An Optimal Checkpointing-Strategy for Real-Time Control Systems Under Transient Faults

Seong Woo Kwak, Byung Jae Choi, and Byung Kook Kim, *Member, IEEE*

***Abstract*—Real-time computer systems are often used in harsh environments, such as aerospace, and in industry. Such systems are subject to many transient faults while in operation. Checkpointing enables a reduction in the recovery time from a transient fault by saving intermediate states of a task in a reliable storage facility, and then, on detection of a fault, restoring from a previously stored state. The interval between checkpoints affects the execution time of the task. Whereas inserting more checkpoints and reducing the interval between them reduces the reprocessing time after faults, checkpoints have associated execution costs, and inserting extra checkpoints increases the overall task execution time. Thus, a trade-off between the reprocessing time and the checkpointing overhead leads to an optimal checkpoint placement strategy that optimizes certain performance measures.**

Real-time control systems are characterized by a timely, and correct, execution of iterative tasks within deadlines. The reliability is the probability that a system functions according to its specification over a period of time. This paper reports on the reliability of a checkpointed real-time control system, where any errors are detected at the checkpointing time. The reliability is used as a performance measure to find the optimal checkpointing strategy. For a single-task control system, the reliability equation over a mission time is derived using the Markov model. Detecting errors at the checkpointing time makes reliability jitter with the number of checkpoints. This forces the need to apply other search algorithms to find the optimal number of checkpoints. By considering the properties of the reliability jittering, a simple algorithm is provided to find the optimal checkpoints effectively. Finally, the reliability model is extended to include multiple tasks by a task allocation algorithm.

***Index Terms*—Fault tolerance, optimal checkpointing, real-time control systems, reliability analysis, rollback recovery.**

Acronyms[1]

CkPt	checkpoint
CS	control system
CT	control task

Manuscript received July 3, 1998; revised March 11, 1999, October 5, 1999, April 30, 2000, and July 12, 2000.

Responsible Editor: W-T. K. Chien.

S. W. Kwak is with the Satellite Technology Research Center, Korea Advanced Institute of Science and Technology (KAIST), 373-1 Kusong-dong, Yousong-Gu, Taejon, 305-701 Korea (e-mail: KSW@rtcl.kaist.ac.kr).

B. J. Choi is with the School of Computer and Communication Engineering, Taegu University, Naeri, Jinryang, Kyungsan, Kyungpook, 712-714 Korea (e-mail: BJChoi@taegu.ac.kr).

B. K. Kim is with the Department of Electrical Engineering, Korea Advanced Institute of Science and Technology (KAIST), 373-1 Kusong-dong, Yousong-Gu, Taejon, 305-701 Korea (e-mail: BKKim@ee.kaist.ac.kr).

Publisher Item Identifier S 0018-9529(01)11173-5.

[1]The singular and plural of an acronym are always spelled the same.

I. Introduction

In recent years there has been a dramatic increase in the use of computers to control life-critical systems. For example, computers are used to control spacecraft, airplanes, power distribution systems, nuclear reactors, and chemical plants. Real-time computers deployed in these life-critical control applications must be designed to meet stringent reliability specifications. The maximum acceptable failure rate range for such systems is from 10^{-5} to 10^{-10} per hour [1], which can be accomplished by providing fault-tolerance through hardware and software redundancy.

Faults encountered by CS are either permanent or transient.

- Permanent-faults are caused mainly by physical defects such as breakage, and the faults remain in the system until the component containing the fault is removed.
- Transient-faults are usually caused by environmental changes such as EMI (electro-magnetic interference), or unstable, or marginally stable, hardware. (Intermittent-faults can be included in transient-faults [2].)

Transient faults disappear after an active period [3]. As hardware manufacturing technology continues to improve, the number of permanent faults is gradually decreasing, and instant malfunctions of computers due to transient faults become the main reason for system failures [3], [4]. Coping with transient faults is the key factor in improving the reliability of a CS, and is the focus of this research.

One fault-tolerance technique commonly used for transient faults is a CkPting scheme [5]. In this scheme, as shown in Fig. 1, the intermediate states of a task are saved periodically in a secure device at each CkPt time. If an error is detected, the saved states are restored and the task re-executed from the CkPt (called rollback) [5]–[7]. Thus, CkPting can greatly reduce the probability of incorrect outputs due to transient faults, and so make the CT more reliable.

For a CkPting scheme, it is important to determine the interval between CkPts [5], [7], [8]. Since each task of a real-time CS should be finished correctly within the specified deadline [4], additional CkPting with shorter CkPt intervals can improve reliability. However, because CkPting needs a finite execution time, it causes some processor execution overhead [5], and consequently limits the improvement of reliability.

Several authors have investigated the problem of selecting a CkPt interval that is optimal with respect to other objectives.

- Reference [9] developed a CkPting strategy that maximizes the probability of the critical task completion on a

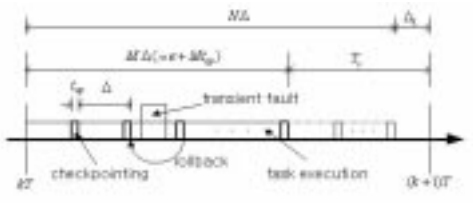


Fig. 1. A CkPted control system.

system with limited repairs. The system considered could undergo failure and repair until the repair time exceeded a specified threshold, at which time the system was deemed to have failed completely. This system is different from a real-time CS, where the execution time of a task, not the repair time, should be maintained within a specified threshold (or deadline). It considered an arbitrary distribution of occurrence and recovery rate of faults. However, the probability so derived, needs very complex computation of convolution integrals and an infinite number of summations.

- Reference [6] developed analytic models for the design and evaluation of CkPting of real-time tasks, and determined an optimal CkPt placement to minimize the mean task-execution time. In the perfect fault-coverage model, it is shown that equidistant inter-CkPt intervals are optimal, assuming a constant fault rate, with the duration of the faults not being considered.
- Reference [10] investigated tradeoffs between configuring a system into duplexes and triplexes given a fixed total number of processors, where the processors use a CkPting scheme. It showed that the duplex–triplex tradeoff is among the effects of permanent, s-independent transient, and s-correlated transient faults. Transient faults were assumed to occur with a Poisson process (constant occurrence rate), and have an arbitrary distribution of duration. Their analysis is represented by complex calculations of integrals, convolutions, and recursive equations.

This paper explores the problem of selecting an optimal CkPt interval for real-time CS that are characterized by timely and correct execution of iterative tasks within deadlines. The duration of a transient fault is considered, as is the reliability which is an objective function for optimizing. Transient faults are assumed to have constant occurrence and recovery rates.

Section II suggests a CS model for a single task.

