REPORT DOCUMENTATION PAGE	*Form Approved* **OMB No. 0704-0188**

1. REPORT DATE *(DD-MM-YYYY)* JUL 06	**2. REPORT TYPE** Conference Paper Postprint	**3. DATES COVERED** *(From - To)* 10 – 12 Jul 06

4. TITLE AND SUBTITLE
PERFORMANCE ANALYSIS OF A CONTROLLED DATABASE UNIT SUBJECT TO DECISION ERRORS AND CONTROL DELAYS

5a. CONTRACT NUMBER
In-house

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER
62702F

6. AUTHOR(S)
N. Eva Wu, James M. Metzler, Mark H Linderman

5d. PROJECT NUMBER
558S

5e. TASK NUMBER
IH

5f. WORK UNIT NUMBER
IM

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Air Force Research Laboratory/IFSB
525 Brooks Rd
Rome NY 13441-4505

Binghamton University
State University of New York
Binghamton, NY 13902-6000

8. PERFORMING ORGANIZATION REPORT NUMBER
AFRL-IF-RS-TP-2006-7

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Air Force Research Laboratory/IFSB
525 Brooks Rd
Rome NY 13441-4505

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSORING/MONITORING AGENCY REPORT NUMBER
AFRL-IF-RS-TP-2006-7

12. DISTRIBUTION AVAILABILITY STATEMENT
Approved for public release: distribution is unlimited. PA# 06-324

13. SUPPLEMENTARY NOTES Paper presented at the 8th International Workshop on Discrete Event Systems, July 10-12, 2006, in Ann Arbor, MI. This material is declared a work of the U. S. Government and is not subject to copyright protection in the United States.

14. ABSTRACT

This paper extends the performance analysis of a controlled database unit studied in Wu, Metzler, and Linderman (2005) to include the cases where errors and delays can occur in state-based control actions as a result of uncertainty in the knowledge of the system state. The paper details the way such errors and delays are captured through augmenting the state space in the Markov model of the database unit. State variable feedback is used to activate the process of restoration upon the failure of one of the database servers in the unit. The performance of the database is evaluated in terms of the resulting mean time to unit failure, the steady state availability, the expected response time, and the service overhead of the database unit. All performance measures are examined with respect to the likelihood of decision error and the amount of control action delay.

15. SUBJECT TERMS
discrete event systems, supervisory control, Markov model, queuing network

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON** James M. Metzler
a. REPORT U	**b. ABSTRACT** U	**c. THIS PAGE** U	UL	7	**19b. TELEPONE NUMBER (*Include area code*)**

Performance Analysis of a Controlled Database Unit Subject to Decision Errors and Control Delays

N. Eva Wu, James M. Metzler, and Mark Linderman

Abstract— **This paper extends the performance analysis of a controlled database unit studied in Wu, Metzler, and Linderman (2005) to include the cases where errors and delays can occur in state-based control actions as a result of uncertainty in the knowledge of the system state. The paper details the way such errors and delays are captured through augmenting the state space in the Markov model of the database unit. State variable feedback is used to activate the process of restoration upon the failure of one of the database servers in the unit. The performance of the database is evaluated in terms of the resulting mean time to unit failure, the steady state availability, the expected response time, and the service overhead of the database unit. All performance measures are examined with respect to the likelihood of decision error and the amount of control action delay.**

I. INTRODUCTION

A recent effort to install and test monitoring tools and to increase the level of redundancy in critical subsystems in air operation centers has provided opportunities for vast performance improvement in its command and control supporting systems. Our previous work on a controlled processing unit [1] has demonstrated that reduced response time to service requests and shortened periods of system unavailability, as a result of automated monitoring and control, can raise significantly the probability to attain the desired outcome in an air operation. A more recent study by Wu, Metzler, and Linderman [2] on a database unit as shown in Fig.1 further revealed the benefits of a conscientious design of redundant architecture, and the application of supervisory control, which were measured in terms of the mean time to unit failure, the steady state availability, the expected response time, and the service overhead of the database unit.

