A New Probablistic Algorithm for Clock Synchronization

K. Arvind

Computer and Information Science Department
University of Massachusetts

COINS Technical Report 89-86
August 1989

A NEW PROBABILISTIC ALGORITHM FOR CLOCK SYNCHRONIZATION [1]

K. Arvind
Department of Computer and Information Science,
University of Massachusetts at Amherst,
Amherst, MA 01003
E-Mail: arvind@cs.umass.edu

COINS Technical Report 89-86

August 1989

[1]This work is part of the Spring Project at the University of Massachusetts funded in part by the Office of Naval Research under contract N00014-85-K-0398 and by the National Science Foundation under grant DCR-8500332

Abstract

We present a new averaging probabilistic clock synchronization algorithm, based on the redundant transmission of multiple synchronization messages, that can guarantee a much lower upper bound on the deviation between clocks than most existing algorithms. Our algorithm is probabilistic in the sense that the upper bound on the deviation that it guarantees has a probability of invalidity associated with it. The probability of invalidity, i.e., the probability that the deviation exceeds the guaranteed maximum deviation, may however be made extremely small by sufficiently increasing the number of messages transmitted. We prove that an upper bound on the probability of invalidity decreases exponentially with the number of messages, i.e., the probability of invalidity itself decreases exponentially or better.

Contents

1. Introduction

A fault-free hardware clock (HC), even if initially synchronized with a standard time reference, tends to drift away from the standard over a period of time. If such a clock is used to measure a time interval (t_1, t_2), the measured length of the interval $(HC(t_2) - HC(t_1))$ tends to differ from the actual length of the interval. However, the fractional error in the measurement of the interval is bounded by a constant ρ_a called the *drift rate* of the clock:

$$\left|\frac{(HC(t_2) - HC(t_1)) - (t_2 - t_1)}{(t_2 - t_1)}\right| < \rho_a. \tag{1}$$

ρ_a is typically of the order of 1 μsec/sec; thus an initially synchronized fault-free clock could drift away from the reference by a few seconds every few days, in the worst case. A *clock synchronization algorithm* is used in a distributed system, to synchronize clocks that can drift apart, to ensure that the maximum deviation between the clocks is bounded.

Clock synchronization has been an active area of research in distributed systems ([1]-[14]). It is of special importance in distributed real-time systems, where an event may spread across multiple nodes each with its own local clock. In order to specify the timing constraints of such events, a common reference of time is necessary in a distributed real-time system [5]. A clock synchronization algorithm is used to generate such a common reference of time.

1.1 Deterministic Algorithms

Several algorithms have been proposed for clock synchronization ([1], [2], [4], [5], [8], [9], [10]). However all these algorithms with the exception of the algorithm in [1] are limited in the upper bound (γ_{max}) on the deviation between clocks in the system that they can guarantee, by certain bounds derived in [3] and [11]. Lundelius and Lynch [11] have shown that, even with no failures and with clocks that do not drift, for a system of N clocks, the γ_{max} that can be guaranteed has a lower bound given by:

$$\gamma_{max} \geq (d_{max} - d_{min}) \left(1 - \frac{1}{N}\right) \qquad (2)$$

(Dolev, Halpern and Strong [3] have proved a related result for the closeness of synchronization obtainable along the real time axis). Here d_{max} and d_{min} refer to the maximum and minimum possible values respectively, of the end-to-end delay (d) of synchronization messages. The end-to-end delay of a message, which is the sum of the time required for a message to be prepared, sent to a receiver and be processed by the receiver, is a random delay determined by random events such as transmission errors that necessitate the retransmission of a message, context switches and page faults.

1.2 Probabilistic Algorithms

The bounds derived in [3] and [11], however apply only to *deterministic* clock synchronization algorithms, i.e., algorithms that guarantee a γ_{max} with *certainity*. If we relax the requirement of certainity and permit an algorithm to provide a probabilistic guarantee (i.e., the guarantee provided by the algorithm may fail to hold good sometimes, but with a *failure probability* that is known or that has a known bound), then we can find algorithms that can guarantee a γ_{max} which need not conform to Eq. (2). However for a probabilistic guarantee to be useful, it must be possible to reduce the failure probability to any desired level by choosing the parameters of the algorithm suitably. Clock synchronization algorithms that can provide such a probabilistic guarantee on γ_{max} are referred to as *probabilistic*[1] clock synchronization algorithms.

[1]The word 'probabilistic' refers to the uncertainity in the guarantee offered by the algorithm and is different from the sense in which the word is used in [3], where a probabilistic algorithm is an algorithm whose behaviour itself is randomly determined.

1.2.1 Cristian's Algorithm

Cristian [1] has proposed one (probably the first) such probabilistic clock synchronization algorithm that is capable of guaranteeing much lower γ_{max}'s than deterministic algorithms. Cristian's algorithm, based on reattempting synchronization if a process does not receive a reply to its request for synchronization within a specified timeout period, guarantees a γ_{max} at synchronization of $U - d_{min}$, where U denotes half the specified timeout period. The γ_{max} can be reduced, for a specified probability of losing synchronization, by increasing the maximum number of attempts permitted.

1.2.2 Our Algorithm

In this paper, we propose an algorithm that represents a new approach to probabilistic clock synchronization, based on averaging end-to-end delays of synchronization messages. Our algorithm makes use of a new probabilistic time transmission protocol, that can be used to transmit the time on the clock of one process to another process with a desired accuracy. The protocol involves the transmission of multiple synchronization messages and works by filtering out the random variations in the end-to-end delays of the synchronization messages through a process of averaging.

Our algorithm, like Cristian's algorithm, can guarantee a much lower γ_{max} than the deterministic algorithms that have to conform to Eq. (2). Also, the probability that the guarantee provided will fail to hold good can be reduced (with an *exponential* rate of decrease) to an arbitrarily small value by increasing the number of synchronization messages. *An interesting feature of our approach is that if a certain minimum level of message traffic can be assumed to exist between nodes, then there is no need for any special synchronization messages to be transmitted at all.*

The rest of the paper is organized as follows. In the next section, we provide an overview of our algorithm. We first state the assumptions underlying our work. We then discuss the proposed time transmission protocol (TTP) and describe how a clock synchronization algorithm can be constructed from it. Next, we state the main properties of TTP and the clock synchronization algorithm and provide examples to illustrate them. We then provide an analysis of TTP and the clock synchronization algorithm and prove the properties stated. We conclude the paper by summarizing it and identifying potential areas of future research. We have provided in Appendix B, a glossary of symbols used in the paper, for the convenience of the reader.

2. Overview of Our Algorithm

2.1 Assumptions

We make the following assumptions about clocks, processes and message delays:

1. The system consists of N hardware clocks HC_i, with one clock synchronization process associated with each clock. The synchronization processes have only read-only access to their own hardware clocks.

2. The hardware clocks are ρ-bounded, i.e., they obey Eq. (1), which can be rewritten as follows:
$$(1-\rho_a)(t_2-t_1) \leq HC(t_2) - HC(t_1) \leq (1+\rho_a)(t_2-t_1) \tag{3}$$
where ρ_a is the drift rate of the clocks. It follows that the rate at which two ρ-bounded clocks can drift apart from each other during an interval of time τ is bounded by $(1 + \rho_a)\tau - (1-\rho_a)\tau = 2\rho_a\tau$. We refer to the term $2\rho_a$ as the *relative clock drift rate* (ρ).

3. Each synchronization process maintains one or more (depending on whether continuous or instantaneous adjustment is used) logical clocks, each of which is a linear function of the time on the hardware clock of the process. In the rest of the paper, unless otherwise stated, the term 'clock' always refers to the logical clock (or the current logical clock if instantaneous adjustment is used) of the process under consideration.

4. Processes can have performance or omission faults [2].

5. A process conveys the time on its clock to another process by transmitting synchronization messages. The end-to-end message delay (d) between two processes, at any instant, is unpredictable and indeterminable and can be modelled as a random variable. The end-to-end delay at different instants of time are assumed to be independent of each other.

6. The end-to-end message delay at any instant has an expected value of $\bar{d}$ and a standard deviation of σ_d. We discuss later how it may be possible to separate successive messages in time if that is necessary to ensure the independence of successive message delays and how occasional transient perturbations in the expected value and standard deviation can be accomodated easily, provided these perturbations do not last too long. Note that we do not make any specific assumptions about the shape of the probability density function of d.

The last two assumptions about the independence of d and invariability of the mean and standard deviation of d are similar to the assumptions made by Cristian in [1] about the independence of successive attempts and the invariability of the probability distribution of the end-to-end message delay.

2.2 The Algorithm

We first describe how a process M can transmit the time on its clock to a process S with a maximum *transmission error* of ϵ_{max}. We then describe how this method of transmitting time can be used to construct a clock synchronization algorithm.