Section III proposes a method to find the mission-time reliability of a CS which uses the state transition of a CkPt interval.

Section IV proposes a simple searching algorithm to find the optimal number of CkPts that maximizes the mission-time reliability.

Section V presents the corresponding numerical results for Section IV.

Section VI provides a heuristic task allocation algorithm for multiple CT; the algorithm allocates tasks of different execution times and deadlines appropriately for the CkPting strategy of a single task to be applied.

Notation

GCDD	greatest common divisor of deadlines
LCMD	least common multiple of deadlines
T	sampling interval/period of the CT
D	deadline of the CT
D_i	deadline of CT $\#i$
Δ	CkPting interval (the time-slot)
M	number of CkPts in the CT
N	maximum number of allowable time-slots in T
Δ_1	remaining time after N time-slots in T
e	execution time of the CT
e_i	execution time of CT $\#i$
T_c	e/T: computation workload in 1 sampling period
T_s	slack time for rollback
$t_{\rm cp}$	CkPting overhead
λ	fault occurrence rate
μ	fault recovery rate
state 0	fault-free (benign)
state 1	fault-active
state u	CS is under fault-free and not in failure
state d	CS is under fault-active and not in failure
state F	CS is in failure
$\phi_{i,\,j}(\Delta)$	$\Pr\{$in state j at $t=\Delta$ given the initial state i at $t=0$, $i, j \in \{0, 1\}\}$
$\phi_0(\Delta)$	$\Pr\{$no fault within $[0, \Delta]$—initial state 0 at the beginning of the interval$\}$
$\psi_{i,\,j}(r, \Delta)$	$\Pr\{r$ consecutive time-slots are corrupted by transient faults, in which: the initial time-slot is corrupted by a transition from state i to state 1 $(\phi_{i,1}(\Delta))$, the final time-slot is corrupted by a transition from state 1 to state j $(\phi_{1,\,j}(\Delta))$, each of the remaining $(r-2)$ time-slots in the middle are corrupted by a transition from state 1 to state 1 $(\phi_{1,1}(\Delta))\}$, $i, j \in \{0, 1\}$, $r \geq 2$
$p_{\alpha,\,\beta}$	state transition probability of a CS over 1 sampling period from state α to state β, $\alpha, \beta \in \{u, d, F\}$
$\pi_\alpha(k)$	$\Pr\{$in state α at $t = k \cdot T$, $\alpha \in \{u, d, F\}\}$
$R(k \cdot T)$	reliability at $t = k \cdot T$
${}_nC_s$	$n!/(n-s)!$

II. Fault and Control-System Model

Of several fault-analysis models used by previous researchers, the Markov model has been the most frequently used to represent the effect of permanent and transient faults [2], [11]–[13]. We have also used the Markov model in modeling transient faults. In the CkPting process, CkPts were placed in equal length, and for simplicity, noninterruptible system calls were not considered.

A. Basic Assumptions

A1. Transient faults arrive according to a Poisson process with rate, λ, and recover with a rate, μ.

A2. Activation of a transient fault always causes errors.
A3. Errors within a CkPt interval are detected perfectly at the CkPting time.
A4. CkPts can be inserted anywhere in a CT.
A5. The sampling period of a CT is equal to its deadline.
A6. CS is in fault-free state at initial time.

- Assumption A1 is common in many papers dealing with transient faults [2], [4], [11]–[14]. Our research also focuses on transient faults that can be described by assumption A1.
- Assumption A2 is conservative, as faults might not necessarily cause errors.
- Assumption A3 implies that transient faults activated within a CkPt interval can be detected by the CkPting process. Thus, rollback is always performed with the length of a CkPt interval. It is not very difficult to use hardware/software error detection techniques, such as error detection codes, acceptance tests, or comparative results from different processors, to ensure that assumption A3 holds.
- Assumption A4, in practice, might be difficult to accomplish; i.e., it is difficult to maintain equal CkPt intervals. However, analysis at equal CkPt intervals [6], [7], [9] can give an insight into how a CkPt should be inserted for reliability improvement.
- Assumptions A5 and A6 are reasonable.

B. CS Model of Single Task

The transient fault in assumption A1 can be modeled using a 2-state, continuous-time Markov chain of fault-active (state 1) and fault-free (or fault-benign) (state 0). These two states are in Fig. 2.

From the Markov model, $\phi_{i,j}(\Delta)$ can be easily derived [15], [17]. For example,

$$\phi_{0,0}(\Delta) = \frac{\lambda}{\mu+\lambda} \cdot \exp[-(\mu+\lambda)\cdot\Delta] + \frac{\mu}{\mu+\lambda},$$
$$\phi_0(\Delta) = \exp(-\lambda\cdot\Delta).$$

Consider a CS that is characterized by a single CT. This CS reads sensor values, computes control inputs using a suitable algorithm, and outputs the results to plants. These steps are repeated in each fixed T [1]. Thus, a mission of the CS consists of sampling periods. In an environment of transient faults, each sampling period of the CS can be represented by 3 states: u, d, F; see Fig. 2.

- State u (d) at $t = k\cdot T$, means that the CT is executed correctly and finished within the sampling period $[(k-1)\cdot T, k\cdot T]$, and transient faults are in fault-free (fault-active) at $t = k\cdot T$.
- State F at $t = k\cdot T$ means that the CT is either executed incorrectly or not finished within the sampling period $[(k-1)\cdot T, k\cdot T]$. Because the CS of a single CT evolves with sampling periods $t = k\cdot T$, $k = 1, 2\ldots$ (see Fig. 2), it can be modeled by a 3-state Markov chain, as shown in Fig. 3.

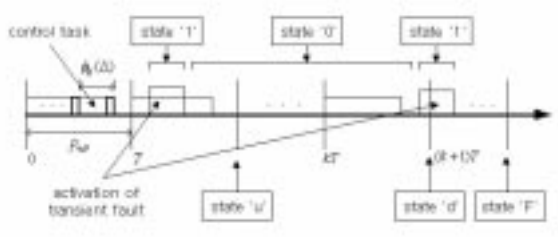


Fig. 2. State description of a fault and a CS.

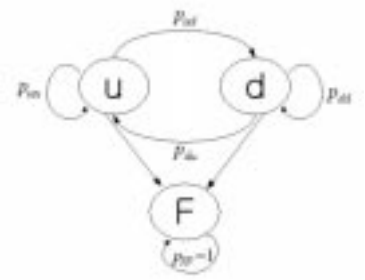


Fig. 3. A sampling interval of a real-time CS.