To assess the performance in a quantified manner, both the processing unit [1] and the database unit (Fig.1) [2] were given the interpretation of a queuing network [3], [4] with specific sets of operating policies and structural parameters. The control authorities considered included the ability to restore the first failed server, and the ability to route service requests. In order to obtain an analytic model of manageable size for scrutinizing the effects of supervisory control, the queuing network was restricted to the closed type [3], [4]. In addition, all the event lifetime distributions were assumed to be exponential. A simulation study was conducted by James Metzler et al., [5] using Arena [7], [8] with all the above restrictions removed.

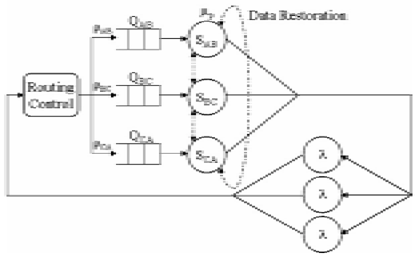

Fig.1 A partitioned database unit

An underlying assumption of the existing study is that the state information in the queuing network model of a given unit is known exactly at any given time. In reality, however, it is not practical to monitor every state variable. As a result, the knowledge on a certain set of states is inferred based on the observables. On the other hand, control actions are likely required at the time of a state transition, such as the occurrence of a component failure, in which case a process of diagnosis must take place before a state-based control action. The time required for diagnosis can be random, and the outcome of the diagnosis can be uncertain. The objectives of this paper, therefore, are to seek for ways to incorporate the effects due to decision errors and control action delays into the Markov model of a queuing network, and to use the model to access the impact of such errors and delays on the performance of the database unit in Fig.1.

The paper is organized as follows. Section II describes the baseline model of the controlled database unit in Fig.1. Section III discusses our approaches to modeling the effects of control delays and decision errors. Section IV presents the results of performance evaluation parameterized with respect to the amount of control action delay and the probability of error.

II. BASELINE MODEL FOR A CONTROLLED DATABASE UNIT

The description of the baseline model, i.e., the model that does not include decision errors and control delays, follows to a large extent that of Wu, Metzler and Linderman [2]. The

This work was supported in part by the U.S. Air Force Research Laboratory Contract F30602-020-C-0225.

N. Eva Wu is with the Department of Electrical and Computer Engineering, Binghamton University, Binghamton, New York, 13902-6000, USA. (telephone: 607-777-4375; fax: 607-777-4464; e-mail: evawu@binghamton.edu).

James M. Metzler and Mark Linderman are with the Air Force Research Laboratory, Rome Research Site, NY 13902-6000, USA. (e-mail: James.Metzler@rl.af.mil, Mark.Linderman@rl.af.mil).

database unit in Fig.1 contains three servers in parallel to answer three classes (A, B, C) of queries for which relevant information can be found in the partitioned sets A, B, C of the database, respectively. Server S_{AB} contains database class *A* as the primary class and database class *B* as the secondary class. Server S_{BC} contains database class *B* as the primary class and database class *C* as the secondary class. Server S_{CA} contains database class *C* as the primary class and database class *A* as the secondary class. The failure of a server implies the loss of two classes of data within the server. A system level failure is declared when two servers fail, in which case one class of data is completely lost. The queues preceding servers S_{AB}, S_{BC}, and S_{CA} are named Q_{AC}, Q_{BC}, and Q_{CA}, respectively. All queues are of sufficient capacity. Service is provided on a FCFS basis at each server.

The three delay elements of average delay $1/\lambda$ imply that there are always three customers present in the unit at any given time. A new query is generated at a delay element upon the completion of the service to a query at one of the servers. The delay elements are intended to be also reflective of the response time to the querying customers by other service nodes in the system that are not explicitly modeled. Any new query is assumed to be equally likely to seek database class A or B or C. Therefore routing probabilities ρ_{AB}, ρ_{BC}, and ρ_{CA} are assigned the same values.

