Hindawi Publishing Corporation
Journal of Control Science and Engineering
Volume 2008, Article ID 310652, 13 pages
doi:10.1155/2008/310652

Research Article

Fault-Tolerant Control of a Distributed Database System

N. Eva Wu,[1] Matthew C. Ruschmann,[1] and Mark H. Linderman[2]

[1] *Department of Electrical and Computer Engineering, Binghamton University, Binghamton, NY 13902-6000, USA*
[2] *US Air Force Research Laboratories at Rome Research Site, Rome, NY 13441-4505, USA*

Correspondence should be addressed to N. Eva Wu, evawu@binghamton.edu

Received 31 December 2006; Accepted 11 September 2007

Recommended by Kemin Zhou

Optimal state information-based control policy for a distributed database system subject to server failures is considered. Fault-tolerance is made possible by the partitioned architecture of the system and data redundancy therein. Control actions include restoration of lost data sets in a single server using redundant data sets in the remaining servers, routing of queries to intact servers, or overhaul of the entire system for renewal. Control policies are determined by solving Markov decision problems with cost criteria that penalize system unavailability and slow query response. Steady-state system availability and expected query response time of the controlled database are evaluated with the Markov model of the database. Robustness is addressed by introducing additional states into the database model to account for control action delays and decision errors. A robust control policy is solved for the Markov decision problem described by the augmented state model.

1. INTRODUCTION

A database, as described in [1], is a shared collection of related data and the description of this data, designed to meet the information needs of a client. A recent study by Wu et al. [2] on a distributed database system, as shown in Figure 1, revealed the benefits of a conscientious design of redundant architecture and the application of state information-based control. Such benefits were quantified in terms of mean time to system failure, steady-state availability, expected response time, and service overhead. The database system was viewed as a queuing network [3, 4] and mathematically modeled as a Markov chain [5]. The control authorities considered included the ability to restore the lost data sets in a single server and the ability to route service requests. In order to obtain an analytic model of manageable size for scrutinizing the effects of control, the queuing network was restricted to the closed type with a query population of three. In addition, all the event lifetime distributions were assumed to be exponential. A simulation study conducted by Metzler [6] using Arena [7, 8] with the above restrictions removed supported the conclusions in [2].

The first objective of this paper is to provide justification that the control policy applied in the aforementioned study [2] is optimal in a well defined sense. To that end, a Markov decision problem [9, 10] is formulated and the solution that minimizes a total expected discounted cost is sought. For the purpose of illustration, a simple problem that disregards the query states is set up, for which the policy developed in [2] is confirmed to be optimal.

In reality, however, it is not practical to monitor every state variable in a network. As a result, knowledge on a certain set of states is inferred based on the observables. On the other hand, a control action, in response to a state transition such as an occurrence of a server failure, must wait until a process of diagnosing the failure state [11] is complete. The time required for diagnosis is assumed to be a random variable and the outcome of the diagnosis usually has some degree of uncertainty as well. If servers must communicate through wireless channels, the likelihood of an erroneous decision and a delayed action is drastically increased. Recognizing that the assumption of instantaneous accessibility of the state information in the database system could lead to overly optimistic conclusions on system performance, Wu et al. [12] took a further step to analyze the effects of control action delays and decision errors for the same database system. Their analysis concluded that delays and errors can significantly degrade the performance of the database system.

Therefore, the second objective of the paper is to seek a robust control policy that mitigates the effects of such control action delays and decision errors. A robust solution obviously has a strong dependence on how uncertainties are

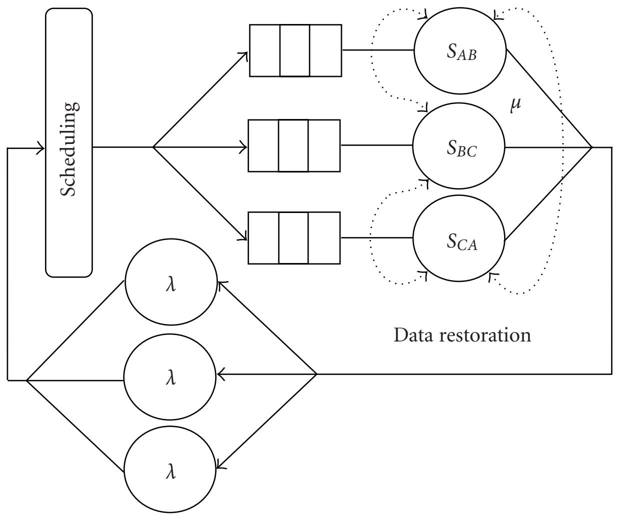


Figure 1: A queuing network representation of a partitioned database system with three servers.

modeled. This paper establishes an uncertain database model following the basic principles presented in [12]. The new model also captures the effect of routing delays of queries from a failed server to remote intact servers. A new Markov decision problem is then formulated and solved. Due to the increased dimension of the problem, approximate solutions are sought via numerical means.

This paper presents a novel model of a replicated data store wherein a set of information is partitioned and each partition is stored on multiple servers. This work is motivated by the recognition of the need for greatly enhanced availability of information management systems in air operations [13]. It addresses the desirability of hardware replication and state-information-controlled restoration, whereas published works in the field of distributed database and replication have discussed specific protocols and software failures [14].

The paper is organized as follows: Section 2 describes the baseline model of the controlled database system shown in Figure 1; Section 3 formulates and solves a Markov decision problem that justifies the control policy applied to the baseline model; Section 4 presents an approach to modeling control action delays and decision errors; Section 5 formulates and solves, using dynamic programming, a Markov decision problem with an uncertain model containing delays and errors, and analyzes the robustly controlled system in terms of system availability and query response time in the presence of control action delays and decision errors.

2. BASELINE MODEL AND NOTATION FOR A CONTROLLED DATABASE SYSTEM

The description of a baseline model for a replicated data store follows to a large extent that of Wu et al. [2]. In particular, a system of three servers is studied, each storing two partitions out of a total of three. Each partition has one "primary" server and one "secondary" server.

The distributed database system in Figure 1 contains three servers in parallel to answer three classes, A, B, and C, of queries for which relevant information can be found in the partitioned data sets, A, B, and C, of the database, respectively. Server S_{MN} would contain the data set corresponding to class M as the primary set and a reproduction of data set N as the secondary set. Alternate secondary data sets are reproduced in order to automate restoration of failed servers within the database. The failure of a server implies the loss of two sets of data within the server. A system level failure is declared when two servers fail, in which case one set 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 in the baseline model. Service is provided on a first-come-first-served (FCFS) basis at each server.

The three delay elements of average delay $1/\lambda$ imply that there are always three queries present in the system at any given time. A new query is generated at a delay element with rate λ upon the completion of the service to a query at one of the servers. The delay elements are also intended to be 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 have a likelihood of ρ_{IJ} to visit server S_{IJ}, where IJ can be AB, BC, or CA.

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 ($\exp(\lambda) \equiv 1 - e^{-\gamma t}$), 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 system overhaul ($\exp(\omega)$) when the failed database system is repaired. All such processes are independent. Standard statistical methods that involve data collection, parameter estimation, and goodness of fit tests exist [15] for identifying event life distributions. Alternative distributions and goodness of these assumptions were investigated in [6]. Since all event lives are assumed to be exponentially distributed, the database system can be conveniently modeled as a Markov chain specified by a state space $\mathcal{X}$, an initial state probability mass function (pmf) $\pi_x(0)$, and a set of state transition rates Λ.

2.1. Model specification

State space $\mathcal{X}$

A state name is coded with a 6-digit number indicative of all queue lengths and server states in the system. With some abuse of notations, a valid state representation is given by $Q_{AB}Q_{BC}Q_{CA}S_{AB}S_{BC}S_{CA}$, where queue length Q_{AB}, Q_{BC}, $Q_{CA} \in \{0, 1, 2, 3\}$ with total length $L \equiv Q_{AB} + Q_{BC} + Q_{CA} \leq 3$ limited by the three entities available in the closed-queue system. The server states S_{AB}, S_{BC}, $S_{CA} \in \{0, 1, 2\}$ are further defined as "2" $\equiv$ data are lost in both the primary and the secondary sets in a server, "1" $\equiv$ the data in the primary set have been restored and data in the secondary set have not been restored, and "0" $\equiv$ data in both primary set and secondary set in a server are intact. A server is said to be in the down state if it is either at states "1" or "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 sets of data lost 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 147 when all the states representing system level failures are aggregated into seven states memorizing the possible queue length distributions and exploiting the symmetry of the three servers. A set of alternative state names are assigned from $\mathcal{X} = \{1, 2, \dots, 147\}$ with 000000 mapped to $x = 1$ and the aggregated system failure state mapped to $x = 141, \dots, 147$. Although the symmetry of the system allows further reduction on the number of states to 56, the 147-state model is retained for clarity of presentation.

Initial state pmf $\{\pi_x(0),\ x = 1, 2, \dots, 147\}$

It is assumed that the database system starts operation from state $x = 1(000000)$, that is, the initial state probability is given by vector $\pi(0) = [1\ 0 \dots 0]$.

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 set of a server are restored with rate γ or repaired with overhaul rate ω. Data in the secondary data set of the server are restored with rate γ, following the restoration of the primary data set. The failed database system is always renewed with overhaul rate ω.