III. Reliability Analysis

The reliability of a CS is the probability that the CT is executed correctly within its deadline over a specified time interval (the mission). Thus, the reliability at $t = k\cdot T$ is the probability that: the Markov chain in Fig. 3 is in state u or d at $t = k\cdot T$. The stochastic states of the Markov chain can be derived using the knowledge of $p_{\alpha,\beta}$, $\alpha, \beta \in \{u, d, F\}$. For a CkPting CS, $p_{\alpha,\beta}$ depends on the number of CkPts in a CT (or the CkPt interval), the $t_{\rm cp}$, the e, the D, and the λ and μ. The derivation of $p_{\alpha,\beta}$ needs the following calculations.

When M CkPts are placed in e, the Δ and T_s are obtained using:

$$\Delta = \frac{e}{M} + t_{\rm cp}, \qquad T_s = T - M\cdot\Delta.$$

The N and Δ_1 are obtained using:

$$N = \text{gilb}\left(\frac{T}{\Delta}\right),$$
$$\Delta_1 = T - N\cdot\Delta, \qquad 0 \le \Delta_1 < \Delta.$$

Owing to the CkPt overhead, the execution time of a CT is increased by $M\cdot t_{\rm cp}$ after placing M CkPts. Fig. 1 schematically shows Δ, T_s, N, and Δ_1. For the successful execution of a CT, at least M time-slots must be fault-free in the N time-slots available for one T; i.e., the CkPted CS can overcome faults in up to $(N-M)$ time-slots. A time-slot is corrupted by transient faults when there is any activation of faults within the length, Δ.

Fig. 4 illustrates $\psi_{i,j}(r, \Delta)$, $r \ge 2$. For $r = 1$, let

$$\psi_{0,1}(r, \Delta) = \phi_{0,1}(\Delta),$$
$$\psi_{1,0}(r, \Delta) = \phi_{1,0}(\Delta),$$
$$\psi_{1,1}(r, \Delta) = \phi_{1,1}(\Delta).$$

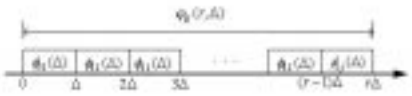


Fig. 4. A transition probability of $\phi_{i,j}(r, \Delta)$.

Then, $\psi_{i,j}(r, \Delta)$ are derived as follows:

$$\begin{aligned}
\psi_{0,0}(r, \Delta) &= \phi_{0,1}(\Delta) \cdot \phi_{1,1}^{r-2}(\Delta) \cdot \phi_{1,0}(\Delta), && r \geq 2;\\
\psi_{0,1}(r, \Delta) &= \phi_{0,1}(\Delta) \cdot \phi_{1,1}^{r-1}(\Delta), && r \geq 1;\\
\psi_{1,0}(r, \Delta) &= \phi_{1,1}^{r-1}(\Delta) \cdot \phi_{1,0}(\Delta), && r \geq 1;\\
\psi_{1,1}(r, \Delta) &= \phi_{1,1}^{r}(\Delta), && r \geq 1.
\end{aligned}$$

Lemmas 1 and 2 are derived using $\psi_{i,j}(r, \Delta)$.

Lemma 1: The probability, $\zeta_{0,0}(s, n, \Delta)$, that an interval consisting of n time-slots, with interval length $n \cdot \Delta$, moves from state 0 at the beginning of the interval into state 0 at the end of the interval, containing s corrupted time-slots in which at least 1 corruption pattern of $\psi_{0,0}(\cdot)$ exists, is:

$$\zeta_{0,0}(s, n, \Delta) = \begin{cases} \displaystyle\sum_{l=1}^{\text{gilb}(s/2)} \sum_{\Omega_l} \Xi_1(l) \cdot \Xi_2(l) & \text{if } s \geq 2,\\ 0 & \text{otherwise;} \end{cases}$$

$$\Xi_1(l) \equiv \frac{(n+l-r)!}{(1!)^{q_1} \cdot (2!)^{q_2} \cdots (l!)^{q_l} \cdot (s-r)! \cdot (n-s)!},$$

$$\Xi_2(l) \equiv \prod_{j=1}^{l} \psi_{0,0}(r_j, \Delta) \cdot [\phi_{0,0}(\Delta) - \phi_0(\Delta)]^{s-r} \cdot \phi_0(\Delta)^{n-s},$$

$$l \equiv \text{number of } \psi_{0,0}(\cdot) \text{ patterns},$$

$$\Omega_l = \left\{ (r_1, r_2, \ldots, r_l) \middle| \sum_{j=1}^{l} r_j \leq s,\ r_1 \geq r_2, \ldots \geq r_l \geq 2 \right\},$$

$$r = \sum_{j=1}^{l} r_j,$$

$$I_l = \{(1, r_1), (2, r_2), \ldots, (l, r_l)\};$$

$$\hat{I}_l = \{(c_v, v) | c_v = |\{(i, v) | (i, v) \in I_l\}|\};$$

$$\hat{u}\text{-}tuple = \Big\{ (v_1, v_2, \ldots, v_u) | v_1 = v_2 = v_3 = \cdots = v_u = v,\ (u, v) \in \hat{I}_l \text{ for } u < v,\ (v, v) \notin \hat{I}_l \Big\}$$

q_u: size of $\hat{u}$-$tuple$.

Proof: See Appendix A.

Lemma 2: The probability, $\eta_{i,j}(s, n, \Delta, \Delta_1)$, $(i, j \in \{0, 1\})$, that an interval consisting of n time-slots and a final slot of length Δ_1, with interval length $n \cdot \Delta + \Delta_1$, moves from state i at the beginning of the interval to state j at the end of the interval containing s corrupted time-slots is:

