THE UNIVERSITY OF MICHIGAN

COMPUTING RESEARCH LABORATORY

ENSURING FAULT-TOLERANCE OF PHASE-LOCKED CLOCKS

C.M. Krishna, Kang G. Shin and Ricky W. Butler

CRL-TR-29-84

June 1984

Room 1079, East Engineering Building
Ann Arbor, Michigan 48109
USA
Tel: (313) 763-8000

ENSURING FAULT-TOLERANCE OF PHASE-LOCKED CLOCKS[1]

C. M. Krishna, Kang G. Shin

Computing Research Laboratory
Department of Electrical Engineering and Computer Science
The University of Michigan
Ann Arbor, Michigan 48109

and

Ricky W. Butler

NASA Langley Research Center
Hampton, VA 23665

ABSTRACT

Processors within a real-time multiprocessor 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 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 and m 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 - Synchronization, interactive consistency and interactive convergence algorithms, phase-locked clocks, malicious failure.

---

[1]The work of Krishna and Shin was supported in part by NASA Grant No. 1-296. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of NASA. All correspondence regarding this report should be addressed to Professor Kang G. Shin.

1. INTRODUCTION

One problem in designing ultra-reliable multiprocessors for the control of real-time systems is synchronizing the processors.

It is difficult to reliably synchronize processors within a multiprocessor because a failed unit, such as a clock or a processor, can behave arbitrarily *maliciously* or *disruptively.* That is to say, when a unit fails, it need not always go dead, or behave otherwise predictably. A failed clock may, for instance, send out conflicting time signals (or *lie*) to different parts of a system. As we shall see in Section 3, even highly redundant systems can, if they are not designed with great care, be vulnerable to just two malicious faults.

In the absence of models for computing the probability of malicious failures,[2] we must assume for safety that malicious behavior is always possible. The most difficult job in synchronizing a highly reliable system is ensuring that synchronization can be achieved in the presence of such malicious units.

The interactive consistency [1], and the interactive convergence [2] algorithms -- both implemented in software -- can be used to synchronize the multiprocessor even in the face of malicious failures. Although these algorithms are elegant and theoretically appealing, they consume a great deal of time for a large N, and time is precious in real-time systems. For example, the SIFT aircraft-control computer [2,3] uses the interactive convergence algorithm to periodically (100ms period) synchronize its processors. Based on data obtained from this sys-

[2] At present, there is no known method of characterizing the behavior of malicious failures.

tem, we projected in [4] the overhead that this algorithm would impose on a similar system for different clock drifts (see Figure 1). The largeness of the overhead imposed by that algorithm is evident. The interactive consistency algorithm performs even worse (see [4] for a detailed overhead analysis).

By comparison, the time overhead of hardware synchronization is insignificant. There are two ways of implementing synchronization in hardware to ensure correct operation in the face of malicious failures. The first is the multi-stage synchronizer of Davies and Wakerly [5], illustrated in Figure 2. Unfortunately, the number of hardware components in that arrangement is a quadratic function of the number of processors (or clocks) to be synchronized. The other way is to use phase-locked clocks which are the subject of this report. As we shall see, phase-locked clocks can be designed without any of the disadvantages mentioned above.

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 Figure 3, 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 with 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 started reasonably in step and their drifts are sufficiently low. A discussion of clock stability can be found in [7].

The arrangement for N=4, m=1 is, to our knowledge, the only phase-locked clock constructed and fully analyzed [8]. Unfortunately, when one attempts to increase m without care, synchronization can be lost due to the presence of malicious faults. In this report, we show how to design phase-locked clocks to tolerate a given arbitrary number of malicious failures. Our work is a generalization of the original design [8] which can tolerate at most one failed clock. As we shall see, the generalization requires to prove two non-trivial theorems, and interestingly it leads to the same necessary condition $N \geq 3m+1$ as the interactive consistency algorithm.

The report is organized as follows. In Section 2, we present preliminary notation and definitions. In Section 3, we show how damaging malicious failures can be. Our main result is contained in Section 4 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 5.

2. NOTATION AND DEFINITIONS

The following notation and definitions are used in this report.