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

$$p_{i,j}(t) \equiv P[X(t) = j \mid X(0) = i], \quad i, j = 1, 2, \dots, 147. \tag{1}$$

The continuous-time Markov chain can be solved from the forward Chapman-Kolmogorov equation [5, 10]

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

and $Q(u(x))$ 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. (See [2] for the complete rate transition.) Control variable $u(x)$ will be defined shortly. State probability mass function at time t,

$$\pi(t) = [\pi_1(t) \quad \pi_2(t) \quad \dots \quad \pi_{147}(t)], \quad t \geq 0, \tag{3}$$

is computed by

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

At this point, a baseline Markov model for the database system of Figure 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, as are the state probabilities.

2.2. Control Policy

Our intention is to eliminate all single point failures. Our approach is to base the control actions on the state information, which effectively alter the transition rates when loss of data occurs in a single server. The possible set of control actions includes restoration, overhaul, and no decision needed. There is one admissible set of control actions at each state. A state of no decision needed has an empty admissible set.

Taking into consideration the symmetry of the model, the control policy considered for this study is summarized as follows:

$$u(x) = \begin{cases} 0, & \text{upon entering the state of one server failure, system overhauls;} \\ 1, & \text{upon entering the state of one server failure, system restores.} \end{cases} \tag{5}$$

The presence of control in the transition rate matrix is seen via $u(x)$ and $\overline{u}(x) = 1 - u(x)$. The values of $u(x)$ represent specific control actions associated with data restoration ($u(x) = 1$) or system overhaul ($\overline{u}(x) = 1$), respectively. Previously in [2], system overhaul is considered only at state $x = 141$ through $x = 147$.

2.3. Performance measures

Two of the four performance measures defined in [2] are reintroduced: steady-state availability A_{sys} and expected response time $E[R]$. These will be used later to validate the control policies that are derived under cost criteria intended to improve both A_{sys} and $E[R]$.

Steady-state availability

Suppose as soon as the database system reaches a system level failure, an overhaul process starts. Suppose, with a rate ω, the system is repaired, and at the completion of the repair, the system immediately starts to operate again. In this case, the Markov chain becomes irreducible, and a unique steady-state distribution exists [5, 10]. The steady-state availability, which can be roughly thought of as the fraction of time the database system is upto, is computed in [2] by

$$A_{\text{sys}} = 1 - \pi_F(\infty), \tag{6}$$

where $\pi_F(\infty)$ is the sum of the system level failure state probabilities determined by solving

$$\pi(\infty)Q = 0, \quad \sum_{x=1}^{147} \pi_x(\infty) = 1. \tag{7}$$

Expected query response time

Query response time is the amount of time elapsing from the instant a query enters a queue until it completes service [10]. With server failures, the average response time $E[R]$ is calculated as the expectation of the ratio of total amount of time that all queries spend waiting for service in queue, plus

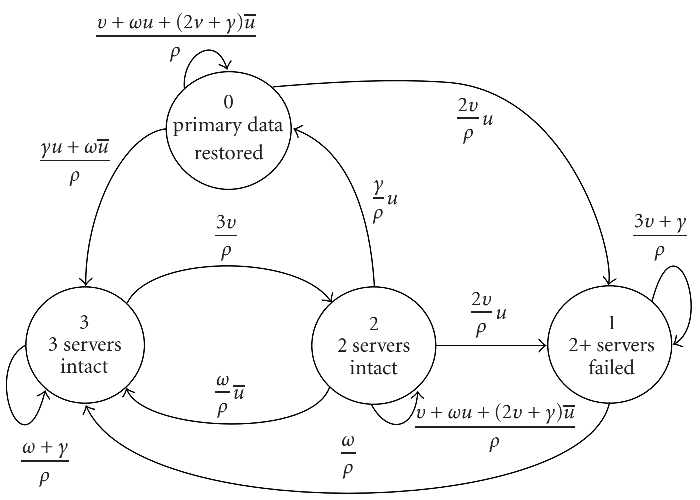

$\frac{v+\omega u+(2v+\gamma)\overline{u}}{\rho}$
0 primary data restored
$\frac{2v}{\rho}u$
$\frac{\gamma u+\omega\overline{u}}{\rho}$
$\frac{\gamma}{\rho}u$
$\frac{3v}{\rho}$
$\frac{3v+\gamma}{\rho}$
$\frac{2v}{\rho}u$
3 3 servers intact
2 2 servers intact
1 2+ servers failed
$\frac{\omega}{\rho}\overline{u}$
$\frac{v+\omega u+(2v+\gamma)\overline{u}}{\rho}$
$\frac{\omega+\gamma}{\rho}$
$\frac{\omega}{\rho}$

FIGURE 2: Markov chain model of the database reflecting only the server states.

their service times to the number of queries that are serviced. Consider, again, the irreducible chain modeling of the system in Figure 1. Let $I_{i,j}$ be the indicator function associated with transition from state i to state j that indicates a query arrival. Let N_i be the total number of queries in queue at state i. Then the total expected number of queries in queue at the steady state is given by

$$E[X] = \sum_{i=1}^{147} \pi_i(\infty) N_i, \tag{8}$$

and the arrival rate at steady-state is

$$\lambda_s = \sum_{i=1}^{147} \pi_i(\infty) \sum_{i=1}^{147} I_{ij} q_{ij}. \tag{9}$$

The calculation of the response time at steady-state then follows Little's Law [4, 10] $E[X] = \lambda_s E[R]$.

3. RESTORATION AS SOLUTION TO MARKOV DECISION PROBLEM

Intuition suggests that by restoring the lost data sets in a single failed server, overhaul can be avoided, and therefore, the stationary control policy $u(x)$ given in (5) ought to render service more available. However, the restoration process occupies one of the remaining servers, and therefore, may prolong the average response time of the system to queries. This section formulates and solves a Markv decision problem (MDP) for the database system to justify the optimality of the restoration policy used in [2].

The Markov decision problem considered in this paper assumes that a cost $C(i,u)$ is incurred at every state transition, where i is the state entered and u is a control action selected from a set of admissible actions [9, 10]. The solution amounts to determining a stationary policy $\pi = \{u_0(x_0), u_1(x_1), \dots\}$ that minimizes the following expected total discounted cost:

$$V_\pi(x_0) = E_\pi \sum_{k=0}^{\infty} \alpha^k C(X_k, u_k), \tag{10}$$

where $0 < \alpha < 1$ is a discount factor.

To simplify the presentation, state information on representing service demand is ignored for the moment. In this case, the inherent symmetry of the database system leads to a very simple 4-state Markov model as shown in Figure 2. As a result, the finite population assumption can be relaxed, that is, the closed queuing network of Figure 1 can either remain closed or can be revised to an open queuing network. In addition, query handling in the event of a server failure becomes completely unrestricted. Two different methods of query handling are to be examined in this section.

(1) Each arrival query has equal likelihood to seek information in data sets A, B, or C, but only the primary data set is available for query service in each server, and the secondary server is there to restore data in a failed server.
(2) Upon a server failure, queries are rerouted to the two remaining servers where the secondary data sets also participate in query service though only one of the two intact servers can provide service to only two of the three classes of queries during restoration.

The distinction in these two cases is captured in transition probabilities and in transition cost $C(i,u)$. Fault-tolerant control policies are now developed for the two cases.

3.1. *Secondary data set reserved for lost data restoration*

This subsection derives the optimal control policy with the first method for handling queries; each arrival query has equal likelihood to seek information in data sets A, B, or C, but only the primary data set is available for query service in

Table 1: One step cost $C(x_k, u_k)$.

State	$u = 1$	$u = 0$
3	0	0
2	$1/\gamma$	$1/\omega$
1	$1/\omega$	$1/\omega$
0	$1/\gamma$	$1/\omega$

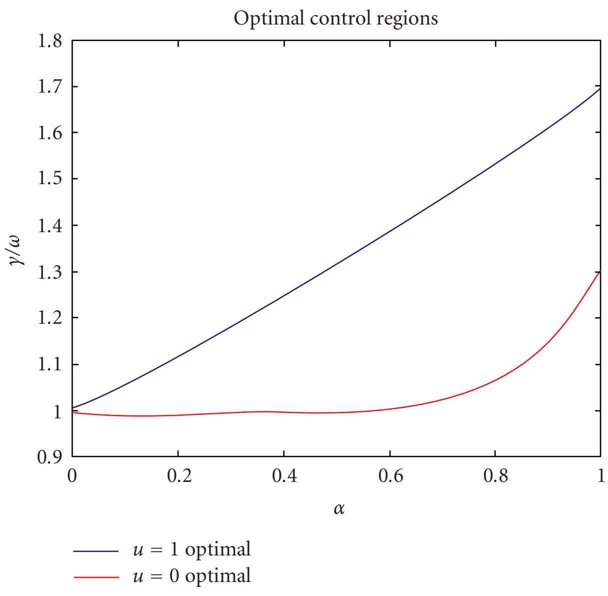


Figure 3: Optimal policy on the $(\alpha, \gamma/\omega)$ graph.

each server, and the secondary server is there to restore data in a failed server.

Figure 2 shows a discrete time Markov chain model for this case. This model is obtained by the application of a uniformization procedure [10] with a uniform rate $\rho = 3\nu+\omega+\gamma$ that is greater than any total outgoing transition rates at any state of the original continuous time Markov process. All parameters in Figure 2 have been defined earlier.

A fault-tolerant control policy essentially determines whether to occupy one of the two working servers to restore the data in the failed server or to overhaul the entire system at the state of one server failure. It is determined by how the designer penalizes a control action at any given state. Table 1 specifies the one step cost at each state.

