[15] A. Schönhage, "Schnelle Multiplikation von Polynomen über Körpern der Charakteristik 2," *Acta Informatica,* vol. 7, pp. 395-398, 1976.
[16] A. Schönhage and V. Strassen, "Schnelle Multiplikation grosser Zahlen," *Computing,* vol. 7, pp. 281-292, 1971.
[17] C. D. Thompson, "Area-time complexity for VLSI," in *Proc. 11th ACM Symp. Theory Comput.,* Apr./May 1979, pp. 81-88.
[18] C. C. Wang, T. K. Truong, H. M. Shao, L. J. Deutsch, J. K. Omura, and I. S. Reed, "VLSI architectures for computing multiplications and inverses in $GF(2^m)$," *IEEE Trans. Comput.,* vol. C-34, pp. 709-717, Aug. 1985.
[19] A. A. Yao, "The entropic limitations on VLSI computations," in *Proc. 13th ACM Symp. Theory Comput.,* May 1981, pp. 308-311.

Comments on "Design and Evaluation of a Fault-Tolerant Multiprocessor Using Hardware Recovery Blocks"

BRUNO CICIANI AND GIOVANNI CANTONE

Abstract—**Referring to the above mentioned paper,**[1] **an error is detected and a correct new solution is proposed and proved.**

Index Terms—**Fault-tolerant multiprocessor, hardware/software recovery blocks, performance of rollback recovery mechanisms, rollback propagation.**

In the paper,[1] the authors define as

TE_k: the accumulated effective computation before the kth rollback when the task can be completed without restart (see line 9, column 2, p. 120)

and they show that TE_k is given by [see formula (A.1)[1]]

$$TE_k = \sum_{j=1}^{k} (X_r^j - T_{\text{roll}}^j) \tag{1.1}$$

in which

X_r^j : is the duration between two consecutive rollbacks (the authors hypothesize that the density function of X_r^j is $\lambda_b e^{-\lambda_b t}$; see line 38, column 1, p. 123) and

T_{roll}^j : the time lost due the jth rollback.

Besides, they define

P_b (P_s) : the probability of rollback (restart) when a failure occurs (see line 16, column 2, p. 120);

$P_r(k)$: the probability of having k rollbacks during the time interval T_{real} (see line 1 and line 21, column 2, p. 120, T_{real} is the total execution time to complete the task when all failures are recovered by rollbacks instead of restarts)

and they show that [see formula (9)[1]]

$$\begin{aligned} P_r(k) &= P(TE_{k+1} > T_{ef}) - P(TE_k > T_{ef}) \\ &= H(T_{ef}, k) - H(T_{ef}, k+1) \end{aligned} \tag{1.2}$$

where

T_{ef}: is the total execution time of a given task under an error-free condition and without the time overhead for generating recovery blocks (see row 42, column 2, p. 118), and

$$H(t, k) = \sum_{i=0}^{k} \binom{k}{i} (1-Z)^i (Z)^{k-i} G_{k-i}(t)$$

$$Z = \sum_{h=1}^{N} e^{-h\lambda_b T_{SS}} P_{st}(h)$$

$$G_{k-i}(t) = 1 - e^{-\lambda_b t} \sum_{h=0}^{k-i-1} ((\lambda_b t)^{k-i-1-h}/(k-i-1-h)!) \qquad (\text{for } k-i>0)$$

$$G_0 = 1$$

with

T_{SS} : the duration state-save invocation interval, and

$P_{st}(h)$: the probability of having an h-step rollback given that the failure is recovered by the rollback.

Therefore, the probability of having 0 (zero) rollbacks during the time interval T_{real}, by use of formula (9)[1] becomes

$$P_r(0) = H(T_{ef}, 0) - H(T_{ef}, 1) = Ze^{-\lambda_b T_{ef}}. \tag{1.3}$$

On the other hand, it is also true that if the probability of restart when a failure occurs is equal to zero (i.e., $P_s = 0$ and $P_b = 1$):

$$P_r(0) = \text{Prob}\{\text{the multiprocessor is failure-free between } t=0 \text{ and } t=T_{ef}\}$$