2.2.1 Transmitting Time

A process M can transmit the time on its clock to a process S by sending n synchronization messages according to the time transmission protocol, TTP, described below. The length of the time period starting at the time the first message is sent and ending at the time the nth message is sent is referred to as the *transmission period* (τ). On receiving the nth message, S makes an estimate T_{est} of the time on M's clock. The *transmission error* ϵ is defined as the difference between the estimate, that S makes of the time on M's clock, and the actual time T_{act} on M's clock at the time S makes the estimate, i.e.,

$$\epsilon = T_{est} - T_{act} \tag{4}$$

TTP works as follows. When it is time for M to transmit its time, M starts sending synchronization messages, msg_i, to S. The ith message, msg_i, which is sent at time T_i on its clock, is of the form "Time is T_i". It sends n such messages within a short transmission period τ (the transmission period τ is chosen to be short enough that the maximum possible clock drift during this period given by $\rho\tau$ is negligible compared to the desired maximum transmission error ϵ_{max}). At the S end, each time a timing message, msg_i, is received, S records its time of receipt, R_i, according to its local clock. After it has received n messages, S computes the averages of the times on the messages and the receipt times as

$$\bar{T}(n) = \frac{1}{n}\sum_{i=1}^{n} T_i$$

and

$$\bar{R}(n) = \frac{1}{n}\sum_{i=1}^{n} R_i.$$

It then estimates the time on M's clock as:

$$T_{est} = R_n - \bar{R}(n) + \bar{T}(n) + \bar{d} \tag{5}$$

We show later that this method results in an average transmission error that is negligible in comparison to the maximum transmission error. We also show that the maximum transmission error (ϵ_{max}) and the probability (p) that the transmission error exceeds this value (known as the *probability of invalidity*) can be made very small by sufficiently increasing the number of synchronization messages transmitted.

2.2.2 Clock Synchronization

The time transmission protocol that we developed in the previous section can be used to construct a clock synchronization algorithm, in a manner similar to how the remote clock reading method is converted to a master/slave clock synchronization algorithm in [1]. As in [1], one process in the system is designated as the *master* and the clocks of the other processes (the *slaves*) are synchronized to the clock of this process through a master/slave protocol. Our algorithm works as follows.

The master process periodically (at intervals R_{synch} according to its clock) transmits its time to the other processes using the time transmission protocol of the previous section. Each slave, when it has received n messages, estimates the time of the master according to the time transmission protocol of the previous section, computes the adjustment $T_{est} - R_n$ that it needs to make to its clock and makes the adjustment.

We did not mention how a slave process adjusts its logical clock at a resynchronization point. Once a slave process has computed an adjustment through the time transmission protocol, it can either modify its logical clock *instantaneously* at the resynchronization point or *continuously* over a period of time. Even though either of these methods can be used with this synchronization protocol, instantaneous adjustment has disadvantages like the possibility of a backward correction of the time or the need to maintain multiple logical clocks [5].

2.3 Main Properties

When the transmission period, τ, is short enough and the number of synchronization messages, n, is not too large, TTP and the clock synchronization protocol above exhibit certain interesting and useful properties. In this section, we summarize these properties. A detailed analysis and derivation of these results is contained in Section 3..

2.3.1 Constraints on τ and n

We first formally state the constraints that must be satisfied by τ and n in order for the properties stated in this section to hold good.

1. The transmission period τ must be sufficiently short. More specifically,

$$\tau \ll \kappa \frac{\epsilon_{max}}{\rho} \tag{6}$$

where κ is determined by the desired probability of invalidity (p). We deduce the value of κ in Section 3.. For now it is sufficient to know that $\kappa \leq 1$. Note that if an ϵ_{max} of the order of 1 millisec is desired, then this constraint means that $\tau \ll 100$ seconds (assuming typical values for ρ ($\sim 10^{-6}$) and κ (~ 0.1)).

2. The number of messages must not be too large. Specifically,

$$n \ll \frac{1}{\rho}. \tag{7}$$

Assuming ρ is of the order of 10^{-6} we get $n \ll 10^6$.

2.3.2 Transmission of Time

The transmission error (ϵ) is later shown to be a random variable that depends both on the clock drift during the transmission period and the average end-to-end delay of the transmitted synchronization messages. However,

Property 1 *The average transmission error ($\bar{\epsilon}$) is negligible compared to the maximum transmission error, i.e.,* $|E[\epsilon]| \ll \epsilon_{max}$.

ϵ has the following interesting property.

Property 2 *The probability distribution of the transmission error, ϵ approaches the Gaussian distribution $N\left(\bar{\epsilon}, \frac{\sigma_d^2}{n}\right)$, as the number, n, of synchronization messages increases.*

We refer to the minimum value of n for $N(\bar{\epsilon}, \frac{\sigma_d^2}{n})$ to approximate $G_n(\epsilon)$, the probability distribution of ϵ, with an accuracy ξ as the *Gaussian cutoff* corresponding to ξ. We denote this cutoff value by n_g. In Section 3. we state why we believe that a $n_g < 10$ would be sufficient to approximate the distribution of ϵ by a Gaussian distribution with a good accuracy.

The following theorem provides an analytical expression for computing the number of synchronization messages required.

Theorem 1 *The minimum number of messages required to guarantee a maximum deviation of ϵ_{max} with a probability of invalidity of p is given by $n_{min} = \max(n_g, n_e)$, where*

$$n_e = \frac{2\sigma_d^2(\text{erfc}^{-1}(p))^2}{\epsilon_{max}^2}. \tag{8}$$

Here $\text{erfc}^{-1}(p)$ *is the inverse of the complementary error function defined as* $\text{erfc}(u) \triangleq 1 - \text{erf}(u)$, *where* $\text{erf}(u) \triangleq \frac{2}{\sqrt{\pi}} \int_0^u e^{-y^2}\, dy$.

On the basis of the above theorem, we deduce, in Section 3., the value of κ that we earlier left unspecified to be $\kappa = \min\left(1, \frac{1}{\sqrt{2}\text{erfc}^{-1}(p)}\right)$.

The erfc function is a decreasing function of its argument and so it can be seen, by solving Eq. (8) for p, that the probability of invalidity decreases, for a given ϵ_{max}, with increasing n.

Complementary Error function			
x	erfc (x)	x	erfc (x)
0.000	1.0×10^{0}	3.459	1.0×10^{-6}
1.163	1.0×10^{-1}	3.766	1.0×10^{-7}
1.822	1.0×10^{-2}	4.052	1.0×10^{-8}
2.327	1.0×10^{-3}	4.320	1.0×10^{-9}
2.751	1.0×10^{-4}	4.572	1.0×10^{-10}
3.123	1.0×10^{-5}	4.812	1.0×10^{-11}

Table 1: erfc(x)

The rate of decrease may be gauged by referring to Table 1. However, the following important result provides a clearer idea of the rate of decrease.

Property 3 *For a given ϵ_{max}, the probability of invalidity, p, decreases exponentially or better with the number of messages, i.e., there exists a bounding function for p of the form $\alpha e^{-n\beta}$ such that the magnitude of α does not increase with n and $\beta > 0$ is independent of n, i.e., $p = \mathbf{P}[|\epsilon| > \epsilon_{max}] < \alpha e^{-n\beta}$*

It will be seen later that the coefficient α is also a decreasing function of n ($\propto \frac{1}{\sqrt{n}}$). Thus the coefficient is also responsible for a decrease, albeit a weak one, in the probability of invalidity with increasing n.

The next theorem is the basis of our claim that our algorithm is exempt from the bounds imposed by Eq. (2). TTP provides a way of determining the time on a remote clock with any desired accuracy in spite of variable message delays.

Theorem 2 *TTP can guarantee any desired maximum transmission error, $\epsilon_{max} > 0$, with any desired probability of invalidity p in the range $0 < p \leq 1$ as long as the choice of values for ϵ_{max}, n and τ is consistent with Eq. (6) and Eq. (7).*

2.3.3 Example

We present an example below to provide an idea of the magnitudes of the various parameters involved in TTP. If the ϵ_{max} desired is of the order of millisecs (say 1.0 msec, which is one hundreth of the d_{max} (= 0.1 sec) quoted in [4]), ρ is taken to be 2 μsec/sec, σ_d is assumed[2] to be 1 millisec and a probability of invalidity of 1×10^{-6} is desired then $\kappa = \min(0.2, 1) = 0.2$ and $\tau \leq \kappa \frac{\epsilon_{max}}{10\rho} = 10$ seconds. The minimum number of messages that have to be transmitted to achieve this accuracy, assuming an $n_g = 10$, is given by Theorem 1 to be 24. Thus the desired accuracy (which is *100 times better* than the accuracy of synchronization at the resynchronization point obtainable with, for example, the algorithm in [4]) is achieved by transmitting just 24

[2][1] describes an experiment involving a measurement of 5000 message roundtrip delays between two processes running on two IBM 4381 processors connected via a channel-to-channel local area network. The observed d_{max} in that experiment was 93.17 millisec, d_{min} was 4.22 millisec, $\bar{d}$ was 4.91 millisec, the median round trip delay was 4.48 millisec and 95% of all observed delays were shorter than 5.2 millisec. These are roundtrip values and have to be scaled down by a factor of 2 for our purposes.

While the above data are sufficient to compute the probability distribution function of the end-to-end message delay for some values of d, they are insufficient to compute the standard deviation of the end-to-end delay, which is an important parameter of our algorithm. However if we approximate the distribution of the end-to-end delays by a Gaussian distribution, we can obtain an approximate value for σ_d. In a later section, we conjecture that the distribution of end-to-end delays is approximately Gaussian and present arguments to justify our conjecture.

