Ensuring Fault Tolerance of Phase-Locked Clocks

C. M. KRISHNA, KANG G. SHIN, AND RICKY W. BUTLER

Abstract — **Processors within a real-time multiprocessor system must be synchronized with as little overhead as possible. Although synchronization can be achieved via both software (e.g., interactive convergence and interactive consistency algorithms) and hardware (e.g., multistage synchronizers and phase-locked clocks), phase-locked clocks are most attractive due to their small overheads.**

Despite the fact that synchronization of the multiprocessor system with phase-locked clocks is totally different in nature from the interactive consistency algorithm [1], we prove that it must satisfy the same condition $N \geq 3m + 1$ where N is the total number of clocks in the multiprocessor system and m is the maximum number of faults tolerable. We also present results showing how to design phase-locked clocks so as to be impervious to up to a given arbitrary number of malicious failures.

Index Terms — **Interactive consistency and interactive convergence algorithms, malicious failure, phase-locked clocks, synchronization.**

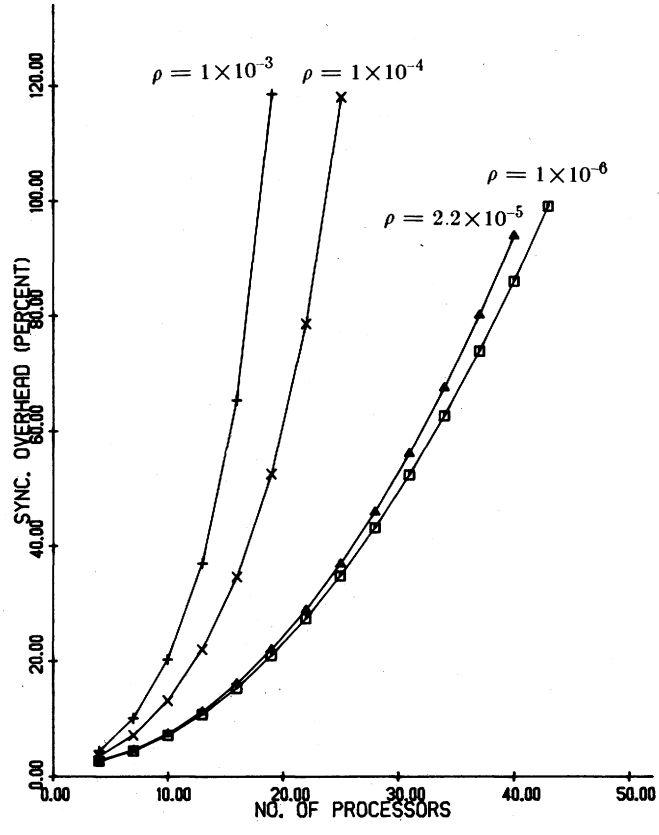
I. Introduction

Synchronizing processors reliably is complicated because individual faulty processors or clocks can act *maliciously* or disruptively. For example, a faulty clock might send conflicting time signals to different receivers. Guaranteeing fault tolerance in the face of such failures is nontrivial.

There are four well-known ways of keeping a system synchronized despite malicious failures: the interactive consistency [1] and the interactive convergence [2] algorithms, the multistage synchronizer [5], and phase-locked clocks [8]. The first two of these impose a significant overhead when a moderate-to-large system is being synchronized. Estimates of the synchronization overhead in a computer such as SIFT [2] show this dramatically (see [4] and Fig. 1). By comparison, the time overheads of the multistage and phase-locked clock designs are negligible. However, the multistage arrangement requires too much hardware to be practical in large systems. This paper concentrates on showing how to synchronize large systems using phase-locked clocks.

Expanding phase-locked clocks is important for two reasons. First, to synchronize a large number of processors with low overhead, it is best for fault-tolerance purposes to have each processor carry its own clock and for these clocks to be synchronized, rather than have a stand-alone fault-tolerant clock providing timing signals to the system. Second, when one requires the kind of reliability called for in aerospace or other life-critical applications, large clocks are called for to ensure adequate fault tolerance.