Definition 1: If the overall system of clocks is properly synchronized, all indivi-

dual non-faulty clocks must agree closely with each other. A well-synchronized system thus has *global clock cycles*. Global clock cycle i is the interval between the i *th* tick of the fastest non-faulty clock (i.e. the non-faulty clock that has its i–*th* tick before those of all the other non-faulty clocks) and the (i+1)–*th* tick of the fastest non-faulty clock. For brevity, we shall denote global clock cycle i by *gcci*.

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 *gcci* by the relation "prior or equal to". Such an ordered set, called a *scenario*, for clock α in *gcci* 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 *gcci*.

If a non-faulty 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 scenario of every non-faulty clock therefore has exactly N elements.

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

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 *gcci*. Such a graph is called the *trigger graph*. For example, in Figure 4,

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 Figure 4, there are two cliques: $\{a,b,c,d\}$ and $\{e,f,g\}$.

Notation: G and NG are the set of clocks and non-faulty 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) $G = G_1 \bigcup G_2$

(ii) $G_1 \bigcap G_2 \bigcap NG = \phi$,

(iii) $G_i \bigcap NG \neq \phi$, 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 non-faulty clock in each of G_1 and G_2.

Definition 7: A clock a is said to be *faster* than a clock b in scenario 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 non-faulty clock in G_1 is faster than every non-faulty clock in G_2.

Notation: Given a partition $P=\{G_1,G_2\}$, NG_1 and NG_2 are the non-faulty clocks in G_1 and G_2, respectively. By definition 6, neither NG_1 nor NG_2 can be empty and $NG_1 \bigcap NG_2 = \phi$.

Definition 8: Cliques A and B (of clocks) are said to be non-overlapping if the

non-faulty 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 scenario S_c^i in *gcci* by p_c^i. 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 satisfactorily, all the non-faulty clocks must have their ticks close together. Also, they should tell good time, i.e. the length of every global clock cycle should be about the length of an ideal (or absolute time) clock's inter-tick interval. These conditions dictate the following two *conditions of correctness* C1 and C2.

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

C1. For all partitions $P=\{G_1,G_2\}$ of the set of clocks G, in which the non-faulty clocks in G_1 are all faster than those in G_2, each of the following (K1 and K2) must apply:

K1. If, in *gcci*, 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 scenario S_k, there is at least one clock $r \in NG_1$ that is slower than the clock h.

K2. If, in *gcci*, 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 non-faulty clock x triggers on a faulty clock y, then there must exist non-faulty 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.

C1 prevents the formation of non-overlapping cliques which would obviously destroy synchrony. C2 ensures that the system keeps good time, i.e. that each global clock cycle is close to being the clock cycle of an ideal clock. The non-faulty clocks define the range of acceptable timing values, and so any faulty clock value between two non-faulty values is also acceptable. If C2 did not hold, the entire system could be subject to the wiles of an erratically fast or slow faulty clock.

Finally, we assume that the transmission of clock signals through the system takes negligible time. This ensures that all non-faulty clocks are seen by all clocks in the same mutual order.

3. MALICIOUS FAILURE AND CORRECT SYNCHRONIZATION

The phase-locked clock system for N=4, m=1 is simple enough to be proved correct by an exhaustive enumeration of all eventualities. It is, to our knowledge,

the only phase-locked clock actually constructed and fully analyzed [8].

Here, the reference used is the second incoming pulse (in temporal order), i.e. the median pulse. Such a clock is proof against the malice of a single faulty clock. To give the reader a feeling for why this is so, and to enhance his intuition about malicious failure, we provide below a simple example.

Call the four clocks *a*, *b*, *c*, and *d*. Let *d* be the maliciously faulty clock. Because *d* is malicious, it may provide different timing signals (i.e. lie) to different receiving clocks. Since the non-faulty clocks by definition send their ticks at the same moment (or do not lie) to all the other receiving clocks, the mutual ordering of the non-faulty clocks within every scenario is the same for all non-faulty clocks. That is to say, if clock *b* sees clock *a* faster than clock *c* in some *gcci* (i.e. clock *a* sends its *i-th* tick to *b* before clock *c* does so), then *a* will appear faster than *c* to both the other non-faulty clocks in the system, i.e. to *a* and *c* in that *gcci*. *d*, however, may appear in different positions in the scenarios of the non-faulty clocks since it is malicious. One way of proving that a four-clock arrangement works despite *d*'s being malicious, is to enumerate all possible actions of *d* and show that the system still continues to satisfy the conditions of correctness.