Let $X_k \in \{1, 2, 3, 4\}$ denote the random state variable at $t = k/\rho$ in the discrete time Markov chain. Control action $u(x_k) = 1$(or 0, or φ) indicates the system's decision to (or not to overhaul, or not to act) restore a failed server. $C(x_k, u_k)$ in Table 1 is the cost incurred when control action u_k is taken based on x_k. It has been shown that under the condition $0 \leq C(j, u) < \infty$ for all j and all u that belongs to some finite admissible sets U_j, the minimal cost $V^*(i)$ satisfies the following optimality equation [9, 10]:

$$V(i) = \min_{u \in U_i} \left\{ C(i, u) + \alpha \sum_j p_{i,j} V(j) \right\}, \tag{11}$$

where $p_{i,j}$ have been marked in Figure 2. In addition, policy π^* is optimal if and only if it yields $V^*(i)$ for all i. The four optimality equations can be expressed explicitly based on (11):

$$\begin{aligned}
V(0) &= \min_u \Bigg\{ \underbrace{\frac{1}{\omega} + \alpha\frac{3\nu+\gamma}{\rho}V(0) + \alpha\frac{\omega}{\rho}V(3),}_{u=0} \\
&\qquad \underbrace{\frac{1}{\gamma} + \alpha\frac{\nu+\omega}{\rho}V(0) + \alpha\frac{2\nu}{\rho}V(1) + \alpha\frac{\gamma}{\rho}V(3)}_{u=1} \Bigg\}; \\
V(1) &= \min_u \left\{ \frac{1}{\omega} + \alpha\frac{3\nu+\gamma}{\rho}V(1) + \alpha\frac{\omega}{\rho}V(3), \right. \\
&\qquad \left. \frac{1}{\omega} + \alpha\frac{3\nu+\gamma}{\rho}V(1) + \alpha\frac{\omega}{\rho}V(3) \right\}; \\
V(2) &= \min_u \left\{ \frac{1}{\omega} + \alpha\frac{2\nu+\gamma}{\rho}V(2) + \alpha\frac{\omega}{\rho}V(3), \right. \\
&\qquad \left. \frac{1}{\gamma} + \alpha\frac{\gamma}{\rho}V(0) + \alpha\frac{2\nu}{\rho}V(1) + \alpha\frac{\nu+\omega}{\rho}V(2) \right\}; \\
V(3) &= \min_u \left\{ \alpha\frac{3\nu}{\rho}V(2) + \alpha\frac{\omega+\gamma}{\rho}V(3), \right. \\
&\qquad \left. \alpha\frac{3\nu}{\rho}V(2) + \alpha\frac{\omega+\gamma}{\rho}V(3) \right\}.
\end{aligned} \tag{12}$$

The above equations are solved for $V^*(i)$, for $i = 0, 1, 2, 3$, using *Mathematica* [16]. Figure 3 is created with $\omega = 10\nu$ and $\alpha \in [0, 1)$. It can be seen that, when the ratio of γ to ω is above the blue curve, $u = 1$ (restoration) is optimal at all states, whereas $u = 0$ (overhaul) is optimal when γ/ω is below the red curve. Between the two curves, $\{u(2) = \varphi,\ u(0) = \varphi\}$ is optimal, for transition from state "2" to state "0" implies restoration of primary data set, which cannot occur with control action $u(2) = 0$. Therefore, the mid-region optimal policy does not take place in the operation of the database system.

Note that $\gamma/\omega = 5$ in [2], which lies above the blue curve in Figure 3 for any $\alpha \in [0, 1)$. Therefore, the always-restore policy implemented in [2] is optimal under the cost structure defined in Table 1.

3.2. *Secondary data set available for both query service and data restoration*

This subsection considers the second method of query handling upon a server failure: overhaul can only occur at state "1," which implies that queries of the failed server are rerouted to the two remaining servers where the secondary data sets also participate in query service though only one of the two intact servers can provide service to only two of the three classes of queries during restoration.

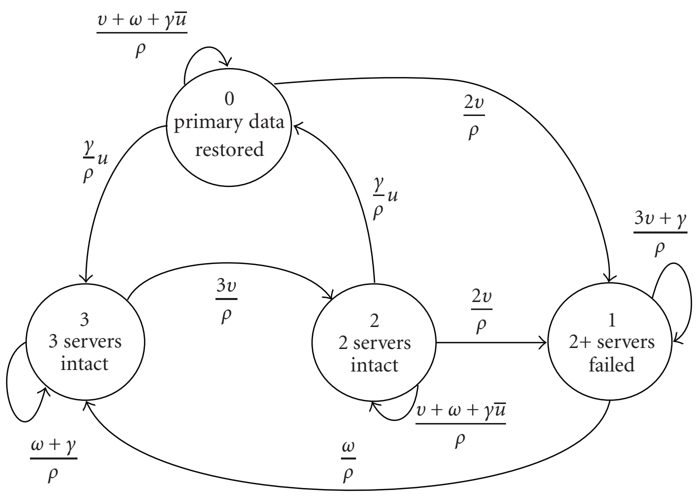


Figure 4: Markov chain model of the database where overhaul does not occur until a second server failure.

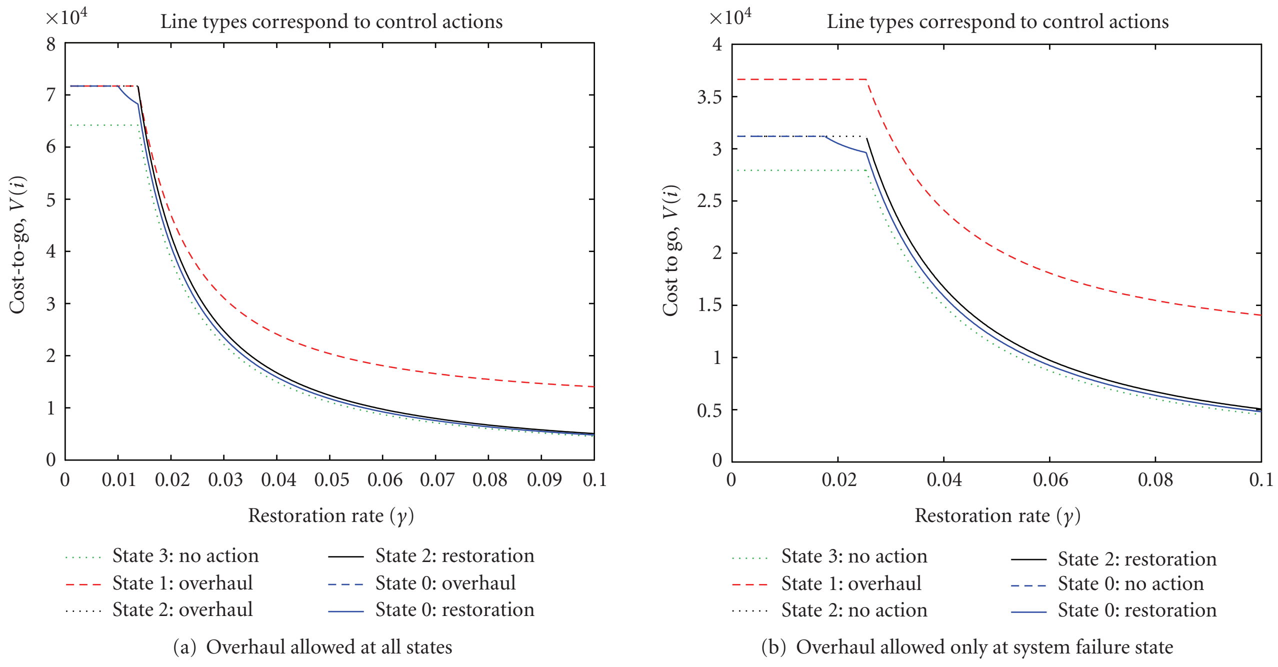


Figure 5: Minimum cost-to-go versus restoration rate for the 4-state model with cost criteria of Table 1.

The uniformized Markov chain model is shown in Figure 4. In this case,

$$u(x)=\begin{cases}0, & \text{upon entering the state of one server failure, system awaits,}\\ 1, & \text{upon entering the state of one server failure, system restores,}\end{cases} \tag{13}$$

overhaul is held until a second server fails, and all classes of queries rely on the service of the two operating servers in the meantime.

Figures 5(a) and 5(b) compare the optimal cost-to-go's of the two methods of query handling as functions of restoration rate γ at fixed $\omega = 10\nu$ and $\nu = 0.001$. Different line types specify different control actions. In Figure 5(b), for example, no control action is taken at state "0" unless $\gamma \geq 1.8\omega$ where restoration takes place; the system is always overhauled at state "1;" no control action is taken at state "2" unless $\gamma \geq 2.6\omega$ where restoration takes place; and no control action is ever taken at state "3." It is seen that control policy change occurs at a higher ratio of γ/ω with the second method (policy change at $\gamma = .026$ in Figure 5(b)) than that with the first method (policy change at $\gamma = .014$ in

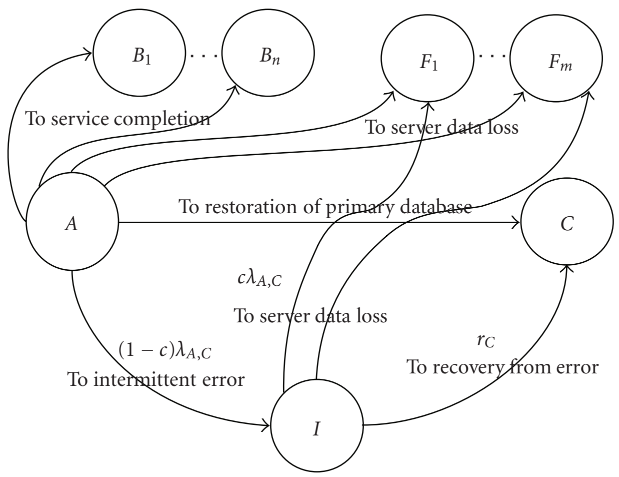


FIGURE 6: Decision error modeling with an intermittent error state.

