Maintaining the Time in a Distributed System

Keith Marzullo
Stanford University Computer Systems Laboratory
Xerox Systems Development Department

Susan Owicki
Stanford University Computer Systems Laboratory

1 Introduction

To a client of a loosely-coupled distributed system, one of the simplest services is a time service. Usually the client simply requests the time from any subset of the time servers making up the service, and uses the first reply. Issues that need to be considered in other services, such as connection establishment or client authentication, need not be considered in a time service. The simplicity of this interaction, however, misrepresents the complexity of implementing such a service.

1.1 Time Services and Clock Synchronization

A time service algorithm is essentially an algorithm that keeps a collection of clocks locally monotonic, synchronized, and adequately accurate with respect to some time standard (such as Greenwich Mean Time). The relative importance of these properties depends on for what purpose the service is used. For example, if the service is used for measurement of time in distributed experiments, absolute accuracy and synchronization are not important issues as long as the relative accuracies and offsets of the clocks are known [Mills 81]. Accuracy is not important if the service is only used to order events occuring within the distributed system. In a system where events both internal and external to the distributed system are ordered, requirements for synchronization and accuracy depend on the behavior of the system, such as the rate at which objects can migrate from one clock's domain to another.

The type of distributed system in which we are interested can be represented by the system in which we have performed our experimentation, the Xerox Research Internet [Boggs 80]. This distributed system contains thousands of personal workstations spread across the United States, Canada and Europe. Located throughout the system are hundreds of "public" processors running various system services such as routing, filing, authentication, printing, naming and electronic mail. Since a time server does not require much in the way of special hardware, most of these servers also act as time servers. Of course, there is no reason to limit time servers to these public machines. Any user who requires it should be able to convert her workstation into a time server. Objects do not move rapidly in this system, so an adequate precision is in the order of tens of seconds.

This type of system imposes certain prerequisites on the time service. The set of servers making up the service is not stable, in that time servers can frequently join or leave the service. The clocks of the servers need not be uniformly accurate. A clock may fail in many ways, such as by stopping, racing ahead, or refusing to change its value when reset. In this paper, we will assume that clocks may have varying accuracies, but are usually stable. Failing clocks will not be dealt with. The work in this paper is extended to deal with failing clocks in [Marzullo 83].

We will consider the problem of keeping a set of clocks synchronized and correct. We will not require clocks to be locally monotonic, so that clocks may be freely set backward as well as forward. A client, however, may require that the local clock is monotonic. Such a clock may be implemented based on a nonmonotonic clock by temporarily running the monotonic clock more slowly when the nonmonotonic clock is set backwards.

1.2 Synchronization Functions

There has been much interest in synchronization in distributed systems, but not much research into time services. Leslie Lamport has developed algorithms which keep clocks synchronized [Lamport 78, 82], but maintain precision by assuming accurate clocks. Other systems which keep accurate clocks synchronized to an external standard under extreme conditions have been developed [Ellingston 73], but these assume a different system configuration and error model then we do.

The problem of a set of processes keeping their clocks synchronized can be simply characterized as each

Reprinted with permission from the *Proceedings of the Second ACM Symposium on Principles of Distributed Computing.*

process i independently performing a calculation across a distributed set of data:

$$C_i(t) \leftarrow F(C_{i_1}(t), C_{i_2}(t), \ldots C_{i_k}(t))$$

This breaks the problem into two problems: specifying an appropriate way to collect the distributed data, and specifying a synchronization function F. In this paper, we will assume that the distributed data is collected by some simple method such as directed broadcasting [Boggs 82], and will concentrate on the specification of a synchronization function.

In the terms of this characterization, several synchronization functions have been specified by others. A simple function that preserves monotonicity is the maximum value of the clocks [Lamport 78]. The median clock value and the mean value of the clocks have also been used to specify very fault-tolerant clock synchronization algorithms [Lamport 82]. Our work differs from these in our assumptions on the properties of the clocks and the use of the time service. We will be concerned with the amount of synchronization a clock achieves with a time standard as well as the other clocks in the time service.

2 Clocks and Maximum Error

2.1 *Properties of Clocks*

A clock $C_i(t)$ may be characterized as a function that maps real time to clock time. It is continuous between the times it is reset. A *perfect clock* (or a *standard*) is one in which $C(t) = t$ for an appropriate definition of t. A clock is *correct* at time t_0 if its value $C_i(t_0)$ is t_0. A clock is *accurate* at time t_0 if its first derivative is one second per second. A clock is *stable* at time t_0 if its second derivative is zero. These properties are all defined in terms of a perfect clock, in that a perfect clock is correct, accurate and stable.

It is physically impossible to construct a time service that can keep a collection of clocks correct with some standard unless there is communication between the system and the standard. A metric for comparing time service algorithms is the rate at which the error grows in the service. In a system where the relative accuracies of the clocks are known, an algorithm should be able to keep the service as accurate as its most accurate clock. The only guarantee any algorithm can make is that the servers will maintain a mutually consistent time.

2.2 *Maximum Error*

In an environment where some clocks are significantly more accurate than others, it is convenient in the construction and analysis of a synchronization function to augment the clock values with some indication of their accuracy. Since a primary concern is maintaining the correctness of the clocks in the service, a convenient error measure is an upper bound on the error in the clock. A server that responds with a time 3:01 and a maximum error of 0:02 asserts that if all of the information it possesses is correct, the correct time must lie in the interval [2:59 .. 3:03]. Conversely, if a server responds at time t to a time request with the pair $<C_i(t), E_i(t)>$, the server's clock is *correct* if the correct time lies in the interval $[C_i(t) - E_i(t) \,..\, C_i(t) + E_i(t)]$.