Assume without loss of generality that *a* is prior or equal to *b* which in turn is prior or equal to *c* in some *gcci*. Consider a sample set of scenarios for our four-clock example. The triggering clock is denoted in bold-face type.

$S_a = a \leq b \leq c \leq d$

$S_b = a \leq d \leq b \leq c$

$S_c = a \leq b \leq d \leq c$

The scenario S_d is irrelevant, since d is faulty.

Notice first that the position of the faulty clock d changes relative to the others, while the mutual ordering of the non-faulty clocks remains unchanged, as indeed it should.

It is easy to see that both conditions of correctness will be satisfied, and that the clock will operate correctly if the above scenario holds. It is not difficult to write down all the 4^3=64 possible scenarios (with the ordering of the non-faulty clocks fixed as above) that are made possible by the arbitrary positioning of d, and to convince oneself that, for all possible scenarios, C1 and C2 are satisfied.

Unfortunately, if we try to allow for m=2,3,..., 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 to 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 *median* clock signal in the scenario, 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 $n=N-2$ non-faulty clocks $a_1, \ldots, a_n$.

Case 1: $N \geq 7$. Consider some *gcci*. Assume that a_k is faster than a_l in *gcci* if $k<l$. Now, let x_1 and x_2 present themselves as the fastest two clocks to $a_1, \ldots, a_p$, and as the slowest two clocks to the other non-faulty clocks, i.e. $a_{p+1}, \ldots a_n$, where $p=\lceil n/2 \rceil = f(N)-1$. Then, the set of scenarios can be represented as in Figure 5.

Recalling that a clock triggers on the $f(N)$-*th* tick in its scenario *not counting itself*, we can draw the trigger graph as in Figure 6. It follows that $\{a_1, \ldots, a_p\}$ and $\{a_{p+1}, \ldots, a_n\}$ will be two non-overlapping 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.

4. MAIN RESULT

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

We begin with the following two lemmas.

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

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

Proof: Let k be a non-faulty clock such that $p_k = x$. We must show that Eq. (1) holds for all x for which p_k is defined iff condition C2 holds.

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

(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 non-faulty, we have nothing to prove. If it is faulty, then since there are at most m faulty clocks, there must be at least one non-faulty clock slower than the reference clock from the right half of the inequality, $f_x(N) < N-m$. Also, from the left half of the inequality, $f_x(N)>x-1$, and since clock k is non-faulty, there is a non-faulty 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 non-faulty clock (the right half of the inequality), and faster than at least one non-faulty clock (the left 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 non-faulty clock that is faster than the reference clock, if the reference clock is faulty. Also, since k is non-faulty, and appears in the x-th (i.e. p_k-th) position, there is at least one non-faulty 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 $1,\ldots,f_x(N)$ of S_k. Similarly, if $f_x(N) \geq \max\{N-m,x\}$, C2 is violated when faulty clocks appear in positions $f_x(N)+1,\ldots,N$ of S_k.[3] **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 non-faulty clocks in G_1.

---

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

(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 scenario, consider the case where they present themselves in the scenario of every $y \in L$ in the $q_1+1,...,q_1+i$ positions. Then, there is no non-faulty 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 report.

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 non-faulty clocks in G_1 trigger

on clocks in G_1 is equivalent to saying that one of the following Eqs. (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 $y\in M$, the following Eq. (3) applies:

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

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

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

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

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

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

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

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

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

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

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

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

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

if Eq. (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 Eq. (3) applies.

We first consider the case where Eq. (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 Eq. (5') the following inequality must hold:

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

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 $p_y \leq N\text{-}m$, and so order themselves in S_z that $p_z > m$. Since Eq. (7) must hold *always*, no matter what

the faulty clocks do, we must have:

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

from which we arrive at the equation

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

Recall that this applies whenever Eq. (2) holds. If, instead, Eq. (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 Eq. (2) always applies (and therefore Eq. (3) never applies), and for which Eq. (8) is satisfied. But, we can always construct $f_x(N)$ to (i) be monotonically non-increasing functions of x and (ii) satisfy Eq. (8): an example of such a construction is provided in the statement of Theorem 2 below. Hence Eq. (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.**

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 non-increasing in x. Clearly, C2 holds. Also, it is easy to see that if $N>3m$ and Eq.