that is $P_r(0)$ is also equal to the *reliability* of the multiprocessor system at time $t = T_{ef}$, we shall denote that by $R(T_{ef})$. The expression of $R(T_{ef})$, under the author's hypothesis, is equal to

$$R(T_{ef}) = e^{-\lambda T_{ef}} \tag{1.4}$$

where λ is the reciprocal of the mean time between failures (see line 3, column 2, p. 120); then, under the hypothesis $P_b = 1$, formula (1.4) becomes

$$R(T_{ef}) = e^{-\lambda_b T_{ef}}. \tag{1.5}$$

Therefore, by comparing (1.3) to (1.5), there results the contradiction:

$$R(T_{ef}) \neq P_r(0). \tag{1.6}$$

This contradiction result is probably due to a wrong interpretation the authors give to TE_k (the accumulated effective computation *before* the kth rollback when the task can be completed without restart) and the confusion that the authors make between the words "after" and "before" the kth rollback.

Indeed in line 40, column 1, p. 121, they assert: "In Appendix A, we show that the distribution function of the accumulated effective computation *after* k rollbacks is Prob ($\text{TE}_k \leq t$) = $H(t, k)$;" while in Appendix A (see line 34, column 1, p. 123) they assert: "thus the accumulated effective computation *before* the kth rollback TE_k is given by" [see formula (A.1)]

$$TE_k = \sum_{j=1}^{k} (X_r^j - T_{\text{roll}}^j). \tag{1.7}$$

In order to overcome the contradiction of (1.6) one should give a different meaning to TE_k. Namely TE_k should be "*the accumulated effective computation after the kth rollback when the task can be completed without restart,*" while "*the accumulated effective*

Manuscript received January 2, 1987; revised June 15, 1987.

B. Ciciani is with the Department of Electronics, University of Rome "Tor Vergata," 00173 Rome, Italy.
G. Cantone is with the Department of Informatica e Sistemistica, University of Naples, 80125 Naples, Italy.

IEEE Log Number 8927549.

[1] Y.-H. Lee and K. G. Shin, *IEEE Trans. Comput.,* vol. C-33, pp. 113-124, Feb. 1984.

computation before the kth rollback when the task can be completed without restart'' could be denoted by CB_k and becomes

$$CB_k \triangleq TE_{k-1} + X_r^k \qquad (1.8)$$

where X_r^k is the time interval between the $(k-1)$th and kth rollback (obviously $CB_1 = X_r^1$ and $CB_0 = 0$).

Then, with the same assumptions made in the paper,[1] the characteristic function of CB_k (for $k \geq 1$) is equal to

$$\Phi_{CB_k}(s) = \Phi_{TE_{k-1}}(s)\Phi_X(s) \qquad (1.9)$$

where [see formula (A.4)[1]]

$$\Phi_{TE_{k-1}}(s) = \sum_{i=0}^{k-1} \binom{k-1}{i} (1-Z)^i (Z)^{k-i-1} (\lambda_b/(\lambda_b+s))^{k-i-1} \qquad (1.10)$$

while

$$\Phi_X(s) = (\lambda_b/(\lambda_b+s)) \qquad (1.11)$$

from which the distribution function of CB_k becomes (for $k \geq 1$)

$$\begin{aligned} D_{CB_k}(t) &= \text{Prob}\ (CB_k \leq t) \\ &= \sum_{i=0}^{k-1} \binom{k-1}{i} (1-Z)^i (Z)^{k-i-1} G_{k-i}(t) \\ &\triangleq L(t, k) \end{aligned} \qquad (1.12)$$

with

$$D_{CB_0}(t) = \text{Prob}\ (CB_0 \leq t) = 1 \triangleq L(t, 0). \qquad (1.13)$$

Therefore,

$$\begin{aligned} P_r(k) &= P(CB_{k+1} > T_{ef}) - P(CB_k > T_{ef}) \\ &= L(T_{ef}, k) - L(T_{ef}, k+1) \end{aligned} \qquad (1.14)$$

The expression of $P_k(k)$ in formula (1.14) is now correct since for $k = 0$ one obtains

$$P_r(0) = L(T_{ef}, 0) - L(T_{ef}, 1) = e^{-\lambda_b T_{ef}} \qquad (1.15)$$

which is indeed equal to the *reliability* at time $t = T_{ef}$, when the probability of restart at occurrence of a failure is equal to zero.