The use of a queuing network model for the database is based on its suitability to involve control actions and to capture their effects on the system performance. The model is built in this study with the premise that event life distributions have been established for the process of query generation $(\exp(\lambda)\equiv 1-e^{-\lambda t})$, the process of service completion $(\exp(\mu))$, the process of server failure $(\exp(\nu))$, the process of data restoration $(\exp(\gamma))$, and the process of unit overhaul $(\exp(\omega))$ when the failed database unit is repaired. All such processes are independent. Standard statistical methods that involve data collection, parameter estimation, and goodness of fit tests exist for identifying event life distributions. Since all event lives are assumed to be exponentially distributed, the database unit can be conveniently modeled as a Markov chain specified by a state space X, an initial state probability mass function (pmf) $\pi_x(0)$, and a set of state transition rates Λ [9,10]. The reader uninterested in the details of model building can advance to the paragraph right above Equation (1).

1) *State space $\mathcal{X}$*

A state name is coded with a *6*-digit number indicative of all queue lengths and server states in the unit. With some abuse of notations, a valid state representation is given by $x=Q_{AB}Q_{BC}Q_{CA}S_{AB}S_{BC}S_{CA}$, where queue length Q_{AB}, Q_{BC}, Q_{CA} $\in$ $\{0, 1, 2, 3\}$ with total length $L \equiv Q_{AB}+Q_{BC}+ Q_{CA} \le 3$, and server state $S_{AB}, S_{BC}, S_{CA} \in \{0, 1, 2\}$. Server state "*2*" ≡ data are lost in both the primary and the secondary classes in a server, "*1*" ≡ the data in the primary class have been restored and data in the secondary class have not been restored, and "*0*" ≡ data in both primary class and secondary class in a server are intact. A server is said to be in the down state if it is either at state "*1*" or at state "*2*". For example, state *110020* indicates that server S_{AB} is up with one customer in its queue, server S_{BC} is down with both classes of data gone and one customer in its queue, and server S_{CA} is up and idle. Note that the queue length includes the customer being served. There are *540* valid states in the baseline system. The total number of states is reduced to *141* when all the states of system level failures are aggregated. A set of alternative state names are assigned from $\mathcal{X} = \{1, 2, \ldots, 141\}$ with *000000* mapped to $x=1$ and the aggregated system failure state mapped to $x=141$.

2) *Initial state pmf* $\{\pi_x(0), x=1,2,\ldots,141\}$

It is assumed that the database unit starts operation from state $x=1$, i.e., the initial state probability is given by vector $\pi(0) = [1\ \ 0\ \ \ldots\ \ 0]$. When overhaul is considered at the occurrence of a system level failure, all customers are flushed out to the delay elements. Once the database unit is renewed and ready for operation again, it starts at the same initial state $x=1$, and a renewal process [10] is formed.

3) *Set of state transition functions* $p_{i,j}(t)$

Events that trigger the transitions and the corresponding transition rates are given as follows. A newly generated query enters one of the servers with rate $(3-L)\times\lambda/3$. A query is answered at a server with rate μ. A complete data loss occurs at a server with rate ν. Data in the primary data class of a server are restored with rate $\gamma_p u_1$, and data in the secondary data class of a server are restored with rate γ_s, where u_1 authorizes whether to restore the lost data for the primary class. Finally, the failed database unit is renewed with rate ωu_3 where u_3 decides whether to repair the failed system.

Let $X \in \mathcal{X}$ denote the random state variable at time t. The set of state transition functions

$$p_{i,j}(t) \equiv P[X(t)=j\,|\,X(0)=i],\ i,j=1,2,\cdots,141 \tag{1}$$

for the continuous-time Markov chain can be solved from the forward Chapman-Kolmogorov equation [7]

$$\dot{P}(t)=P(t)Q(u_1,u_3),\ P(0)=I,\ P(t)=[p_{i,j}(t)]\ , \tag{2}$$

where $Q(u_1,u_3)$ is called an infinitesimal generator or a rate transition matrix whose $(i,j)^{th}$ entry is given by the rate associated with the transition from current state *i* to next state *j* in the rate transition table. State probability mass function at time *t*

$$\pi(t)=[\pi_1(t)\quad \pi_2(t)\quad \cdots\quad \pi_{141}(t)],\ t\ge 0 \tag{3}$$

is computed by

$$\pi(t)=\pi(0)P(t)\,. \tag{4}$$

At this point a baseline Markov model for the database unit of Fig.1 has been established. Since transition rate matrix *Q* is dependent on control actions, the state transition functions $p_{i,j}(t)$ are being controlled, and so are the state probabilities.