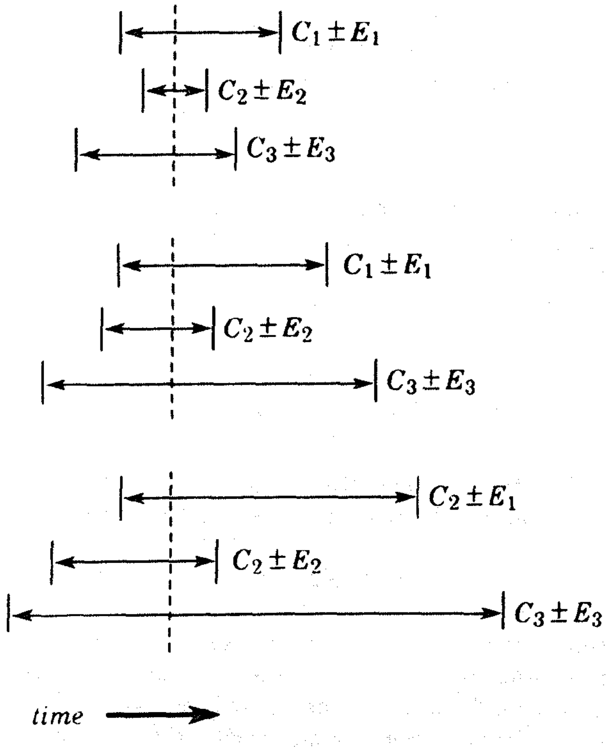

Figure 1
Growth of Maximum Errors

This formulation can be viewed as having servers respond with intervals of time rather than points. Figure 1 shows these intervals for three time servers $\{S_1, S_2, S_3\}$ at three different times. The horizontal direction is the real time axis with the correct time indicated by a dashed line. The *leading edge* of an interval is the edge $C_i + E_i$, and the *trailing edge* is the edge $C_i - E_i$. The clocks of the three servers are all correct. As the system runs (the vertical axis), the individual intervals both grow and shift with respect to the correct time.

A time server must be able to calculate a bound on its maximum error. In the case of continuous clocks, there are three sources of error:

- The error the clock inherits when it is reset
- The delay between when a clock is read and when another clock is set to this value.
- The deterioration in the time maintained by a clock between resets.

Since the time server is only calculating an upper bound on the error, the unknown component in the delay may be safely estimated by measuring the time between sending a time request and receiving the reply. In this paper, this delay is assumed to be nondeterministic and bounded by ξ. The symbol σ_j will be used to denote the amount of time from when server S_i sends a time request to server S_j and when S_j receives the request, and ρ_j to denote the amount of time from when server S_j sends the reply to server S_j and when S_i receives this reply. The minimum message delay will be assumed to be zero. The two algorithms presented here can easily be extended to take into account nonzero minimum message delay times.

To estimate the deterioration, the server must have an upper bound on the inaccuracy of the clock. will be acclompished by assuming that the clock has a known *maximum drift rate* δ, such that

$$\left| 1 - \frac{dC_i(t)}{dt} \right| \leq \delta_i$$

If $C_i(t)$ is continuous over the range $t_0 \leq t \leq t_0 + \Delta$, the above relation may be rearranged and integrated:

$$C_i(t_0) + \Delta - \delta_i\Delta \leq C_i(t_0 + \Delta) \leq C_i(t_0) + \Delta + \delta_i\Delta$$

The deterioration of a server's clock can be estimated by augmenting its error by $s\delta_i$ if its clock has not been reset for s seconds. In this paper, δ_i will be assumed to be small enough to ignore terms of order δ_i^2

2.3 *Consistency of a Time Service*

Having a clock report its maximum error also allows a precise definition of the *consistency* of a time service. If one server responds with a time of 3:01 and an error of 0:02, and another server responds with a time of 3:06 and an error of 0:02, at least one of them must be wrong, since the correct time cannot lie in both of these intervals. Correspondingly, a time service is *consistent* if the intersection of the intervals defined by all of the servers is non-empty. In terms of the values maintained by a time server, two servers are consistant at time t_0 if

$$|C_i(t_0) - C_j(t_0)| \leq E_i(t_0) + E_j(t_0)$$

Consistency is a weaker condition than correctness. As will be proven later, the algorithms in this paper maintain correctness under the assumption that each clock has a known valid upper bound on its drift rate. In an actual service, these valid bounds may not be known for all of the clocks, which can lead to an incorrect service. Since there is no perfect clock in the system, an algorithm implementing the service can not check to see if the clocks are remaining correct; however, it can check to see if the clocks are remaining consistent.

3 Minimization of the Maximum Error as a Synchronization Function

Given that a time server reports its maximum error, a client of the time service could collect a set of times and use the response with the smallest error rather than the first reply it receives. This suggests a synchronization function that chooses the time with the smallest maximum error. Define a graph in which time servers are nodes and communication paths are edges. We assume this graph is connected and define an algorithm in which each server periodically synchronizes with the neighbor that has the smallest maximum error. This can be expressed in two rules:

MM-1: A time server S_i maintains a clock C_i, the time when last reset r_i, and an inherited error ε_i. If S_i receives a time request at time t, it responds with the pair $[C_i(t), E_i(t)]$ where

$$E_i(t) = \varepsilon_i + (C_i(t) - r_i)\delta_i$$

and $C_i(t)$ is the value of the clock C_i at time t.

MM-2: Each time server sends a time request to its neighbors at least once every τ seconds. Let t be the time at which S_i sends a request to S_j and $\xi^i{}_j$ be the time as measured by C_i between S_i sending the request and receiving the response. Any reply that is inconsistent with S_i is ignored. When a consistent reply is received from S_j, the following predicate is evaluated:

$$E_j + (1+\delta_i)\xi^i_j \leq E_i$$

If this predicate is true, S_i sets $\varepsilon_i \leftarrow E_j + (1+\delta_i)\xi^i{}_j$, $C_i \leftarrow C_j$ and $r_i \leftarrow C_j$.

The first rule describes how a time server behaves when asked the time, and the second rule how it synchronizes with its neighbors. We require this algorithm to preserve correctness (and therefore, consistency). That it does so is proven by the following theorem:

Theorem 1 If all of the δ_i are valid upper bounds on the drift rates of the clocks C_i, then an initially correct time service running algorithm *MM* will remain correct.

Proof: This proof will require the following property of the error of a server that has not been reset:

Lemma 1 If time server S_i is not reset over the interval $t_0 \leq t \leq t_0 + \Delta$, then $E_i(t_0+\Delta) = E_i(t_0) + \delta_i\Delta$

Proof: Since S_i has not been reset, the values of ε_i and r_i are the same at t_0 and $t_0 + \Delta$. From rule *MM*-1,

$$E_i(t_0+\Delta) - E_i(\mathrm{t_0}) = \delta_i(C\mathrm{i}(\mathrm{t_0}+\Delta) - C_i(\mathrm{t_0}))$$

The lemma follows from the definition of δ and dropping the term $\delta_i{}^2\Delta$. □