Figure 5(a)). Despite the slight favor toward overhaul, the optimality of the "always-restore" policy applied in [2] still holds with the second method at the nominal parameter values $\mu = 12$, $\gamma = 0.05$, and $\omega = 0.01$, where $\gamma/\omega = 5 > 2.6$.

4. AUGMENTED MODEL INCLUDING CONTROL DELAYS AND DECISION ERRORS

This section establishes a full-state model to include the effects of decision errors and control action delays upon entering a state of a single server failure. The first two subsections follow [12] that treated these separately as the effect of decision errors when a control action is taken incorrectly but immediately upon entering a state, and the effect of delayed control actions when a correct control action is taken but after some time delay. There are deterministically diagnosable systems for which the only cost of diagnosis is time [11]. The third subsection presents a new model to be used in robust control policy design that combines the two augmented models and introduces also delays due to rerouting queries from failed sever to intact servers.

4.1. Modeling the effect of erroneous decisions

The control action considered in this study is state information based. Upon entering a state, for instance, 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 [17].

A decision error in the database system could include the possibility that upon a server failure, the wrong server is identified as being failed. More specifically, S_{AB}, for instance, has failed. However, S_{CA} is mistakenly observed as the failed server. Based on the false information, the control action would be for S_{BC} to restore data set C in S_{CA}, whereas S_{AB} would be expected to continue to work. As a consequence, none of the servers can process queries for a period of time, and the database system is said to have entered an intermittent error state. It is assumed that from this state, only transitions representing service completion can occur. Figure 6 depicts a generic representation of such a case.

Without loss of generality, let A be a state that is entered upon the loss of both data sets in a server. Let C be the state entered upon the completion of primary data set restoration associated with the data loss. Let B_1 through B_n be the states representing completion of services at other n servers. Let $G_1, \dots, G_l$ be the state entered upon the arrival of a new query in one of the queues. (G_i are not shown explicitly in Figure 6.) Let F_1 through F_m be the states entered upon data loss at other m servers. An intermittent state I is introduced, as shown in Figure 6, 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 system of Figure 1, 60 states are added to the original 147 states of baseline model. It is assumed that once the primary data set restoration takes place for a particular server, the secondary data set restoration proceeds without a decision error.

Let $\lambda_{A,C}$ denote the transition rate from state A to state C in the absence of decision error in the restoration of the primary database associated with the most recent data loss. Let c be the probability of successful restoration, given that the event of restoration occurs. $(1-c)$ then is referred to as the thinning [5] of the Poisson arrival process associated with the restoration. The split of rate $\lambda_{A,C}$ into rate $c\lambda_{A,C}$ and rate $(1-c)\lambda_{A,C}$ is sometimes also called a decomposition of a Poisson arrival process into type 1 with probability w and type 2 with probability $(1-c)$.

An imperfect decision corresponds to the value of c being less than unity. As a consequence, the authority of control that is supposed to reinforce the restoration process is weakened. The smaller the value of c, 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 ($c = 1$) 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, that is, 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 current state A. It turns out that this probability is readily obtained for a Markov chain

$$P_{AC} = \frac{c\lambda_{AC}}{\Lambda(A)}, \tag{14}$$

where

$$\Lambda(A) = \lambda_{AB_1} + \cdots + \lambda_{AB_n} + \cdots + \lambda_{AF_1} + \cdots + \lambda_{AF_m} + \lambda_{AC} \tag{15}$$

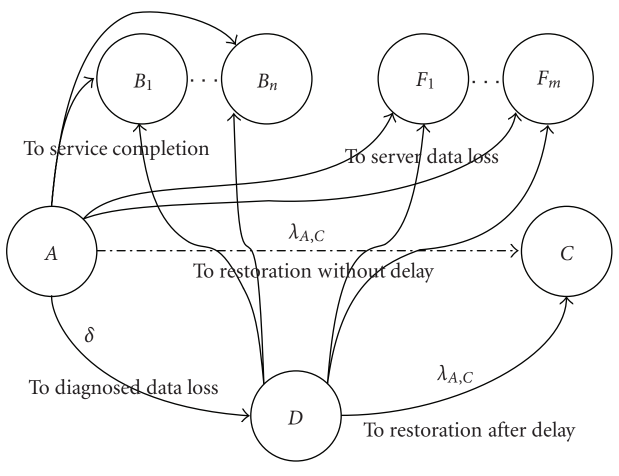


FIGURE 7: Control action delay modeling with a single-stage delay state.

without decision error, in which case $w = 1$ in (14), and

$$\Lambda(A) = \lambda_{AB_1} + \cdots + \lambda_{AB_n} + \lambda_{AF_1} + \cdots + \lambda_{AF_m} + c\lambda_{AC} + (1-c)\lambda_{AC} \tag{16}$$

with decision error, in which case $c < 1$. Note that (15) and (16) are the same, and both enter (14). Therefore, (14) is proportional to c, and is largest at $c = 1$ when there is no decision error.

4.2. Modeling the effect of delayed control actions

Time required for diagnosis [11] can be regarded as the universal cause of a control action delay. An example of the control action delay in the database system shown in Figure 1 would be that a total loss of data in 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. Figure 7 depicts a proposed model capable of describing a delayed restoration action by an exponentially distributed random amount with average δ^{-1} units of time upon entering state A. With a single-stage delay for each state entered upon a total loss of data in a server, another 60 states are added to the baseline model.

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 [5];

$$\mathcal{L}^{-1}\left\{\prod_{i=1}^{N} \frac{\delta_i}{s+\delta_i}\right\}. \tag{17}$$

One may use an N-stage Erlang to approach a constant delay, an N-state hyperexponential to approach a highly uncertain delay, or a mixture of the two to acquire more general properties [10] in its distribution.

Note that there are two significant differences between the decision error model of Figure 6 and the control delay model of Figure 7. First, the link to *restoration of primary database* is present in Figure 7 with a smaller likelihood of transition, whereas the link to *restoration without delay* is absent in Figure 7. In addition, all links to *service completion* are absent in Figure 6, but are present in Figure 7. Therefore, each case has its distinct nature.

4.3. Full-state model of the controlled database system

Referring again to the closed queuing network view of the distributed database system in Figure 1, this section presents its augmented model that incorporates all three sources of uncertainties: decision errors (Section 4.1), control action delays (Section 4.2), and routing delays. Routing delays are incurred when queries at a failed server are rerouted to the remaining intact servers.

Rerouting of queries becomes desirable when the queries observe a server failure after they have entered the queue preceding the server. An exponentially distributed random routing time is introduced with rate τ/sec for this purpose. A routing delay is assumed independent of a control action delay. The former captures the random time of diagnosis, whereas the latter captures random time of transmission of queries among servers. Model augmentation amounts to adding new transitions among existing states without the need for new states.

In order to establish a full state model with all uncertainty types, the representation of the composite state variable is modified to $x = Q_{AB}Q_{BC}Q_{CA}S_{AB}S_{BC}S_{CA}U$, where $Q_{IJ} \in \{0,1,2,3\}$ and $S_{IJ} \in \{0,1,2\}$ as in the baseline model described in Section 2; newly introduced uncertainty variable $U \in \{0,1,2\}$ with "1" = control delayed and "2" = wrong decision made. This results in a 267 state-model. By exploiting symmetry, the 256 $(147+60+60)$ state model can be reduced to a 96-state model. The binary control variables are defined as follows: $u_1 = 1$ to restore, $u_2 = 1$ to overhaul, and $u_3 = 1$ to reroute queries.

The states, the transitions, and the transition rates of the uncertain model are summarized in Figure 8, based on which transition matrix Q of a Markov chain can be built and used in the next section for robust control policy design. τ in Figure 8 is the newly introduced query transmission rate when the action for rerouting is called for. Error probability $\bar{c}$ relates to c in Figure 6 through $\bar{c} = 1 - c$. Subscript "p" denotes "primary" and "s" denotes "secondary." Use of symmetry is reflected in server state S_f and arrival rates λ_1, λ_2, and λ_3.