B. *Restoration and overhaul*

Our ultimate goal is to eliminate all single point failures, and to mitigate the effects of a single server failure on the performance of the database unit. Our approach is to base the supervisory control actions on the state information, which effectively alter the transition rates when loss of data occurs in a single server.

Taking into consideration the symmetry of the model, the control policy is described only for the case of a failed server S_{AB}. The control policies considered for this study are summarized as follows.

$$u_1 = \begin{cases} 0, & S_{AB} = 2, S_{BC} \text{ serves}, S_{CA} \text{ serves (no restoration)} \\ 1, & S_{AB} = 2, S_{BC} \text{ serves}, S_{CA} \text{ restores class A data} \end{cases} \qquad (5)$$

The presence of supervisory control in the transition rate matrix is seen via u_1, u_3, $1\text{-}u_1$, and $1\text{-}u_3$. The values of u_1, u_3 represent specific control actions associated with data restoration, and unit overhaul, respectively. Unit overhaul occurs only at the unit failure state *141*.

The complete baseline model is provided in [2] in the form of a rate transition table, where an additional control variable u_2 was present. u_2 controls routing probabilities when data loss occurs in a server. u_2 is removed in this paper because the small number of queries in the system makes the additional benefit afforded by routing control less obvious to observe.

III. Model augmentation to include errors & delays

This section focuses on modeling the effects of decision errors and control action delays upon entering a state. These two undesirable effects can be intertwined. To quantify their individual impact on performance, they are separated into the class of decision errors when a control action is taken incorrectly but immediately upon entering a state, and the class of delayed control actions when a correct control action is taken but after some time delay. In addition, there are deterministically diagnosable systems for which the only cost of diagnosis is time [9]. Two augmented models will be generated in this section representing a controlled database unit with decision error, and one with control action delay, respectively. Each model will contain *201* states.

A. *Effect of decision error*

The supervisory control considered in this study is state information-based. Upon entering a state, say, *A*, any information deficiency can result in uncertainty in decision making as to whether to take a control action or what control actions to take. In this case, every decision carries a risk.

An example of a decision error with the database unit would be that upon a server failure a wrong server is being identified as having failed. More specifically, S_{AB}, for instance, has failed. S_{CA}, however, is mistakenly thought to be the failed one. Based on the false information, the control action would be for S_{BC} to restore data class *C* in S_{CA}, whereas S_{AB} would be expected to continue to work. As a consequence of a wrong decision, none of the servers can process queries for a period of time. The database unit is said to have entered an intermittent error state. It is assumed that from this state, only transitions to more server failures, or to the recovery to original destination state can occur. Fig.2 depicts a generalized representation of such a case.

Without loss of generality, let *A* be a state that is entered upon a total data loss in a server. Let *C* be the state entered upon the completion of primary database restoration associated with the data loss. Let B_1 through B_n be the states representing completions of services at other *n* servers. Let G_1, …, G_l be the state entered upon the arrival of a new query in one of the server queues. Let F_1 through F_m be the states entered upon data loss at other *m* servers. The notion of intermittent state *I* is introduced, as shown in Fig.2, to allow the representation of imperfect decision making upon entering *A*. Therefore, there is an intermittent error state for each state that involves outgoing transitions with weakened control authorities due to some decision errors. In the database unit of Fig.1, altogether *60* states are added to the original *141* state baseline model. Note that states G_i's are not shown explicitly in Fig.2, and they can be regarded as part of F_i's from this point on. It is assumed that once the primary database restoration takes place for a particular server, the secondary restoration is error free.

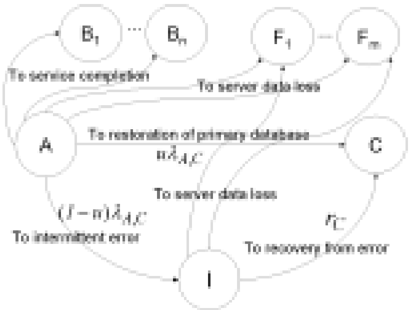

Fig.2 Decision error modeling w. an intermittent error state

Let $\lambda_{A,C}$ denote the transition rate from state *A* to state *C* in the absence of decision error to restoration of primary database associated with the most recent data loss. Let *u* be the probability of successful restoration given that the event of restoration occurs. $(1-u)$ then is referred to as the *thinning* [9] of the Poisson arrival process associated with the restoration. The split of rate $\lambda_{A,C}$ into rate $u\lambda_{A,C}$ and rate $(1-u)\lambda_{A,C}$ is sometimes also called a decomposition [10] of a Poisson arrival process into type 1 with probability *u* and type 2 with probability (*1-u*).