(2) implies Eq. (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 Eq. (4) if Eq. (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 $x = 1,...,N$.

(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 non-faulty clock g in the system (this clock must have the maximum value of $f_x(N)$) in clock g's scenario is that it is in the first $m+1$ clocks in that scenario. Similarly, all that we can tell about the position of the slowest non-faulty 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 non-increasing 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 non-increasing functions of x, then the following condition C3 holds.

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

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

Consider the fastest non-faulty clock. In the course of proving Theorem 1, it was established that $N-m > f_x(N) > m$ for all $x=1,...,N$, 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 non-faulty clocks. From the monotonic nature (in p_k) of the $f_{p_k}(N)$, the trigger for the fastest non-faulty clock must lie in the interval $2m+1$, ..., $N-m$. But, since $N \geq 3m+1$, any faulty clock in this interval must be sandwiched between non-faulty clocks.

The proof for the slowest non-faulty clock is similar. **Q.E.D.**

5. DISCUSSION

In this report, 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 the Introduction, 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 1 is a comparison of the techniques for synchronization in the face of malicious failure.

The phase-locked theorem here is completely different from the other ways -- the interactive convergence and interactive consistency algorithms -- of handling malicious failure. However, we have proved in this report that the theorem requires the same necessary condition $N \geq 3m+1$ as in software synchronization. It would be interesting to make a comparative study of the various algorithms that handle malicious failure and try to establish a unified theory from which the proofs for the interactive convergence and interactive consistency algorithms, as well as that of Theorems 1 and 2 of this report, would follow. We believe that the $N>3m$ requirement, which is common to all the 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, April 1980.

[2] J. Goldberg, *et al.*, "Development and Analysis of the Software Implemented Fault-Tolerance (SIFT) Computer," *NASA 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, No. 10, pp. 1240-1255, October 1978.*

[4] C. M. Krishna, K. G. Shin, and R. W. Butler, "Synchronization and Fault-Masking in Redundant Real-Time Systems," *Digest of Papers, FTCS-14,* pp. 152-157, June 1984.

