Probabilistic Clock Synchronization in Distributed Systems

K. Arvind, Member, IEEE

Abstract—This paper presents and analyzes a new probabilistic clock synchronization algorithm that can guarantee a much smaller bound on the clock skew than most existing algorithms. The algorithm is probabilistic in the sense that the bound on the clock skew that it guarantees has a probability of invalidity associated with it. However, the probability of invalidity may be made extremely small by transmitting a sufficient number of synchronization messages. It is shown that an upper bound on the probability of invalidity decreases exponentially with the number of synchronization messages transmitted. A closed-form expression that relates the probability of invalidity to the clock skew and the number of synchronization messages is also derived.

Index Terms—Clock synchronization, deterministic algorithm, distributed systems, master-slave scheme, probabilistic algorithm, probability of invalidity, time transmission protocol

I. INTRODUCTION

A FAULT-FREE hardware clock, even if initially synchronized with a standard time reference, tends to drift away from the standard over a period of time. As a result, an interval of time measured with such a clock tends to be in error. However, the rate at which the hardware clock deviates from the standard is bounded by a constant. This constant, known as the maximum drift rate of the clock, is typically of the order of 1 μs/s. A direct consequence of the phenomenon of clock drift is that clocks in a distributed system gradually deviate from each other over a period of time. Closely synchronized clocks are necessary in several important distributed systems applications, including financial transactions, stock trading, airline reservations, hard real-time systems [10], [16], distributed file systems, authentication, and performance evaluation. A clock synchronization algorithm is used in a distributed system to ensure that the skew that develops between clocks remains bounded. Several clock synchronization algorithms have been proposed in the literature [3], [6], [8], [9], [12]–[14], [17], [18]. This paper proposes and analyzes a new clock synchronization algorithm based on a probabilistic approach. The proposed algorithm can guarantee a much smaller bound on the clock skew than most existing clock synchronization algorithms.

A. Deterministic Clock Synchronization Algorithms

Most clock synchronization algorithms proposed in the literature [6], [8], [9], [12]–[14], [17], [18] try to guarantee an upper bound on the clock skew with certainty. However, a theoretical limit derived by Lundelius and Lynch [15] limits the maximum clock skew that these deterministic algorithms can guarantee. It is shown in [15], that the upper bound on the clock skew that can be deterministically guaranteed by any clock synchronization algorithm can be no smaller than $(d_{\max} - d_{\min})(1 - \frac{1}{N})$. Here N is the number of nodes in the system, and $d_{\max}$ and $d_{\min}$, respectively, denote the maximum and minimum values of message delays in the system.

B. Probabilistic Clock Synchronization Algorithms

The theoretical limit established in [15] constrains only those algorithms that provide a deterministic guarantee on the maximum clock skew. Clock skews that are significantly smaller than this theoretical limit can be achieved if we are willing to relax the requirement of determinism and accept a probabilistic guarantee. A guarantee is said to be probabilistic if it fails to hold sometimes, but with a failure probability that can be determined or bounded. A clock synchronization algorithm that provides a probabilistic guarantee on the maximum clock skew, is referred to as a probabilistic clock synchronization algorithm. Note that the word "probabilistic," as used here, connotes the uncertainty in the guarantee offered by the algorithm, rather than any randomness in the actions of the algorithm [7].

Cristian's Probabilistic Algorithm: The idea of probabilistic clock synchronization was proposed by Cristian [3]. Cristian also proposed the first probabilistic clock synchronization algorithm [3], referred to as CRI in this paper. Cristian's algorithm is based on a remote clock reading (RCR). RCR is used by a node to read the clock at a remote node with a specified minimum accuracy. RCR involves querying a target node for the time on its clock. The querying node then estimates the time on the target node's clock from the response received. RCR guarantees that the maximum estimation error is approximately $D - d_{\min}$, where $D$ is half the response time and $d_{\min}$ is the minimum response time. Note that if the maximum error is specified as $U - d_{\min}$, $D$ has to be less than or equal to $U$. RCR is said to have achieved rapport when $D \leq U$. RCR repeatedly queries the target node until rapport is achieved. Note that if there is a constraint on the maximum number of attempts to achieve rapport, rapport may never be achieved.
CRI is a master-slave algorithm that makes use of RCR to achieve synchronization. One node in the system is designated as the master, and the remaining nodes are designated as slaves. Each slave periodically resynchronizes with the master by estimating the reading on the master’s clock by using RCR and adjusting its own clock accordingly. However, resynchronization is not guaranteed, because RCR may fail to achieve rapport. The probability that algorithm CRI fails to resynchronize can be determined analytically. Thus, CRI is a probabilistic clock synchronization algorithm, and is not subject to the limit established in [15]. As a result, CRI can guarantee much smaller maximum clock skews than deterministic algorithms.