An imperfect decision corresponds to the value of *u* being less than unity. As a consequence, the authority of supervisory control that is supposed to reinforce the restoration process has been weakened. The smaller the value of *u*, the weaker the control authority is.

The rate of recovery from decision error is denoted by r_C.

To state the fact that recovery from an intermittent error state to restoration cannot be faster than the error-free ($u=1$) restoration process, $r_C \le \lambda_{A,C}$ is enforced. On the other hand, the outgoing transition rates from the intermittent error state to the states of data loss in other servers, i.e., from I to F_i, $i=1, 2, \dots , m$, are bounded below by the corresponding rates going from A to F_i. These transitions further reduce the likelihood of reaching state C.

It is now shown that decision errors always degrade the performance in terms of the state transition probability P_{AC} which is the probability that restoration to state C occurs given that the state is A. It turns out that this probability is readily obtained for a Markov chain [9].

$$P_{AC} = \frac{u\lambda_{AC}}{\Lambda(A)} \ , \tag{6}$$

where

$$\Lambda(A) = \lambda_{AB_1} + \cdots + \lambda_{AB_n} + \lambda_{AF_1} + \cdots + \lambda_{AF_m} + \lambda_{AC} \tag{7}$$

without decision error, in which case $u = 1$ in (6), and

$$\Lambda(A) = \lambda_{AB_1} + \cdots + \lambda_{AB_n} + \lambda_{AF_1} + \cdots + \lambda_{AF_m} + u\lambda_{AC} + (1-u)\lambda_{AC} \tag{8}$$

with decision error, in which case $u < 1$. The denominators of (7) and (8) are the same. Apparently, (6) is proportional to u, and is the largest at $u=1$ when there is no decision error. On the other hand, flow balance at state I yields

$$\dot{\pi}_I = (1-u)\lambda_{A,C}\pi_A - (\sum_{i=1}^{m} \lambda_{I,F_i} + r_C)\pi_I \ , \tag{9}$$

from which the following expression for $\pi_I(t)$ in terms of $\pi_A(t)$ at steady state is obtained

$$\pi_I(\infty) = \frac{(1-u)\lambda_{AC}}{\sum_{i=1}^{m} \lambda_{IF_i} + r_C} \pi_A(\infty) \ . \tag{10}$$

(10) is proportional to $1-u$.

Some results of numerical calculation will be presented in Section IV based on the state-augmented model of the database unit of Fig.1 that show how certain performance measures depend on the probability of the restoration decision error.

B. *Effect of delayed control actions*

Time required for diagnosis can be regarded as the universal cause of a control action delay. Time delay can be traded off in some applications with the decision error to minimize their combined effects. This subsection focuses on the discussion of the effect of time delay alone.

An example on the control action delay with the database unit of Fig.1 would be that a total loss of data on a server is not immediately observed. As a result, the action of data restoration is delayed.

As in the previous subsection, let A be a state that is entered upon a total loss of data in a server. Let C be the state entered upon the completion of primary database restoration associated with the data loss. States B_1 through B_n, and states F_1 through F_m also follow the earlier definitions. Fig.3 depicts a proposed model capable of describing a delayed restoration action by an exponentially distributed random amount with average δ^{-1} upon entering state A.

In a more general case, there can be an N-phased delay implemented in the augmented model by inserting N states D_1 through D_N in series between states A and C. Each state D_i retains outgoing transitions to all B_1 through B_n, and F_1 through F_m, in addition to transition to D_{i+1}. The total amount of delay before restoration action is bounded below by random variable $D = D_1 + \cdots + D_N$, with a generalized Erlang distribution [10]

$$L^{-1}\{\sum_{i=1}^{N} \frac{\delta_i}{s+\delta_i}\} \ . \tag{11}$$

One may use an N-stage Erlang to approach a constant delay, or an N-stage hyper-exponential to approach a highly uncertain delay, or a mixture of the two to acquire more general properties [9].