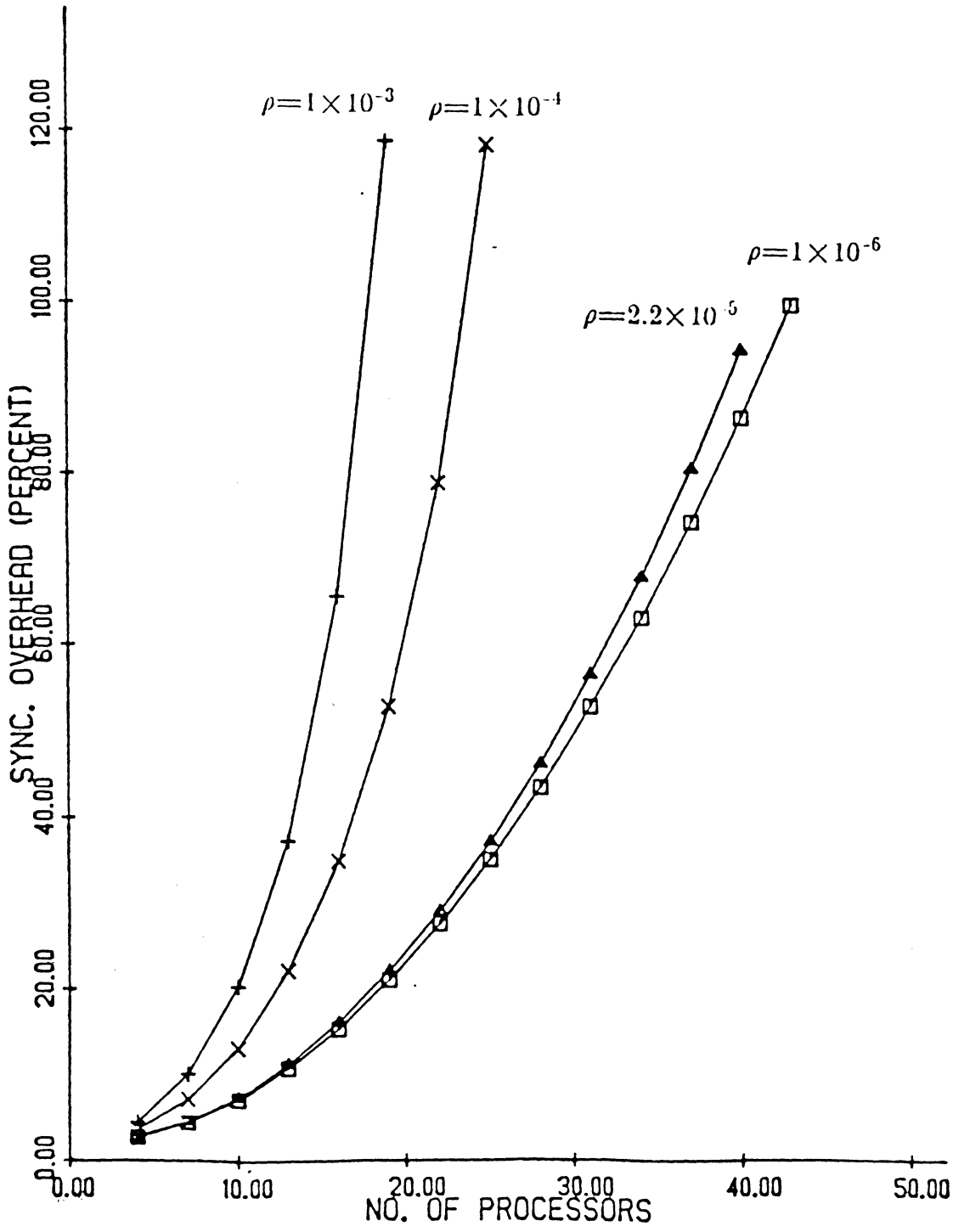
[5] D. Davies and J. F. Wakerly, "Synchronization and Matching in Redundant Systems," *IEEE Trans. Comput.*, Vol. C-27, No. 6, pp. 531-539, June 1978.

[6] A. L. Hopkins, *et al.*, "FTMP -- A Highly Reliable Fault-Tolerant Multiprocessor for Aircraft," *Proc. IEEE, Vol. 66, No. 10, pp. 1221-1239, October 1978.*

[7] A. W. Holt and J. M. Myers, "An Approach to the Analysis of Clock Networks," *NASA Contract Report,* CR-166028, November 1982.

[8] T. B. Smith, "Fault-Tolerant Clocking System," *Digest of Papers, FTCS-11,* pp. 262-264, 1981.

[9] F. Harary, *Graph Theory,* Addison Wesley Publishing Co., New York, 1969.



Resynchronization Interval = 100 ms

Figure 1. Overhead of Interactive Convergence Algorithm.