Theorem 1 will be proven by showing that no individual server can become incorrect. First, rule *MM*-1 will be shown to preserve correctness. Assume that the service is correct at time t_0, and that server S_i does not reset over the interval $t_0 \leq t \leq t_0 + \Delta$:

$$C_i(t_0+\Delta) - E_i(t_0+\Delta) \leq C_i(t_0) - E_i(t_0) + \Delta \quad [\textit{definition of } \delta \textit{ and lemma 1}]$$

$$C_i(t_0) - E_i(t_0) + \Delta \leq t_0 + \Delta \quad [\textit{hypothesis and definition of } \textbf{correct}]$$

$$C_i(t_0+\Delta) + E_i(t_0+\Delta) \geq C_i(t_0) + E_i(t_0) + \Delta \quad [\textit{definition of } \delta \textit{ and lemma 1}]$$

$$C_i(t_0) + E_i(t_0) + \Delta \geq t_0 + \Delta \quad [\textit{hypothesis and definition of } \textbf{correct}]$$

Combining the above inequalities gives

$$C_i(t_0+\Delta) - E_i(t_0+\Delta) \leq t_0 + \Delta \leq C_i(t_0+\Delta) + E_i(t_0+\Delta)$$

We also need to show that rule *MM*-2 also preserves correctness. Assume that S_i sends a time request to S_j at time t_0, and at time $t_0 + \sigma_j$ time server S_j responds with the correct interval $<C_j(t_0+\sigma_j), E_j(t_0+\sigma_j)>$. Let the delay in receiving the reply be ξ_j as measured by a perfect clock and $\xi^i{}_j$ as measured by C_i. If S_i is reset using this value, then[†]:

$$E_i(t_0+\xi_j+0) = E_j(t_0+\sigma_j) + (1+\delta_i)\xi^i{}_j \quad [\textit{rule MM-2}]$$

$$(1-\delta_i)\xi_j \leq \xi^i{}_j \leq (1+\delta_i)\xi_j \quad [\textit{definition of } \delta] \; (1)$$

Combining these two inequalities gives (after dropping a $\xi_j\delta_i{}^2$ term)

$$E_j(t_0+\sigma_j) + \xi_j \leq E_i(t_0+\xi_j+0) \leq E_j(t_0+\sigma_j) + (1+2\delta_i)\xi_j \quad (2)$$

$$C_i(t_0+\xi_j+0) = C_j(t_0+\sigma_j) \quad [\textit{rule MM-2}] \; (3)$$

$$C_i(t_0+\xi_j+0) - E_i(t_0+\xi_j+0) \leq C_j(t_0+\sigma_j) - E_j(t_0+\sigma_j) - \xi_j \quad [\textit{equations 2 and 3}]$$

$$C_j(t_0+\sigma_j) - E_j(t_0+\sigma_j) \leq t_0 + \sigma_j \quad [\textit{hypothesis and definition of } \textbf{correct}]$$

Since ξ_j is positive and $0 \leq \sigma_j \leq \xi_j$, the last two equations give the condition that the lower bound of the resulting interval of S_i is correct:

$$C_i(t_0+\xi_j+0) - E_i(t_0+\xi_j+0) \leq t_0 + \xi_j$$

$$C_i(t_0+\xi_j+0) + E_i(t_0+\xi_j+0) \geq C_j(t_0+\sigma_j) + E_j(t_0+\sigma_j) + \xi_j \quad [\textit{equations 2 and 3}]$$

$$C_j(t_0+\sigma_j) + E_j(t_0+\sigma_j) + \xi_j \geq t_0 + \sigma_j + \xi_j \quad [\textit{hypothesis and definition of } \textbf{correct}]$$

Since σ_j is positive, the last two equations give the condition that the upper bound of the resulting interval of S_i is correct:

$$C_i(t_0+\xi_j+0) + E_i(t_0+\xi_j+0) \geq t_0 + \xi_j$$

□

The maximum error of any clock in a fully-connected service running algorithm *MM* will be equal to the smallest error in the system plus any error accumulated during or after the last reset. A more surprising property is that the resulting synchronization of the clocks is not very good. This is

[†] In this paper, $\lim\limits_{a\to\pm 0} F(x+a)$ is written as $F(x\pm 0)$

because the algorithm cannot guarantee that different servers will pick the same server as having the smallest error. They remain relatively well synchronized only because the clocks which possess the smallest errors are consistent. These claims are proven as theorems 2 and 3 below:

Theorem 2 If all of the δ_i are accurate upper bounds on the absolute value of the drift rates of the clocks C_i, then in a fully-connected time service S running algorithm MM in which $(\forall S_i \in S)[0 \le \delta_i < 1]$, the error in any server will be bounded by

$$E_i(t) < E_M(t) + \xi + \delta_i(\tau + 2\xi)$$

where $E_M(t)$ is the smallest error in the service at time t.

Proof: For this proof we will assume that a time server answers its own request with zero delay. Since this self-reply satisfies the predicate in rule MM 2, there will always be at least one reply that satisfies MM-2. This reply will not change the value of C_i or E_i, so the behavior of a service under this assumption should not be different than one in which there are no self-replies.

Lemma 2 $E_i(t_0 + \Delta) \le E_i(t_0) + \delta_i \Delta$

Proof: This will be proven using induction on the number of times C_i has reset during the interval $t_0 \le t \le t_0 + \Delta$. If the clock has not reset, then lemma 2 reduces to lemma 1. Let ${t_0}^k$ be the k^{th} time S_i reset C_i since t_0. By hypothesis, $E_i(t_0 + \Delta_k) \le E_i(t_0) + \delta_i \Delta_k$ for the interval $0 \le \Delta_k < {t_0}^{k+1} - t_0$.

$$E_i({t_0}^{k+1} - 0) \le E_i(t_0) + \delta_i({t_0}^{k+1} - t_0) \quad [\textit{hypothesis}]$$
$$E_i({t_0}^{k+1} + 0) \le E_i({t_0}^{k+1} - 0) \quad [\textit{Rule MM-2}]$$
$$E_i(t_0 + \Delta_{k+1}) \le E_i({t_0}^{k+1} + 0) + \delta_i((\Delta_{k+1} + t_0) - {t_0}^{k+1}) \quad [\textit{lemma 1}]$$

Combining these intervals gives:

$$E_i(t_0 + \Delta_{k+1}) \le E_i(t_0) + \delta_i \Delta_{k+1}$$

for the interval $t_0 \le \Delta_k < {t_0}^{k+2} - t_0$. □

Lemma 3 The minimum error $E_M(t)$ in a time service S in which $(\forall S_i \in S)[0 \le \delta_i < 1]$ will never decrease.