$$\begin{aligned}
\eta_{0,0}(s, n, \Delta, \Delta_1) =& {}_nC_s \cdot \Xi_3(\Delta)^s \cdot \phi_0(\Delta)^{n-s} \cdot \phi_{0,0}(\Delta_1)\\
&+ \sum_{r=1}^{s} {}_{n-r}C_{s-r} \cdot \Xi_3(\Delta)^{s-r} \cdot \phi_0(\Delta)^{n-s}\\
&\cdot \psi_{0,1}(r, \Delta) \cdot \phi_{1,0}(\Delta_1) + \zeta_{0,0}(s, n, \Delta)\\
&\cdot \phi_{0,0}(\Delta_1) + \sum_{r=1}^{s-2} \zeta_{0,0}(s-r, n-r, \Delta)\\
&\cdot \psi_{0,1}(r, \Delta) \cdot \phi_{1,0}(\Delta_1), \qquad s \geq 0\\
\Xi_3(\Delta) \equiv& \phi_{0,0}(\Delta) - \phi_0(\Delta)\\
\eta_{0,1}(s, n, \Delta, \Delta_1) =& \sum_{r=1}^{s} \eta_{0,0}(s-r, n-r, \Delta, 0) \cdot \psi_{0,1}(r, \Delta)\\
&\cdot \phi_{1,1}(\Delta_1) + \eta_{0,0}(s, n, \Delta, 0)\\
&\cdot \phi_{0,1}(\Delta_1), \qquad s \geq 0\\
\eta_{1,0}(s, n, \Delta, \Delta_1) =& \begin{cases} \Xi_4(s; 0) & 1 \leq s < n\\ \Xi_4(n; 0) + \psi_{1,1}(n, \Delta) \cdot \phi_{1,0}(\Delta_1), & s = n \end{cases}\\
\eta_{1,1}(s, n, \Delta, \Delta_1) =& \begin{cases} \Xi_4(s; 1) & 1 \leq s < n\\ \Xi_4(n; 1) + \psi_{1,1}(n, \Delta) \cdot \phi_{1,1}(\Delta_1), & s = n \end{cases}\\
\Xi_4(x; j) \equiv& \sum_{r=1}^{x} \psi_{1,0}(r, \Delta) \cdot \eta_{0,j}(s-r, n-r, \Delta, \Delta_1).
\end{aligned}$$

Proof: See Appendix B.

Now, the state transition probabilities, $p_{\alpha,\beta}$, $(\alpha, \beta \in \{u, d, F\})$, can be easily obtained using the $\eta_{i,j}(s, n, \Delta, \Delta_1)$ of lemma 2. Because the CkPted CS can execute the CT successfully, even under faults in up to $(N - M)$ time-slots, the $p_{\alpha,\beta}$ are represented as:

$$\begin{aligned}
p_{u,u} &= \sum_{s=0}^{N-M} \eta_{0,0}(s, N, \Delta, \Delta_1),\\
p_{u,d} &= \sum_{s=0}^{N-M} \eta_{0,1}(s, N, \Delta, \Delta_1),\\
p_{d,u} &= \sum_{s=0}^{N-M} \eta_{1,0}(s, N, \Delta, \Delta_1),\\
p_{d,d} &= \sum_{s=0}^{N-M} \eta_{1,1}(s, N, \Delta, \Delta_1).
\end{aligned}$$

Other transition probabilities are:

$$\begin{aligned}
&p_{u,F} = 1 - p_{u,u} - p_{u,d}, \quad p_{d,F} = 1 - p_{d,u} - p_{d,d},\\
&p_{F,u} = p_{F,d} = 0, \quad p_{F,F} = 1.
\end{aligned}$$

By defining the state vector $\Pi(k) = [\pi_u(k), \pi_d(k), \pi_F(k)]^T$, the solution of the Markov model in Fig. 3 is:

$$\Pi(k) = P^k \cdot \Pi(0), \qquad P \equiv \begin{bmatrix} p_{u,u} & p_{d,u} & 0\\ p_{u,d} & p_{d,d} & 0\\ p_{u,F} & p_{d,F} & 1 \end{bmatrix}. \tag{1}$$

Because the reliability at $t = k \cdot T$ is $\pi_u(k) + \pi_d(k)$, then $R(k \cdot T) = C \cdot P^k \cdot \Pi(0)$, $C \equiv [1\ 1\ 0]$, $\Pi(0) = [1\ 0\ 0]^T$.

IV. Optimal Checkpointing

The system reliability varies greatly, depending upon the CkPt interval. Thus, the CkPt interval should be appropriately chosen when designing a CkPted CS. This section provides a simple algorithm to find the optimal number of CkPts, which maximizes the mission-time reliability.

Proposition 1: When

$$(p_{u,\,u} - p_{d,\,d})^2 + 4p_{u,\,d}p_{d,\,u} \neq 0,$$

the solution of (1), is:

$$\begin{aligned} R(k \cdot T) &= \frac{p_{u,\,d}}{\zeta(u,\,d)}\,(\Xi_5 - \Xi_6); \\ \Xi_5 &\equiv \left(\frac{p_{u,\,u} + p_{d,\,d} + \zeta(u,\,d)}{2}\right)^k \\ &\quad \cdot \left(1 - \frac{2p_{d,\,u}}{p_{u,u} - p_{d,\,d} - \zeta(u,\,d)}\right), \\ \Xi_6 &\equiv \left(\frac{p_{u,\,u} + p_{d,\,d} - \zeta(u,\,d)}{2}\right)^k \\ &\quad \cdot \left(1 - \frac{2p_{d,\,u}}{p_{u,\,u} - p_{d,\,d} + \zeta(u,\,d)}\right), \\ \zeta(u,\,d) &\equiv \sqrt{(p_{u,\,u} - p_{d,\,d})^2 + 4p_{u,\,d} \cdot p_{d,\,u}}. \end{aligned} \tag{2}$$

Proof: See Appendix C.

The CkPt interval depends on the number of CkPts within the CT. Thus, to find the CkPt interval, it is sufficient to know the number of CkPts in the CT. However, it is very difficult to find a closed-form solution for the optimal number of CkPts, because $R(k \cdot T)$ is represented by a complicated variation of (2).

Generally, if the CkPting overhead is zero, the reliability increases in proportion to the number of CkPts. In this case, an infinite number of CkPts are the best. However, when the CkPting overhead is nonzero, there is a finite optimal number of CkPts. When errors are detected at the CkPting time, reliabilities jitter along with the number of CkPts, even though the reliability trend increases (or decreases) before (or after) the optimal number of CkPts.

The jittering phenomenon is caused by: the rollback is always performed with the length of a CkPt interval. Hence, the number of CkPts available in the slack-time, and not the slack-time, is more closely related to the rollback capability (or reliability). Because the number of CkPts available in the slack-time increases in a stepwise fashion according to the number of CkPts in the CT, the reliability decreases in the flat-step region as the number of CkPts is increased, due to the CkPting overhead. The reliability increases at the step-jump, because the number of CkPts available in the slack-time increases at these points. This makes reliability jitter according to the number of CkPts. When detecting an error at its occurrence, there is no jittering.

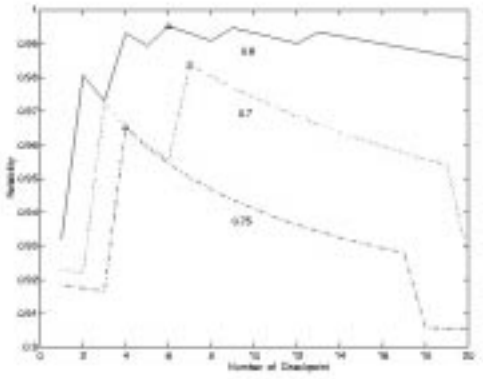


Fig. 5. Reliability *vs* T_c and CkPts.