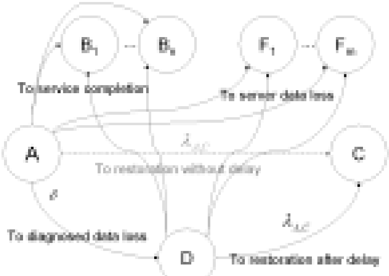

Fig.3 Control delay modeling w. a single-stage delay state

Note that there are two significant differences between the decision error model of Fig.2 and the control delay model of Fig.3. First, the link *to restoration of primary database* is present in Fig.2 with a smaller likelihood of transition, whereas the link *to restoration without delay* is absent in Fig.3. In addition, all links *to service completion* are absent in Fig. 2, but present in Fig.3. Therefore, these are two cases of different nature.

With a single-stage delay for each state entered upon a total loss of data in a server, *60* states are added to the baseline model. Numerical results on the effect of control action delay will be presented in the next section.

IV. Performance Analysis and Discussion

A. *Time to system failure*

When $u_3=0$, the augmented Markov chain model for the database unit contains one absorbing state $x=201$ at which the chain remains forever once it is entered. This is the state of system level failure. The rest of *200* states are transient states. Decompose the state probability vector

$$\pi(t) \equiv [\underbrace{\pi_\tau(t)}_{1\times 200} \quad \underbrace{\pi_\alpha(t)}_{1\times 1}], \tag{12}$$

where vector $\pi_\tau(t)$ contains the transient state probabilities, and $\pi_\alpha(t)$ is the absorbing state probability. Decomposing the rate transition matrix Q and the state transition function matrix $P(t)$ solved from (2) accordingly yields

$$Q = \begin{bmatrix} Q_{11} & Q_{12} \\ 0 & 0 \end{bmatrix}, P(t) = \begin{bmatrix} P_{11}(t) & P_{12}(t) \\ 0 & 1 \end{bmatrix} . \qquad (13)$$

From (2), (4), and (12), it can be determined that the probability density function of time to system failure, or time to absorption, is given by

$$\dot{\pi}_\alpha(t) = \pi_\tau(0) P_{11}(t) Q_{12}, \pi_\alpha(0) = 0, \qquad (14)$$

where

$$\pi_\tau(0) = [1 \quad 0 \quad \cdots], \ P_{11}(t) = e^{Q_{11}t}. \qquad (15)$$

In addition, the mean time to failure of the database unit can be shown to be [9]

$$MTTF = -\pi_\tau(0) Q_{11}^{-1} 1_\tau, 1_\tau = \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix}. \qquad (16)$$

Fig. 4 below shows the dependence of mean time to failure of the database unit on probability of correct control action for data restoration with restoration rate γ as a parameter. The plot indicates that MTTF is sensitive to restoration rate, and becomes more sensitive to supervisory control coverage at a higher restoration rate. The relative robustness of MTTF with respect to supervisory control coverage can be attributed to the fact that recovery has taken a most optimistic path with $r_C = \lambda_{A,C}$, after a decision error has been made.