A Design Approach for Self-Diagnosis of Fault-Tolerant Clock Synchronization

MEILIU LU, DU ZHANG, AND TADAO MURATA

Abstract—**Fault-tolerant clock synchronization (FTCS) is crucial for a distributed system with a desired degree of fault tolerance. Recently, various FTCS algorithms have been proposed to mask malicious faults of clocking modules. A common weakness of the mask-based algorithms is their inability to identify faulty clocking modules in the system for the purpose of repair. As a result, the usually required availability of correct clocking modules could not be maintained. A general design approach for self-diagnosis of faulty clocking modules in an FTCS system is presented. The approach is based on a statistical testing method. The major advantages are better self-stability control and lower overhead. With the support of a necessary repair technique, integration of the self-diagnosis algorithm into a mask-based FTCS algorithm can help achieve an ideal level of clock availability. A self-stability evaluation method is also discussed.**

Index Terms—**Clock synchronization, fault tolerance, hypothesis testing, self-diagnosis, self-stabilization.**

Manuscript received December 23, 1986; revised April 19, 1988. This work was supported in part by the National Science Foundation under Grant DMC 85-10208.

M. Lu was with the Department of Mathematics, Statistics, and Computer Science, University of Illinois at Chicago, Chicago, IL 60680. He is now with the Department of Computer Science, California State University, Sacramento, CA 95819.

D. Zhang was with the Department of Electrical Engineering and Computer Science, University of Illinois at Chicago, Chicago, IL 60680. He is now with the Department of Computer Science, California State University, Sacramento, CA 95819.

T. Murata is with the Department of Electrical Engineering and Computer Science, University of Illinois, Chicago, IL 60680.

IEEE Log Number 8927550.

I. Introduction

The *ultimate purpose* of a fault-tolerant clock synchronization (FTCS) algorithm is to assure that a coordinated action can be taken in a distributed system in the presence of clock failures. Most existing FTCS algorithms cannot achieve such high systemwide performance. The naturally decreasing number of correct clocks can eventually break the assumptions of an FTCS algorithm on the availability of correct clocks. This implies that execution of the FTCS algorithm will fail to synchronize the correct clocks. For example, one restriction on all FTCS algorithms without authentication is that the number of failing clocks may not exceed a limit, say f, such that the minimum number of clocks in the system that guarantees a fault-tolerant operation is $3f+1$ [3]. The inability to maintain required clock availability is mainly due to the lack of a system function responsible for detecting and repairing potentially harmful defective clocks. Detection and repair (or replacement) features should be added to FTCS systems in which a certain level of the clock availability must be maintained. We call an FTCS algorithm without the detection function a mask-based FTCS algorithm. This paper presents an adaptive self-diagnosis technique and its associated design methodology. With the support of a necessary repair technique, integration of this self-diagnosis algorithm into a mask-based FTCS algorithm can help achieve an ideal level of clock availability.

The approach taken in this paper is outlined as follows. Suppose there is a distributed system of m nodes with each node having its own local clock. At each node of the system, 1) the difference between the local clock reading and a clock observation from another node is modeled as a random variable with certain probability distribution; 2) a sample of the clock observations is then collected with a sampling strategy; 3) a statistical test, based on the sample, is conducted; and 4) the test result is used to support the self-diagnosis of its clock's correctness [6].

It is possible to use the above statistical sampling test in developing the proposed self-diagnosis technique because clock observations are independent. Such independence is due to an inherent property of the distributed system. Another advantage of using a statistical method is that the resulting algorithm is analytically traceable in evaluating certain interesting properties such as availability.

This paper is organized as follows. Some background information is given in the next section. The general design method for the self-diagnosis algorithm is discussed in Section III. Section IV presents an adaptive diagnosis algorithm to illustrate the design method. An analysis of the algorithm is described in Section V. Finally, Section VI concludes the paper.

II. Background

An FTCS system consists of a group of clocking modules such that each node i in the main system must have its own clocking module, which is denoted as C_i, responsible for keeping synchronous time for node i. A *clocking module* C_i consists of three major components: a timer D, a time adjustment register TAR, and a program which executes a predefined FTCS algorithm. The logical clock time of