Probabilistic Algorithm PCS: This paper proposes a new clock synchronization algorithm, referred to as PCS. PCS represents a different approach to probabilistic clock synchronization. The approach is based on a time transmission protocol called TTP. TTP is used by a node to communicate the time on its clock to a target node. At the core of TTP is a simple averaging algorithm that filters out variations in message delays. PCS is derived from TTP by incorporating TTP into a master-slave structure similar to that in CRI.

Algorithm PCS, being a probabilistic algorithm, offers an important advantage over deterministic clock synchronization algorithms. It can guarantee significantly smaller maximum clock skews than these algorithms. The maximum clock skew and the probability of invalidity guaranteed by algorithm PCS are comparable to those guaranteed by the probabilistic algorithm CRI. Also, in a broadcast network such as Ethernet, algorithm PCS requires a smaller number of synchronization messages than algorithm CRI to guarantee a specified maximum skew.

C. Organization of the Presentation

The rest of the paper is organized as follows. The next section discusses the assumptions underlying algorithm PCS. Section III describes the details of the time transmission protocol TTP and the algorithm PCS itself. Section IV summarizes the main properties of the algorithm. Section V presents schemes to accommodate systems that do not satisfy some of the assumptions made by the algorithm. Section VI concludes the paper with a summary. Appendix A contains the details of the proofs of various results presented in Section IV. Appendix B provides a glossary of symbols used in the paper.

II. Assumptions Underlying Algorithm PCS

The first part of this section lists the assumptions underlying algorithm PCS. The assumptions are examined and discussed in the second part.

A. List of the Assumptions

The probabilistic clock synchronization algorithm PCS presented in this paper makes the following assumptions.

1) Each hardware clock in the system has a clock synchronization process associated with it. A synchronization process has read-only access to its own clock.

2) The hardware clocks have a resolution that is much higher than the desired maximum clock skew and the message delays involved in the algorithm.

3) The rate at which skew develops between two hardware clocks is bounded. This bound is referred to as the relative drift rate, and is denoted by the symbol $\rho$. In order to simplify the analysis, the algorithm PCS also assumes that the derivative of the rate at which clock skew develops is negligible.

4) Each synchronization process implements one or more logical clocks in software. A logical clock is a linear function of the time on the hardware clock of the process. The term “clock,” as used in the rest of this paper, always refers to a logical clock.

5) Processes and communication links are free from faults.

6) The message delay between two processes can be modeled as a random variable, and estimates of its mean and standard deviation are known. Delays of successive synchronization messages are independent of each other.

B. Discussion of the Assumptions

Assumptions 1 and 4 can be realized without much difficulty. The bound on the drift rate mentioned in assumption 3 is normally guaranteed by the manufacturer of a hardware clock. The second part of assumption 3, namely, that the rate of change of the drift rate of a clock is negligible, is also reasonable. Changes in the drift rate are usually caused by changes in the ambient physical conditions. If the ambient physical conditions do not change abruptly, the drift rate also will not change abruptly.

Assumption 2, which is made by algorithm CRI also, implies two things. First, the smallest clock skew that PCS can guarantee is much higher than the resolution of hardware clocks. Quartz clocks typically have an accuracy of the order of microseconds. Thus, the clock skews that algorithm PCS can guarantee is of the order of tenths of milliseconds at best. This is still much better than the accuracy that can be guaranteed by deterministic clock synchronization algorithms. Second, the message delays in the system have to be significantly larger than the resolution of hardware clocks. Thus, the delays have to be at least of the order of tenths of milliseconds. This requirement is easily satisfied in real systems [3].