Proof: By lemma 1, as long as no time server is reset, no error in the service will decrease, so E_M can not decrease. Let S_i be the first server to reset. Assume that at the end of its reset it has the smallest error, and

$$E_M(t_0 + \xi_j + 0) = E_i(t_0 + \xi_j + 0) < E_M(t_0 + \xi_j - 0)$$

$$E_i(t_0 + \xi_j + 0) = E_j(t_0 + \sigma_j) + (1 + \delta_i)\xi^{i}_{j} \quad [\textit{Rule MM-2}]$$
$$= E_j(t_0 + \xi_j - 0) - \delta_j \rho_j + (1 + \delta_i)\xi^{i}_{j} \quad [\textit{Lemma 1 and definition of } \rho]$$
$$\ge E_M(t_0 + \xi_j - 0) - \delta_j \rho_j + \xi_j \quad [\textit{equation 1 and definition of } E_M]$$

Thus, for $E_M(t_0 + \xi_j + 0) < E_M(t_0 + \xi_j - 0)$, $\xi_j - \delta_j \rho_j < 0$. Since $\xi_j = \sigma_j + \rho_j$ and both σ_j and ρ_j are positive, this inequality is true only if $\delta_i > 1$. We have our contradiction. □

Lemma 4 If a time service is consistent, then $(\forall S_i, S_j \in S)[|C_i(t) - C_j(t)| \le E_i(t) + E_j(t)]$.

Proof: This follows directly from the definition of **consistent**. □

Assume that at time t, time server S_i broadcasts a time request to the n time servers making up the time service. Define t_r to be the time S_i receives the n^{th} response. Let the set of indices k order the replies:

$$\xi_{k_1} \le \xi_{k_2} \le \ldots \le \xi_{k_{n-1}} \le \xi_{k_n}$$

S_i will examine the replies until it finds one which satisfies MM-2. By assumption, there will be at least one such reply. Call this server from which this first accepted reply originated S_{k_x}.

$$(0 < j < x)[E_{k_j}(t + \sigma_{k_j}) + (1 + \delta_i)\xi^{i}_{k_j} > E_i(t + \xi_{k_j})] \quad [\textit{rule MM-2}]$$
$$E_{k_j}(t + \sigma_{k_j}) + (1 + 2\delta_i)\xi_{k_j} > E_{k_j}(t + \sigma_{k_j}) + (1 + \delta_i)\xi^{i}_{k_j} \quad [\textit{equation 2}]$$
$$(0 < j < x)[E_i(t + \xi_{k_j}) = E_i(t + \xi_{k_x}) - (\xi_{k_x} - \xi_{k_j})\delta_i] \quad [\textit{lemma 1}]$$

Combining the above three equations gives

$$(0 < j < x)[E_{k_j}(t + \sigma_{k_j}) + (1 + 2\delta_i)\xi_{k_j} > E_i(t + \xi_{k_x}) - (\xi_{k_x} - \xi_{k_j})\delta_i] \tag{4}$$

$$E_{k_x}(t + \sigma_{k_x}) + (1 + \delta_i)\xi^{i}_{k_x} \le E_i(t + \xi_{k_x}) \quad [\textit{Rule MM-2}]$$
$$E_{k_x}(t + \sigma_{k_x}) + \xi_{k_x} \le E_{k_x}(t + \sigma_{k_x}) + (1 + \delta_i)\xi^{i}_{k_x} \quad [\textit{equation 1}]$$

Combining the above two equations gives

$$E_{k_x}(t + \sigma_{k_x}) + \xi_{k_x} \le E_i(t + \xi_{k_x}) \tag{5}$$

Combining equations 4 and 5 gives

$$(0 < j < x)[E_{k_x}(t + \sigma_{k_x}) < E_{k_j}(t + \sigma_{k_j}) - (1 + \delta_i)(\xi_{k_x} - \xi_{k_j})] \tag{6}$$

S_i will continue to examine the replies. Either at least one more reply will be found that satisfies rule *MM*-2, or no such reply will be found. Assume there is a reply from S_{k_y} which satisfies *MM*-2, with $k_y > k_x$. Using the same argument that led to equation 6, we get

$$\begin{aligned}
&(x<j<y)[E_{k_y}(t+\sigma_{k_y}) \\
&\quad < E_{k_j}(t+\sigma_{k_j}) - (1+\delta_i)(\xi_{k_y} - \xi_{k_j})] && (7) \\
&E_{k_y}(t+\sigma_{k_y}) \leq E_i(t+\xi_{k_y}) - (1+\delta_i)\xi^{i}_{k_y} && [\textit{rule MM-2}] \\
&\quad = E_i(t+\xi_{k_x}+0) + (\xi_{k_y} - \xi_{k_x})\delta_i - (1+\delta_i)\xi^{i}_{k_y} && [\textit{lemma 1}] \\
&\quad = E_{k_x}(t+\sigma_{k_x}) + \\
&\qquad (1+\delta_i)\xi^{i}_{k_x} + (\xi_{k_y} - \xi_{k_x})\delta_i - (1+\delta_i)\xi^{i}_{k_y} && [\textit{rule MM-2}] \\
&\quad \leq E_{k_x}(t+\sigma_{k_x}) + (1+2\delta_i)\xi_{k_x} + (\xi_{k_y} - \xi_{k_x})\delta_i - \xi_{k_y} && [\textit{equation 1 twice}]
\end{aligned}$$

By algebraically rearranging the last relation, we get

$$E_{k_y}(t+\sigma_{k_y}) \leq E_{k_x}(t+\sigma_{k_x}) - (1+\delta_i)(\xi_{k_y} - \xi_{k_x}) + 2\delta_i\xi_{k_y}$$

Since δ_i and ξ_{k_y} are positive, this relation along with equations 6 and 7 gives

$$\begin{aligned}
&(0<j<y)[E_{k_y}(t+\sigma_{k_y}) \\
&\quad < E_{k_j}(t+\sigma_{k_j}) - (1+\delta_i)(\xi_{k_y} - \xi_{k_j})] && (8)
\end{aligned}$$

Equation 8 is equation 6 with y substituted for x. Therefore, for each server from which S_i resets its clock, there will be a corresponding equation of the form of equation 8. In particular, there will be such an equation for the last time server from which S_i resets. Call this server S_{k_m}. The relationship is

$$\begin{aligned}
&(0<j<m)[E_{k_m}(t+\sigma_{k_m}) \\
&\quad < E_{k_j}(t+\sigma_{k_j}) - (1+\delta_i)(\xi_{k_m} - \xi_{k_j})] && (9)
\end{aligned}$$

