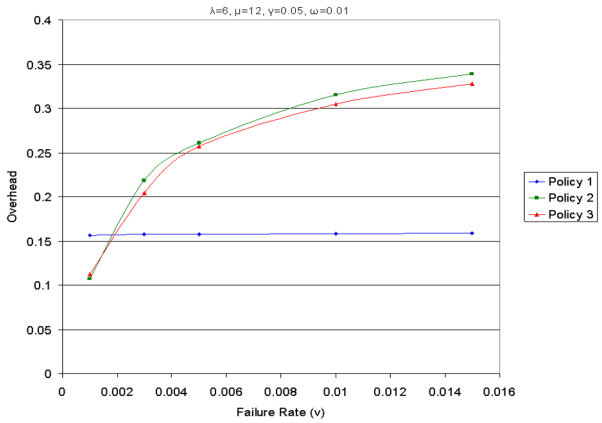

Fig. 6. Overhead versus failure rate using generalized distributions.

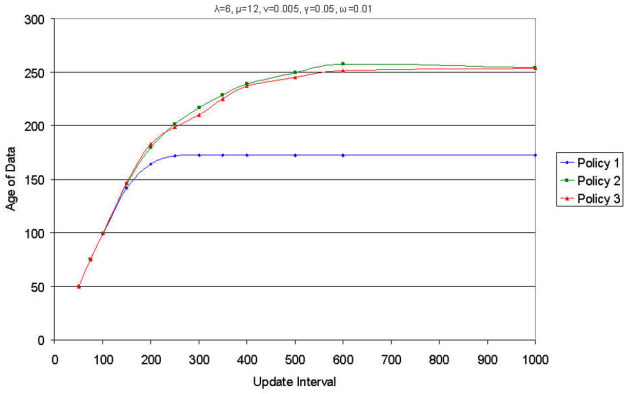

Fig. 7. Age of data versus the update interval.

The update interval only partially determines the age of the data in the unit. The data, on average, will only be as old as the minimum of the update inter-arrival time or the MTTF, which will result in the system being overhauled with current data. The age of the data contained within the database unit is shown in Fig. 7 versus the update interval. At low intervals, the data is as old as the inter-arrival time. As the interval increases, the age of the data reaches a maximum of the MTTF of each policy given in Table 2. At these high intervals, the data is being updated only by the system overhaul process.

The update interval has a significant impact on the availability of the system, as shown in Fig. 8. As the update interval increases, the system is more available to answer queries. Availability increases until the update interval is

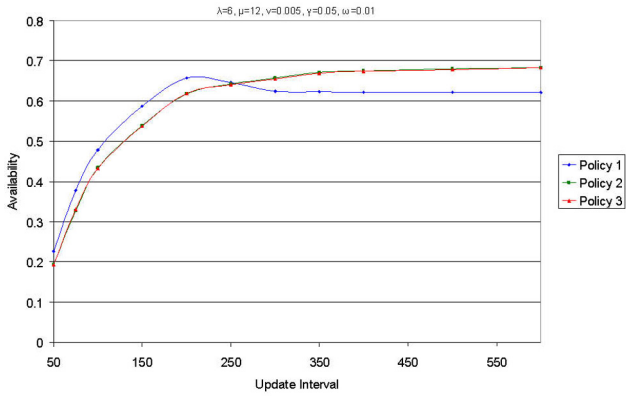

Fig. 8. Availability versus the update interval.

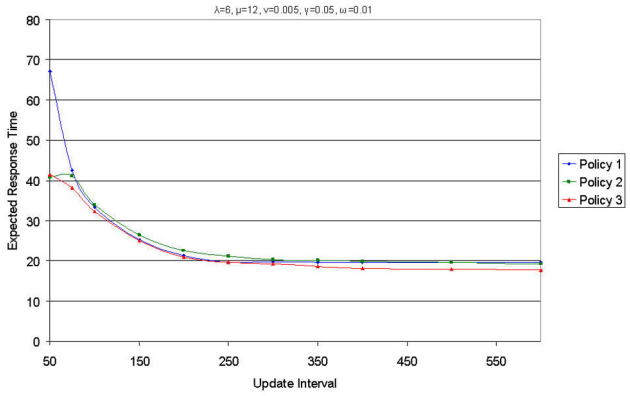

Fig. 9. Expected response time versus the update interval.

so large it neither improves MTTF nor hinders processing queries, and it reaches the steady-state value given in Table 2. Policy 1 enjoys a higher availability at low update intervals because failed servers are being restored by the update process, which, on average, is 2.5 times faster than the overhaul process. Beyond an update interval equal to its MTTF, policy 1 no longer benefits from restoration provided by the update process and its availability diminishes slightly.