	State #	States					Arrivals						Completions			Failures			Delay	Restorations				Overhaul	Reroute
		$\mathcal{Q}_{AB}$	$\mathcal{Q}_{BC}$	$\mathcal{Q}_{CA}$	S_f	U	λ_1	λ_2	λ_3	Overhaul			μ_p			ν			δ	$c\gamma_p$	$c\gamma_s$	$\bar{c}\gamma_p$	$\bar{c}\gamma_s$	ω	τ
Available	**1**	0	0	0	0	0	2	2	2	0	0	0	0	0	0	29	29	29	0	0	0	0	0	0	0
	2	1	0	0	0	0	4	3	3	0	0	0	1	0	0	32	30	31	0	0	0	0	0	0	0
	3	1	1	0	0	0	7	6	5	0	0	0	2	2	0	38	37	36	0	0	0	0	0	0	0
	4	2	0	0	0	0	8	7	6	0	0	0	2	0	0	35	33	34	0	0	0	0	0	0	0
	5	1	1	1	0	0	0	0	0	0	0	0	3	3	3	39	39	39	0	0	0	0	0	0	0
	6	1	2	0	0	0	0	0	0	0	0	0	4	3	0	44	43	40	0	0	0	0	0	0	0
	7	2	1	0	0	0	0	0	0	0	0	0	3	4	0	45	41	42	0	0	0	0	0	0	0
	8	3	0	0	0	0	0	0	0	0	0	0	4	0	0	48	46	47	0	0	0	0	0	0	0
Secondary restore	**9**	0	0	0	1	0	12	11	10	50	50	50	0	0	0	0	49	49	0	0	1	0	0	1	0
	10	0	0	1	1	0	17	16	13	51	51	52	0	0	9	0	50	50	0	0	2	0	0	2	0
	11	0	**1**	0	1	0	18	14	16	51	52	51	0	9	0	0	50	50	0	0	2	0	0	2	0
	12	1	0	0	1	0	15	18	17	52	51	51	0	0	0	0	50	50	0	0	2	0	0	2	10
	13	0	0	2	1	0	21	20	26	55	54	56	0	0	10	0	52	52	0	0	4	0	0	4	0
	14	0	**2**	0	1	0	24	27	22	54	56	55	0	11	0	0	52	52	0	0	4	0	0	4	0
	15	2	0	0	1	0	28	25	23	56	55	54	0	0	0	0	52	52	0	0	4	0	0	4	17
	16	0	**1**	1	1	0	19	22	20	53	55	54	0	10	11	0	51	51	0	0	3	0	0	3	0
	17	1	0	1	1	0	23	19	21	54	53	55	0	0	12	0	51	51	0	0	3	0	0	3	13
	18	1	**1**	0	1	0	25	24	19	55	54	53	0	12	0	0	51	51	0	0	3	0	0	3	16
	19	1	**1**	1	1	0	0	0	0	0	0	0	0	17	18	0	53	53	0	0	5	0	0	5	20
	20	0	**1**	2	1	0	0	0	0	0	0	0	0	13	16	0	54	54	0	0	6	0	0	6	0
	21	1	0	2	1	0	0	0	0	0	0	0	0	0	17	0	55	55	0	0	7	0	0	7	26
	22	0	**2**	1	1	0	0	0	0	0	0	0	0	16	14	0	55	55	0	0	7	0	0	7	0
	23	2	0	1	1	0	0	0	0	0	0	0	0	0	15	0	54	54	0	0	6	0	0	6	21
	24	1	**2**	0	1	0	0	0	0	0	0	0	0	18	0	0	54	54	0	0	6	0	0	6	22
	25	2	**1**	0	1	0	0	0	0	0	0	0	0	15	0	0	55	55	0	0	7	0	0	7	19
	26	0	0	3	1	0	0	0	0	0	0	0	0	0	13	0	56	56	0	0	8	0	0	8	0
	27	0	**3**	0	1	0	0	0	0	0	0	0	0	14	0	0	56	56	0	0	8	0	0	8	0
	28	3	0	0	1	0	0	0	0	0	0	0	0	0	0	0	56	56	0	0	8	0	0	8	23
Observation delay	**29**	0	0	0	2	0	32	31	30	50	50	50	0	0	0	0	49	49	57	0	0	0	0	1	0
	30	0	0	1	2	0	37	36	33	51	51	52	0	0	29	0	50	50	58	0	0	0	0	2	0
	31	0	1	0	2	0	38	34	36	51	52	51	0	29	0	0	50	50	59	0	0	0	0	2	0
	32	1	0	0	2	0	35	38	37	52	51	51	0	0	0	0	50	50	60	0	0	0	0	2	0
	33	0	0	2	2	0	41	40	46	55	54	56	0	0	30	0	52	52	61	0	0	0	0	4	0
	34	0	2	0	2	0	44	47	42	54	56	55	0	31	0	0	52	52	62	0	0	0	0	4	0
	35	**2**	0	0	2	0	48	45	43	56	55	54	0	0	0	0	52	52	63	0	0	0	0	4	0
	36	0	1	1	2	0	39	42	40	53	55	54	0	30	31	0	51	51	64	0	0	0	0	3	0
	37	1	0	1	2	0	43	39	41	54	53	55	0	0	32	0	51	51	65	0	0	0	0	3	0
	38	1	1	0	2	0	45	44	39	55	54	53	0	32	0	0	51	51	66	0	0	0	0	3	0
	39	1	1	1	2	0	0	0	0	0	0	0	0	37	38	0	53	53	67	0	0	0	0	5	0
	40	0	1	2	2	0	0	0	0	0	0	0	0	33	36	0	54	54	68	0	0	0	0	6	0
	41	1	0	2	2	0	0	0	0	0	0	0	0	0	37	0	55	55	69	0	0	0	0	7	0
	42	0	2	1	2	0	0	0	0	0	0	0	0	36	34	0	55	55	70	0	0	0	0	7	0
	43	**2**	0	1	2	0	0	0	0	0	0	0	0	0	35	0	54	54	71	0	0	0	0	6	0
	44	1	2	0	2	0	0	0	0	0	0	0	0	38	0	0	54	54	72	0	0	0	0	6	0
	45	**2**	1	0	2	0	0	0	0	0	0	0	0	35	0	0	55	55	73	0	0	0	0	7	0
	46	0	0	3	2	0	0	0	0	0	0	0	0	0	33	0	56	56	74	0	0	0	0	8	0
	47	0	3	0	2	0	0	0	0	0	0	0	0	34	0	0	56	56	75	0	0	0	0	8	0
	48	**3**	0	0	2	0	0	0	0	0	0	0	0	0	0	0	56	56	76	0	0	0	0	8	0
Failure	**49**	0	0	0	3	0	50	50	50	50	50	50	0	0	0	0	49	49	0	0	0	0	0	1	0
	50	1	0	0	3	0	52	51	51	52	51	51	0	0	0	0	50	50	0	0	0	0	0	2	0
	51	1	1	0	3	0	55	54	53	55	54	53	0	0	0	0	51	51	0	0	0	0	0	3	0
	52	**2**	0	0	3	0	56	55	54	56	55	54	0	0	0	0	52	52	0	0	0	0	0	4	0
	53	1	1	1	3	0	0	0	0	0	0	0	0	0	0	0	53	53	0	0	0	0	0	5	0
	54	1	**2**	0	3	0	0	0	0	0	0	0	0	0	0	0	54	54	0	0	0	0	0	6	0
	55	**2**	1	0	3	0	0	0	0	0	0	0	0	0	0	0	55	55	0	0	0	0	0	7	0
	56	**3**	0	0	3	0	0	0	0	0	0	0	0	0	0	0	56	56	0	0	0	0	0	8	0
Primary restoration	**57**	0	0	0	2	1	60	59	58	50	50	50	0	0	0	0	49	49	0	9	0	77	0	1	0
	58	0	0	**1**	2	1	65	64	61	51	51	52	0	0	57	0	50	50	0	10	0	78	0	2	0
	59	0	1	0	2	1	66	62	64	51	52	51	0	57	0	0	50	50	0	11	0	79	0	2	0
	60	1	0	0	2	1	63	66	65	52	51	51	0	0	0	0	50	50	0	12	0	80	0	2	58
	61	0	0	**2**	2	1	69	68	74	55	54	56	0	0	58	0	52	52	0	13	0	81	0	4	0
	62	0	2	0	2	1	72	75	70	54	56	55	0	59	0	0	52	52	0	14	0	82	0	4	0
	63	**2**	0	0	2	1	76	73	71	56	55	54	0	0	0	0	52	52	0	15	0	83	0	4	65
	64	0	1	**1**	2	1	67	70	68	53	55	54	0	58	59	0	51	51	0	16	0	84	0	3	0
	65	1	0	**1**	2	1	71	67	69	54	53	55	0	0	60	0	51	51	0	17	0	85	0	3	61
	66	1	1	0	2	1	73	72	67	55	54	53	0	60	0	0	51	51	0	18	0	86	0	3	64
	67	1	1	**1**	2	1	0	0	0	0	0	0	0	65	66	0	53	53	0	19	0	87	0	5	68
	68	0	1	**2**	2	1	0	0	0	0	0	0	0	61	64	0	54	54	0	20	0	88	0	6	0
	69	1	0	**2**	2	1	0	0	0	0	0	0	0	0	65	0	55	55	0	21	0	89	0	7	74
	70	0	2	**1**	2	1	0	0	0	0	0	0	0	64	62	0	55	55	0	22	0	90	0	7	0
	71	**2**	0	**1**	2	1	0	0	0	0	0	0	0	0	63	0	54	54	0	23	0	91	0	6	69
	72	1	2	0	2	1	0	0	0	0	0	0	0	66	0	0	54	54	0	24	0	92	0	6	70
	73	**2**	1	0	2	1	0	0	0	0	0	0	0	63	0	0	55	55	0	25	0	93	0	7	67
	74	0	0	**3**	2	1	0	0	0	0	0	0	0	0	61	0	56	56	0	26	0	94	0	8	0
	75	0	3	0	2	1	0	0	0	0	0	0	0	62	0	0	56	56	0	27	0	95	0	8	0
	76	**3**	0	0	2	1	0	0	0	0	0	0	0	0	0	0	56	56	0	28	0	96	0	8	71
Rework primary restoration	**77**	0	0	0	2	2	80	79	78	50	50	50	0	0	0	0	49	49	0	9	0	57	0	1	0
	78	0	0	**1**	2	2	85	84	81	51	51	52	0	0	77	0	50	50	0	10	0	58	0	2	0
	79	0	1	0	2	2	86	82	84	51	52	51	0	77	0	0	50	50	0	11	0	59	0	2	0
	80	1	0	0	2	2	83	86	85	52	51	51	0	0	0	0	50	50	0	12	0	60	0	2	78
	81	0	0	**2**	2	2	89	88	94	55	54	56	0	0	78	0	52	52	0	13	0	61	0	4	0
	82	0	2	0	2	2	92	95	90	54	56	55	0	79	0	0	52	52	0	14	0	62	0	4	0
	83	**2**	0	0	2	2	96	93	91	56	55	54	0	0	0	0	52	52	0	15	0	63	0	4	85
	84	0	1	**1**	2	2	87	90	88	53	55	54	0	78	79	0	51	51	0	16	0	64	0	3	0
	85	1	0	**1**	2	2	91	87	89	54	53	55	0	0	80	0	51	51	0	17	0	65	0	3	81
	86	1	1	0	2	2	93	92	87	55	54	53	0	80	0	0	51	51	0	18	0	66	0	3	84
	87	1	1	**1**	2	2	0	0	0	0	0	0	0	85	86	0	53	53	0	19	0	67	0	5	88
	88	0	1	2	2	2	0	0	0	0	0	0	0	81	84	0	54	54	0	20	0	68	0	6	0
	89	1	0	**2**	2	2	0	0	0	0	0	0	0	0	85	0	55	55	0	21	0	69	0	7	94
	90	0	2	**1**	2	2	0	0	0	0	0	0	0	84	82	0	55	55	0	22	0	70	0	7	0
	91	**2**	0	**1**	2	2	0	0	0	0	0	0	0	0	83	0	54	54	0	23	0	71	0	6	89
	92	1	2	0	2	2	0	0	0	0	0	0	0	86	0	0	54	54	0	24	0	72	0	6	90
	93	**2**	1	0	2	2	0	0	0	0	0	0	0	83	0	0	55	55	0	25	0	73	0	7	87
	94	0	0	**3**	2	2	0	0	0	0	0	0	0	0	81	0	56	56	0	26	0	74	0	8	0
	95	0	3	0	2	2	0	0	0	0	0	0	0	82	0	0	56	56	0	27	0	75	0	8	0
	96	3	0	0	2	2	0	0	0	0	0	0	0	0	0	0	56	56	0	28	0	76	0	8	91