Note that the maximum deviation from the median for 95% of the cases in the above data is less than 0.36 millisec. It can be shown that, if the probability of the event that a normally distributed random variable differs from its median by more than 0.36 is 0.05, then the standard deviation of the variable is 0.184. Hence if we approximate the distribution of the end-to-end delays by a Gaussian distribution that is centered at the median value of the end-to-end delay quoted above and whose spread is such that the area under the density function curve within ± 0.36 millisec of the median point is 0.95, then *we can estimate σ_d to be* 0.184.

n vs. p ($\frac{\epsilon_{max}}{\sigma_d} = 1$)		n vs. $\frac{\epsilon_{max}}{\sigma_d}$ ($p = 1 \times 10^{-6}$)	
p	n	$\frac{\epsilon_{max}}{\sigma_d}$	n
1.0×10^{-6}	24	10.0	10
1.0×10^{-7}	29	3.0	10
1.0×10^{-8}	33	2.0	10
1.0×10^{-9}	38	1.0	24
1.0×10^{-10}	42	0.75	43
1.0×10^{-11}	47	0.50	96

Table 2: Number of synchronization messages versus p and $\frac{\epsilon_{max}}{\sigma_d}(n_g = 10)$

messages over a period of length not exceeding 10 seconds[3]. A better accuracy and/or a smaller probability of invalidity would require more messages (Table 2). Note however that there is an upper limit on n given by Eq. (7). In practice, Eq. (6) also limits the maximum number of messages that can be transmitted, because, transmission of messages requires instruction execution. Instruction execution takes a finite amount of time and hence there is a limit on the number of messages that can be transmitted within a period τ.

An interesting point to note is that if a minimum level of message traffic always exists between M and S, so that we can assume that at least n messages, apart from the clock synchronization messages, are sent from M to S in a period τ, then there is no need for special clock synchronization messages at all. M can stamp each message that it sends to S with the time on its clock, and these timestamped messages now carry the synchronization information that S requires.

[3]This period is the transmission period and is not to be confused with the resynchronization interval, R_{synch}, that arises when this transmission protocol is used to construct a clock synchronization protocol. R_{synch} is much larger than this period, as we show later

2.3.4 Clock Synchronization

The following property describes the most important characteristic of our clock synchronization algorithm.

Property 4 *Our clock synchronization algorithm can guarantee any desired upper bound, γ_{max}, $\gamma_{max} > 0$, between any two clocks in the system i.e.,*

$$\forall i, j, t\, |L_i(t) - L_j(t)| \leq \gamma_{max}$$

with any desired probability of invalidity, p ($0 < p \leq 1$), of the guarantee, as long as the choice of values for γ_{max}, p and τ is consistent with Eq. (6) and Eq. (7). Here i, j denote processes and $L_q(t)$ denotes the time on process q's logical clock at real time t.

Proof: Theorem 2 guarantees that the maximum deviation between the master and a slave immediately after a resynchronization is no more than ϵ_{max} with a probability of invalidity of p. The durations between successive resynchronizations at a slave can never exceed $R_{synch} + d_{max} - d_{min}$, since the worst case occurs when a slave receives the last synchronization message with minimum delay (i.e., at time $T_1 + \tau + d_{min}$, where T_1 is the time of transmission of the first synchronization message) in one synchronization cycle and with maximum delay (i.e., at time $T_1 + R_{synch} + \tau + d_{max}$) in the next cycle and for this case the duration between successive resynchronizations is given by $R_{synch} + d_{max} - d_{min}$. Thus the maximum deviation (due to clock drift) that can develop between the clocks of the master and a slave during the interval between two successive resynchronizations is given by $\rho(R_{synch} + d_{max} - d_{min})$. The maximum deviation between a slave and the master at any time is therefore given by:

$$\gamma_{ms}^{max} = \epsilon_{max} + \rho(R_{synch} + d_{max} - d_{min}) \tag{9}$$

The maximum deviation between any two clocks in the system is given by $2\gamma_{ms}^{max}$:

$$\gamma_{max} = 2\gamma_{ms}^{max} = 2(\epsilon_{max} + \rho(R_{synch} + d_{max} - d_{min})) \tag{10}$$

By choosing suitable ϵ_{max}, R_{synch} and n, any desired γ_{max} and p can be guaranteed as long as the choice does not result in the violation of Eq. (6) or Eq. (7).

Q.E.D.

2.3.5 Comparison

If the γ_{max} desired is 4 millisec and the desired probability of invalidity is 1×10^{-6}, we can choose $\epsilon_{max} = 1$ millisec and $\rho(R_{synch} + d_{max} - d_{min}) = 1$ millisec which for $\rho = 2\mu$sec/sec, $n_g = 10$ and $\sigma_d = 1$ millisec gives an R_{synch} of ≈ 500 sec and $n = 24$ messages. A less tight γ_{max} of 6 millisec can be achieved with the same probability of invalidity, the same resynchronization interval and $n = n_g = 10$ messages, by choosing $\epsilon_{max} = 2$ millisec and $\rho R_{synch} = 1$ millisec. The γ_{max}'s in these examples are better than the best γ_{max} of $\frac{1}{2}(d_{max} - d_{min}) \approx 50$ millisec achievable with the clock synchronization algorithms described in [2], [4], [5], [6], [8], [9] and [10] which have to conform to Eq. (2), and are comparable to the γ_{max} achievable with the probabilistic algorithm in [1]. Also, by Property 3 it should be possible to reduce the probability of invalidity substantially by increasing the number of messages by a small number. This is illustrated in Table 2. Note that p can be reduced by an order of magnitude by increasing the number of synchronization messages by just 5.

We are unable to provide any meaningful comparison of the number of messages that the algorithm in [1] requires against the number that our algorithm requires, to guarantee a specified γ_{max} and p. This is because the data on round trip delays reported in [1] is not sufficient to determine σ_d accurately. We can, no doubt, try to provide an approximate comparison by making use of the approximate estimate of σ_d that we obtained earlier. However this comparison would not be very reliable because the number of messages required by our algorithm is quite sensitive to the ratio of σ_d and ϵ_{max} ($n_e \propto \left(\frac{\sigma_d}{\epsilon_{max}}\right)^2$). If σ_d is of the order of a few tenths of milliseconds and the desired ϵ_{max} is also of the same order, an error of a few tenths of milliseconds in estimating σ_d will cause a large error in the estimate of the number of messages. For example, if $\sigma_d = 0.184$ and if the ϵ_{max} desired is 0.184 millisec, then the number of messages required by our algorithm to guarantee a p of 1.0×10^{-9} is 38. If σ_d is 0.25 millisec instead, the number of messages required is 69. Thus we are off from the actual number of messages required by 31 if we make an error of 0.066 millisec in estimating σ_d. Note that an overestimation of σ_d results in an overestimation of n. Thus our earlier examples actually overestimate the number of messages required.

If however the ϵ_{max} desired is somewhat larger than σ_d, then small errors in estimating σ_d do not cause significant errors in the estimate of n. For example, if we assume n_g to be 10, then, by Theorem 1, $n = 10$ for all values of $\epsilon_{max} > 1.949\sigma_d$, and $n = 2$ for all values of $\epsilon_{max} > 4.36\sigma_d$, if we assume a $n_g = 2$. Thus if a ϵ_{max} of 1 millisec is desired, and we estimate σ_d as 0.184, the number of messages required remains 10 (assuming $n_g = 10$), even if we have made an error of 0.3 millisec in estimating σ_d. If we assume a approximately Gaussian distribution for the end-to-end delays, then n_g is close to 1. The number of messages required to guarantee an $\epsilon_{max} > 4.36\sigma_d$ millisec and a $p = 1.0 \times 10^{-9}$ is then equal to $max(n_g, 2)$ (assuming $\sigma_d = 0.184$). This is comparable to the average number of messages (≈ 2) required by the algorithm in [1] to guarantee the same ϵ_{max} and p.

An additional point to be noted is that, if a minimum level of message traffic always exists between M and S, *our algorithm requires no synchronization messages at all*. For example, if an ϵ_{max} of 1 millisec and a probability of invalidity of 1.0×10^{-9} are desired, the minimum traffic level required for operating without any synchronization messages is just 3.8 messages per second (at least 38 messages in a span of $\tau = 10$ seconds), even if we assume a σ_d of 1.0 millisec.

3. Analysis

In this section we provide a detailed analysis of TTP and the clock synchronization protocol and establish their properties.

Lemma 1 *Let d_i and δ_i respectively denote the end-to-end delay of the ith message and the deviation between the clocks of S and M when the ith message is received at S. The transmission error ϵ is given by:*

$$\epsilon = \Delta\delta_n - \Delta\bar{d}(n) \tag{11}$$

where $\Delta\delta_n \triangleq \delta_n - \bar{\delta}(n)$, $\Delta\bar{d}(n) \triangleq \bar{d}(n) - \bar{d}$, $\bar{\delta}(n) = \frac{1}{n}\sum_{i=1}^{n}\delta_i$ and $\bar{d}(n) = \frac{1}{n}\sum_{i=1}^{n} d_i$.

<u>Proof</u>: The ith message is received at S d_i time units after it was stamped with the time on it. When it arrives at S, the clock of S is ahead of the clock of M by δ_i time units. Hence

$$R_i = T_i + d_i + \delta_i$$

and

$$\bar{R}(n) = \bar{T}(n) + \bar{d}(n) + \bar{\delta}(n).$$

The latter equation can be rewritten as

$$\bar{T}(n) - \bar{R}(n) = -\left(\bar{d}(n) + \bar{\delta}(n)\right). \tag{12}$$

The nth message was sent at time T_n on M's clock and it took d_n units of time to arrive at S. Hence the actual time (T_{act}) on M's clock when the nth message is received by S is equal to the sum of the time stamped on the message and its delay. Alternatively, T_{act} may be obtained by deducting the deviation (δ_n) between S and M at the time of arrival of the nth message at S, from the time of arrival. Hence

$$T_{act} = T_n + d_n = R_n - \delta_n \tag{13}$$

The transmission error is then given by equations (4), (5) and (13) as

$$\begin{aligned}
\epsilon &= T_{est} - T_{act} \\
&= (R_n - \bar{R}(n) + \bar{T}(n) + \bar{d}) - (R_n - \delta_n) \\
&= \bar{d} + \delta_n - (\bar{d}(n) + \bar{\delta}(n)) \quad \text{(from Eq. (12))} \\
&= (\delta_n - \bar{\delta}(n)) - (\bar{d}(n) - \bar{d}) \\
&= \Delta\delta_n - \Delta\bar{d}(n)
\end{aligned}$$

Q.E.D.