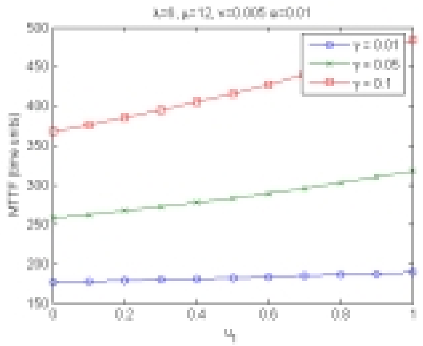

Fig.4 Unit MTTF versus control coverage

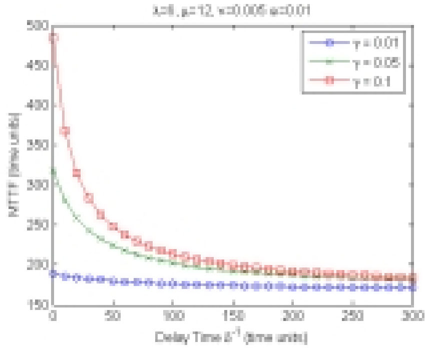

Fig.5 Unit MTTF versus control delay

Fig. 5 above shows the dependence of mean time to failure of the database unit on expected control action delay for data restoration with restoration rate γ as a parameter. It is expected that control action delay affects MTTF more drastically when restoration rate is high. Control action delay becomes dominant in how long it takes to restore data when it becomes comparable to average time required to perform data restoration.

B. Steady-state availability

Suppose as soon as the database unit reaches a system level failure, an overhaul process starts with all the customers flushed out to the delay elements. Suppose with a rate ω the unit is repaired. At the completion of the repair to condition $\pi(0)$, the unit immediately starts to operate again. In this case u_3 is set to 1 in the model, whereas it is set to 0 in the case of an absorbing chain. The existence of a unique steady-state distribution of the Markov chain when $u_3=1$ is guaranteed if the chain is irreducible (or ergodic) [10]. The steady state availability, which can be roughly thought of as the fraction of time the database unit is up, is given by

$$A_{sys} = 1 - \pi_{201}(\infty), \qquad (17)$$

where $\pi_{201}(\infty)$ is determined by solving

$$\pi(\infty)Q = 0, \text{ and } \textstyle\sum_{x=1}^{201} \pi_x(\infty) = 1. \qquad (18)$$

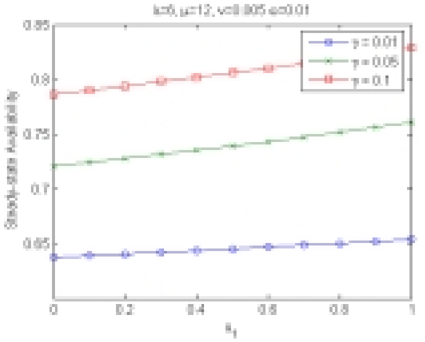

Fig.6 Steady-state availability versus control coverage

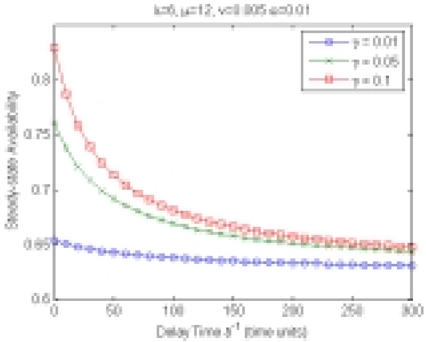

Fig.7 Steady-state availability versus control delay

Fig.6 and Fig.7 show the steady-state availability as a function of supervisory control coverage and a function of expected control action delay. It can be seen that both long delays and slow restoration reduce the availability to unacceptable levels. Explanations on the insensitivity of the availability with respect to coverage and delay under slow restoration conditions follow those for Fig.4 and Fig.5.

C. *Response time*

Consider again the irreducible chain studied in the previous section. Let $I_{i,j}$ be the indicator function associated with transition from state *i* to state *j*, and q_{ij} be the corresponding entry in transition rate matrix *Q*. Let N_i be the total number of queries in queue at state *i*. Then the total expected number of queries in queue at steady-state is given by

$$E[X] = \sum_{i=1}^{201} \pi_i(\infty) N_i \ , \tag{19}$$

and the arrival rate at steady-state is

$$\lambda_s = \sum_{i=1}^{201} \pi_i(\infty) \sum_{j=1}^{201} I_{ij} q_{ij} \ . \tag{20}$$

The calculation of the response time at steady-state then follows Little's Theorem $E[X] = \lambda_s E[R]$. [4]

Fig.7 and Fig.8 show the average response time as a function of supervisory control coverage and a function of control action delay, respectively. Unlike the other performance measures, the sensitivity of the average response time remains relatively significant at a low restoration rate.