Figure 8: Transitions and transition rates of the uncertain database state model.

5. ROBUST CONTROL POLICY DESIGN

This section seeks robust control policies as solutions to the Markov decision problem:

$$V_{\pi^*}(x_0) = \min_{\pi} E_{\pi}\sum_{k=0}^{\infty}\alpha^k C(X_k, \mathbf{u}_k), \quad x_0 \in \mathcal{X} = \{1,2,\dots,95,96\}, \tag{18}$$

where $0 < \alpha < 1$, $\pi = \{\mathbf{u}_0, \mathbf{u}_1, \dots\}$ is the control policy sought, $\mathbf{u} = (u_1, u_2, u_3)$, and $u_i \in \{0,1\}$. $u_1 = 1$ to restore, $u_2 = 1$ to overhaul, and $u_3 = 1$ to reroute queries, as defined in Section 4.3. Note that the full-state model enables the designer to consider service demand and to weigh availability against response time. Thus two cost criteria are established. The first criterion,

$$C(x_k, \mathbf{u}_k) = \frac{Q_1(x_k) + Q_2(x_k) + Q_3(x_k)}{\mu}, \tag{19}$$

penalizes long queues that cannot effectively reduce in time due to server failure, and thus favors response time. The second criterion, shown in the following table, penalizes prolonged service time, again, due to server failure, and thus favors availability.

The size of the state space suggests numerical means for solutions. Mathematical programs will be applied to obtain the solutions. The steady-state availability and the expected query response time of the controlled database system with the optimal policy will then be examined under various conditions.

5.1. *Optimal policy design via mathematical programming*

The rate transition matrix $Q(\mathbf{u}(x))$ of the 96-state model can be obtained based on Figure 8 established in Section 4.3. This $Q(\mathbf{u}(x))$ depends on $\mathbf{u}(x) = (u_1(x), u_2(x), u_3(x))$, $u_i \in \{0,1\}$. State probability equation

$$\dot{\pi}(t) = \pi(t)Q(\mathbf{u}(x)) \tag{20}$$

originated from the forward Chapman-Kolmogorov (2) can now be uniformized to yield a discrete time Markov chain

$$\pi(k+1) = \pi(k)\left[I + \frac{1}{\rho}Q(\mathbf{u}(x))\right], \tag{21}$$

where uniform rate ρ can be chosen to be

$$\rho = 3\lambda + 3\nu + \tau + \delta + \gamma + \mu + \omega. \tag{22}$$

Recall optimality (11)

$$V(i) = \min_{\mathbf{u}\in\mathbf{U}_i}\left\{C(i,\mathbf{u}) + \alpha\sum_j p_{i,j}V(j)\right\}, \quad i \in \mathcal{X}, \tag{23}$$

as an alternative characterization of the solution to Markov decision problem (18), which produces a system of 96 equations.

Dynamic programming is the most natural numerical approach to policy design (18) because (11) is derived through taking limit of a finite horizon dynamic program [9, 10]

$$V_{k+1}(j) = \min_{\mathbf{u}\in\mathbf{U}_j}\left\{C(j,\mathbf{u}) + \alpha\sum_{r\in\mathcal{X}} p_{j,r}V_k(r)\right\}, \quad k = 1,2,\dots,N-1, \tag{24}$$

where $\alpha < 1$, and terminal cost $V_0(j) = 0$, forall $j \in \mathcal{X}$. In this case the optimal cost is given by $V_N(x_0)$, $x_0 \in \mathcal{X}$. More specifically, with $\mathbf{u}$ taking values in a finite set, the minimal cost-to-go from x_0 of the 96-state Markov decision process satisfies

$$\lim_{N\to\infty} V_N(x_0) = V^*(x_0), \quad x_0 \in \mathcal{X} = \{1,2,\dots,95,96\}, \tag{25}$$

where$V_N(x_0)$ is the minimal cost-to-go from x_0 of an N-step finite horizon process.

The solution to a dynamic program results from an iterative calculation backwards along the horizon from $V_0(j)$ to the first step $V_N(j)$. For the dynamic programming calculation to converge to the true cost-to-go, N must be significantly large, and must be less than 1.

Linear programming [18] can be considered as an alternative numerical approach to the solution of the Markov decision problem. In this case, the set of optimality equations is turned into a set of affine constraints on the set of optimization variables $\{V(i)\}$, and the problem can be formally stated as follows:

$$\text{Maximize} \quad V(1) + V(2) + \cdots + V(95) + V(96) \tag{26}$$

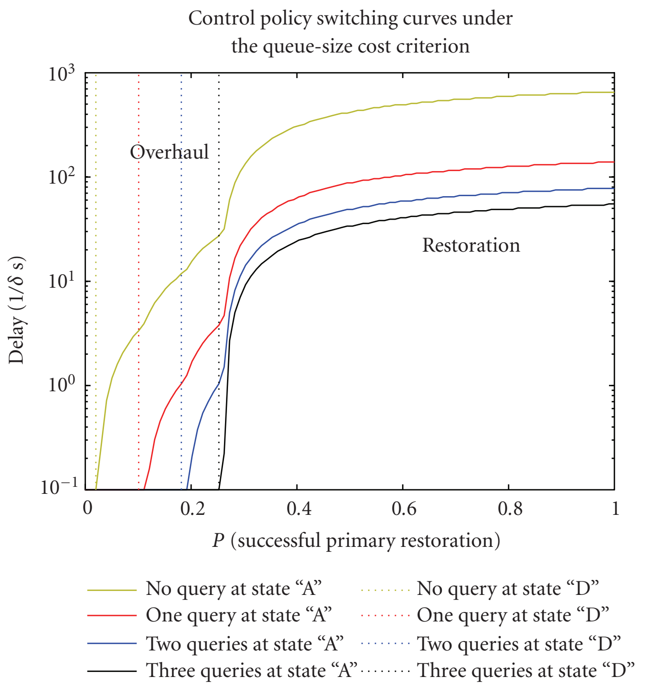
$$\text{subject to} \quad V(i) \geq 0, \quad i \in \mathcal{X} = \{1,\dots,96\}, \tag{27}$$

$$V(i) \leq \left[C(i,\mathbf{u}) + \alpha\sum_i p_{i,j}V(j)\right]\Bigg|_{\mathbf{u}} \quad \forall\mathbf{u}\in\mathbf{U}_i, i \in \mathcal{X}. \tag{28}$$

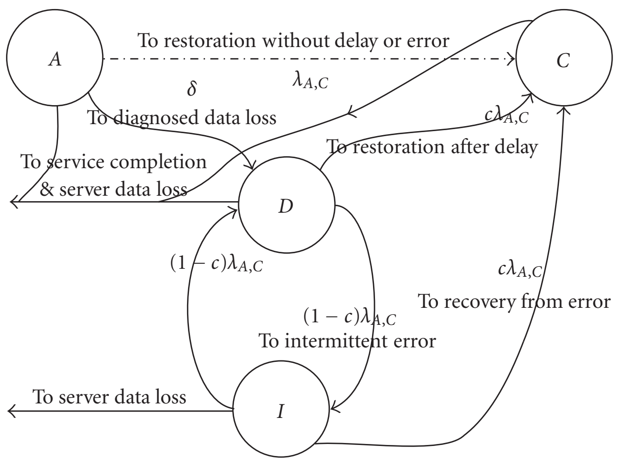
The equivalence of the linear program formulation (26)–(28) and the optimality equation formulation can be easily established. First, (27) is trivially satisfied for all i in both formulations because one-step cost $C(i,\mathbf{u})$ is always nonnegative.