It should be noted that $\bar{d}(n)$ and $\Delta\bar{d}(n)$ are themselves random variables; It is easy to see that, the expected value of $\bar{d}(n)$ is the same as the expected value of d while the expected value of $\Delta\bar{d}(n)$ is zero. However the variances of $\bar{d}(n)$ and $\Delta\bar{d}(n)$ are identical and smaller than σ_d^2 by a factor n.

$$V\left[\Delta\bar{d}(n)\right] = V\left[\bar{d}(n)\right] = \frac{\sigma_d^2}{n}. \tag{14}$$

We have not made use of the assumptions in Eq. (6) or (7) in our analysis so far. We make use of Eq. (7) in the following lemma to derive an useful approximation for ϵ.

Lemma 2 *Let δ_0 denote the deviation between the clocks of S and M at the start of transmission of msg_1, δ_i^T denote the increase in deviation between the clocks between the times of transmission of msg_1 and msg_i, and δ_i^d denote the increase in deviation during the time it takes msg_i to reach S after it has been transmitted (i.e., the increase in deviation during the message delay d_i of msg_i). Under the assumption that Eq. (7) holds good, ϵ can be approximated as*

$$\epsilon = \Delta\delta_n^T - \Delta\bar{d}(n) \tag{15}$$

where $\Delta\delta_n^T = \delta_n^T - \bar{\delta}_T(n)$ and $\bar{\delta}_T(n) = \frac{1}{n}\sum_{i=1}^{n}\delta_i^T$.

Proof: The quantity δ_i represents the amount by which the clock of S is ahead of the clock of M when the ith message arrives at S. It can be decomposed into three components, viz., the deviation between the clocks of S and M at the start of transmission of msg_1 (δ_0), the increase in deviation between the clocks of S and M between the times of transmission of msg_1 and msg_i (δ_i^T), and δ_i^d, the increase in deviation during the time it takes message msg_i to reach S after it has been transmitted (δ_i^d). Thus

$$\delta_i = \delta_0 + \delta_i^T + \delta_i^d \quad (1 \le i \le n)$$

Hence

$$\bar{\delta}(n) = \frac{1}{n}\sum_{i=1}^{n}\delta_i = \delta_0 + \bar{\delta}_T(n) + \frac{1}{n}\sum_{i=1}^{n}\delta_i^d$$

Therefore

$$\Delta\delta_n = \delta_n - \bar{\delta}(n) = \delta_n^T - \bar{\delta}_T(n) + \delta_n^d - \frac{1}{n}\sum_{i=1}^{n}\delta_i^d \tag{16}$$

Each δ_i^d can be written as $\delta_i^d = \hat{\rho}_i d_i$, where $\hat{\rho}_i \le \rho$ is the average drift rate during the end-to-end delay of message msg_i. Eq. (16) can then be rewritten as,

$$\Delta\delta_n = \delta_n^T - \bar{\delta}_T(n) + \hat{\rho}_n d_n - \frac{1}{n}\sum_{i=1}^{n}\hat{\rho}_i d_i \tag{17}$$

It follows from Lemma 1 that

$$\begin{aligned}
\epsilon &= \Delta\delta_n - \Delta\bar{d}(n) \\
&= \delta_n^T - \bar{\delta}_T(n) + \hat{\rho}_n d_n - \frac{1}{n}\sum_{i=1}^{n}\hat{\rho}_i d_i - \frac{1}{n}\sum_{i=1}^{n} d_i + \bar{d} \\
&= \delta_n^T - \bar{\delta}_T(n) - \frac{1}{n}\sum_{i=1}^{n} d_i + \bar{d} \\
&= \delta_n^T - \bar{\delta}_T(n) - \Delta\bar{d}(n) \\
&= \Delta\delta_n^T - \Delta\bar{d}(n)
\end{aligned}$$

In the third step above, we made use of Eq. (7) and neglected $\hat{\rho}_i$ and $\frac{\hat{\rho}_i}{n}$ in comparison to $\frac{1}{n}$.

Q.E.D.

Note that ϵ may be written as the sum of several independent random variables. We make use of this fact later. We look at some useful properties of ϵ now.

Property 1 *The magnitude of the average transmission error ($\bar{\epsilon}$) is negligible compared to the maximum transmission error, i.e.,*

$$|E[\epsilon]| \ll \epsilon_{max}.$$

<u>Proof</u>: The average transmission error $\bar{\epsilon}$ is given by $E[\epsilon]$. By Lemma 2,

$$\begin{aligned} |E[\epsilon]| &= |E[\Delta\delta_n^\tau - \Delta\bar{d}(n)]| \\ &= |E[\Delta\delta_n^\tau]| \\ &= |E[\delta_n^\tau - \bar{\delta}_\tau(n)]| \end{aligned}$$

Note that $\delta_n^\tau - \bar{\delta}_\tau(n) \leq \rho\tau$ since the maximum possible clock drift during an interval τ is given by $\rho\tau$. Hence

$$|E[\epsilon]| \leq \rho\tau \ll \kappa\epsilon_{max} \leq \epsilon_{max}$$

The second inequality above follows from Eq. (6).

Q.E.D.

Lemma 3 *If $\rho\tau \ll \frac{\sigma_d}{\sqrt{n}}$, the variance of the transmission error $V[\epsilon]$ is given by*

$$V[\epsilon] = \frac{\sigma_d^2}{n} \tag{18}$$

__Proof__: The variance of the transmission error $V[\epsilon]$ is given by

$$V[\epsilon] = V\left[\delta_n^\tau - \bar{\delta}_\tau(n) - \Delta\bar{d}_n\right] = V\left[\delta_n^\tau - \bar{\delta}_\tau(n)\right] + V\left[\Delta\bar{d}_n\right].$$

We have, by Eq. (14),

$$V\left[\Delta\bar{d}_n\right] = \frac{\sigma_d^2}{n}.$$

Also, since $\delta_n^\tau - \bar{\delta}_\tau(n) \le \rho\tau$,

$$V\left[\delta_n^\tau - \frac{1}{n}\sum \delta_i^\tau\right] \le \rho^2\tau^2.$$

Since $\rho\tau \ll \frac{\sigma_d}{\sqrt{n}}$, $\rho^2\tau^2$ can be neglected in comparison to $\frac{\sigma_d^2}{n}$. The result follows.

Q.E.D.

We show later that the condition $\rho\tau \ll \frac{\sigma_d}{\sqrt{n}}$ holds good whenever Eq. (6) holds good, and hence can be dropped from the statement of this and other theorems.

It should be noted that if only one message is sent (which we refer to as our baseline case), the average transmission error is again zero, but the variance remains approximately equal to the variance of the message delay σ_d^2. The variance is an important property because the probability that ϵ exceeds a given ϵ_{max} (the probability of invalidity) decreases with decreasing variance. An upper bound (not the least upper bound) on this probability is given by Tchebysheff's inequality,

$$\mathbf{P}\left[|\epsilon - \bar{\epsilon}| \ge \epsilon_{max}\right] \le \frac{V[\epsilon]}{\epsilon_{max}^2},$$

which by Property 1 and Lemma 3 reduces to

$$\mathbf{P}\left[|\epsilon| \ge \epsilon_{max}\right] \le \frac{\sigma_d^2}{n\epsilon_{max}^2}.$$

Thus by transmitting n messages, we have reduced this probability by *at least* a factor of n, compared to the baseline case. We have emphasized "at least" because, Tchebysheff's inequality holds generally for any probability distribution and the upper bound that it provides may be conservative for specific probability distributions. This statement is especially true of the Gaussian distribution which is of particular interest to us since the distribution of ϵ tends to becomes Gaussian for large n. This is proved in the following theorem.

Property 2 *If $\rho\tau \ll \frac{\sigma_d}{\sqrt{n}}$, the probability distribution of the transmission error (ϵ) approaches the Gaussian distribution $N\left(\bar{\epsilon}, \frac{\sigma_d^2}{n}\right)$, as the number of messages n increases.*

Proof: $\bar{d}(n)$ approaches $N(\bar{d}, \frac{\sigma_d^2}{n})$ as n becomes large, by the central limit theorem. Hence $\Delta\bar{d}(n) = \bar{d}(n) - \bar{d}$ approaches the Normal distribution $N(0, \frac{\sigma_d^2}{n})$. ϵ being the sum of $\Delta\bar{d}(n)$ and the independent random variables that constitute $\delta_n^\tau - \bar{\delta}_\tau(n)$, itself approaches the normal distribution $N(E[\epsilon], V[\epsilon])$ as n becomes large.
Q.E.D.