By definition, all of the replies after the one from S_{k_m} do not cause a reset:

$$\begin{aligned}
&(\mathrm{m}<j\leq n)[E_{k_j}(t+\sigma_{k_j}) > E_i(t+\xi_{k_j}) - (1+\delta_i)\xi^{i}_{k_j}] && [\textit{rule MM-2}] \\
&\quad = E_i(t+\xi_{k_m}+0) + (\xi_{k_j} - \xi_{k_m})\delta_i - (1+\delta_i)\xi^{i}_{k_j} && [\textit{lemma 1}] \\
&\quad > E_i(t+\xi_{k_m}+0) + (\xi_{k_j} - \xi_{k_m})\delta_i - \xi_{k_j} && [\textit{equation 1}]
\end{aligned}$$

By algebraically rearranging the last relation, we get

$$\begin{aligned}
&(\mathrm{m}<j\leq n)[E_i(t+\xi_{k_m}+0) \\
&\quad < E_{k_j}(t+\sigma_{k_j}) + \xi_{k_j} - (\xi_{k_j} - \xi_{k_m}\delta_i]
\end{aligned}$$

$$\begin{aligned}
&(\mathrm{m}<j\leq n)[E_i(t+\xi_{k_m}+0) < E_{k_j}(t+\sigma_{k_j}) + \xi] \\
&\qquad [\textit{Rule MM-2 and definition of } \xi] \; (10)
\end{aligned}$$

$$\begin{aligned}
&E_i(t+\xi_{k_m}+0) - (1+\delta_i)\xi^{i}_{k_m} = E_{k_m}(t+\sigma_{k_m}) && [\textit{Rule MM-2}] \\
&E_i(t+\xi_{k_m}+0) - (1+2\delta_i)\xi_{k_m} \leq E_{k_m}(t+\sigma_{k_m}) && [\textit{equation 1}]
\end{aligned}$$

By combining equation 9 and the last equation, we get

$$\begin{aligned}
&(0<j<m)[E_i(t+\xi_{k_m}+0) \\
&\quad < E_{k_j}(t+\sigma_{k_j}) + (1+\delta_i)\xi_{k_j} + \delta_i\xi_{k_m}] \\
&\quad < E_{k_j}(t+\sigma_{k_j}) + (1+2\delta_i)\xi && [\textit{definition of } \xi]
\end{aligned}$$

By combining equation 10 and the last equation, we get

$$\begin{aligned}
&(\forall j)[E_i(t+\xi_{k_m}+0) < E_j(t+\sigma_j) + (1+2\delta_i)\xi] \\
&(\forall j)[E_i(t_r) < E_j(t+\sigma_j) + (1+2\delta_i)\xi - (t_r - t - \xi_{k_m})\delta_i] && [\textit{lemma 1}] \; (11) \\
&E_M(t+\sigma_M) \leq E_M(t_r) && [\textit{lemma 3 and definition of } t_r] \\
&E_i(t_r) < E_M(t_r) + (1+2\delta_i)\xi - (t_r - t - \xi_{k_m})\delta_i && [\textit{above and equation 11 with } j=M] \\
&E_i(t_r) < E_M(t_r) + (1+2\delta_i)\xi && [\delta_i > 0] \\
&E_i(t' \geq t_r) < E_M(t) + (1+2\delta_i)\xi + \delta_i\tau && [\textit{lemma 1 and definition of } \tau]
\end{aligned}$$

□

Theorem 3 If all of the δ_i are accurate upper bounds on the absolute value of the drift rates of the clocks C_i, then in a fully-connected time service running algorithm *MM*, the asynchronism will be bounded by

$$|C_i(t) - C_j(t)| < 2E_M(t) + 2\xi + (\delta_i + \delta_j)(\tau + 2\xi)$$

Proof: The theorem follows from theorem 2 and lemma 4. □

Theorems 2 and 3 are expressed in terms of the most precise clock; that is, the clock with the smallest error in the service at some time t. There is no *a priori* reason that the most accurate clock will also be the most precise. A time service in any initial state with bounded errors, however, will eventually reach the state where the most accurate clock is also the most precise. This means that eventually the time service will derive its behavior from the most accurate clocks in the service.

Theorem 4 Let $\delta_{min} \equiv$ **minimum** $(\forall S_i \in \mathbf{S})[\delta_i]$ and $\mathbf{S}_{min}$ be the subset of a finite time service $\mathbf{S}$ with $\delta = \delta_{min}$ Let $\mathbf{S}_{min}$ be the set of servers in $\mathbf{S}$ with the smallest error. If no time server resets to a clock with an error worse than its own, then there exists a time t_x for which if $t \geq t_x$, $S_M(t_x) \in \mathbf{S}_{min}$.

Proof: Consider a time service at time t_0. First assume that no clock in the system is to reset after t_0. Then, the error of each time server increases linearly according to lemma 1. If $S_i \in \mathbf{S}_{min}$ and $S_j \notin \mathbf{S}_{min}$, then

$$E_i(t_0+\Delta)=E_i(t_0)+\delta_i\Delta \text{ and } E_j(t_0+\Delta)=E_j(t_0)+\delta_j\Delta \quad \textit{[lemma 1]}$$

$$E_j(t_0+\Delta)-E_i(t_0+\Delta)=E_j(t_0)-E_i(t_0)+(\delta_j-\delta_i)\Delta \quad \textit{[subtracting the above two equations]}$$

Rearranging the above relations, we see that E_i is less than E_j for all times

$$t \geq t_0 + (E_i(t_0)-E_j(t_0))/(\delta_j-\delta_i).$$

If both $S_i, S_j \in \mathbf{S}_{min}$, then under the same conditions, the server with the smaller $E(t_0)$ continues to have the smallest error. Therefore, for all $t \geq t^0{}_x$, with

$$t_x^0 \equiv t_0 + \mathbf{max}\left[\frac{E_i(t_0)-E_k(t_0)}{\delta_k-\delta_i}\right]$$

where $S_i \in \mathbf{S}_{min}$ and $S_k \in \mathbf{S}-\mathbf{S}_{min}$, the element of $\mathbf{S}_{min}$ with the smallest initial error would be S_M.

Now consider what happens at time $t_0+\Delta_1$ when the first time server resets. Let S_r be this server (if more than one server simultaneously resets, arbitrarily choose one of them to be S_r). If we again assume that no servers reset after Δ_1, then by the time

$$t_x^1 \equiv t_0 + \mathbf{max}\left[\frac{E_i(t_0+\Delta_1)-E_k(t_0+\Delta_1)}{\delta_k-\delta_i}\right]$$

the element of $\mathbf{S}_{min}$ with the smallest initial error would be S_M.