Phase-locked clocks were first used to ensure that the processors of FTMP [6] operated in lock step. We consider here a total of N clocks to be synchronized in the face of up to m faulty clocks. The basic theory behind their operation is simple. In Fig. 2, we provide a schematic diagram of an individual clock. Each clock consists of a receiver, which monitors the clock pulses of the $N - 1$ other clocks in the arrangement, and these are used to generate a reference signal. By comparing this reference to its own pulse, the receiving clock computes an estimate of its own phase error. This estimated phase error is then put into an appropriate filter, and the output of the filter controls the clock oscillator's frequency. By thus controlling the frequency of the individual clocks, they can be kept in phase lock and therefore synchronized for as long as the initial phase error is below a prescribed bound, i.e., for as long as the clocks start reasonably in step and their drifts are sufficiently low. A discussion of clock stability can be found in [7].



Resynchronization Interval = 100 ms

Fig. 1. Overhead of interactive convergence algorithm.

The arrangement for $N = 4$, $m = 1$ is, to our knowledge, the only phase-locked clock constructed [8]. The four-clock arrangement is simple enough for a proof of correctness to be obtained by brute-force enumeration of all possible actions of the single malicious clock. Expanding the clock is nontrivial. In fact, if we try to allow for $m = 2, 3, \cdots$, by expanding the system arbitrarily without sufficient care, synchronization can be lost. We provide an example of this in Section III.

The correspondence is organized as follows. In Section II, we present preliminary notation and definitions. In Section III, we show how damaging malicious failures can be. Our main result is contained in Section IV where we prove two important theorems related to the design of phase-locked clocks so as to tolerate up to a given arbitrary number of faults, and we conclude with Section V.

II. Notation and Definitions

The following notation and definitions are used in this correspondence.

Definition 1: If the overall system of clocks is properly synchronized, all individual nonfaulty clocks must agree closely with

Manuscript received December 1, 1983; revised March 1, 1985. This work was supported in part by NASA under Grant 1-296. Any opinions, findings, and conclusions or recommendations expressed in this correspondence are those of the authors and do not necessarily reflect the views of NASA.

C. M. Krishna was with the Department of Electrical Engineering and Computer Science, University of Michigan, Ann Arbor, MI 48109. He is now with the Department of Electrical and Computer Engineering, University of Massachusetts, Amherst, MA 01102.

K. G. Shin is with the Department of Electrical Engineering and Computer Science, University of Michigan, Ann Arbor, MI 48109.

R. W. Butler is with NASA Airlab, NASA Langley Research Center, Hampton, VA 23665.

0018-9340/85/0800-0752$01.00

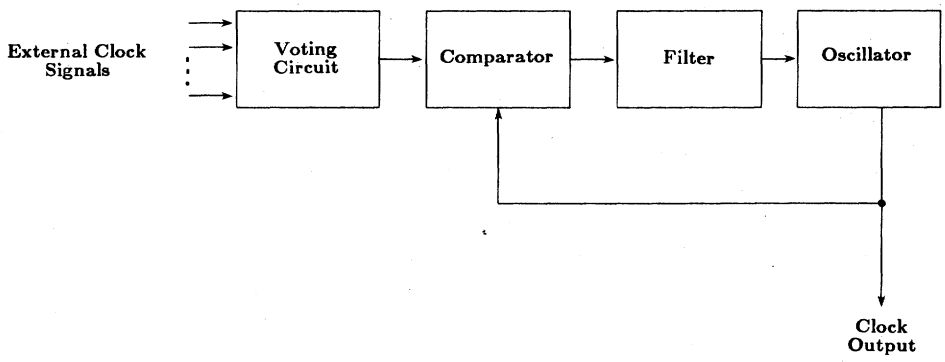


Fig. 2. Component of a phase-locked clock.

each other. A well-synchronized system thus has *global clock cycles*. Global clock cycle i is the interval between the ith tick of the fastest nonfaulty clock (i.e., the nonfaulty clock that has its ith tick before those of all the other nonfaulty clocks) and the $(i + 1)$th tick of the fastest nonfaulty clock. For brevity, we shall denote global clock cycle i by gcc$i$.

Definition 2: Each of the clocks "sees" through its receiving circuitry the ticks of the other clocks. These ticks, together with the receiving clock's own tick, can be totally ordered in any gcc$i$ by the relation "prior or equal to." Such an ordered set, called a *tick sequence,* for clock α in gcc$i$ is denoted by S^i_α. We shall frequently drop the superscript for convenience: where this is done, it will be understood that we are talking about some gcc$i$.

If a nonfaulty clock c does not receive a tick from clock d within a given timeout period in any global clock cycle, the tick for d is arbitrarily assumed by c to be at the end of that timeout period. The tick sequence of every nonfaulty clock therefore has exactly N elements.

Definition 3: If clock a has clock b as its reference in some gcc$i$, it is said to *trigger* on b in that gcc$i$.