This result can be rewritten as follows. Let $G_n(\epsilon)$ denote the probability distribution function of ϵ, $\Phi(\epsilon)$ denote the Gaussian probability distribution function given by $\Phi(\epsilon) \triangleq \frac{1}{\sqrt{2\pi}\sigma} \int_{-\infty}^{\epsilon} e^{-\frac{t^2}{2\sigma^2}} dt$ and $\sigma \triangleq \frac{\sigma_d}{\sqrt{n}}$. Then for all $n > n_g$,

$$\left|\frac{G_n(\epsilon) - \Phi(\epsilon)}{\Phi(\epsilon)}\right| < \xi$$

where $\xi > 0$ can be made arbitrarily small by choosing n_g large enough. As mentioned in an earlier section, we refer to n_g, the minimum value of n for $N(\bar{\epsilon}, \sigma^2)$ to approximate $G_n(\epsilon)$ with an accuracy ξ as the *Gaussian cutoff*. If d itself has an approximately Gaussian distribution, then the Gaussian cutoff would be close to 1. If the distribution of d is not close to Gaussian, then n_g would be larger and can be determined by experiment. In a simulation study of the relationship between n_g and ξ, in which we assumed an uniform distribution for d and $\tau = 0$, we found that an $n_g > 5$ results in an error due to approximation of less than 5% within the first standard deviation, less than 3% between the first and second standard deviations and less than 0.5% beyond the second standard deviation. Thus an $n_g > 5$ (say $n_g = 10$, to be safe) would be sufficient (probably more than sufficient) to approximate the distribution of ϵ by a Gaussian distribution with a good accuracy, even when the distribution of d is uniform (i.e., not close to Gaussian). But we conjecture that the distribution of d would be approximately Gaussian and hence a smaller n_g would suffice. Our conjecture is based on the fact that the end-to-end message delay d itself is the sum of several independent random delays [5]:

1. *Send time*: the variable time required by the sender to assemble and send a synchronization message

2. *Access-time*: the variable access time of the sender to the communication medium, depending on the access strategy.

3. *Propagation delay*: the variable propagation delay of the message depending on distance and channel.

4. *Receive time*: the variable time required to switch contexts and schedule the receiving process to run at the receiving node and the time to process the message.

Hence by the central limit theorem the distribution of d itself is likely to be somewhat Gaussian.

We next prove a theorem that provides an analytical expression for computing the number of synchronization messages required.

Theorem 1 *The minimum number of messages required to guarantee a maximum deviation of ϵ_{max} with a probability of invalidity of p is given by $n_{min} = \max(n_g, n_e)$, provided $p\tau \ll \frac{\sigma_d}{\sqrt{n}}$. Here,*

$$n_e = \frac{2\sigma_d^2(\text{erfc}^{-1}(p))^2}{\epsilon_{max}^2} \tag{19}$$

and $\operatorname{erfc}^{-1}(p)$ *is the inverse of the complementary error function defined as* $\operatorname{erfc}(u) \triangleq 1 - \operatorname{erf}(u)$, *where*

$$\operatorname{erf}(u) \triangleq \frac{2}{\sqrt{\pi}} \int_0^u e^{-y^2}\, dy.$$

Proof: For a normally distributed random variable X, with distribution $N(\bar{X}, \sigma_X^2)$, the probability that X lies within ϵ_{max} of its mean μ_X is given by

$$\begin{aligned} \mathbf{P}\left[|X - \mu_X| \le \epsilon_{max}\right] &= \int_{-\epsilon_{max}+\bar{X}}^{\epsilon_{max}+\bar{X}} \frac{1}{\sqrt{2\pi}\sigma_X} e^{\frac{1}{2}\left(\frac{x-\bar{X}}{\sigma_X}\right)^2} dx \\ &= 2\int_{\bar{X}}^{\epsilon_{max}+\bar{X}} \frac{1}{\sqrt{2\pi}\sigma_X} e^{\frac{1}{2}\left(\frac{x-\bar{X}}{\sigma_X}\right)^2} dx \\ &= \frac{2}{\sqrt{\pi}} \int_0^{\frac{\epsilon_{max}}{\sqrt{2}\sigma_X}} e^{-y^2}\, dy \\ &= \operatorname{erf}\left(\frac{\epsilon_{max}}{\sqrt{2}\sigma_X}\right) \end{aligned} \tag{20}$$

The following approximation is valid by Property 1:

$$\mathbf{P}\left[|\epsilon - E[\epsilon]| \le \epsilon_{max}\right] = \mathbf{P}\left[-\epsilon_{max} + E[\epsilon] \le \epsilon \le \epsilon_{max} + E[\epsilon]\right] \approx \mathbf{P}\left[|\epsilon| \le \epsilon_{max}\right]$$

For $n > n_g$ we can approximate $G_n(\epsilon)$ by $N(\bar{\epsilon}, \frac{\sigma_d^2}{n})$. Hence by Eq. (20) and the above approximations, we have

$$\mathbf{P}\left[|\epsilon| \le \epsilon_{max}\right] = \operatorname{erf}\left(\frac{\sqrt{n}\epsilon_{max}}{\sqrt{2}\sigma_d}\right)$$

and therefore

$$\mathbf{P}\left[|\epsilon| > \epsilon_{max}\right] = \operatorname{erfc}\left(\frac{\sqrt{n}\epsilon_{max}}{\sqrt{2}\sigma_d}\right) \tag{21}$$

To guarantee a maximum deviation of ϵ_{max} with a probability of invalidity p, we must have $\mathbf{P}\left[|\epsilon| > \epsilon_{max}\right] \le p$, i.e.,

$$\operatorname{erfc}\left(\frac{\sqrt{n}\epsilon_{max}}{\sqrt{2}\sigma_d}\right) \le p$$

which when solved for n yields the n_c of Eq. (19). Since $n > n_g$ for the Gaussian approximation to be accurate, $n_{min} = \max(n_g, n_c)$.

Q.E.D.

Notice that if at least n_{min} messages are transmitted then the condition $\rho\tau \ll \frac{\sigma_d}{\sqrt{n}}$ can be rewritten as $\tau \ll \frac{1}{\sqrt{2}\text{erfc}^{-1}(p)} \frac{\epsilon_{max}}{\rho}$. This condition holds good whenever Eq. (6), holds good, if we define κ to be less than or equal to $\frac{1}{\sqrt{2}\text{erfc}^{-1}(p)}$. However, since we have made use of the assumption that $\kappa \leq 1$ in proving Property 1, we must make sure that the κ that we define satisfies the assumption $\kappa \leq 1$ also. Hence we define κ as follows:

$$\kappa = \min\left(1, \frac{1}{\sqrt{2}\text{erfc}^{-1}(p)}\right) \tag{22}$$

With κ thus defined, we may drop the condition $\rho\tau \ll \frac{\sigma_d}{\sqrt{n}}$ from Property 2 and Theorem 1.

We prove next that the probability of invalidity decreases exponentially or better with the number of messages. We first establish the following lemma that simplifies the proof.

Lemma 4 *The probability that the magnitude of the transmission error, ϵ, exceeds ϵ_{max} is bounded by the function $\sqrt{\frac{2}{\pi}} \frac{\sigma e^{\frac{-\epsilon_{max}^2}{2\sigma^2}}}{\epsilon_{max}}$, i.e.,*

$$\mathbf{P}\left[|\epsilon| > \epsilon_{max}\right] < \sqrt{\frac{2}{\pi}} \frac{\sigma e^{\frac{-\epsilon_{max}^2}{2\sigma^2}}}{\epsilon_{max}},$$

where $\sigma = \frac{\sigma_d}{\sqrt{n}}$.

Proof: By Property 2, ϵ is distributed $N(\bar{\epsilon}, \sigma)$. Therefore

$$\mathbf{P}\left[|\epsilon| > \epsilon_{max}\right] \approx \mathbf{P}\left[|\epsilon - \bar{\epsilon}| > \epsilon_{max}\right]$$

$$= \frac{2}{\sqrt{2\pi}\sigma}\int_{\epsilon_{max}}^{\infty} e^{-\frac{t^2}{2\sigma^2}}dt$$

$$= \sqrt{\frac{2}{\pi}}\int_{\frac{\epsilon_{max}}{\sigma}}^{\infty} e^{-\frac{y^2}{2}}dy \tag{23}$$

Let

$$Q(z) \triangleq \int_{z}^{\infty} e^{-\frac{y^2}{2}}dy \tag{24}$$

A bound can be derived [15] for $Q(z)$ as follows. For $z > 0$,

$$\begin{aligned} Q(z) &= \int_z^\infty \frac{1}{y}\left(ye^{-\frac{y^2}{2}}\right)dy \\ &= -\frac{1}{y}e^{-\frac{y^2}{2}}dy|_z^\infty - \int_z^\infty \frac{e^{-\frac{y^2}{2}}}{y^2}dy \\ &= \frac{1}{z}e^{-\frac{z^2}{2}} - \int_z^\infty \frac{e^{-\frac{y^2}{2}}}{y^2}dy \end{aligned}$$

The second term in the last equation is always positive. Hence

$$Q(z) < \frac{1}{z}e^{-\frac{z^2}{2}} \tag{25}$$

From equations (23), (24) and (25), it follows that

$$\mathbf{P}[|\epsilon| > \epsilon_{max}] = \sqrt{\frac{2}{\pi}}Q\left(\frac{\epsilon_{max}}{\sigma}\right) < \sqrt{\frac{2}{\pi}}\frac{\sigma e^{\frac{-\epsilon_{max}^2}{2\sigma^2}}}{\epsilon_{max}}$$

Q.E.D.

Property 3 *For a given ϵ_{max}, the probability of invalidity, p, decreases exponentially or better with the number of messages, i.e., there exists a bounding function for p of the form $\alpha e^{-n\beta}$ such that the magnitude of α does not increase with n and $\beta > 0$ is independent of n, i.e.,*

$$p = \mathbf{P}[|\epsilon| > \epsilon_{max}] < \alpha e^{-n\beta}$$

<u>Proof</u>: By Lemma 4,

$$\mathbf{P}\left[|\epsilon| > \epsilon_{max}\right] < \sqrt{\frac{2}{\pi}} \frac{\sigma_d e^{-\frac{n\epsilon_{max}^2}{2\sigma_d^2}}}{\sqrt{n}\epsilon_{max}} \simeq \alpha e^{-n\beta}$$

Q.E.D.