If it were possible for $t^1{}_x > t^0{}_x$, there would be a series of resets such that $t^{m+1}{}_x > t^m{}_x$, and the theorem would be false. The largest values of $t^1{}_x$ would be obtained for the largest $E_i(t_0+\Delta_1)$ and the smallest $E_k(t_0+\Delta_1)$. Since $S_i \in \mathbf{S}_{min}$, $E_i(t_0+\Delta_1)=E_M(t_0+\Delta_1)$, so $t^1{}_x$ is maximized for the smallest $E_k(t_0+\Delta_1)$.

$$E_M(t_0+\Delta_1) \leq E_j(t_0+\Delta_1+0) \quad \textit{[lemma 3]}$$

Thus, the largest value of $t^1{}_x$ is obtained for $E_k=E_j$ and $E_j(t_0+\Delta_1+0)=E_M(t_0+\Delta_1)$.

$$E_i(t_0+\Delta_1)=E_i(t_0)+\delta_i\Delta-\beta_i \text{ with } \beta_i \geq 0 \quad \textit{[lemma 2]}$$

$$E_M(t_0+\Delta_1)=E_M(t_0)+\delta_M\Delta \quad \textit{[hypothesis and lemma 1]}$$

Substituting the last three equations into the definition of $t^1{}_x$, we get, after some algebra:

$$t_x^1 \leq t_0 + \mathbf{max}\left[\frac{E_i(t_0)-E_M(t_0)}{\delta_k-\delta} + \left(\frac{\delta_i-\delta_M-\frac{\beta_i}{\Delta_i}}{\delta_k-\delta_i}\right)\Delta_1\right]$$

Since $S_i \in \mathbf{S}_{min}$, $\delta_M \geq \delta_i$ and $\beta_i \geq 0$, the coefficient of Δ_1 in this equation is less than or equal to zero. If we call this coefficient $-\alpha$, then we get from the above equation the following relationship:

$$t_x^1 \leq t_x^0 - \alpha\Delta_1 \text{ for } \alpha \geq 0$$

This was the largest $t^1{}_x$ we could construct, so this is true for any one server resetting. The same argument can be made for any $t^{m+1}{}_x$ given $t^m{}_x$, so

$$t_x^0 \geq t_x^1 \geq \ldots \geq t_x^m \geq t_x^{m+1} \geq \ldots$$

The latest time in which the service will have $S_M \in \mathbf{S}_{min}$ is $t^0{}_x$. Since the diference in the errors is bounded, this is a finite value. □

From theorems 3 and 4, it is easy to prove that eventually the error of all of the clocks in the time service will be bounded by some linear function $E_i(t) \leq a + b\delta_{min}$. This means that the "long term" growth of the error in the system (or the average growth of the error over a base that is large in in terms of τ) is equal to the accuracy of the most accurate clock in the system.

Experimentation with this algorithm has shown that it behaves as expected when servers have good bounds on their maximum drift rates. This algorithm, however, has some flaws; in particular, it is not resiliant to some clock possessing an invalid upper bound on its drift rate. It is not clear what a server should do when it finds itself inconsistent with another server. Even though it knows that at least one of them must be incorrect, it cannot easily tell which. A majority voting scheme may not work since the consistency property is not transitive.

A simple solution is to assume that the probability of any time server being incorrect is small. When a server finds itself inconsistent with another server, it assumes that the probability of a third time server also being incorrect is very small, so the original server resets to the value of any third server. This recovery algorithm can work very well. For example, in one experiment there was a network of two servers in which one server assumed its maximum drift rate was bounded by one second a day and whose actual drift rate was closer to one hour a day (about four percent fast). Each time either of the two clocks decided to reset, it found itself inconsistent with its neighbor and obtained the time from a server on some other network. The main problem was that the servers did not check their neighbor very often, so the time of the inaccurate clock would be very far off by the time it reset.

This recovery algorithm can break down as soon as there is more than one incorrect server directly connected to a server. In this case, the service can partition into different consistency groups (Figure 4). This is discussed further in Section 5.

4 Intersection as a Synchronization Function

A time service using algorithm *MM* can not be any more accurate than the most accurate clock in the system. This is an obvious property of all algorithms that select a time from one clock in the system. In addition, algorithm *MM* does not guarantee that each server will select the same clock with which to synchronize, so the asynchronism is limited only by the consistency of the system.

The consistency of the clocks, however, suggests another synchronization function. Since the time intervals must intersect in order to mutually define a correct time, the correct time must lie in the interval defined to be the intersection of all of the individual intervals. For example, Figure 2 shows two different intersections of intervals. In both cases, if the individual intervals are correct, the correct time must lie within the shaded region.

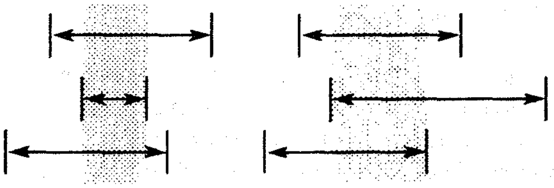

Figure 2
Intersections of Maximum Errors

The intersection of the intervals is determined by the servers with the latest trailing edge and earliest leading edge. There are two ways in which this intersection can occur: either both edges are defined by the same interval (as is the case with the left side of Figure 2), or the edges are defined by different servers (as with the right side of Figure 2). The first case reduces to algorithm *MM*, while the second case produces an interval that is smaller than the smallest pre-existing interval. One would expect that in some cases an algorithm using intersection will perform better than algorithm *MM*.

More formally, the *intersection* of the time intervals of two time servers S_i and S_j is defined to be

$$[\mathbf{max}[C_i - E_i, C_j - E_j] .. \mathbf{min}[C_i + E_i, C_j + E_j]] \quad (12)$$

If one interval is not a subset of the other, and $C_i - E_i \le C_j - E_j$, then it can be shown from equation 12 that $C_i + E_i \le C_j + E_j$. The following relationships can be derived from these two inequalities:

$$C_i \le C_j \quad (13)$$

$$|C_i - C_j| \le |E_i - E_j| \quad (14)$$

An algorithm using intersection as a synchronization function can be defined by the following two rules:

IM-1 The same as *MM-1*.

IM-2 Transform each reply $<C_j, E_j>$ into the interval $[T_j .. L_j]$ where

$$T_j \leftarrow C_j - E_j - C_i$$
$$L_j \leftarrow C_j + E_j + (1+\delta_i)\xi^i{}_j - C_i$$

The interval $[a .. b]$ is constructed as $a \leftarrow \mathbf{max}\ T_j$ and $b \leftarrow \mathbf{min}\ L_j$ over all of the replies. If $b > a$, then the time service is consistent, and S_i sets $\varepsilon_i \leftarrow (b-a)/2$, $C_i \leftarrow (a+b)/2$ and $r_i \leftarrow (a+b)/2$.