Assumption 5 is relaxed later on in this paper. Section V sketches schemes to handle performance, omission, and crash failures. This paper, however, does not consider other types of faults, such as Byzantine faults. The probability of observing these types of faults during the lifetime of a system can usually be made very small by building error-detecting and error-correcting redundancy into the components of the system [6].

Assumption 6 merits detailed examination. The delay incurred by a message is determined by several random factors. These factors include the current computational and communication load in the system, and random events such as context switches, page faults, and communication errors that result in retransmissions. Thus, modeling message delays as random variables is justifiable. The assumption that message
delays are independent of each other can be justified by imposing the following requirement: The interval between successive synchronization messages must be much larger than the average message delay. This requirement ensures independence as follows. If successive synchronization messages are sufficiently separated, then the random events that determine the corresponding message delays will be sufficiently separated. The correlation between the events is minimized as a consequence. A separation of about 2 s is used for algorithm CRI in [3].

Assumption 6 also states that estimates of the average and standard deviation of the message delay are available. Methods for determining these estimates for systems operating in a steady state are outlined in Section V. For dynamic systems in which these statistics can change with time, a priori estimates do not make sense. If worst case estimates of the statistics are available, algorithm PCS permits these to be used instead of the current estimates. However, this will require a tradeoff on the desired maximum clock skew. If worst case estimates are not available, auxiliary mechanisms to dynamically determine the statistics will have to be employed. These mechanisms are explored further in Section V.

Most of the assumptions above are made by algorithm CRI also. Instead of estimates of message delay statistics, algorithm CRI requires estimates of the probability distribution of round-trip message delays. The probability distribution and the assumption that message delays are independent of each other are used to precompute one of the algorithm parameters, namely, the number of attempts to achieve rapport. However, the maximum clock skew guaranteed by algorithm CRI is not dependent on the knowledge of the distribution or on the independence assumption.

III. DESCRIPTION OF ALGORITHM PCS

This section describes algorithm PCS. First, the time transmission protocol TTP on which algorithm PCS is based is presented in Section III-A. Section III-B then shows how to incorporate TTP into a master-slave scheme to derive algorithm PCS.

A. Time Transmission Protocol TTP

The time transmission protocol TTP is used by a node to communicate the time on its clock to a target node. TTP involves the transmission of a sequence of time-stamped messages to the target node. The target node estimates the time on the transmitting node’s clock, on the basis of the time-stamps on the messages and the message delay statistics.

TTP works as follows. When a node \( M \) wants to communicate the time on its clock to a target node \( S \), it sends a sequence of \( n \) synchronization messages to \( S \). The \( i \)-th message \( \text{msg}_i \), sent at time \( T_i \) on \( M \)'s clock, is of the following form: “Time is \( T_i \).” The separation between successive messages is a parameter of the time transmission protocol. The target node \( S \) records the time \( R_i \) according to its clock, at which it receives each message. After it has received the \( n \) messages, \( S \) estimates the current time on \( M \)'s clock by using the following equation:

\[
T_{est} = R_n - \bar{R}(n) + \bar{T}(n) + \bar{d},
\]

where

\[
\bar{T}(n) = \frac{1}{n} \sum_{i=1}^{n} T_i,
\]

and

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

The notation used in the above equations is as follows: \( T_i \) is the time stamp recorded by \( M \) on the \( i \)-th message transmitted; \( R_i \) is the time, according to \( S \)'s clock, at which the \( i \)-th message is received by \( S \); \( \bar{d} \) is an estimate of the expected value of the message delay.

The intuition behind the estimation equation, (1), is explained below. Let \( d_i \) and \( \delta_i \), respectively, denote the delay of the \( i \)-th message and the deviation between the clocks of \( S \) and \( M \), when the \( i \)-th message is received by \( S \). The actual time \( T_{act} \) on \( M \)'s clock at the time \( S \) makes the estimate given by the following:

\[
T_{act} = R_n - \delta_n.
\]

However, \( \delta_n \) is not determinable. Therefore, an approximation \( T_{approx} \) for \( T_{act} \) is obtained by replacing \( \delta_n \) in the above equation with \( \bar{d}(n) = \frac{1}{n} \sum_{i=1}^{n} \delta_i \), the average value of \( \delta_i \) over \( n \) message receipts.

\[
T_{approx} = R_n - \bar{d}(n).
\]