Definition 4: Given the various triggers, we can draw a directed graph with the clocks as the vertices, and the directed arcs reflecting the relationship "triggers" in some gcc$i$. Such a graph is called the *trigger graph.* For example, in Fig. 3, a triggers b and c, and is itself triggered by d, while d is triggered by b. A *clique* of clocks is a component [9] of the trigger graph, i.e., a set of connected vertices. In Fig. 3, there are two cliques: $\{a, b, c, d\}$ and $\{e, f, g\}$.

Notation: G and NG are the sets of clocks and nonfaulty clocks, respectively, in the system. There are N clocks in all, and up to m failures must be sustained.

Definition 6: A *partition* of G is defined as $P = \{G_1, G_2\}$ where G_1 and G_2 are subsets of G with the following properties.

i) $G = G_1 \cup G_2$,
ii) $G_1 \cap G_2 \cap NG = \phi$,
iii) $G_i \cap NG \neq \phi$, $i = 1, 2$.

From i), each clock must belong to at least one of G_1 and G_2. From ii), only faulty clocks may belong to both G_1 and G_2. From iii), there must be at least one nonfaulty clock in each of G_1 and G_2.

Definition 7: A clock a is said to be *faster* than a clock b in tick sequence S if a precedes b in S. In a partition $P = \{G_1, G_2\}$, G_1 is said to be faster than G_2 if every nonfaulty clock in G_1 is faster than every nonfaulty clock in G_2.

Notation: Given a partition $P = \{G_1, G_2\}$, NG_1 and NG_2 are the nonfaulty clocks in G_1 and G_2, respectively. By Definition 6, neither NG_1 nor NG_2 can be empty, and $NG_1 \cap NG_2 = \phi$.

Definition 8: Cliques A and B (of clocks) are said to be nonoverlapping if the nonfaulty clocks of A are either all faster than those of B, or vice versa.

Notation: Denote the position of a clock c in its own tick sequence S^i_c in gcc$i$ by p^i_c. Again, we shall frequently drop the superscript for convenience. The reference signal (i.e., the trigger) is a function of N and of p_c. It is denoted by $f_{p_c}(N)$. By this, we mean that clock c triggers on the $f_{p_c}(N)$th signal in S_c, *not* counting itself.

For the system to operate correctly, the nonfaulty clocks must be

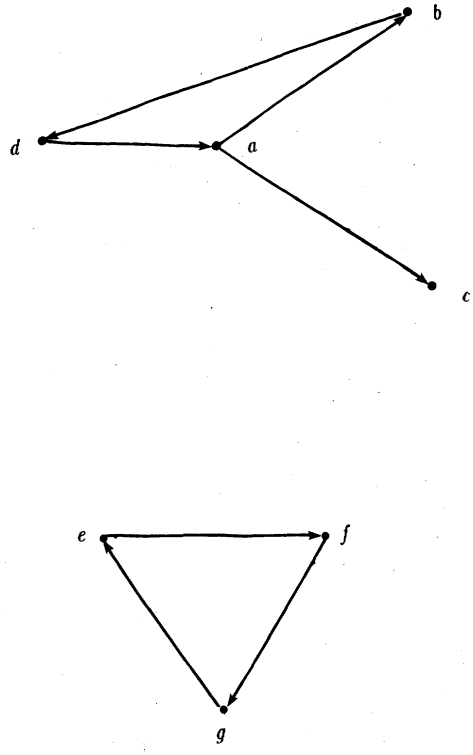


Fig. 3. Trigger graph: an example.

synchronized, i.e., have their ticks close together. This can only be if nonoverlapping cliques are not allowed to form. Clearly, A and B are nonoverlapping cliques if every nonfaulty clock in A is faster than all nonfaulty clocks in B (or vice versa), and no nonfaulty clock in either clique triggers on a clock in the other.

Also, correctness should imply that the clocks keep good time, i.e., the length of every global clock cycle should be about that of an ideal clock cycle. This dictates that a reference clock should be either a nonfaulty clock, or a faulty clock sandwiched between nonfaulty clocks. If this is not the case, and at least one nonfaulty clock triggers on a faulty clock slower or faster than all the nonfaulty clocks, and if there are no nonoverlapping cliques, then the entire set of nonfaulty clocks would be abnormally slowed down or sped up. We, therefore, have the following conditions of correctness.

Definition 9: Each of the following *conditions of correctness* must be satisfied in gcc$i$ if the system is to be correctly operating in every gcc$i$.