As with algorithm *MM*, this algorithm will be useful only if it preserves the correctness of the clocks running the service

Theorem 5 A correct time service which follows algorithm *IM* and in which all the servers have valid bounds on their maximum drift rates will remain correct.

Proof: From rule *IM*-1 and theorem 1, a correct time server that is not reset will remain correct. Since a time server only changes its clock to the intersection of the intervals $[C_i - E_i .. C_i + E_i]$ and $[C_j - E_j .. C_j + E_j + (1+\delta_i)\xi^i{}_j]$ (rule *IM*-2), all we need to show is that both of these intervals are correct at time $t + \xi_j$. The first interval is just the value of the unchanged clock C_i, so from theorem 1 it is correct. We need to show that if C_j is correct at $t + \sigma_j$,

$$C_j(t+\sigma_j)-E_j(t+\sigma_j)\leq t+\xi_j$$
$$\leq C_j(t+\sigma_j)+E_j(t+\sigma_j)+(1+\delta_i)\xi^i_j$$

$$C_j(t+\sigma_j)-E_j(t+\sigma_j)\leq t+\sigma_j \quad [\textit{definition of } \textbf{correct}]$$
$$\leq t+\xi_j \quad [\textit{definition of } \xi_j]$$
$$C_j(t+\sigma_j)+E_j(t+\sigma_j)+(1+\delta_i)\xi^i_j\geq t+\sigma_j+(1+\delta_i)\xi^i_j$$
$$[\textit{definition of } \textbf{correct}]$$
$$\geq t+\sigma_j+\xi_j \quad [\textit{equation 1}]$$
$$\geq t+\xi_j \quad [\sigma_j\geq 0]$$

□

One way to compare algorithm *IM* with *MM* is to compare the final intervals that are calculated by these algorithms given the same input. This is equivalent to comparing the minimum of a set of intervals to the intersection of these intervals. The intuitive comparison given in the beginning of this section is proven in theorem 6.

Theorem 6 The intersection of the intervals of a time service is at least as small as the smallest interval.

Proof: The intersection of the intervals is defined by equation 12. Either the server with the maximum trailing edge will also have the minimum leading edge or it will not. If server S_i does, then

$$C_i-E_i\geq C_j-E_j \quad [\textit{equation 12}]$$
$$C_i+E_i\leq C_j+E_j \quad [\textit{equation 12}]$$

Rearranging and combining these two inequalities gives $E_i\leq E_j$. Since this is true for all E_j, the intersection is the smallest interval.

Assume that S_i has the minimum leading edge and S_j has the maximum trailing edge with $S_i\neq S_j$.

$$C_j-E_j\geq C_k-E_k \quad [\textit{equation 12}]$$
$$C_i+E_i\leq C_k+E_k \quad [\textit{equation 12}]$$

Rearranging and combining these two inequalities gives $E_i+E_j+C_i-C_j\leq 2E_k$. From equation 12, the length of the intersection is $F\ +E_j+C_i-C_j$. Since this is true for all k, the intersection is at least as small as the smallest interval □

Since the interval to which a server sets its clock is derived rather than selected, algorithm *IM* will in general keep clocks much better synchronized than algorithm *MM*. This is proven in the following theorem:

Theorem 7 In a time service running algorithm *IM*, the asynchronism is bounded by

$$|C_i(t)-C_j(t)|\leq\xi+(\delta_i+\delta_j)\tau$$

Proof: At some time t in the time service, a clock has a value $Ci(t)=t+\varepsilon_i$ and an error $E_i(t)=e_i$. If clock $C_i(t)$ is correct, $|\varepsilon_i|\leq e_i$. If a server sends a time request to S_i at time t, then the reply from S_i will produce the interval

$$[t+\varepsilon_i+\sigma_i-e_i\ .\ t+\varepsilon_i+2\sigma_i+\rho_i+e_i]$$

This reply is received by the requestor at time $t'=t+\sigma_i+\rho_i$. The transformed interval $[T_i\ ..\ L_i]$ is $[\varepsilon_i-e_i-\rho_i\ ..\ \varepsilon_i+\sigma_i+e_i]$. The intersection of all of the replies is

$$[\textbf{max}\ \varepsilon_i-e_i-\rho_i\ ..\ \textbf{min}\ \varepsilon_i+\sigma_i+e_i]$$
$$=[-(\textbf{min}\ \tau_i+\sigma_i)\ ..\ \textbf{min}\ \lambda_i+\sigma_i]$$

where $\tau_i\equiv e_i-\varepsilon_i$ and $\lambda_i\equiv e_i+\varepsilon_i$. Let λ_{min} and τ_{min} be the minimum values of λ_i and τ_i, respectively. The largest possible asynchronism between two servers resetting at the same time would be for one to have its intersection interval edges as early as possible and the other's edges as late as possible. The largest possible value of **min** $\tau_i+\sigma_i$ is $\tau_{\mathrm{min}}+\xi$ and the smallest possible value is τ_{min}. Similarly, the largest possible value of **min** $\lambda_i+\sigma_i$ is $\lambda_{\mathrm{min}}+\xi$ and the smallest possible value is λ_{min}. Thus, the intervals $[-\tau_{min}-\xi\ ..\ \lambda_{min}]$ and $[-\tau_{min}\ ..\ \lambda_{min}+\xi]$ result in the largest asynchronism The midpoints of these intervals differ by

$$|-\tau_{min}-\xi+\lambda_{min}-(-\tau_{min}+\lambda_{min}+\xi)|/2\ =\ \xi$$

The theorem follows from the definition of δ and τ. □

As was mentioned before, the error in a time service running algorithm *IM* should in the best case grow more slowly than a time service running algorithm *MM*. The next theorem states that the expected growth of the error in the system may also be slower.

Theorem 8 Let the actual drift rate a clock C_i exhibits between two successive readings of its value relative to a standard be the random variable α. Let this random variable be distributed by some probability density function that is nonzero only over the range $-s_i\delta\leq\alpha\leq s_i\delta$ where $\delta=\delta_i/s_i$. Assume that at time t_0 the clocks are synchronized and have the same error e_0, and the intersection of the intervals of the n servers at time $t\geq t_0$ is e. If the α are identically distributed and independent and the clocks are not reset in this interval,

$$\lim_{n\to\infty} E(e)=e_0$$

Proof The following lemma, given without proof, will be used in the proof of theorem 7. A proof of lemma 5 can be found in [Marzullo 83].