Note that the coefficient α is a decreasing function of n ($\propto \frac{1}{\sqrt{n}}$). Thus the coefficient is also responsible for a decrease, albeit a weak one, in the probability of invalidity with increasing n.

The next theorem is the basis of our claim that our algorithm is exempt from the bounds imposed by Eq. (2). TTP provides a way of determining the time on a remote clock with any desired accuracy in spite of variable message delays.

Theorem 2 *Our transmission protocol can guarantee any desired maximum transmission error, $\epsilon_{max} > 0$, with any desired probability of invalidity p ($0 < p \leq 1$), as long as the choice of values for ϵ_{max}, p and τ is consistent with Eq. (6) and Eq. (7).*

<u>Proof</u>: For a given ϵ_{max} and p, we can determine n_{min} as defined in Theorem 1. If n_{min} satisfies Eq. (7), then the desired maximum transmission error and probability of invalidity are guaranteed by Theorem 1, as long as at least n_{min} messages are transmitted within a period τ that does not violate Eq. (6).

Q.E.D.

Note that for a given ϵ_{max}, the smallest p that can be guaranteed is determined by Eq. (7). Solving Eq. (8) for p, we have

$$p = \text{erfc}\left(\frac{\epsilon_{max}}{\sigma_d}\sqrt{\frac{n_c}{2}}\right)$$

For a given ϵ_{max}, we can decrease p by increasing the number of messages transmitted. However since Eq. (7) should not be violated, p cannot be reduced indefinitely. Assuming a typical value of 10^{-6} for ρ, the largest number of messages that can be transmitted without violating Eq. (8) is of the order of $\frac{1}{10\rho} = 10^5$. If a ϵ_{max} of 1 millisec is desired, then the smallest probability of invalidity that can be achieved is of the order of erfc $(10^{2.5})$ which is extremely small due to the nature of the erfc function.

In a similar manner, for a given probability of invalidity (p), we can decrease the maximum transmission error, ϵ_{max} by increasing the number of messages. Again due to Eq. (7), there is a limit on the smallest ϵ_{max} that can be guaranteed. For typical values (that we have been assuming in our examples) of the other parameters, we can see from Eq. (8) that $\epsilon_{max} \gg 10^{-3}$ millisec.

We restate the following property which represents the most important characteristic of our clock synchronization algorithm, for completeness. We had stated and proved this property in Section 2.3

Property 4 *Our clock synchronization algorithm can guarantee any desired upper bound, γ_{max}, $\gamma_{max} > 0$, between any two clocks in the system, i.e.,*

$$\forall i, j, t \, |L_i(t) - L_j(t)| \leq \gamma_{max}$$

with any desired probability of invalidity, p ($0 < p \leq 1$), of the guarantee, as long as the choice of values for ϵ_{max}, p and τ is consistent with Eq. (6) and Eq. (7). Here i, j denote processes and $L_q(t)$ denotes the logical time on process q's clock at real time t.

It should be pointed out that the remarks that we made after Theorem 2 regarding the lower bound on ϵ_{max} apply to γ_{max} also; i.e., there is a lower bound on γ_{max} (for a given p) and a lower bound on p (for a given γ_{max}) imposed by Eq. (7). The bound on γ_{max} is still much lower than that imposed by Eq. (2) as the examples in Section 2.3 should convince.

4. Discussion

4.1 Transmission of Time

We had illustrated in an example in Section 2.3 that τ is of the order of several seconds for typical values of the other parameters. When the transmission period is so large, it is possible to separate successive message transmissions by a reasonable interval, if that is necessary to ensure the independence of the end-to-end message delays of successive messages. Also it is possible to relax the assumption that the mean and variance of the end-to-end message delay be unvarying. We can allow *occasional* transient communication traffic bursts that might result in temporarily establishing a new probability distribution with a different mean and variance, during the period that they last, provided the bursts do not last for too long. If a traffic burst is allowed to last for a maximum of W milliseconds after which the probability distribution returns to the steady state distribution, then by sending messages at intervals separated by a duration greater than W, one can ensure that most of the messages delays conform to their steady state probability distribution (provided the traffic bursts are not very frequent). It should be noted however, that separation of successive messages by an interval limits the number of messages that can be transmitted within the interval τ, and by a reasoning similar to that used in the remarks following Theorem 2 the lower bound on ϵ_{max} is increased as a consequence.

Maximum Wait Period

In our protocol, we have implicitly assumed that all the n messages sent by M are received by S when it estimates the time on M's clock. What if some of the messages are lost? What if a d_{max} does not exist? How long does S wait before it starts the estimation process?

We can schedule the transmission such that, it starts sufficiently ahead of the time at which resynchronization must be done, and such that the nth message can afford an end-to-end delay of upto T_l time units. The estimation process is started when n messages have been received or when the time for resynchronization has arrived, whichever is earlier. If the latter event occurs first, then there will be fewer than n messages to make the estimate and the accuracy and probability of validity correspondingly deteriorate. In order to ensure that this does not occur, T_l is chosen such that

$$\mathbf{P}\left[d_n > T_l\right] < \eta_1 \qquad\qquad (0 \leq \eta_1 \leq 1)$$

where η_1 is the desired probability that this undesirable event does not occur.

Alternatively, an extra n_x messages may be transmitted over and above the original n messages. The number n_x is chosen so that, the probability that more than n_x of the transmitted messages don't arrive within time, is smaller than a desired bound (η_2), i.e.,

$$\mathbf{P}\left[Ev(n_x)\right] < \eta_2 \qquad\qquad (0 \leq \eta_2 \leq 1)$$

where $Ev(n_x)$ denotes the event that n_x or more messages have missed the deadline.

4.2 Clock Synchronization

We have assumed in Eq. (10) that the distribution of the end-to-end message delay between the master process and every slave process is the same. It is possible to extend it to the case where the distributions are different for different slave processes, but we do not consider it further here.

We have assumed that processes can have performance or omission faults. Performace faults which cause a process to respond slowly are automatically handled as long as the resultant end-to-end message delays conform to the assumed probability distribution. Omission faults which arise when a process crashes do not affect the performance of the algorithm, as long as the process that crashed is not the master process; if it is a slave process that has crashed, then the clocks of running processes still remain mutually synchronized.

We have addressed the problem of handling master process failures in Appendix A, where we discuss two schemes to improve the fault-tolerance of the clock synchronization algorithm proposed in this paper. The two schemes are adaptations to suit our algorithm, of schemes proposed in [1] to handle master node failures. The first scheme is based on passive redundancy and can handle only one kind of fault, namely master node failure. The second scheme is based on active redundancy and can handle more general classes of faults. This scheme requires multiple master processes each of which is synchronized with a common external reference of time. The reader is referred to Appendix A for further details regarding these schemes.

5. Conclusion

We presented a new time transmission protocol that can achieve very small transmission errors, and described a probabilistic clock synchronization algorithm that incorporates this protocol. The clock synchronization algorithm presented, which is not subject to the bounds imposed by Eq. (2), was shown to perform much better than the algorithms that are constrained to obey Eq. (2). We also showed that the probability of invalidity of the guarantee on γ_{max} that the algorithm provides, drops exponentially (or better) with the number of synchronization messages transmitted.

There are a number of aspects of the algorithm that can be improved through further research. We have assumed steady mean and variance for the end-to-end message delays. While this may be a fairly realistic assumption for process control systems in steady state, other systems may be characterized by probability distributions that can change with time. A meta-level process that keeps track of the current probability distribution (specifically, the current $\bar{d}$ and σ_d since these are the only quantities of interest to us) would be necessary to handle dynamic systems characterized by changing probability distributions. Another area of research is in the improvement of the fault-tolerance of the algorithm by identifying new schemes to handle process and clock failures. A third area of research would be to identify how the time transmission protocol presented here can be used with the algorithms in [2], [4], [5], [6], [8], [9] and [10] to overcome the limitations imposed by Eq. (2). Finally, it would be interesting to identify other kinds of probabilistic algorithms and to determine the theoretical and practical bounds on the γ_{max} that can be guaranteed by probabilistic clock synchronization algorithms.

Acknowledgements

I would like to thank Prof. Krithi Ramamritham, Prof. Jack Stankovic, Dr. Tom Philips (IBM) and Dr. B.R. Badrinath for their comments, suggestions and encouragement.

APPENDICES

A. Fault Tolerance

The master/slave clock synchronization algorithm that we proposed in this paper is critically dependent on the proper functioning of the master[4] node. We describe schemes based on passive and active redundancy to provide fault-tolerance to our master/slave clock synchronization algorithm. These schemes are adaptations of schemes proposed by Cristian, to suit our algorithm.

A.1 Scheme I

The first scheme that we examine is based on passive redundancy, in the sense that though any of the nodes in the system can potentially serve as the master node, only one node actually does so at any instant of time. The other nodes are in effect back-up nodes that are called in to serve as master nodes when required. This scheme is similar to the *Ranked Master Group* scheme proposed in [1] except that none of the clocks are required to be synchronized with an external reference of time.

The scheme assumes that the nodes in the system are ordered into a *deputation order*. At any instant of time, the highest ranking functional node in the deputation order serves as the master node. When the node currently serving as the master node fails, the next highest ranking functional node takes over as the new master.

[4]We refer to a node that runs a master synchronization process as a master node and a node that runs a slave synchronization process as a slave node.