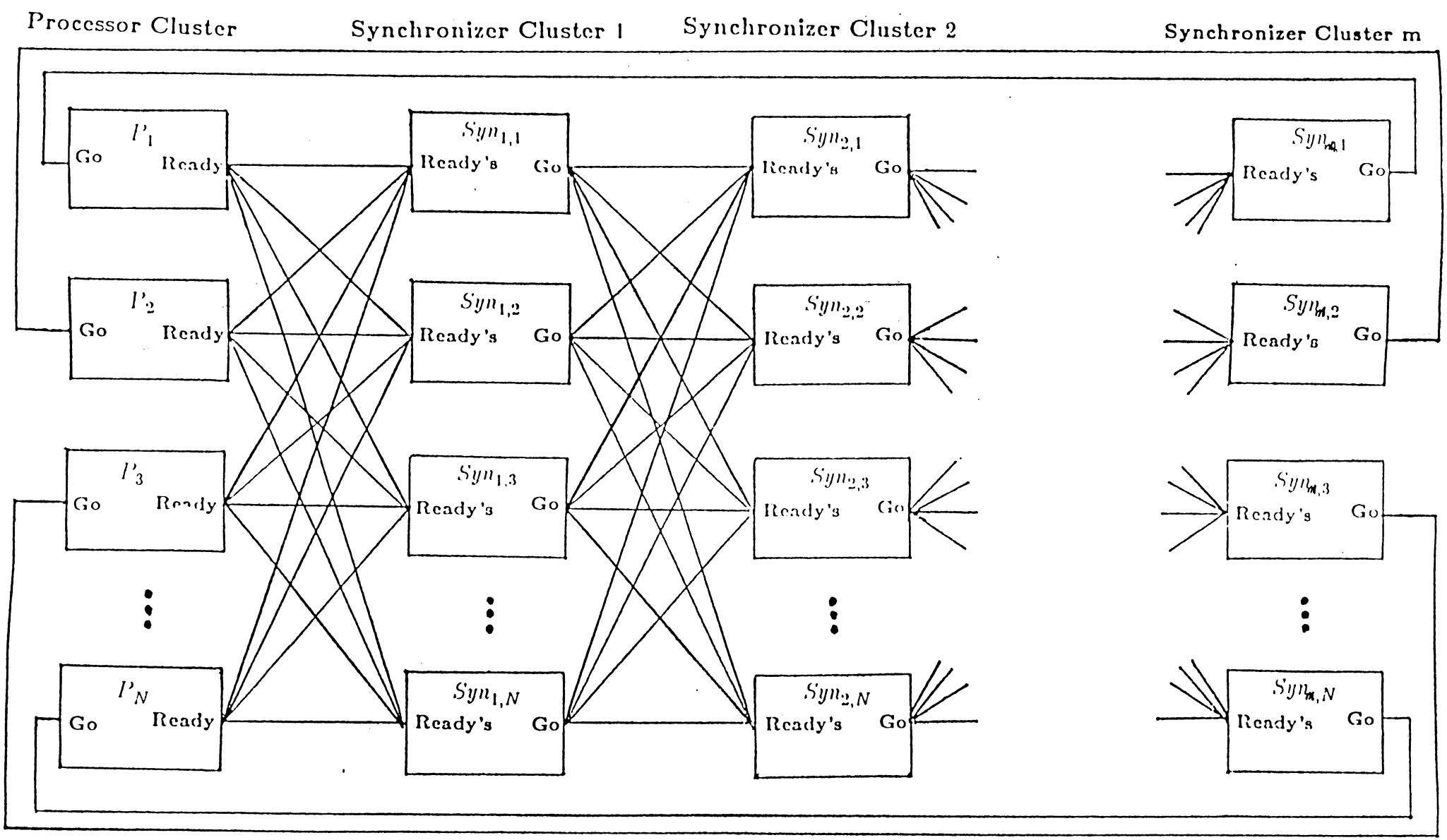


Figure 2. Davies and Wakerly's Multistage Synchronizer Design