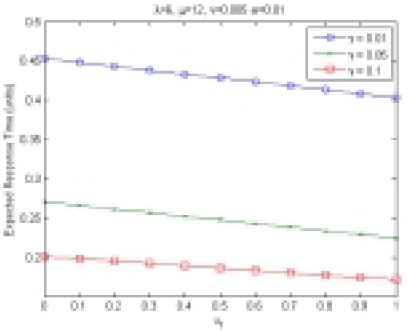

Fig.8 Average query response time versus control coverage

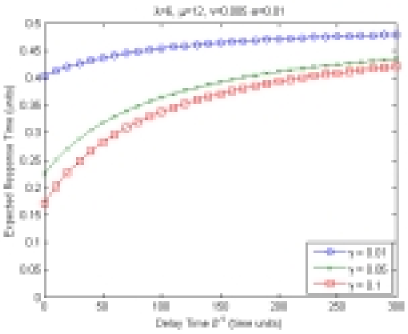

Fig.9 Average query response time versus supervisory control delay

D. *Overhead*

Overhead is a quantity introduced to reflect the ratio of the time invested on helping the database unit to survive longer to its overall busy time. It is a measure of the cost of supervisory control. More specifically,

$$\theta \equiv \frac{\Pr[S_{AB} \text{ restores or fails} \,|\, \text{unit is not failed}]}{\Pr[S_{AB} \text{ restores or fails or serves} \,|\, \text{unit is not failed}]} \tag{21}$$

Overhead θ is calculated for the irreducible chain (u_3=*1*) as a function of supervisory control coverage and a function of supervisory control delay. These are shown in Fig.10 and Fig.11. As in the case of availability, overhead at the steady-state becomes unacceptably high at low restoration rate. It is also sensitive to control coverage and delay when restoration rate is high.

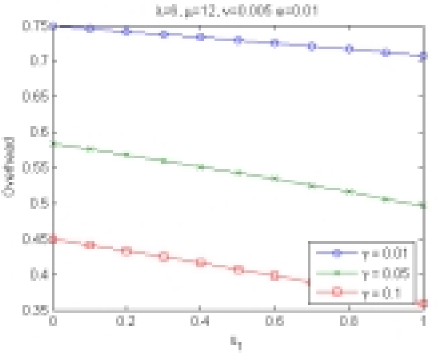

Fig.10 Service overhead versus control coverage

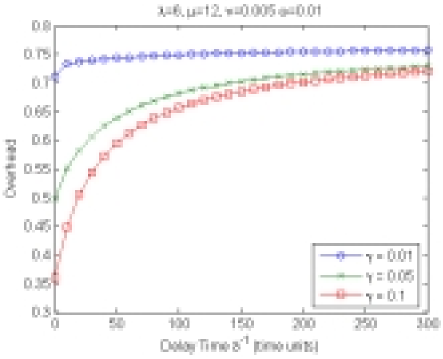

Fig.11 Service overhead versus control delay

References

[1] N. E. Wu, and T. Busch, "An example of supervisory control in C2," *Proc. IEEE Conference on Decision and Control*, 2004.

[2] N.E. Wu, J. M. Metzler, M. Linderman, "Supervisory Control of a Database Unit", *Proc. IEEE Conference on Decision and Control*, 2005.

[3] L. Kleinrock, *Queuing Systems Volumn II: Computer Applications*, John Wiley & Sons, 1976.

[4] G. Bolch, S. Greiner, H. de Meer, and K.S. Trivedi, *Queuing Networks and Markov Chains*, John Weiley & Sons, 1998.

[5] J.M. Metzler, N.E. Wu, and T. Busch, "A Simulation Study of the Effect of Supervisory Control on a Redundant Database Unit," submitted to the 2006 American Control Conference.

[6] Rockwell Software, Inc. Arena, Academic Version 7.01.00, 2004.

[7] W. D. Kelton, R. P. Sadowski and D. T. Sturrock, Simulation with Arena, 3rd Ed., McGraw-Hill, 2004.

[8] S. Zacks, Introduction to Reliability Analysis: Probability Models and Statistics Methods, Springer-Verlag, 1992.

[9] C.G. Cassandras and S. Lafortune, Introduction to Discrete Event Systems, Kluwer, 1999.

[10] E.P.C., Kao, An Introduction to Stochastic Processes, Duxbury Press, 1997.