Frequent system updates tax the resources of the database unit causing an increase in the expected response time, as shown in Fig. 9. As the update interval increases, the customer service interruption caused becomes negligible, and the expected response time reaches the values shown in Fig. 5. Policy 1 experiences a substantial increase in expected response time at a low update interval because many times only two servers are available to process queries. The impact on expected response time is more severe when those servers are interrupted to process update entities.

Updating the system is considered time invested in maintaining the database unit. Therefore, the expression for overhead θ in Wu et al. [2005] is modified to

$$\theta \equiv \frac{\Pr[M|N]}{\Pr[P|N]} \tag{4}$$

where $M \equiv \mathrm{S}_{AB}$ restores or fails or updates, $N \equiv$ unit is not failed, and $P \equiv \mathrm{S}_{AB}$ restores or fails or updates or serves.

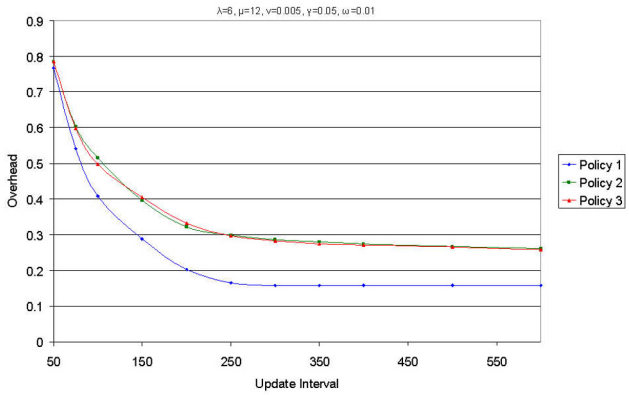

Fig. 10. Overhead versus the update interval.

Overhead improves as the update inter-arrival time increases, as shown in Fig. 10. Restoration of failed servers by an update entity is not assessed as overhead for policy 1 because the database unit is failed during this time. Therefore, overhead is significantly lower for policy 1.

5. CONCLUSION

The use of discrete event system simulation allows the removal of limitations imposed by having to represent a database unit analytically as a Markov model. A more practical system may be evaluated that includes an open queuing network, dynamic event lifetime distribution rates, and an update process. This paper has modeled and evaluated a database unit representative of a C2 supporting system under several supervisory control policies. The effects of restoration (u_1) and routing (u_2) were assessed based on measures of fault-tolerance and responsiveness. While restoration remains more beneficial than routing, the benefits of routing control are slightly more visible for an unlimited-sized population as compared to the results obtained in Wu et al. [2005]. The addition of an update process, while necessary, weighs heavily on the performance of the system.

REFERENCES

J. Banks, J.S. Carson, B.L. Nelson, and D.M. Nicol. *Discrete-Event System Simulation.* Prentice Hall, 3rd edition, 2001.

C.G. Cassandras and S. Lafortune. *Introduction to Discrete Event Systems.* Kulwer Academic Publishers, Boston, 1999.

Rockwell Software Inc. *Arena Professional Edition, Version 7.01.00.* 2003.

W.D. Kelton, R.P. Sadowski, and D.T. Sturrock. *Simulation with Arena.* McGraw-Hill M.G., 3rd edition, 2004.

A.M. Law and W.D. Kelton. *Simulation Modeling and Analysis.* McGraw-Hill, 3rd edition, 2000.

K.S. Trivedi. *Probability and Statistics with Reliability, Queuing, and Computer Science Applications.* Prentice Hall, 1982.

N. Eva Wu, James M. Metzler, and Mark H. Linderman. Supervisory control of a database unit. *Proc. IEEE Conference on Decision and Control*, pages 7615–7620, 2005.