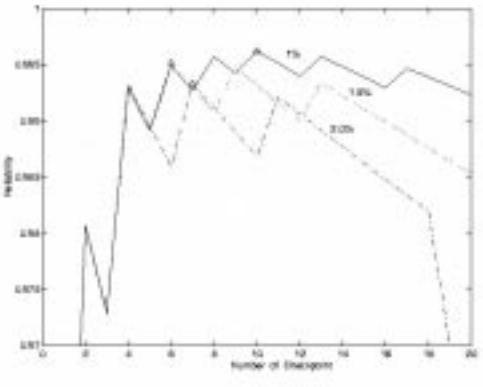


Fig. 6. Reliability *vs* $t_{\rm cp}$ and CkPts.

The jittering phenomenon can be seen in the simulation examples of Figs. 5–8 (see Section V). The jittering reliability makes it difficult to apply other search algorithms to find the optimal number of CkPts. A simple peak-detection algorithm is given here for this problem. This peak-detection algorithm:

a) uses the property that the envelope (the piece-wise linear curve connecting the local maximum points of the reliability curve) of jittering reliability is a convex function of the number of CkPts;
b) seeks peak reliability values until the next local peak after the maximum peak is detected;
c) determines the optimal number of CkPts to maximize reliability.

`Peak-Detection Algorithm`

Notation

R_M reliability after a mission when M CkPts are placed in the CT

$M_{\max}$ maximum allowable number of CkPts

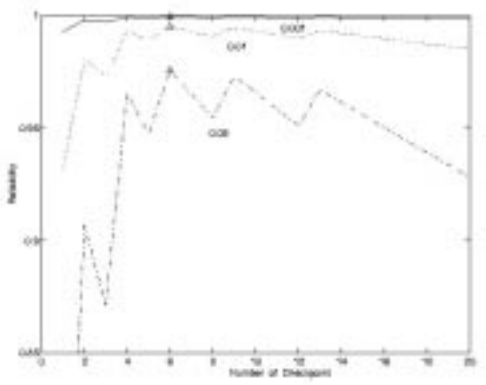
For a nonzero CkPting overhead,
$M_{\max} = \text{gilb}(T - e/t_{\rm cp})$.

`Algorithm Steps`

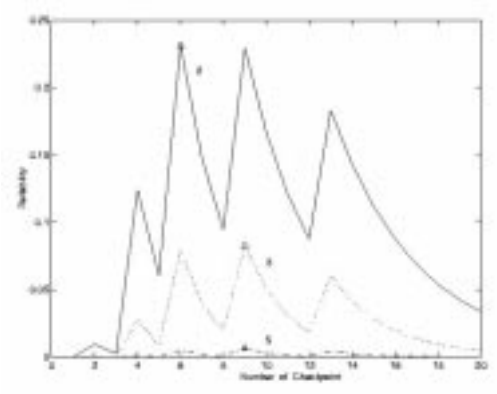
$\text{Max\_R} = \text{Peak\_R} = \max(R_1,\, R_2)$;

$\text{Max\_M} = \begin{cases} 1 & \text{if } R_1 \geq R_2 \\ 2 & \text{otherwise} \end{cases}$

$M = 3$;

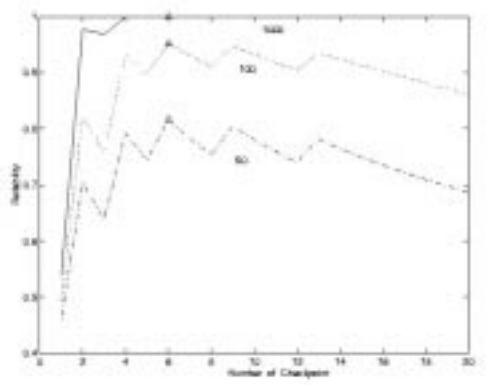


(a)

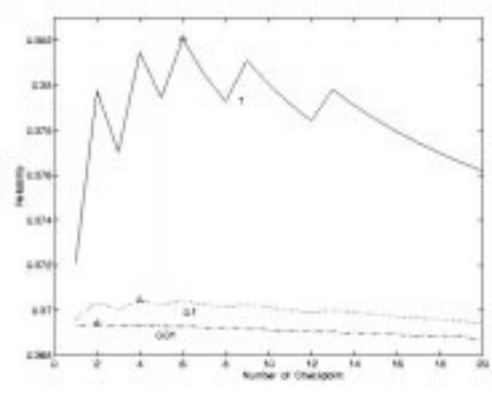


(b)

Fig. 7. Reliability *vs* λ and CkPts.



(a)



(b)

Fig. 8. Reliability *vs* μ and CkPts.

```
While (Maximum-Peak not detected) and (M ≤ M_max) Do
  If (R_{M-1} > R_{M-2}) and (R_{M-1} > R_M) Then
    Peak_R = R_{M-1}; /* peak detected */
    If (Max_R > Peak_R) Then
      Maximum Peak detected;
      CONTINUE;
      End_If
    Max_R = Peak_R; /* save local peak */
    Max_M = M - 1; /* save index of local peak */
    End_If
  M = M + 1;
  End_While
Optimal_Number=Max_M;/* optimal number of checkpoints */
  End_Algorithm
```

V. Numerical Results

Fig. 5 shows how the system reliability varies with workload (T_c) and the number of CkPts.

Assumptions:

1) The CkPt overhead is 1.5% of c.
2) The sampling period of T is 0.1.
3) The mission time is $100T$.
4) $\lambda = 0.01$; $\mu = 100$.

The system reliability improves at first, as the number of CkPts is increased, but later drops off because "more CkPts" means more "CkPting overhead." Also, the system reliability decreases with an increase in workload because of less slack time. Fig. 5 illustrates these facts. The optimal number of CkPts can be obtained using the Algorithm in Section IV (marked by a triangle in Fig. 5).

Fig. 6 shows the variation of reliability vs the CkPting overhead. The reliability decreases with increasing overhead, because the actual workload increases due to the CkPting overhead.

Figs. 7 and 8 show how the system reliability varies with λ and μ, respectively.

Fig. 7(a) shows that the optimal number of CkPts is nearly s-independent of λ for a range of occurrence rates, where the probability of more than 1 fault in T is negligible, e.g., $\lambda = 0.001, 0.01, 0.05$ or 2. However, for very frequent fault occurrences where the probability of more than one fault within T can not be neglected, e.g., $\lambda = 3$ or 5, the optimal number of CkPts varies with λ, as shown in Fig. 7(b).

The reliability decreases with increasing $1/\mu$. Fig. 8(a) and (b) show the system reliability at various values of μ. The optimal number of CkPts varies with μ, even though it does not heavily depend on μ (as shown in Fig. 8).