Suppose $(V(1),\dots,V(96))$ is the linear program solution. Then there must be one active (equality achieved) constraint for each of the affine inequality constraints of the form $V(i) \leq \cdots$ for each i. Suppose for some j, the constraint(s) $V(j)$ is not active. Then $V(j)$ can be increased until one of the inequality constraints becomes active without violating the rest of the inequality constraints because $\alpha p_{i,j} < 1$ as coefficient of $V(j)$ on the right side of the inequality constraints (28). This, however, contradicts the assumption that $\sum_i V(i)$ is maximum. Therefore, $(V(1),\dots,V(96))$ is also the solution to the optimality equations (28).

Assume now that $(V(1),\dots,V(96))$ satisfies the optimality equations. It then automatically satisfies the inequality



(a)



(b)

FIGURE 9: (a) Switching curves of the optimal policy under discounted queue size. (b) Partial database model containing both control action delay and decision error.

constraints (28), of which 96 are active, one for each $V(i)$ appearing on the left side. Suppose $\sum_i V(i)$ is not maximum. There is at least a $V(j)$ for some j that is smaller than the corresponding cost in max $\sum_i V(i)$, which implies that the corresponding constraint(s) for $V(j) < \cdots$ is (are) slack or inactive. This contradicts that $V(j)$ satisfies the optimality equation. Therefore, $(V(1),\ldots,V(96))$ must also be the solution of the linear program formulation (28). The equivalence is thus established.

The function linprog in MATLAB's Optimization Toolbox [19] solves the maximization problem above. The active constraints are checked with a MATLAB script to determine the optimal control policy.

The computational complexity of the dynamic program and that of the linear program are now compared. Finding the solution to a linear program generally requires a computation time proportional to n^2m [18] when $m \geq n$, where n is the number of optimization variables, and m is the number of constraints. The computational complexity of an iterative dynamic programming solution can be approximated by assuming that each iteration is a series of linear programs. The linear programming solution to the set of optimality equations is of course a single linear program.

The number of control variables, u, the number of states, s, and the horizon length, N, are critical to the computation time of these methods. First, consider the iterative method as a series of linear programs. Each individual iteration along the N-step horizon consists of s individual linear programs. Each individual linear program has u variables and $2u$ constraints. Therefore, the computation time is proportional to $Ns(u^2 2u) = Ns(2u^3)$. Now, consider the method of solving the optimality equations through linear programming. The single linear program has s variables and $2^u s$ constraints. Hence, its computation time is proportional to $s^2 2^u s = s^3 2^u$.

Although the computation time grows faster for the linear program as the number of states increases, the horizon N is typically much larger than s^2 for small discount factorin β in $\alpha = \rho/(\beta+\rho)$. Therefore, the linear program is more efficient for moderate numbers of states and small discount factors.

5.2. Availability and response time under robust control policy

A selected set of results on the robust control policies solved via mathematical programming are presented in this subsection, and the system availability and query response time under some of the optimal policies are examined.

5.2.1. Restoration-overhaul switching

Under the cost criterion (19) (minimum total discounted queue size), the optimal policy depends on the number of queries in the queue behind the failed server. No action is taken to overhaul the system until the two active queues are empty and the buildup of queries behind the failed server is significant. Figure 9(a) depicts a switching curve of of the control policy between overhaul and restoration before (solid) and after (dotted) state D in Figure 9(b) is reached. Policy switching is determined by the amount of control action delay, the decision error probability, and the number of queries in the failed server. It can be seen that, while the two active queues are occupied or after the primary data is successfully restored, restoration is performed on the failed server as long as the server performing the restoration does not have any customers waiting in its queue.

Under the cost criterion stated in Table 2 (minimally reduced service time), the optimal policy always attempts to

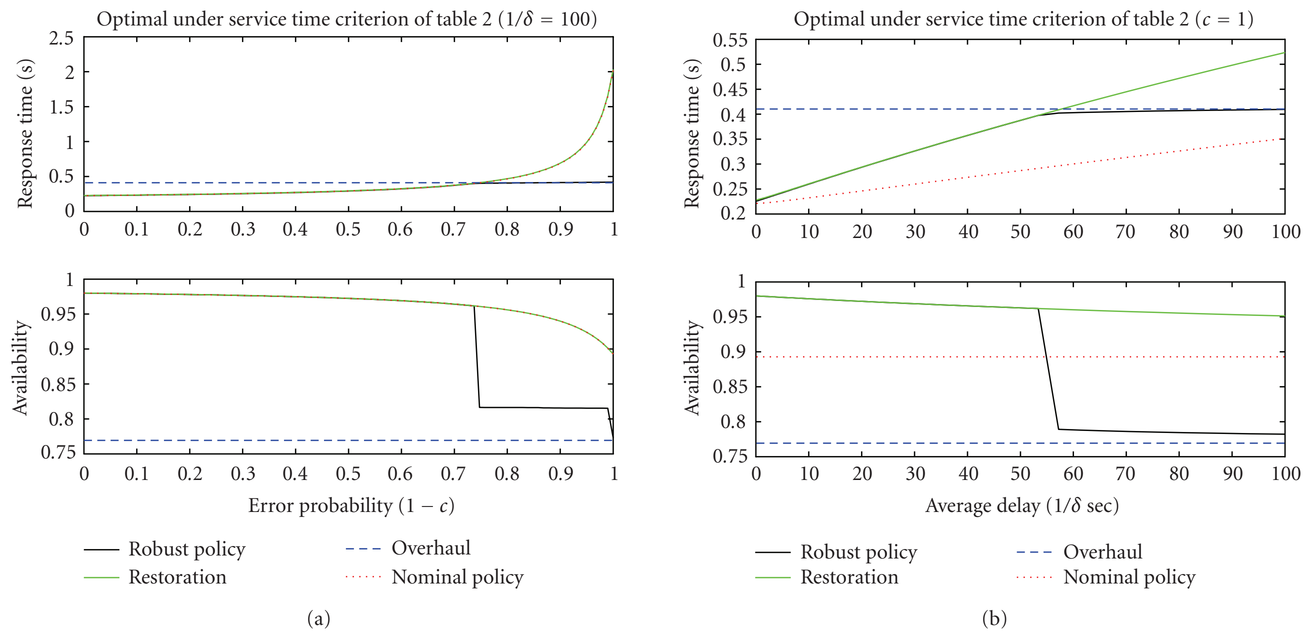


Figure 10: Response time (upper panel) and availability (lower panel) resulting from robust control policy (solid black) and nominal control policy (dotted red) versus (a) decision error $(1 - c)$ and (b) control delay $(1/\delta)$.

Table 2: Discounted service rate with service demand consideration.

Current state x_k	One-step cost $C(x_k, \mathbf{u}_k)$
The database is fully functional	0
One unavailable server has a queue	$1/\mu$
Two unavailable servers have a queue	$2/\mu$
All servers are unavailable and have a queue	$3/\mu$

restore the failed server as long as the server performing the restoration does not have any queries waiting in its queue. The only exception is when three queries are piled into any single queue. In this case, overhaul occurs when the uncertainties are significant.

5.2.2. *Performance under nominal and robust policies, and effect of routing delay*

This subsection examines the system steady-state availability and the expected query response under the robust policy, where random control delay and decision error are explicitly modeled, and under nominal policy where uncertainties are ignored. The results are similar for policies derived with either the queue size criterion (19) or the service time criterion (Table 2). The robust policy shows two distinct features in Figures 10(a) and 10(b); it switches control action when uncertainties (delay and error) becomes significant, and it balances between availability and response time in this situation.

The routing only policy does not attempt to restore the single failed server. Instead, queries are routed to an empty queue whenever the subsequent server contains the data for the query. The system is overhauled upon a second server failure. It offers some advantage in response time over the always-restore policy when there is no routing delay, as shown in Figure 11(a). It is also seen that the robust optimal policy experience improved performance with rerouting authority. However, a routing delay of about one second is significant enough to discourage the use of the routing-only policy, as shown in Figure 11(b).

6. CONCLUSIONS

Uncertainties due to control delays, transmission delays, and decision errors in the distributed database system degrade the performance of the database system performance in terms of availability and response time. Restoration remains to be the optimal policy over a significant range of uncertainties. Beyond boundaries of the range, however, the optimal control policy switches to overhaul. By formulating and solving a Markov decision problem, the robustness of the control policies is investigated. Boundaries for which optimal actions alter are shown to exist and are quantified. The robust policies are shown to provide the best compromise among competing interests.