C1. For all partitions $P = \{G_1, G_2\}$ of the set of clocks G, in which the nonfaulty clocks in G_1 are all faster than those in G_2, each of the following (K1 and K2) must apply.
  K1. If, in gcc$i$, all clocks in NG_1 trigger on clocks in G_1, then there is at least one clock in NG_2 that triggers on a clock in G_1. Furthermore, if no clock in NG_2 triggers on a clock

in NG_1, at least one clock $k \in NG_2$ must trigger on a faulty clock $h \in G_1$ such that in the tick sequence S_k there is at least one clock $r \in NG_1$ that is slower than the clock h.

K2. If, in gcc$i$, all clocks in NG_2 trigger on clocks in G_2, then there is at least one clock in NG_1 that triggers on a clock in G_2. Furthermore, if no clock in NG_1 triggers on a clock in NG_2, at least one clock $k \in NG_1$ must trigger on a faulty clock $h \in G_2$ such that in S_k there is at least one clock $r \in NG_2$ that is faster than h.

C2. If a nonfaulty clock x triggers on a faulty clock y, then there must exist nonfaulty clocks z_1 and z_2 such that z_1 is faster than or equal to y, and y is faster than or equal to z_2. Either z_1 or z_2 may be x itself.

Finally, we assume that the transmission of clock signals through the system takes negligible time and that the clocks are completely connected. This ensures that all nonfaulty clocks are seen by all clocks in the same mutual order.

III. Malicious Failure and Correct Synchronization

As remarked in Section I, the clock $N = 4, m = 1$ is the only one actually constructed.

Unfortunately, if we try to allow for $m = 2, 3, \cdots$, by expanding the system arbitrarily without sufficient care, the conditions of correctness can be violated. In fact, it is even possible for a system to contain an arbitrarily large number of clocks and still be vulnerable to just two malicious failures.

To see this, consider the following example. Let us choose, for each clock y in the system, $f_{p_y}(N)$ as the *medium* clock signal in the tick sequence, not counting clock y. If N is odd (and there is thus an even number of "other" clocks), choose the slower of the two middle clocks. Then, $f_{p_y}(N)$ is only a function of N. We therefore drop the subscript for this example. Choosing the median signal is certainly good intuition.

Let there be only two faulty clocks, x_1 and x_2, and $n = N - 2$ nonfaulty clocks $a_1, \cdots, a_n$.

Case 1: $N \geq 7$. Consider some gcc$i$. Assume that a_k is faster than a_l in gcc$i$ if $k < l$. Now, let x_1 and x_2 present themselves as the fastest two clocks to $a_1, \cdots, a_p$, and as the slowest two clocks to the other nonfaulty clocks, i.e., $a_{p+1}, \cdots, a_n$, where $p = \lceil n/2 \rceil = f(N) - 1$. Then, the set of tick sequences can be represented as in Fig. 4.

Recalling that a clock triggers on the $f(N)$th tick in its tick sequence *not counting itself,* we can draw the trigger graph as in Fig. 5. It follows that $\{a_1, \cdots, a_p\}$ and $\{a_{p+1}, \cdots, a_n\}$ will be two nonoverlapping cliques, no matter how large n may be. It is easy to work out the case for $N = 7$ to convince oneself of this fact.

Case 2: $N \leq 7$. This is trivial, and showing that the system is incapable of sustaining even two maliciously faulty clocks is left to the reader.

This has been a cautionary tale of the unbridled use of intuition in designing phase-locked clocks. Assured now that a more careful approach is needed, we turn in the following section to showing how to expand phase-locked clocks.

IV. Main Result

Our job is to i) find the lower bound on the size N of a system of clocks that must sustain up to m maliciously faulty clocks, and ii) find the functions $f_x(N)$ for $x = 1, \cdots, N$.

We begin with the following two lemmas.

Lemma 1: Condition C2 is satisfied if and only if there exist functions $f_x(N)$ for $x = 1, \cdots, N$, such that

$$\min\{m, x - 1\} < f_x(N) < \max\{N - m, x\}. \qquad (1)$$

Proof: Let k be a nonfaulty clock such that $p_k = x$. We must show that (1) holds for all x for which p_k is defined if and only if condition C2 holds.

Suppose that there exist functions $f_x(N)$ for $x = 1, \cdots, N$ satisfying (1). This implies $\min\{m, x - 1\} + 1 < \max\{N - m, x\}$ for all $x \in \{1, 2, \cdots, N\}$, leading to $N > 2m + 1$. Hence, it is sufficient to consider the following three cases.