A.1.1 Analysis

The underlying master/slave clock synchronization algorithm guarantees that for any node N_i in the system,

$$|L_i - L_m| \leq \gamma_{ms}^{max}, \tag{26}$$

where L_p, for any p, denotes the local time at node N_p and N_m is the node that is currently serving as the master node. It is clear that as long as the current master node is functional, for any two nodes N_i and N_j,

$$\begin{aligned} |L_i - L_j| &= |(L_i - L_m) - (L_j - L_m)| \\ &\leq |L_i - L_m| + |L_j - L_m| \\ &= 2\gamma_{ms}^{max}. \end{aligned} \tag{27}$$

However when the node that is currently serving as the master node fails, there is a gap of length $T_{reconfig}$ between the instant of failure and the time at which all the nodes have got synchronized to the new master node. During this period, the clock of any node can drift by an additional amount equal to $\rho T_{reconfig}$. Hence the bound γ_{ms}^{max} in Eq. (26) will have to be replaced by the value $\gamma_{ms}^{max} + \rho T_{reconfig}^{max}$, where $T_{reconfig}^{max}$ is the assumed upper bound on $T_{reconfig}$.

The time $T_{reconfig}$ consists of the time it takes for the new master node to learn about the failure and the time it takes for it to get all the other nodes to synchronize with it. Suppose it takes a maximum of T_{nf} units of time for information about the failure of a node to reach all other nodes, and a time $T_{resynch}$ for the new master node to transmit its time to all the other nodes, then $T_{reconfig}^{max} = T_{nf} + T_{resynch}$.

A.1.2 Discussion

The scheme described so far handles only one kind of fault, namely *master node failure*. The scheme assumes that nodes are fail-safe, i.e., if they are faulty they do not affect the rest of the system. Thus there is the implicit assumption that the master node cannot transmit incorrect synchronization messages. Another assumption made is that the communication channel between the master and a slave does not corrupt or lose messages and that it conforms to the assumptions made about end-to-end message delays by the underlying clock synchronization algorithm. These assumptions are necessary for Eq. (26) to be valid. Thus this scheme is limited in the kinds of faults it can handle.

In this scheme, since the number of master nodes is always one, the costs incurred are the same as the costs incurred by the underlying master/slave clock synchronization algorithm. No additional price is paid for the fault-tolerance provided.

In the next section, we describe a scheme that is less limited in the kinds of faults it can tolerate.

A.2 Scheme II

The second scheme is based on active redundancy, in the sense that there are multiple nodes that concurrently function as master nodes. It is an adaptation of the *Active Master Set* architecture proposed in [1]. We have extended it to provide fault-tolerance at join time, i.e., when a slave node first joins the system.

The scheme requires that the master nodes be externally synchronized to an external reference of time such as the Universal Time Coordinated (UTC) time signals broadcast by the WWV radio station of the National Bureau of Standards. Thus the local clock L_m at each master node N_m satisfies the following condition:

$$|L_m - C_{ext}| \leq \gamma_{me}, \tag{28}$$

where C_{ext} is the external time reference used. It follows that, for any two *master* nodes N_p and N_q,

$$\begin{aligned} |L_p - L_q| &= |(L_p - C_{ext}) - (L_q - C_{ext})| \\ &\leq |L_p - C_{ext}| + |L_q - C_{ext}| \\ &= 2\gamma_{me}. \end{aligned} \tag{29}$$

In this scheme, it is possible that different slave nodes are synchronized to different master nodes at a given time. Hence for any two nodes N_i and N_j,

$$\begin{aligned} |L_i - L_j| &= |(L_i - L_p) - (L_j - L_q) + (L_p - L_q)| \\ &\leq |L_i - L_p| + |L_j - L_q| + |L_p - L_q| \\ &= 2(\gamma_{ms}^{max} + \gamma_{me}), \end{aligned} \tag{30}$$

where nodes N_p and N_q are the current masters of N_i and N_j respectively. The last step above follows from Eqs. (26) and (29).

In this scheme, at the end of each resynchronization interval each slave node N_s receives synchronization messages from *each* master node N_m. The node N_s then makes estimates C_m^{est} of the local clocks at each master node N_m on the basis of these synchronization messages (this estimation is part of the master/slave clock synchronization algorithm used by the nodes). N_s applies a *test* to each of these estimates to determine if the estimate is *acceptable* and uses the first acceptable estimate to resynchronize its local clock.

We next explain the *test of acceptability*. If everything is fine, i.e., if there is no faulty behaviour on the part of any component, then the estimate C_m^{est} of the clock at master node N_m made by slave node N_s satisfies the following condition:

$$\left|L_s - C_m^{est}\right| \leq \gamma_{ms}^{max} + 2\gamma_{me} + \epsilon_{max}. \tag{31}$$

Here ϵ_{max} is the maximum transmission error (the error in estimating the master clock) guaranteed by the underlying time transmission protocol. The term $2\gamma_{me}$ is present because N_s could have synchronized with a different master node during the previous resynchronization interval and two master nodes can be out of synchrony by as much as $2\gamma_{me}$ units. Eq. (31) is referred to as the test of acceptability.

If an estimate is not acceptable, i.e., it does not satisfy the test of acceptability, then it means that some of the assumptions made by the clock synchronization algorithm have been violated. The assumptions made by the clock synchronization algorithm could be violated, for instance, if

1. the clock at the master node has become faulty, or
2. the master node has transmitted incorrect synchronization messages, or
3. the channel connecting the master node and the slave node has lost or corrupted messages, or
4. the assumptions made about the end-to-end delays of messages does not hold good.

In any case, if an estimate violates Eq. (31) and if that estimate is used to adjust the local clock for resynchronization, then the functional nodes in the system can no longer be guaranteed to remain synchronized, i.e., Eq. (30) may not always be satisfied. Hence such an estimate is termed unacceptable and is not used to adjust the local clock.

Note that the test of acceptability cannot be applied when a slave node joins the system, i.e., when it is synchronized to a master for the first time. Prior to the first synchronization, the local clock at the slave node can have an arbitrary value and Eq. (31) may not be satisfied even if there are no faults in the system. The problem of masking faults when a slave joins the system is addressed in Section A.2.2.

A.2.1 Discussion

This scheme tolerates any combination of faults in the system as long as a slave node is able to obtain an acceptable estimate of the clock of at least one master node. However since there are multiple (say μ) master nodes, each resynchronization requires μ times as many synchronization messages as the basic master/slave algorithm that makes use of only a single master node. The scheme also incurs additional costs due to the special hardware receiver units required for external synchronization with the UTC signals.

A.2.2 Fault-Tolerance at Join Time

In this section, we consider the problem of tolerating faults that could cause incorrect estimates to be made at the point of time when a slave node first joins the system. At the time of joining, the estimate of the clock at a master node made by a slave node could be incorrect or unusable due to any of the factors listed earlier as possible causes for the violation of Eq. (31). However since the clock at the slave node can have an arbitrary value at join time, it is not possible to use Eq. (31) to determine whether an estimate made of the clock at a master node is correct or not. However if we make the assumption that a *majority of the estimates of master node clocks made by a slave node at join time will be correct* , then we can use the scheme described below to ensure that the clock at a joining node will be synchronized with the clocks at the other nodes in spite of faults in the system.

We define

$$f \triangleq \left\lfloor \frac{\mu - 1}{2} \right\rfloor ,$$

where μ is the number of master nodes. At join time, each slave node N_i receives synchronization messages from each of the master nodes and estimates the clock values at the master nodes. Each slave node then sets its clock L_i as follows.

$$L_i = \text{oneof}\left(\{\text{est}\,(i,k)\,, f+1 \leq k \leq \mu - f\}\right)$$

where the function oneof takes a set of real numbers as its argument and returns some randomly chosen element from this set as its value and the function est (i,k) returns the kth element in the ***ascending*** ordering of the set of estimates of master node clocks made by slave node N_i. If a slave node does not hear from some master node before it sets its clock, it uses an arbitrary value as the estimate of that master node clock. Note that L_i corresponds to the estimate (or one of the two estimates) that remains (remain) after discarding the largest f and smallest f estimates.

In the next subsection, we show that if a majority of the estimates obtained by a joining node are correct, then the node is synchronized to all the other nodes that have already joined the system.

A.2.3 Analysis

Let C_{ij} denote the estimate made by slave node N_i of the clock at master node N_j at join time. If there is no faulty behaviour on the part of any component, then the underlying time transmission protocol guarantees that

$$|C_{ij} - L_j| \le \epsilon_{max},$$

where ϵ_{max} is the maximum transmission error. Also, Eq. (28) holds good. Hence

$$\begin{aligned} |C_{ij} - C_{ext}| &= |(C_{ij} - L_j) + (L_j - C_{ext})| \\ &\le |C_{ij} - L_j| + |L_j - C_{ext}| \\ &= \epsilon_{max} + \gamma_{me}. \end{aligned} \tag{32}$$

In the context of join, we define an estimate C_{ij} to be correct if if it satisfies Eq. (32).

Lemma 5 *If a majority of the estimates in a joining slave node N_i are correct, then the estimates* est (i,k), $f+1 \le k \le \mu - f$, *are correct.*

Proof: Suppose some estimate est$(i,k), f+1 \leq k \leq \mu - f$ is not correct, then

$$|\text{est}(i,k) - C_{ext}| > \epsilon_{max} + \gamma_{me}.$$

But then either all of the estimates est$(i,p), 1 \leq p \leq k$ are incorrect or all of the estimates est$(i,q), k \leq q \leq \mu$ are incorrect, since these sets of estimates are respectively less than (or equal to) and greater than (or equal to) est(i,k). In either case the number of incorrect estimates must be at least $f+1$, since $|\{p : 1 \leq p \leq k\}| > f$ and $|\{q : k \leq q \leq \mu\}| > f$. This contradicts the fact that a majority of the estimates are correct.

Q.E.D.