In summary, the optimal number of CkPts depends on c, T, $t_{\rm cp}$, λ, μ.

VI. Extension to Multiple Tasks

CS usually have multiple CT that are characterized by different execution times and deadlines. In CkPting such systems,

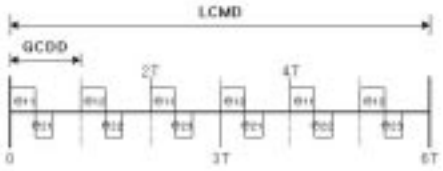


Fig. 9. Task allocation with 2 different deadlines.

CT should be allocated in appropriate order to improve reliability. This section extends the results in selecting the number of CkPts, to the case of multiple CT, by providing a task-allocation algorithm.

Assumptions:

1) Sufficient time redundancy is available for the efficiency of the CkPting scheme.
2) CT can be sliced to an arbitrary length.
3) CT deadlines are equal to periods.

`Task Allocation Algorithm`
1. Find the GCDD of D_i.
2a. Divide each CT, e, into $n_i = D_i/GCDD$ subtasks with equal length e_i/n_i: $e_{i,1}, e_{i,2}, \ldots, e_{i,n_i}$.
2b. Assign each subtask sequentially on each GCDD interval with priority (usually earlier deadline first).
2c. After allocating all CT repeatedly until the LCMD, the resultant allocation pattern is repeated for each LCMD interval.
`End_Algorithm`

This algorithm gives the CT allocation, assuming the worst fault occurrence case to have occurred. For a system with 2 CT, $D_1 = 2T$ and $D_2 = 3T$, this algorithm provides an allocation of CT during the LCMD, as shown in Fig. 9. Because a deadline consists of GCDD, the interval of a GCDD can be used as a basic interval for reliability analysis, similar to a sampling period for a single CT. Because GCDD are made to have the same workload by the Task-Allocation algorithm, the identical CkPting scheme as for the single CT can be applied to all GCDD. The Markov model of Fig. 3 can be used to evaluate the reliability. This evolves with each GCDD interval in this case. The reliability is measured in multiple LCMD because all CT complete their execution at these points. For simplicity, task rescheduling during I/O operations is not considered.

Appendix A
Proof of Lemma 1

Obviously, $\zeta_{0,0}(s, n, \Delta) = 0$ for $s < 2$. Since the corruption pattern $\psi_{0,0}(\cdot)$ contains at least 2 time-slots, the maximum number of groups of $\psi_{0,0}(\cdot)$ corresponding to s corrupted time-slots is $\mathrm{gilb}(s/2)$. Each $\psi_{0,0}(\cdot)$ group can have 2 to s corrupted time-slots, and the total number of corrupted time-slots (r) by the groups should not exceed s. Thus, considering l, $(1 \le l \le \mathrm{gilb}(s/2))$, groups of $\psi_{0,0}(\cdot)$, the set of the vector whose element (r_j) represents the number of corrupted time-slots in each $\psi_{0,0}(\cdot)$ group j, is:

$$\Omega_l = \left\{ (r_1, r_2, \ldots, r_l) \,\middle|\, \sum_{j=1}^{l} r_j \le s, r_1 \ge r_2, \ldots, \ge r_l \ge 2 \right\}.$$

The index for l groups is assigned in decreasing order of corrupted time-slots. The probability that l groups of $\psi_{0,0}(\cdot)$ contain the number of corrupted time-slots indicated in the set Ω_l is:

$$\sum_{\Omega_l} \prod_{j=1}^{l} \cdot \psi_{0,0}(r_j, \Delta).$$

From the definition of $\phi_{i,j}(\Delta)$ and $\phi_0(\Delta)$, the probability that a time-slot is corrupted only by the faults that occur and disappear within the time-slot is: $\phi_{0,0}(\Delta) - \phi_0(\Delta)$.

Notation:

$\hat{u}$-*tuple* set whose element is composed of $u-tuples$ of the same r_j in $(r_1, r_2, \ldots, r_l)$

q_u size (number of elements) of $\hat{u}$-*tuple*.

For example, in $(r_1, r_2, \ldots, r_8) = (2, 3, 3, 4, 4, 5, 5, 5)$, there are 3 $\hat{u}$-*tuple*:

$$\hat{1}\text{-}tuple = \{2\},$$
$$\hat{2}\text{-}tuple = \{(3, 3), (4, 4)\},$$
$$\hat{3}\text{-}tuple = \{(5, 5, 5)\}.$$

Thus, $q_1 = 1$, $q_2 = 2$, $q_3 = 1$ in this example. Then there are

$$\frac{(n+l-r)!}{(1!)^{q_1} \cdot (2!)^{q_2} \cdots (l!)^{q_l} \cdot (s-r)! \cdot (n-s)!}$$

combinations for ordering l groups of $\psi_{0,0}(\cdot)$, $(s-r)$ time-slots of $\phi_{0,0}(\Delta) - \phi_0(\Delta)$, $(n-s)$ time-slots of $\phi_0(\Delta)$. Therefore, the probability that: r time-slots are corrupted by l groups of $\psi_{0,0}(\cdot)$, $(s-r)$ time-slots are corrupted by the transition pattern of $\phi_{0,0}(\Delta) - \phi_0(\Delta)$, the other $(n-s)$ time-slots are fault-free, is:

$$\sum_{\Omega_l} \zeta_{1,l} \cdot \zeta_{2,l};$$

$$\zeta_{1,l} \equiv \frac{(n+l-r)!}{(1!)^{q_1} \cdot (2!)^{q_2} \cdots (l!)^{q_l} \cdot (s-r)! \cdot (n-s)!},$$

$$\zeta_{2,l} \equiv \prod_{j=1}^{l} \psi_{0,0}(r_j, \Delta) \cdot [\phi_{0,0}(\Delta) - \phi_0(\Delta)]^{s-r} \cdot \phi_0(\Delta)^{n-s}.$$

The sum of these probabilities for l gives $\zeta_{0,0}(s, n, \Delta)$.

Appendix B
Proof of Lemma 2

Four probability terms constitute $\eta_{0,0}(s, n, \Delta, \Delta_1)$.