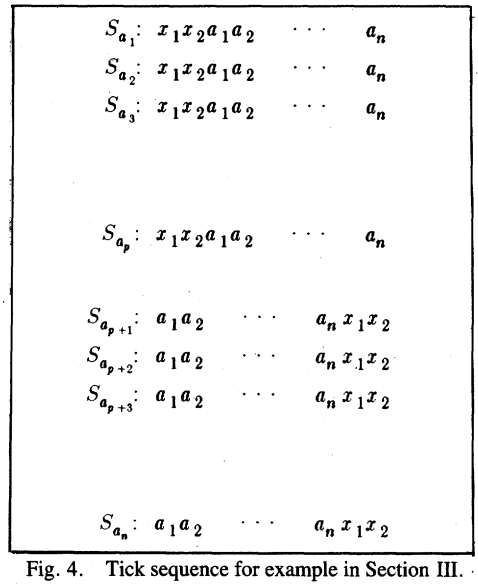


Fig. 4. Tick sequence for example in Section III.

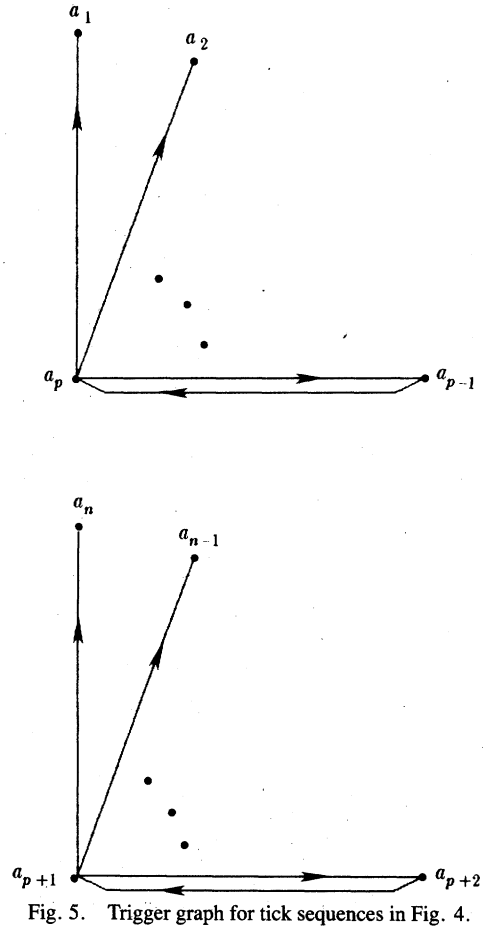


Fig. 5. Trigger graph for tick sequences in Fig. 4.

i) $x \leq m$:

Clearly, $\max\{N - m, x\} = N - m$, $\min\{m, x - 1\} = x - 1$, and therefore $x - 1 < f_x(N) < N - m$. If the reference clock is nonfaulty, we have nothing to prove. If it is faulty, then since there are at most m faulty clocks, there must be at least one nonfaulty clock slower than the reference clock from the right-hand half of the inequality, $f_x(N) < N - m$. Also, from the left-hand half of the inequality, $f_x(N) > x - 1$, and since clock k is nonfaulty, there is

a nonfaulty clock (i.e., k itself) faster than the reference clock. So, C2 is satisfied.

ii) $N - m \geq x > m$:

$\min\{m, x-1\} = m$, $\max\{N-m, x\} = N - m$, and therefore $m + 1 \leq f_x(N) \leq N - m - 1$. Since at most m faulty clocks exist, if the reference clock in S_k were faulty, it must appear in S_k as slower than at least one nonfaulty clock (the right-hand half of the inequality) and faster than at least one nonfaulty clock (the left-hand half of the inequality), and C2 is satisfied.

iii) $N \geq x > N - m$:

$\min\{m, x-1\} = m$, $\max\{N-m, x\} = x$, and $m + 1 \leq f_x(N) \leq x - 1$. As with the previous cases, there must appear in S_k at least one nonfaulty clock that is faster than the reference clock, if the reference clock is faulty. Also, since k is nonfaulty, and appears in the xth (i.e., p_kth) position, there is at least one nonfaulty clock, in particular clock k, that is slower than the reference clock in S_k, thus satisfying C2.