Lemma 5 Define n independent random variables θ, identically distributed according to a probability density function that is nonzero only over the range $0 \leq \theta \leq \Theta$. If θ_{max} is the maximum value of the random variables and θ_{max} the minimum value, and $E(\theta)$ is the expected value of θ, then

$$\lim_{n \to \infty} E(\theta_{max}) = \Theta \text{ and } \lim_{n \to \infty} E(\theta_{min}) = 0$$

Define the random variable θ to be $\alpha + \delta$, so $0 \leq \theta \leq 2\delta$. At time t,

$$C_i(t) = t_0 + (t - t_0)(1 + s_i(\theta - \delta)) \quad [\textit{definition of } \delta, \theta, \alpha]$$
$$E_i(t) = e_0 + s_i(t - t_0)\delta \quad [\textit{lemma 1}]$$

Let $T_i(t)$ be the trailing edge of the interval $<C_i(t), E_i(t)>$ and $L_i(t)$ the leading edge. The above two equations give expressions for $T_i(t)$ and $L_i(t)$:

$$T_i(t) = t - e_0 + s_i(t - t_0)(\theta - 2\delta)$$
$$L_i(t) = t + e_0 + s_i(t - t_0)\theta$$

From lemma 5, in the limit of $n \to \infty$, $E(\theta_{max}) = 2\delta$ and $E(\theta_{min} | \theta_{max}) = 0$. Substituting into the two equations above,

$$E(e) = [t + e_0 + s_i(t - t_0)E(\theta_{min} | \theta_{max}) - (t - e_0 + s_i(t - t_0)(E(\theta_{max}) - 2\delta))]/2 = e_0.$$

□

This algorithm uses the information about how far the servers have drifted apart as compared to their possible drift in order to construct the more accurate interval. For example, consider a system consisting of a clock accurate to at least one second a day and a server accurate to at least two seconds a day. If these clocks were initially synchronized and were after one day three seconds apart, then the clocks could be correctly resynchronized The argument of theorem 8 is that given enough servers, the probability of having one server drift at $+\delta_i$ and another drift at $-\delta_j$ is large given that the maximum drift rates are valid. If the maximum drift rates are overspecified, then the expected growth in the error is the amount they are overspecified. This is equivalent to a service in which all of the clocks have a bias with respect to some time standard with a rate different from one second per second.

Experimental observations of this algorithm have shown that in correct systems the error does grow much slower than in algorithm *MM*. In one test of a small system where the δ_i were chosen casually, the error grew ten times slower than it would have under algorithm *MM*. This algorithm, however, has proven to be even less fault-tolerant than algorithm *MM*. There are consistent states in which algorithm *MM* will recover correctness while algorithm *IM* will not. The converse is not true. Figure 3 is an example of one of these states. The dashed line indicates the correct time, so that even though the servers are consistent, only S_1 and S_3 are correct. Under *MM*, a server would choose S_3, while under *IM*, a server would choose the incorrect interval $S_2 \cap S_3$. Algorithm *IM* is particularly susceptible to servers drifting slightly slower or faster than their assumed maximum drift rates

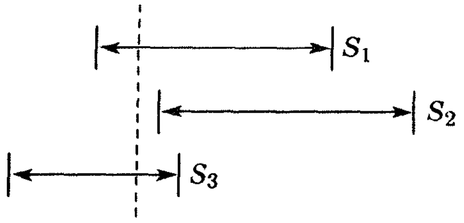

Figure 3

5 Inconsistency

The two algorithms presented here have been analyzed mathematically and experimentally in order to understand their behavior. Their greatest weakness is the need for each server to have a correct upper bound on the magnitude of its drift rate. If servers do not have valid bounds, the system can become inconsistent, leaving little information on which servers are incorrect. For example, Figure 4 shows an inconsistent six-server time service. There are three sets of consistent servers whose intersections are shown by the shaded areas. It is not apparent which set of servers (if any) is the correct one.

There is not enough information in the static arrangement of the time server intervals to determine why the system is inconsistent. Instead, the *rates* of the servers must be examined in order to determine how to recover. An interesting approach to this is to apply the algorithms in this paper to the rates of the clocks as well as to their values. Two

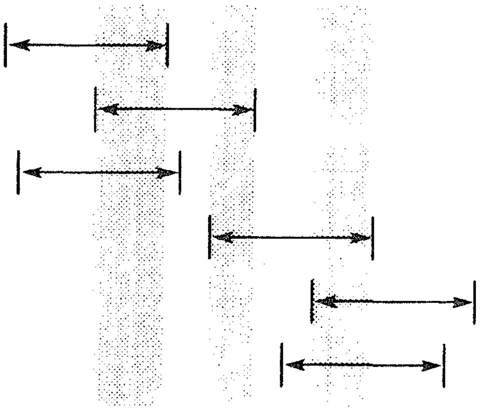

Figure 4
An Inconsistent Time Service

clocks have consistent rates (are *consonant*) at some time t_0 if their rate of separation is less than the sum of their maximum drift rates:

$$\left| \frac{d}{dt}\left(C_i(t)-C_j(t)\right) \right| \le \delta_i + \delta_j$$

A rate interval that is equivalent to the time interval used in the previous algorithms can be defined based on this definition of consonance. Algorithms *MM* and *IM* can then be applied to maintain a consonant set of δ_i, just as they were previously used to maintain a consistent set of t_i. The details are presented in [Marzullo 83] along with a more full development of the two algorithms described here

References

[Boggs 80] David R. Boggs, John F. Shoch, Edward A. Taft and Richard M. Metcalfe. Pup: an internetwork architecture. *IEEE Trans. on Comm. COM-22*, 5 (April 1980), 612-624

[Ellingston 73] C. Ellingston and R. J. Kulpinski. Dissemination of system-time. *IEEE Trans. Comm. Com- 23*, 5 (May 1973), 605-624.*of the ACM 27*, 7 (July 1978), 558-565.

[Lamport 78] Leslie Lamport. Time, clocks and the ordering of events in a distributed system. *Comm. of the ACM 27*, 7 (July 1978), 558-565.

[Lamport 82] Leslie Lamport and P.M. Melliar-Smith. Synchronizing clocks in the presence of faults. SRI International CSL (Lamport Opus 60), March 1982.

[Marzullo 83] Keith Marzullo Loosely-Coupled Distributed Services: A Distributed Time Service. Ph.D. dissertation Stanford University Computer Systems Laboratory, 1983 (draft)

[Mills 81] David L. Mills Time synchronization in DCNET hosts. COMSAT Laboratories (IEN 173), February 25, 1981.