Term #1 is the probability that s time-slots are all corrupted by the pattern of $\phi_{0,0}(\Delta) - \phi_0(\Delta)$; the other $(n-s)$ time-slots are fault-free. The final slot of length Δ_1, might or might not be corrupted by faults, but it should transit from state 0 into state

0, *viz*, with the transition pattern of $\phi_{0,0}(\Delta_1)$. This is because the final time-slot always ends with state 0, and the $n \cdot \Delta + \Delta_1$ interval ends with state 0. The number of combinations to choose s corrupted time-slots among n time-slots is ${}_nC_s$. Thus, the probability is:

$$ {}_nC_s \cdot [\phi_{0,0}(\Delta) - \phi_0(\Delta)]^s \cdot \phi_0(\Delta)^{n-s} \cdot \phi_{0,0}(\Delta_1).$$

Term #2 is the probability that the final consecutive r time-slots are corrupted by the transition pattern of $\psi_{0,1}(r, \Delta)$, $(s-r)$ time-slots are corrupted by the pattern of $\phi_{0,0}(\Delta) - \phi_0(\Delta)$, the other $(n-s)$ time-slots are fault-free. The final slot of length Δ_1 begins with state 1, because of the 1 state at the end of the pattern $\psi_{0,1}(r, \Delta)$, and ends with state 0 because of the 0 state at the end of the interval, $n \cdot \Delta + \Delta_1$, i.e., the final slot has the transition pattern of $\phi_{0,0}(\Delta_1)$. There are ${}_{n-r}C_{s-r}$ combinations to choose $(s-r)$ corrupted time-slots of pattern $\phi_{0,0}(\Delta) - \phi_0(\Delta)$ among $(n-r)$ time-slots. Thus, the probability is:

$$\left[\sum_{r=1}^{s} {}_{n-r}C_{s-r} \cdot [\phi_{0,0}(\Delta) - \phi_0(\Delta)]^{s-r} \cdot \psi_{0,1}(r, \Delta)\right] \cdot \phi_0(\Delta)^{n-s} \cdot \phi_{1,0}(\Delta_1).$$

Term #3 is the probability that s time-slots are corrupted by at least 1 corruption pattern of $\psi_{0,0}(\cdot)$. The final slot, of length Δ_1, should have the transition pattern of $\phi_{0,0}(\Delta_1)$. From lemma 1, this probability is: $\zeta_{0,0}(s, n, \Delta) \cdot \phi_{0,0}(\Delta_1)$.

Term #4 is the probability that the final consecutive r time-slots are corrupted by the pattern of $\psi_{0,1}(r, \Delta)$, and $(s-r)$ time-slots among the other $(n-r)$ time-slots are corrupted by at least 1 corruption pattern of $\psi_{0,0}(\cdot)$. The final slot of length Δ_1 should have the transition pattern of $\phi_{1,0}(\Delta_1)$. Using $\zeta_{0,0}(\cdot)$, the probability is:

$$\sum_{r=1}^{s-2} \zeta_{0,0}(s-r, n-r, \Delta) \cdot \psi_{0,1}(r, \Delta) \cdot \phi_{1,0}(\Delta_1).$$

The sum of the 4 terms is: $\eta_{0,0}(s, n, \Delta, \Delta_1)$. This completes the proof of $\eta_{0,0}(s, n, \Delta, \Delta_1)$.

$\eta_{0,1}(s, n, \Delta, \Delta_1)$ can be derived easily by using $\eta_{0,0}(s, n, \Delta, \Delta_1)$. Consider two cases for the corrupted time-slots in $\eta_{0,1}(s, n, \Delta, \Delta_1)$:

1) the final r consecutive time-slots are corrupted by the pattern of $\psi_{0,1}(\cdot)$;
2) there is no final consecutive pattern of $\psi_{0,1}(\cdot)$.

The probability of case 1 can be derived by using $\eta_{0,0}(\cdot)$:

$$\sum_{r=1}^{s} \eta_{0,0}(s-r, n-r, \Delta, 0) \cdot \psi_{0,1}(r, \Delta) \cdot \phi_{1,1}(\Delta_1).$$

The probability of case 2 is: $\eta_{0,0}(s, n, \Delta, 0) \cdot \phi_{0,1}(\Delta_1)$. Thus, the sum of these 2 probabilities is: $\eta_{0,1}(s, n, \Delta, \Delta_1)$. This completes the proof of $\eta_{0,1}(s, n, \Delta, \Delta_1)$.

Because the beginning state is 1 in $\eta_{1,0}(s, n, \Delta, \Delta_1)$, the first time-slot should begin with state 1. Thus, $\eta_{1,0}(s, n, \Delta, \Delta_1)$ for $1 \le s < n$ can be represented by the probability that the first r consecutive time-slots are corrupted by the transition pattern of $\psi_{1,0}(\cdot)$, and the other $(s-r)$ time-slots are corrupted by the pattern of $\eta_{0,0}(\cdot)$:

$$\sum_{r=1}^{s} \psi_{1,0}(r, \Delta) \cdot \eta_{0,0}(s-r, n-r, \Delta, \Delta_1).$$

For $s = n$, all n time-slots can be corrupted by the pattern of $\psi_{1,1}(\cdot)$. Thus, this term should be added to the previous equation, which gives:

$$\sum_{r=1}^{n} \psi_{1,0}(r, \Delta) \cdot \eta_{0,0}(s-r, n-r, \Delta, \Delta_1) + \psi_{1,1}(n, \Delta) \cdot \phi_{1,0}(\Delta_1).$$

This completes the proof of $\eta_{1,0}(s, n, \Delta, \Delta_1)$.

Similarly $\eta_{1,1}(s, n, \Delta, \Delta_1)$ can be proved.

Appendix C
Solution of (1)

Since the transition probability $p_{u,d}$ and $p_{d,u}$ cannot be 0 when $\mu \neq \infty$ and $\Delta \neq 0$, then $(p_{u,u} - p_{d,d})^2 + 4p_{u,d} \cdot p_{d,u} \neq 0$.

Consider the system:

$$\mathbf{X}(k) = \mathbf{A} \times \mathbf{X}(k-1).$$

Let the eigenpairs of $\mathbf{A}$ be (λ_1, u_1), (λ_2, u_2), ..., (λ_m, u_m), then the solution of this difference equation is [16]:

$$\begin{aligned} \mathbf{X}(k) &= \sum_{i=1}^{m} c_i \cdot \lambda_i^k \cdot u_i \\ \mathbf{C} &\equiv [c_1 \;\; c_2 \;\; \cdots \;\; c_m]^T = \mathbf{S}^{-1} \times \mathbf{X}(0), \\ \mathbf{S} &\equiv [u_1 \;\; u_2 \;\; \cdots \;\; u_m]. \end{aligned} \tag{3}$$