Conversely, suppose $f_x(N) \leq \min\{m, x-1\}$. Then, C2 is violated when faulty clocks appear in positions $1, \cdots, f_x(N)$ of S_k. Similarly, if $f_x(N) \geq \max\{N-m, x\}$, C2 is violated when faulty clocks appear in positions $f_x(N) + 1, \cdots, N$ of S_k.[1]

Q.E.D.

Lemma 2: If all clocks in NG_1 trigger only on clocks in G_1 (where the notation is the same as in Definition 9), then the following are equivalent.

i) $q_1 \geq \min_{k \in NG_2} f_{p_k}(N)$ where q_1 is the number of nonfaulty clocks in G_1.

ii) K1 is satisfied.

Proof:

i) implies ii): If i) holds, then it is easy to see that no matter how the up to m faulty clocks in G arrange themselves, K1 is satisfied.

ii) implies i): Suppose, to the contrary, that $q_1 < \min_{k \in NG_2} f_{p_k}(N)$. Consider the nonempty set $L = \{y : y \in NG_2 \text{ and } f_{p_y}(N) = \min_{k \in NG_2} f_{p_k}(N)\}$. Assume that there are $i \leq m$ faulty clocks in G_1. Since the faulty clocks may present themselves in any position in any tick sequence, consider the case where they present themselves in the tick sequence of every $y \in L$ in the $q_1 + 1, \cdots, q_1 + i$ positions. Then, there is no nonfaulty clock in G_1 that is slower than the reference clock of any clock in NG_2, a contradiction.

Q.E.D.

The two theorems below yield the main result of this correspondence.

Theorem 1: To ensure that, despite up to m malicious failures, the conditions of correctness are satisfied, the system must have $N \geq 3m + 1$ clocks.

Proof: We will only consider here the case of partitions $P = \{G_1, G_2\}$ in which all clocks in NG_1 trigger on clocks in G_1. The other case (i.e., K2) can similarly be dealt with.

Let there be q_1 and q_2 clocks, respectively, in NG_1 and NG_2. Let $M = \{y : y \in NG_1 \text{ and } f_{p_y}(N) = \max_{k \in NG_1} f_{p_k}(N)\}$. Let $i \leq m$ be the number of faulty clocks that belong to G_1. Then, the assumption that all nonfaulty clocks in G_1 trigger on clocks in G_1 is equivalent to saying that one of (2) and (3) must apply:

$$q_1 + i \geq \max_{k \in NG_1} f_{p_k}(N) + 1 = f_{p_y}(N) + 1, \tag{2}$$

which applies if there exists at least one p_y, $y \in M$, such that $p_y < f_{p_y}(N)$. The addition of 1 follows from the fact that clock y does not count itself when counting to $f_{p_y}(N)$. If $p_y \geq f_{p_y}(N)$ for all $y \in M$, (3) applies:

$$q_1 + i \geq \max_{k \in NG_1} f_{p_k}(N). \tag{3}$$

First consider the case where (2) applies. The condition that C1 (more specifically, K1) holds implies, from Lemma 2, that

$$q_1 \geq \min_{k \in NG_2} f_{p_k}(N). \tag{4}$$

[1]Once again, the addition of 1 occurs because a clock does not count itself when counting to $f_x(N)$.

Since this must be true for all partitions of G, we have for all $q_1 \in \{1, \cdots, N-1\}$:

$$q_1 \geq \max_{k \in NG_1} f_{p_k}(N) - i + 1$$

if $q_1 \geq \min_{k \in NG_2} f_{p_k}(N)$. Hence, K1 can be written as

For all $q_1 = 1, \cdots, N-1$,

$$\left\{q_1 \geq \max_{k \in NG_1} f_{p_k}(N) - i + 1 \supset q_1 \geq \min_{k \in NG_2} f_{p_k}(N)\right\}.$$

In particular, this is true for $q_1 = \max_{k \in NG_1} f_{p_k}(N) - i + 1$. Thus,

$$\max_{k \in NG_1} f_{p_k}(N) - i + 1 \geq \min_{k \in NG_2} f_{p_k}(N)$$

or

$$\max_{k \in NG_1} f_{p_k}(N) - \min_{k \in NG_2} f_{p_k}(N) \geq i - 1. \tag{5}$$

Recall that this is true if (2) applies. Similarly, if (3) applies, we have from an identical argument

$$\max_{k \in NG_1} f_{p_k}(N) - \min_{k \in NG_2} f_{p_k}(N) \geq i. \tag{6}$$

Equations (2)–(6) must hold for all possible i. Since there are at most m faulty clocks, we must have

$$\max_{k \in NG_1} f_{p_k}(N) - \min_{k \in NG_2} f_{p_k}(N) \geq m - 1 \tag{5'}$$

if (2) applies, and

$$\max_{k \in NG_1} f_{p_k}(N) - \min_{k \in NG_2} f_{p_k}(N) \geq m \tag{6'}$$

if (3) applies.