The \( i \)-th message is received at \( S \) \( d_i \) time units after it is time-stamped and transmitted by \( M \). When it arrives at \( S \), \( S \)'s clock is ahead of \( M \)'s clock by \( \delta_i \) time units. Hence, we get the following:

\[
R_i = T_i + d_i + \delta_i.
\]

It follows that

\[
\bar{d}(n) = \bar{R}(n) - \bar{T}(n) - \bar{d}(n),
\]

where \( \bar{d}(n) = \frac{1}{n} \sum_{i=1}^{n} d_i \). Therefore, from (2), we get the following:

\[
T_{approx} = R_n - \bar{R}(n) + \bar{T}(n) + \bar{d}(n).
\]

However, it is not possible to compute \( \bar{d}(n) \), because the individual message delays \( d_i \) are not known. Therefore, \( T_{approx} \) itself is approximated by replacing \( \bar{d}(n) \) with \( \bar{d} \), an estimate of the expected value of the message delay. The resultant approximation \( T_{est} \) is given by the following:

\[
T_{est} = R_n - \bar{R}(n) + \bar{T}(n) + \bar{d},
\]

which is nothing but (1).

Two other relevant quantities associated with TTP are the transmission error \( \epsilon \) and the transmission period \( \tau \). The transmission error \( \epsilon \) is defined as the error that \( S \) makes in estimating the time on \( M \)'s clock:

\[
\epsilon = T_{est} - T_{act}.
\]
where $T_{est}$ is the estimate that $S$ makes of the time on $M$’s clock, and $T_{act}$ is the actual time on $M$’s clock when the estimate is made. The transmission period $\tau$ is defined as the length of the interval over which $M$ transmits the $n$ synchronization messages.

**B. Derivation of PCS from TTP**

The time transmission protocol TTP can be incorporated into a master-slave scheme to derive the clock synchronization algorithm PCS. One node in the system is designated as the master. The clocks of the other nodes (the slaves) are synchronized to the master node using TTP.

The clock of a slave is synchronized with the master’s clock as follows. The master node communicates the time on its clock to the slave node using TTP. The slave estimates the time on the master’s clock as described in the previous section, and computes the difference between the estimate and the current time on its own clock. It then makes the necessary adjustment to its clock to synchronize it with the master’s clock. This synchronization procedure is repeated periodically in order to compensate for the drift that develops over a period of time. The length of the interval between successive repetitions of the synchronization procedure is called the resynchronization interval, denoted $R_{synch}$. Note that the resynchronization interval is at least as long as the transmission period $\tau$.

Figs. 1 and 2 formalize the above description in self-documenting pseudocode. The code for the master node makes use of two timers. The first one, called the resynchronization timer, is used to restart the synchronization procedure at the end of each resynchronization interval. The second timer, called the message timer, is used to separate successive messages by a specified interval $W$. $W$ is chosen large enough to ensure independence of successive synchronization message delays.

When there are multiple slaves, the master executes the procedure for each slave in the system. The master ensures that each slave’s clock is synchronized with its own clock. This action automatically ensures that the clocks of the slaves are mutually synchronized. The skew between any two clocks in the system is always less than or equal to twice the maximum skew between the master’s clock and any slave’s clock. Note that if the system is based on a broadcast network such as Ethernet, the master need transmit only a single set of messages per resynchronization interval, even if there are multiple slaves. Thus, the number of messages required in such a system will be independent of the number of nodes in the system.

In the synchronization procedure outlined above, the slave can make the adjustment to its clock in two ways. It can instantaneously adjust its clock at the point in time (the resynchronization point) when it estimates the adjustment to be made. This technique is called instantaneous adjustment. Alternatively, it can use the continuous adjustment technique, in which the adjustment is amortized over a period of time. Continuous adjustment offers some advantages, such as monotonically increasing time and the avoidance of multiple logical clocks. If instantaneous adjustment is used, the slave’s clock reads $T_{est}$, which is its estimate of the time on the Master’s clock, immediately after synchronization. The skew between the slave’s clock and the master’s clock at this point is then equal to $T_{act} - T_{est}$, which is identical to the transmission delay estimation error $\epsilon$. The transmission error represents the skew between the slave’s clock and the master’s clock at the resynchronization point. Hence, the transmission error $\epsilon$ is also referred to as the skew at resynchronization.