For

$$\mathbf{A} = \begin{bmatrix} a_{1,1}a_{1,2} \\ a_{2,1}a_{2,2} \end{bmatrix}, \qquad (a_{1,1} - a_{2,2})^2 + 4a_{1,2}a_{2,1} \neq 0,$$

the eigenvalues of $\mathbf{A}$ are

$$\lambda_1 = \frac{a_{1,1} + a_{2,2} + \sqrt{(a_{1,1} - a_{2,2})^2 + 4a_{1,2}a_{2,1}}}{2}$$
$$\lambda_2 = \frac{a_{1,1} + a_{2,2} - \sqrt{(a_{1,1} - a_{2,2})^2 + 4a_{1,2}a_{2,1}}}{2}.$$

The corresponding eigenvectors are:

$$u_i = \begin{bmatrix} -\dfrac{a_{1,2}}{a_{1,1} - \lambda_i} \\ 1 \end{bmatrix}, \qquad \text{for } i = 1, 2. \tag{4}$$

Thus, $\mathbf{C} = \mathbf{S}^{-1} \times \mathbf{X}(0)$ is:

$$\mathbf{C} = \mathbf{S}^{-1} \times \mathbf{X}(0) = [\zeta \quad -\zeta],$$
$$\zeta \equiv \frac{(a_{1,1} - \lambda_2) \cdot (a_{1,1} - \lambda_1)}{a_{1,2} \cdot (\lambda_2 - \lambda_1)}. \tag{5}$$

From (3)–(5):

$$X_1(k) = \zeta \cdot \lambda_1^k \cdot \frac{-a_{1,2}}{a_{1,1} - \lambda_1} - \zeta \cdot \lambda_2^k \cdot \frac{-a_{1,2}}{a_{1,1} - \lambda_2}, \tag{6}$$
$$X_2(k) = \zeta \cdot \lambda_1^k - \zeta \cdot \lambda_2^k. \tag{7}$$

The reliability at $t = k \cdot T$ is:

$$R(k \cdot T) = \pi_u(k) + \pi_d(k).$$

By replacing

$$a_{1,1} \text{ with } p_{u,u},$$
$$a_{1,2} \text{ with } p_{d,u},$$
$$a_{2,1} \text{ with } p_{u,d},$$
$$a_{2,2} \text{ with } p_{d,d},$$

the $\pi_u(k)$ and $\pi_d(k)$ can be obtained from (6) and (7), respectively. Arranging the resultant equation gives (2).

REFERENCES

[1] C. M. Krishna and A. D. Singh, "Reliability of checkpointed real-time systems using time redundancy," *IEEE Trans. Reliability*, vol. 42, pp. 427–435, Sept. 1993.

[2] A. M. Saleh and J. H. Patel, "Transient fault analysis for retry techniques," *IEEE Trans. Reliability*, vol. 37, pp. 323–330, Aug. 1988.

[3] D. P. Siewiorek and R. S. Swarz, *Reliable Computer Systems*: Digital Press, 1992.

[4] H. Kim and K. G. Shin, "Design and analysis of an optimal instruction retry policy for TMR controller computers," *IEEE Trans. Computers*, vol. 45, pp. 1217–1225, Nov. 1996.

[5] A. Ziv and J. Bruck, "An on-line algorithm for checkpoint placement," *IEEE Trans. Computers*, vol. 46, pp. 976–984, Sept. 1997.

[6] K. G. Shin, T-H. Lin, and Y-H. Lee, "Optimal checkpointing of real-time tasks," *IEEE Trans. Computers*, vol. C-36, pp. 1328–1341, Nov. 1987.

[7] J. W. Young, "A first order approximation to the optimal checkpoint intervals," *Comm. ACM*, vol. 17, pp. 530–531, Nov. 1974.

[8] E. Gelenbe and D. Derochette, "Performance of rollback recovery systems under intermittent failures," *Comm. ACM*, vol. 21, pp. 493–499, June 1978.

[9] R. Geist, R. Reynolds, and J. Westall, "Selection of a checkpoint interval in a critical-task environment," *IEEE Trans. Reliability*, vol. 37, pp. 395–400, Oct. 1988.

[10] C. M. Krishna and A. D. Singh, "Optimal configuration of redundant real-time systems in the face of correlated failure," *IEEE Trans. Reliability*, vol. 44, pp. 587–594, Dec. 1995.

[11] I. Koren and S. Y. H. Su, "Reliability analysis of N-Modular redundancy systems with intermittent and permanent faults," *IEEE Trans. Computers*, vol. C-28, pp. 514–520, July 1979.

[12] K-J. Chung, "Optimal test-times for intermittent faults," *IEEE Trans. Reliability*, vol. 44, pp. 645–647, Dec. 1995.

[13] H. Kim and K. G. Shin, "Modeling of externally-induced/common-cause faults in fault-tolerant systems," in *IEEE/AIAA Digital Avionics System Conf.*, Oct. 1994, pp. 402–407.

[14] C-J. Hou and K. G. Shin, "Determination of an optimal retry time in multiple-module computing systems," *IEEE Trans. Computers*, vol. 45, pp. 374–379, Mar. 1996.

[15] C. M. Krishina and K. G. Shin, *Real-Time Systems*: McGraw-Hill, 1997.

[16] J. L. Goldberg, *Matrix Theory With Applications*: McGraw-Hill, 1992.

[17] S. W. Kwak and B. K. Kim, "Task scheduling strategies for reliable TMR controllers using task grouping and assignment," *IEEE Trans. Reliability*, vol. 49, pp. 355–362, Dec. 2000.

Seong Woo Kwak was born on March 10, 1970 in Korea. He received the B.S. (1993), M.S. (1995), and Ph.D. (2000), all in electrical engineering from Korea Advanced Institute of Science and Technology (KAIST). He is a Research Professor at Satellite Technology Research Center (SaTReC) in KAIST. His research interests are in intelligent control, fault tolerant systems, and real-time systems.

Byung Jae Choi was born on August 20, 1965 in Korea. He received the B.S. (1987) in electronics from Kyung-Book National University, and M.S. (1989) in nuclear engineering, and Ph.D. (1998) in electrical engineering from Korea Advanced Institute of Science and Technology (KAIST). He is an Assistant Professor at Taegu University, Kyungpook, Korea. His research interests are in intelligent control, variable structure control, and fuzzy control.

Byung Kook Kim was born on October 5, 1952 in Korea. He received the B.S. (1975) from Seoul National University, Seoul, Korea, and the M.S. (1977) and Ph.D. (1981) in electrical engineering from Korea Advanced Institute of Science and Technology (KAIST). From 1982 to 1984 and from 1996 to 1997, he was a research fellow in the Department of Electrical Engineering and Computer Science at the University of Michigan. He joined the Department of Electrical Engineering at KAIST in 1986 as an Assistant Professor, where he is currently Professor. His research interests are in robotics, process control, reliable and intelligent control, fault-tolerant systems, real-time systems, and robot vision.