We first consider the case where (2) applies. We claim that it implies that $N > 3m$.

To see why, let y be the slowest clock in M and z the slowest clock in L (with L defined as in Lemma 2). Then, due to Lemma 1 and (5') the following inequality must hold:

$$\begin{aligned}\max\{N-m, p_y\} > \max_{k \in NG_1} f_{p_k}(N) &\geq m - 1 + \min_{k \in NG_2} f_{p_k}(N) \\ &> m - 1 + \min\{m, p_z - 1\}.\end{aligned} \tag{7}$$

Then up to m faulty clocks in the system can arrange themselves in any order. In particular, they can so order themselves in S_y that $p_y \leq N - m$, and so order themselves in S_z that $p_z > m$. Since (7) must hold *always*, no matter what the faulty clocks do, we must have

$$\begin{aligned}N - m > \max_{k \in NG_1} f_{p_k}(N) &\geq m - 1 + \min_{k \in NG_2} f_{p_k}(N) \\ &\geq (m-1) + (m+1),\end{aligned} \tag{8}$$

from which we arrive at the inequality

$$N > 3m. \tag{9}$$

Recall that this applies whenever (2) holds. If, instead, (3) applies, we can similarly show that

$$N > 3m + 1. \tag{10}$$

Since we seek the smallest N to satisfy the conditions of correctness, we have done if we can show that there exist functions $f_x(N)$ such that (2) always applies (and therefore (3) never applies), and for which (8) is satisfied. But, we can always construct $f_x(N)$ to i) be monotonically nonincreasing functions of x, and ii) satisfy (8): an example of such a construction is provided in the statement of Theorem 2 below. Hence, (2) always applies, and $N \geq 3m + 1$ is the necessary condition.

The case when all clocks in NG_2 trigger on clocks in G_2 can be similarly treated.

Q.E.D.

TABLE I
COMPARISON OF SYNCHRONIZATION METHODS

Technique	Min.Cluster	I/O Ports	Overhead
Expanded Phase-Locked Clock	$3m+1$	$9m^2+3m$	Negligible
Multistage Synchronizers (Davies and Wakerly	$2m^2+3m+1$	$8m^3+16m^2+10m+2$	Negligible
Interactive Convergence Algorithm (Goldberg, et al.)	$3m+1$	$9m^2+3m$	Considerable

m = number of faulty processors accommodated

Theorem 2: If $N \geq 3m + 1$ and $f_x(N) = \begin{cases} 2m & \text{if } x < N - m, \\ m + 1 & \text{if } x \geq N - m, \end{cases}$ then the conditions of correctness are satisfied.

Proof: $f_x(N)$ as defined here satisfies Lemmas 1 and 2 and is monotonically nonincreasing in x. Clearly, C2 holds. Also, it is easy to see that *if* $N > 3m$, and (2) implies (4), *then* case K1 in Definition 8 will hold. We therefore only have to show that the definition of $f_x(N)$ as given above satisfies (4) if (2) is satisfied. This can easily be verified by a direct substitution.

Case K2 can be similarly seen to hold.

Q.E.D.

It should be noted that the set of functions $f_x(N)$ is not always unique. From the proofs of Theorems 1 and 2, the following inequalities are sufficient:

i) $m + 1 \leq f_x(N) \leq N - m - 1$ for all $x = 1, \cdots, N$,

ii) $f_x(N) \geq m - 1 + f_{N-m}(N)$ for all $x \leq m + 1$,

iii) $f_{N-m}(N) \leq f_x(N) \leq f_{m+1}(N)$ for $N - m > x > m + 1$,

iv) $f_i(N) \geq f_j(N)$ iff $i \leq j$.

The intervals $x \geq N - m$ and $x \leq m + 1$ arise from the up to m faulty clocks in the system. All that we can tell about the fastest nonfaulty clock g in the system [this clock must have the maximum value of $f_x(N)$] in clock g's tick sequence is that it is in the first $m + 1$ clocks in that tick sequence. Similarly, all that we can tell about the position of the slowest nonfaulty clock s in the system [which must have the minimum value of $f_x(N)$] is that it occupies a place in the last $m + 1$ clocks. This leads at once to the intervals $x \geq N - m$ and $x \leq m + 1$.