**IV. MATHEMATICAL ANALYSIS OF ALGORITHM PCS**

The preceding section describes how algorithm PCS can be used to keep clocks in a distributed system synchronized. However, the algorithm will not be very useful unless it can be shown that it can guarantee a small skew with a low probability of losing synchronization. Further, a technique that can be used to convert the service specifications (maximum clock skew, probability of invalidity) to the algorithm parameters (number of messages, interval between messages, and resynchronization interval), is required in order to use the algorithm. This section addresses both of these issues. First, Section IV-A presents relevant results from an analytical evaluation of the algorithm. Next Section IV-B illustrates the performance of the algorithm through numerical examples.

**A. Results from an Analysis of Algorithm PCS**

This section presents several important results relevant to algorithm PCS. The statement of each result is followed by a simple proof based on intuitive arguments. The proofs are meant to convince the reader why the result is to be expected, without subjecting him/her to the details. Rigorous proofs of the results are provided in Appendix A.
1) The clock skew at resynchronization is a random variable. The clock skew at resynchronization is identical to the transmission error $c$. The transmission error is primarily determined by the average of the deviations of the synchronization message delays from the expected value of the delay. This will be obvious after some thought and examination of the equations that were presented in Section III-A. Message delays may be modeled as random variables, as was argued in Section II. The result follows from the fact that the sum of a set of random variables is itself a random variable.

2) The magnitude of the expected value of the skew at resynchronization is negligible compared to the maximum skew at resynchronization.

   The principal component of the skew at resynchronization ($c$) is the average difference between the message delay and the expected value of the message delay. The expected value of this difference is zero. However, there are other components, such as the clock drift during the transmission period and the error in estimating the expected value of the message delay, that contribute to the value of $c$. Therefore, the expected value of $c$ is not zero, but is relatively negligible. This result assumes that the transmission period is sufficiently short that the drift that develops during this period is negligible compared to the specified maximum skew. Even a transmission period of the order of minutes typically satisfies this assumption. The result also assumes that the error in the estimate of the expected value of the message delay is negligible compared to the specified maximum skew.

3) The probability distribution of the normalized transmission error ($Z_n$) defined as follows:

   \[ Z_n \triangleq \frac{c - E[c]}{\sqrt{V[c]}} \]

   approaches the standard normal distribution $N(0,1)$ in limit as the number of messages $n$ increases. Here $E[c]$ and $V[c]$, respectively, denote the expected value and variance of the skew at resynchronization $c$. As argued earlier, the skew at resynchronization $c$ is primarily a sum of several random message delays. As stated earlier, it is assumed that synchronization message delays are independent and identically distributed during a transmission period. Thus, $c$ is a sum of several independent, identically distributed random variables. The result then follows from the central limit theorem. The larger the number of message delays involved, the closer is the distribution of $Z_n$ to $N(0,1)$. The minimum value of $n$ for $N(0,1)$ to approximate the probability distribution of $Z_n$, with an accuracy $\xi$, is referred to as the Gaussian cutoff corresponding to $\xi$. The Gaussian cutoff is denoted as $n_\xi$. Appendix A explains why $n_\xi > 10$ is sufficient to approximate the distribution of $c$ by a Gaussian distribution with good accuracy. If the distribution of the message delay is itself approximately Gaussian, $n_\xi$ will be even smaller.

4) The minimum number of messages required to guarantee a maximum skew at resynchronization of $\epsilon_{\text{max}}$ with a probability of invalidity $p$ is given by $n_{\text{min}} = \max(n_g, n_e)$, where the following expression is true:

   \[ n_e = \frac{2\sigma_d^2 (\text{erfc}^{-1}(p))^2}{\epsilon_{\text{max}}^2}. \]  

   Here $\sigma_d$ is the standard deviation of the message delay, and $\text{erfc}^{-1}(p)$ is the inverse of the complementary error function defined as $\text{erfc}(u) \triangleq 1 - \text{erf}(u)$, where $\text{erf}(u) \triangleq \frac{2}{\sqrt{\pi}} \int_0^u e^{-y^2} dy$. This result has its basis in the well-known relationship between the normal distribution and the error function. This is an important result that is used to determine the parameter $n$ (number of messages) of algorithm PCS from a specification of the desired maximum clock skew and probability of invalidity.