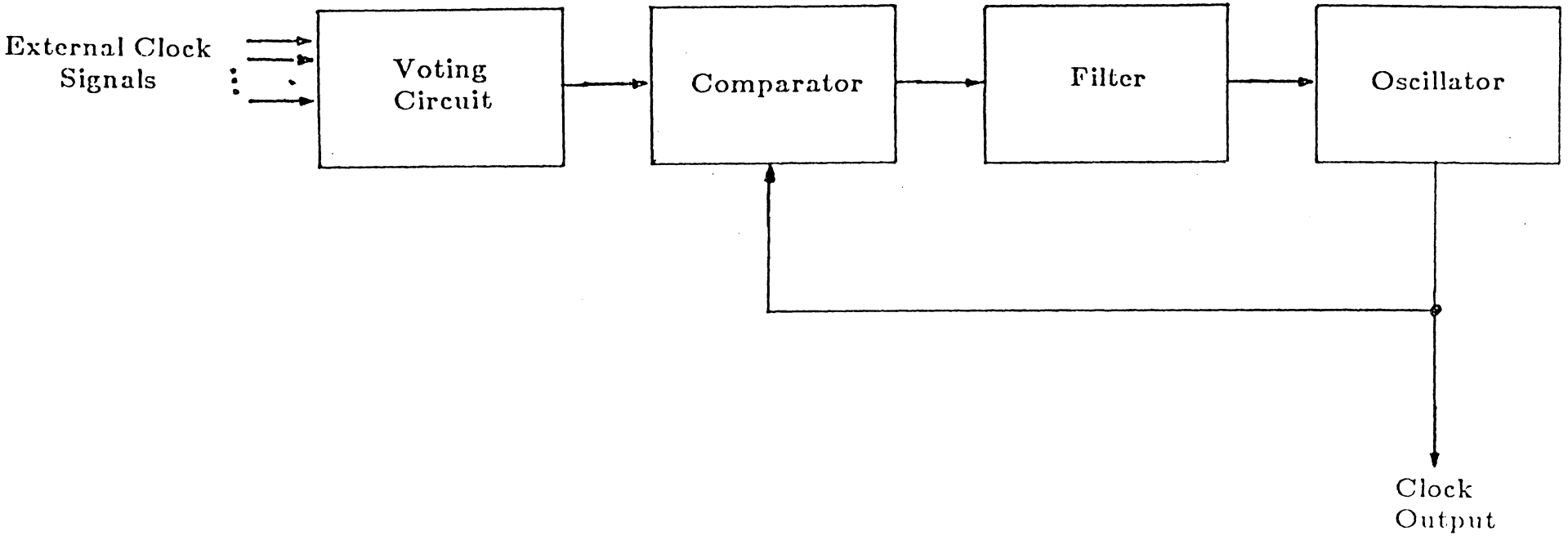
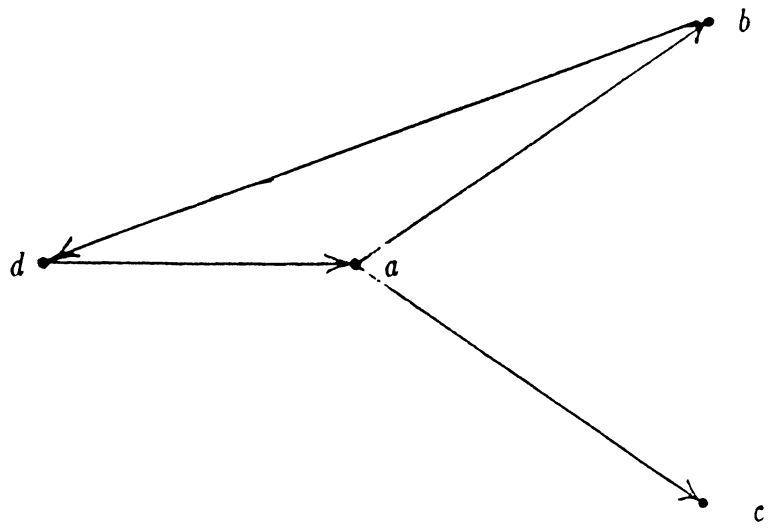