The authors have also investigated the optimal control policy for the database under the open queuing network setting in the face of delays and errors. Simulations with SimEvents [20] show that response time further depends on the arrival rate of queries. Simulation results will be reported

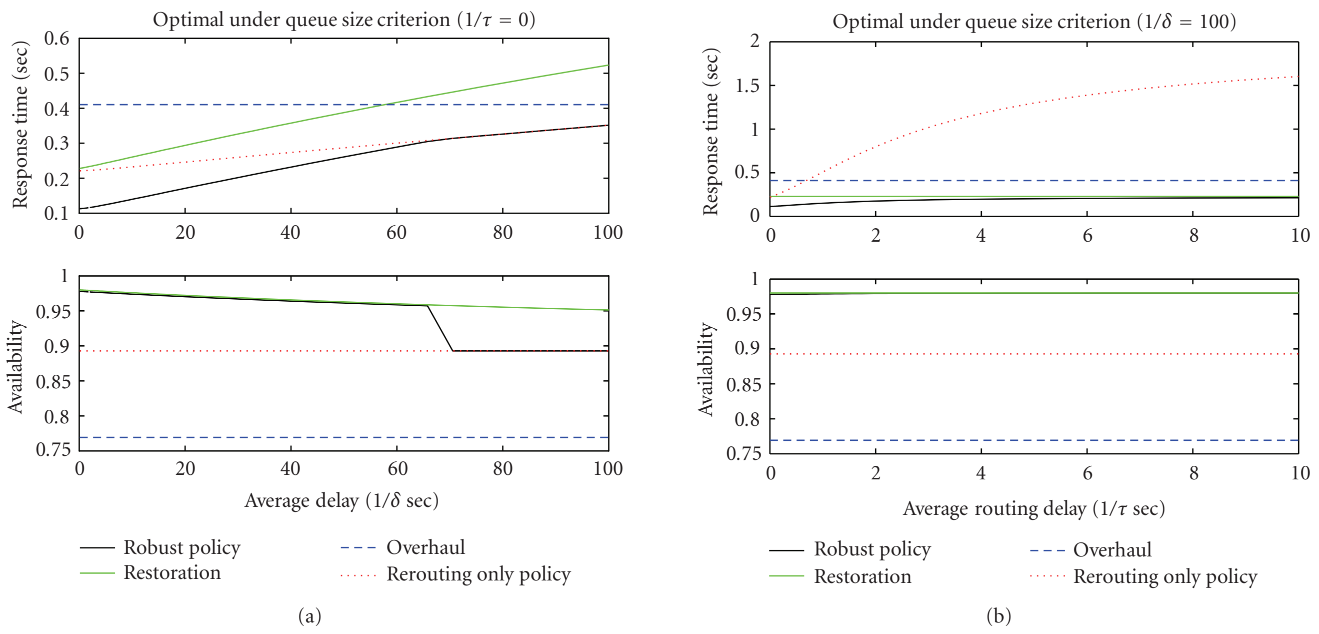


FIGURE 11: Response time (upper panel) and availability (lower panel) resulting from robust control policy (solid black) and routing only policy (dotted red) versus (a) control delay ($1/\delta$) and (b) routing delay ($1/\tau$).

separately. Simulation study of larger networks has also been planned.

ACKNOWLEDGMENT

This work was supported in part by AFOSR under Grants FA9550-06-0456 and FA9550-06-10249.

REFERENCES

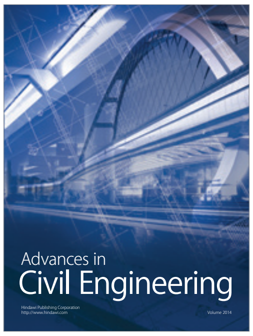
[1] T. Connolly and C. Begg, *Database Solutions: A Step by Step Guide to Building Databases*, Pearson/Addison Wesley, New York, NY, USA, 2nd edition, 2004.
[2] N. E. Wu, J. M. Metzler, and M. H. Linderman, "Supervisory control of a database unit," in *Proceedings of the 44th IEEE Conference on Decision and Control and European Control Conference (CDC-ECC '05)*, pp. 7615–7620, Seville, Spain, December 2005.
[3] L. Kleinrock, *Queueing Systems: Volume 2: Computer Applications*, John Wiley & Sons, New York, NY, USA, 1976.
[4] G. Bolch, S. Greiner, H. de Meer, and K. S. Trivedi, *Queueing Networks and Markov Chains: Modeling and Performance Evaluation with Computer Science Applications*, John Wiley & Sons, New York, NY, USA, 1998.
[5] E. P. C. Kao, *An Introduction to Stochastic Processes*, Duxbury Press, New York, NY, USA, 1997.
[6] J. M. Metzler, "The effect of supervisory control on a redundant database unit," M.S. thesis, Binghamton University, Vestal, NY, USA, 2005.
[7] Arena, Academic Version 7.01.00, Rockwell Software, 2004.
[8] W. D. Kelton, R. P. Sadowski, and D. T. Sturrock, *Simulation with Arena*, McGraw-Hill, New York, NY, USA, 3rd edition, 2004.
[9] D. P. Bertsekas, *Dynamic Programming and Optimal Control*, vol. 1, 2, Athena Scientific, Belmont, Mass, USA, 1995.
[10] C. G. Cassandras and S. Lafortune, *Introduction to Discrete Event Systems*, Kluwer Academic Publishers, Dordrecht, The Netherlands, 1999.
[11] D. Thorsley and D. Teneketzis, "Diagnosability of stochastic discrete-event systems," *IEEE Transactions on Automatic Control*, vol. 50, no. 4, pp. 476–492, 2005.
[12] N. E. Wu, J. M. Metzler, and M. H. Linderman, "Controlled database unit subject to control delays and decision errors," in *Proceedings of the 8th International Workshop on of Discrete Event Systems (WODES '06)*, pp. 131–136, Michigan, Mich, USA, July 2006.
[13] N. E. Wu and T. Busch, "Reconfiguration of C2 architecture for improved availability to support air operations," *IEEE Transactions on Aerospace and Electronics*, vol. 43, no. 2, pp. 795–805, 2007.
[14] A. A. Helal, A. A. Heddaya, and B. K. Bhargava, *Replication Techniques in Distributed Systems*, Kluwer Academic Publishers, Norwell, Mass, USA, 1996.
[15] S. Zacks, *Introduction to Reliability Analysis: Probability Models and Statistics Methods*, Springer, New York, NY, USA, 1992.
[16] S. Wolfram, *Mathematica 5.2*, Wolfram Media, Champaign, Ill, USA, 3rd edition, 2005.
[17] N. E. Wu, "Coverage in fault-tolerant control," *Automatica*, vol. 40, no. 4, pp. 537–548, 2004.
[18] S. Boyd and L. Vandenberghe, *Convex Optimization*, Cambridge University Press, New York, NY, USA, 2004.
[19] MathWorks, "Optimization Toolbox User's Guide, for Use with MATLAB, Version 3," The MathWorks, 2006.
[20] MathWorks, "SimEvents User's Guide, for Use with Simulink," The MathWorks, 2006.

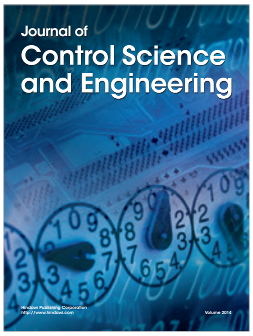
Journal of
Engineering

The Scientific World Journal

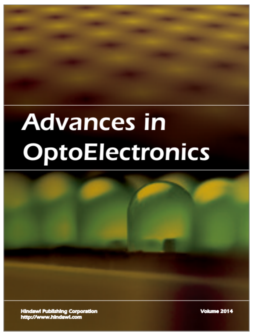
International Journal of
Rotating Machinery

Journal of
Sensors

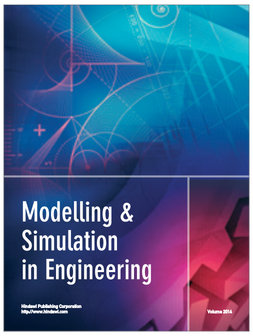
International Journal of
Distributed Sensor Networks

Advances in
Civil Engineering

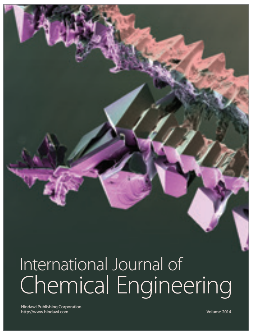
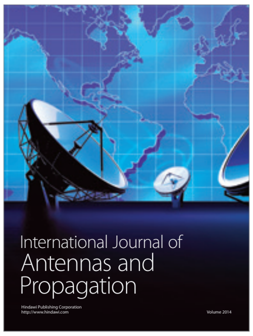

Journal of
Control Science and Engineering

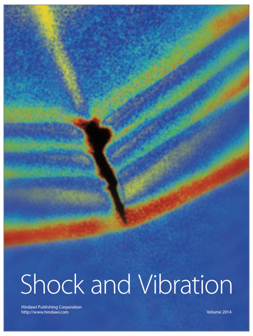
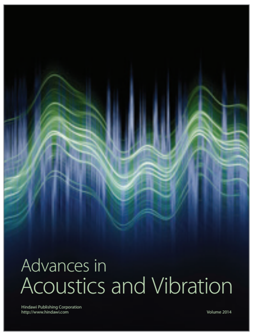
Journal of
Robotics

Journal of
Electrical and Computer Engineering

Advances in
OptoElectronics

VLSI Design

International Journal of
Navigation and Observation

Modelling & Simulation in Engineering

International Journal of
Aerospace Engineering

International Journal of
Chemical Engineering

International Journal of
Antennas and Propagation

Active and Passive Electronic Components

Shock and Vibration

Advances in
Acoustics and Vibration