It is interesting to note that if conditions C1 and C2 are both satisfied, and the functions $f_x(N)$ are monotonically nonincreasing in x, then a stronger condition than C2 automatically holds.

Corollary: If the conditions of correctness are satisfied, with the $f_x(N)$ being defined as monotonically nonincreasing functions of x, then the following condition C3 holds.

C3. Every nonfaulty clock necessarily triggers on either a nonfaulty clock or a faulty clock that is sandwiched between the *other* nonfaulty clocks.

Proof: Now, C3 follows immediately from C2 for all but the fastest and slowest nonfaulty clocks.

Consider the fastest nonfaulty clock. In the course of proving Theorem 1, it was established that $N - m > f_x(N) > m$ for all $x = 1, \cdots, N$, and that $\max_{k \in G} f_{p_k}(N) - \min_{k \in G} f_{p_k}(N) \geq m - 1$, leading to $2m$ as the smallest value for $\max_{k \in NG} f_{p_k}(N)$ where $NG \subset G$ is the set of nonfaulty clocks. From the monotonic nature (in p_k) of the $f_{p_k}(N)$, the trigger for the fastest nonfaulty clock must lie in the interval $2m + 1, \cdots, N - m$. But, since $N \geq 3m + 1$, any faulty clock in this interval must be sandwiched between nonfaulty clocks.

The proof for the slowest nonfaulty clock is similar.

Q.E.D.

V. DISCUSSION

In this correspondence, we have shown how to construct phase-locked clocks that operate correctly in the face of up to a given arbitrary number of malicious failures. As we saw in Section I, the high overhead of the other methods of synchronization — time overhead for software synchronization (i.e., interactive consistency and interactive convergence algorithms) and hardware overhead for the Davies and Wakerly arrangement — makes phase-locked clocks very attractive candidates for fault-tolerant clocking arrangements due to their small overheads. Table I is a comparison of the techniques for synchronization in the face of malicious failure.

It would be interesting to make a comparative study of the various algorithms that handle malicious failure and to try to establish a unified theory from which the proofs for the interactive convergence and interactive consistency algorithms, as well as those of Theorems 1 and 2 of this correspondence, would follow. We believe that the $N > 3m$ requirement, which is common to all three algorithms, might be a fruitful starting point in the search for such a unified theory.

REFERENCES

[1] M. Pease *et al.*, "Reaching agreement in the presence of faults," *J. ACM*, vol. 27, no. 2, pp. 228–234, Apr. 1980.

[2] J. Goldberg *et al.*, "Development and analysis of the software implemented fault-tolerance (SIFT) computer," NASA Contract Rep. CR-172146, June 1983.

[3] J. H. Wensley *et al.*, "SIFT: Design and analysis of a fault-tolerant computer for aircraft control," *Proc. IEEE*, vol. 66, pp. 1240–1255, Oct. 1978.

[4] C. M. Krishna, K. G. Shin, and R. W. Butler, "Synchronization and fault-masking in redundant real-time systems," in *Dig. Papers, FTCS-14*, June 1984, pp. 152–157.

[5] D. Davies and J. F. Wakerly, "Synchronization and matching in redundant systems," *IEEE Trans. Comput.*, vol. C-27, pp. 531–539, June 1978.

[6] A. L. Hopkins *et al.*, "FTMP — A highly reliable fault-tolerant multiprocessor for aircraft," *Proc. IEEE*, vol. 66, pp. 1221–1239, Oct. 1978.

[7] A. W. Holt and J. M. Myers, "An approach to the analysis of clock networks," NASA Contract Rep. CR-166028, Nov. 1982.

[8] T. B. Smith, "Fault-tolerant clocking system," in *Dig. Papers, FTCS-11*, 1981, pp. 262–264.

[9] F. Harary, *Graph Theory*. New York: Addison-Wesley, 1969.

Note on a Proposed Test for Random Number Generators

GEORGE MARSAGLIA

Abstract **— A recently proposed test for uniform random number generators is based on the mean and variance of the outcome of a sequence of iterations. This note points out that many nonuniform random number generators would pass such a test, and derives an improved test based on the exact distribution of the outcome.**

Manuscript received February 1, 1984; revised January 1, 1985.

The author is with the Department of Computer Science, Washington State University, Pullman, WA 99164.