Lemma 6 *The clock in a joining node is within $\gamma_{ms}^{max} + 2\gamma_{me}$ of external time until the first resynchronization.*

By Lemma 5, the estimate chosen to set the local clock at the slave node is correct and hence by Eq. (32) the local clock is within $\epsilon_{max} + \gamma_{me}$ of external time. The maximum drift, within the current resynchronization interval of the slave clock with respect to the master clock with which it is synchronized is given by $\gamma_{ms}^{max} - \epsilon_{max}$ by the definition of γ_{ms}^{max} (Eq. 9). Hence the maximum drift, during the current resynchronization interval, with respect to external time is equal to $\gamma_{ms}^{max} - \epsilon_{max} + \gamma_{me}$. Hence the local clock is within $(\epsilon_{max} + \gamma_{me}) + (\gamma_{ms}^{max} - \epsilon_{max} + \gamma_{me}) = \gamma_{ms}^{max} + 2\gamma_{me}$ of external time throughout the current resynchronization interval.

Q.E.D.

Lemma 7 *The clock at a node that has just joined will be within $2\gamma_{ms}^{max} + 3\gamma_{me}$ of the clock at any node that has gone through at least one resynchronization, and within $2(\gamma_{ms}^{max} + 2\gamma_{me})$ of the clock at any node that has joined but has not yet completed its first resynchronization interval.*

After the first resynchronization interval after join, a slave clock is within $\gamma_{ms}^{max} + \gamma_{me}$ of external time since the clock synchronization algorithm guarantees that the local clock will be within γ_{ms}^{max} of the master clock and Eq. (28) guarantees that the master clock will be within γ_{me} of external time. The result follows from this observation and Lemma 6.

Q.E.D.

Thus the maximum skew, between clocks in the system, that can be guaranteed has a slightly higher value before the instant when all the nodes have completed their first resynchronization intervals, than after.

A point to be noted is that the test of acceptability used at the first resynchronization point after join (i.e., at the end of the first resynchronization interval) will be different from Eq. (31) which is used at subsequent resynchronization points. This is because the maximum skew between the clocks at a slave node N_s and a master node N_m at the end of the first resynchronization interval can be greater than $\gamma_{ms}^{max} + 2\gamma_{me}$. The maximum skew is given by

$$\begin{aligned} |L_s - L_m| &= |(L_s - C_{ext}) - (L_m - C_{ext})| \\ &\leq |L_s - C_{ext}| + |L_m - C_{ext}| \\ &= \gamma_{ms}^{max} + 2\gamma_{me} + \gamma_{me} \\ &= \gamma_{ms}^{max} + 3\gamma_{me}. \end{aligned}$$

The third step above makes use of Lemma (6).

Hence the following condition has to be used instead of Eq. (32) at the first resynchronization point for testing acceptability.

$$\left|L_s - C_m^{est}\right| \leq \gamma_{ms}^{max} + 3\gamma_{me} + \epsilon_{max}.$$

B. Notation

$A \triangleq B$: A is defined as B

$A \simeq B$: A is of the form B

$A \sim B$: A is of the order of B

C_{ext}: external time reference

C_m^{est}: estimate of clock at master node N_m. d: end-to-end message delay

$\bar{d}$: expected value of d

d_{max}: upper bound on the end-to-end message delay

d_{min}: lower bound on the end-to-end message delay

d_i: end-to-end delay for the ith message

$\bar{d}(n)$: average of d computed using n samples

δ_0: deviation between the clocks of S and M at the start of transmission of message msg_1

δ_i: deviation between the clocks of S and M, when the ith message from M is received by S.

δ_i^T: increase in deviation between the clocks of S and M between the times of transmission of msg_1 and msg_i

δ_i^d: increase in deviation during the message delay d_i of msg_i

$\bar{\delta}(n)$: average deviation between the clocks of S and M over n message receipts

$\bar{\delta}_T(n)$: average increase in deviation prior to transmission

$\Delta\bar{d}(n)$: deviation of the average of d computed using n samples from the average computed using an infinite number of samples.

$\Delta\delta_n$: deviation of δ_n from $\bar{\delta}_n$

e: base of the natural logarithms

$\text{erf}(x)$: error function

$\text{erfc}(x)$: complementary error function

$\text{est}(i,k)$: defined in Section A.2.2

$E[X]$: expected value of random variable X

η_1, η_2: probability bounds

ϵ: transmission error

ϵ_{max}: desired maximum transmission error

G_n: probability distribution of the transmission error

γ_{max}: guaranteed maximum deviation between any two clocks in the system

γ_{me}: maximum deviation between master clock and the external reference of time

γ_{ms}: deviation between the master clock and a slave clock

γ_{ms}^{max}: maximum deviation between the master clock and any slave clock in the system

HC, HC_i: hardware clocks

κ: a parameter (dependent on p) that determines τ

L_i: logical clock of process i

M: sender

μ: number of master nodes

msg_i: ith synchronization message

n: number of synchronization messages

n_g: Gaussian cutoff

n_x: number of extra messages needed to compensate for delayed messages

N: number of clock synchronization processes in the system

$N(m, k^2)$: standard normal distribution with mean m and variance k^2

p: probability of invalidity

$\mathbf{P}$ [event]: probability of the event occuring

π: ratio of the circumference of a circle to its diameter

$\Phi(x)$: Gaussian probability distribution function corresponding to $N(0, \sigma^2)$

$Q(z)$: defined in Lemma 4

R_i: time of receipt of the ith message according to the receiver's clock

$\bar{R}(n)$: average of the message receipt times

R_{synch}: resynchronization interval

ρ: relative clock drift rate

ρ_a: absolute clock drift rate

$\hat{\rho}_i$: average relative clock drift rate during the end-to-end delay of message msg_i

S: receiver

σ: $\frac{\sigma_d}{\sqrt{n}}$

σ_d: standard deviation of the end-to-end message delay

t_1, t_2: time instants

T_i: time stamped on the ith message

T_l: latency available for the nth message

$\bar{T}(n)$: average of the times stamped on the n messages received

τ: duration of message transmission measured on the sender's clock

U: half time-out period in Cristian's algorithm

$V[X]$: variance of random variable X

W: maximum duration of a transient traffic burst

x, y, t: integration variables

$\bar{X}$: expected value of the random variable X

ξ: error bound

REFERENCES

[1] Cristian, F., *A Probabilistic Approach to Distributed Clock Synchronization*. Proc. of the Ninth International Conference on Distributed Computing Systems, May 1989. Also, Research Report RJ 6432 (62550), IBM Almaden Research Center, San Jose, California, September 1988.

[2] Cristian, F., Aghili, H., Strong, R., *Clock Synchronization in the Presence of Omission and Performance Faults, and Processor Joins*. Proc. of the International Conference on Fault-Tolerant Computing, 1987.

[3] Dolev, D., Halpern, J.Y., Strong, H.R., *On the Possibility and Impossibility of Achieving Clock Synchronization*. Journal of Computer and System Sciences, Vol. 32, No. 2, 1986, pp. 230-250.

[4] Halpern, J., Simons, B., Strong, R., *Fault-Tolerant Clock Synchronization*. Proc. of the Third Annual ACM Symposium on Principles of Distributed Computing, August 1984.

[5] Kopetz, H. and Ochsenreiter, W., *Clock Synchronization in Distributed Real-Time Systems*. IEEE Transactions on Computers, 36(8):933-40, August 1987.

[6] Kopetz, H., Damm, A., Koza, Ch., Mulazzani, M., Schwabl, W., Senft, Ch., Zainlinger, R., *Distributed Fault-Tolerant Real-Time Systems: The MARS Approach*. MARS Report Nr. 4/88, Institut für Technische Informatik, Technische Universität Wien, Austria, 1988.

[7] Lamport, L., *Time, Clocks and the Ordering of Events in a Distributed System*. Communications of the ACM, Vol. 21, No. 7, July 1978, pp. 558-565.

[8] Lamport, L., Melliar-Smith, P.M., *Byzantine Clock Synchronization*. Proc. of the Third Annual ACM Symposium on Principles of Distributed Computing, August 1984.

[9] Lamport, L., Melliar-Smith, P.M., *Synchronizing Clocks in the Presence of Faults*. Journal of the Association for Computing Machinery, Vol. 32, No. 1, January 1985, pp. 52-78.

[10] Lundelius, J., Lynch, N., *A New Faul-Tolerant Algorithm for Clock Synchronization*. Proc. of the Third Annual ACM Symposium on Principles of Distributed Computing, August 1984.

[11] Lundelius, L., Lynch, N., *An Upper and Lower Bound for Clock Synchronization*. Information and Control, Vol. 62, 1984, pp. 190-204.

[12] Stankovic, J.A., Ramamritham, K., *Design of the Spring Kernel*. COINS Technical Report 88-85, Department of Computer and Information Science, University of Massachusetts at Amherst, Amherst, Massachusetts, September 1988.

[13] Walter, C.J., Kieckhafer, R.M., Finn, A.M., *MAFT: A Multicomputer Architecture for Fault-Tolerance in Real-Time Control Systems*. Proc. of the Real-Time Systems Symposium, December 1985, pp. 133-140.

[14] J.W. Wensley, L. Lamport, J. Goldberg, M. Green, K.N. Levitt, P.M. Melliar-Smith, R.E. Shostak, C.B. Weinstock, *SIFT: Design and Analysis of a Fault-Tolerant Computer for Aircraft Control*. Proc. of the IEEE 66(11), October 1978, pp. 1240-1255.

[15] Wozencraft, J.M., Jacobs, I.M., *Principles of Communication Engineering*. John Wiley, 1987.