Figure 3. Component of a Phase-Locked Clock



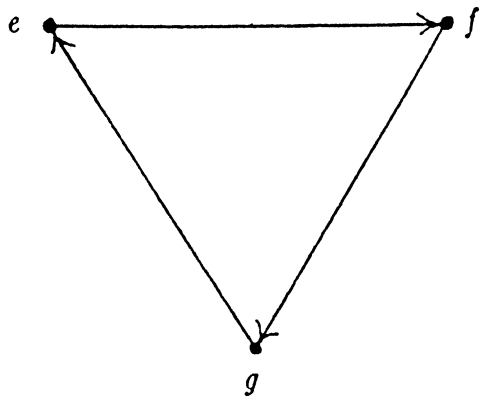


Figure 4. Trigger Graph: An Example

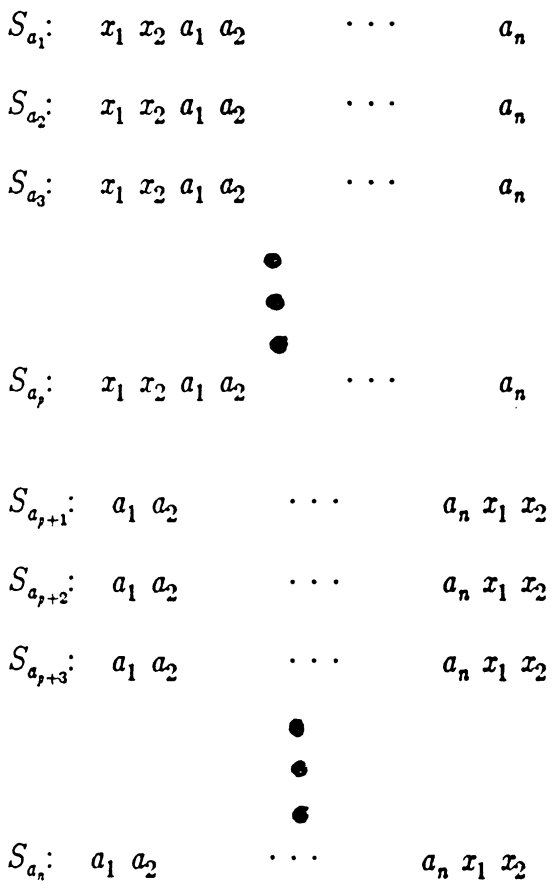


Figure 5: Scenario for Example in Section 3

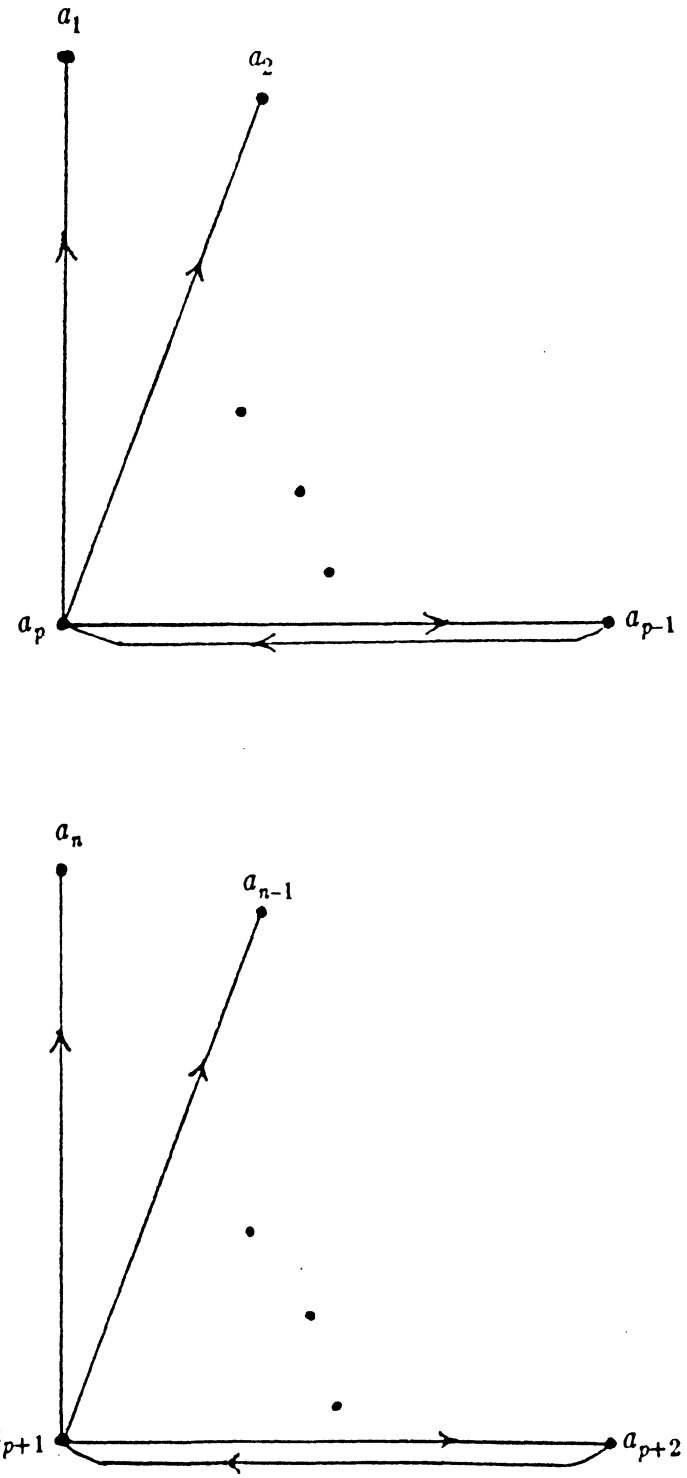


Figure 6. Trigger Graph for Scenarios in Figure 5

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

Table 1. Comparison of Synchronization Methods