5) The probability of invalidity $p$ decreases exponentially or better with the number of messages, for a specified maximum clock skew at resynchronization $\epsilon_{\text{max}} (\epsilon_{\text{max}} > |E[c]|)$. In other words, there exists a bounding function for $p$ of the form $e^{-n\alpha}$, i.e., $p = P(|c| > \epsilon_{\text{max}}) \leq e^{-n\alpha}$, where $\alpha$ is a nonincreasing function of $n$, and $\beta$ is independent of $n$. Here $E[c]$ denotes the expected value of the clock skew at resynchronization $c$. It follows from the previous result that the probability of invalidity is related to the number of messages through the $\text{erfc}$ function. Table I lists the values of the $\text{erfc}$ function for various values of its argument. As the table shows, even small changes in the argument cause order of magnitude changes in the value of the $\text{erfc}$ function. This suggests an exponential relationship between the probability of invalidity and the number of messages. The relationship is rigorously shown in Appendix A.

6) The maximum clock skew between any two clocks in the system that algorithm PCS can guarantee is equal to $\gamma_{\text{max}}$, defined as follows:

   \[ \gamma_{\text{max}} = 2(\epsilon_{\text{max}} + (R_{\text{synch}} + d_{\text{max}} - d_{\text{min}}) \rho). \]

   Here $\epsilon_{\text{max}}$ denotes the desired maximum skew at resynchronization, $R_{\text{synch}}$ is the specified resynchronization interval, $d_{\text{max}}$ is the maximum message delay, $d_{\text{min}}$ is the minimum message delay, and $\rho$ is the relative clock drift. The term $(R_{\text{synch}} + d_{\text{max}} - d_{\text{min}})$ represents the worst case length of the interval between two resynchronization points. The amount by which a slave's clock can drift from the master's clock during this interval is given by $(R_{\text{synch}} + d_{\text{max}} - d_{\text{min}}) \rho$. The maximum skew that can develop between the master's clock and a slave's clock is the sum of this drift and the skew at resynchronization. The maximum skew between any two clocks in the system is twice the maximum skew between the master's clock and a slave's clock, because clocks can drift in opposite directions.

**B. Performance of Algorithm PCS**

This section presents some performance data for algorithm PCS. The performance is illustrated in terms of the magnitude of the maximum clock skew, the associated probability of invalidity, and the cost of the guarantee. Section IV-B presents two
### TABLE I

<table>
<thead>
<tr>
<th>$x$</th>
<th>$p = \text{erfc}(x)$</th>
<th>$x$</th>
<th>$p = \text{erfc}(x)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.000</td>
<td>$1.0 \times 10^0$</td>
<td>3.459</td>
<td>$1.0 \times 10^{-6}$</td>
</tr>
<tr>
<td>1.163</td>
<td>$1.0 \times 10^{-1}$</td>
<td>3.766</td>
<td>$1.0 \times 10^{-7}$</td>
</tr>
<tr>
<td>1.822</td>
<td>$1.0 \times 10^{-2}$</td>
<td>4.052</td>
<td>$1.0 \times 10^{-8}$</td>
</tr>
<tr>
<td>2.327</td>
<td>$1.0 \times 10^{-3}$</td>
<td>4.320</td>
<td>$1.0 \times 10^{-9}$</td>
</tr>
<tr>
<td>2.751</td>
<td>$1.0 \times 10^{-4}$</td>
<td>4.572</td>
<td>$1.0 \times 10^{-9}$</td>
</tr>
<tr>
<td>3.123</td>
<td>$1.0 \times 10^{-5}$</td>
<td>4.812</td>
<td>$1.0 \times 10^{-11}$</td>
</tr>
</tbody>
</table>

### TABLE II

#### $n$ vs. $\frac{\text{\text{max}}}{\text{\text{d}}}$ ($p = 1 \times 10^{-9}$)

<table>
<thead>
<tr>
<th>$n$</th>
<th>$\frac{\text{\text{max}}}{\text{\text{d}}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0 $\times 10^{-6}$</td>
<td>24</td>
</tr>
<tr>
<td>1.0 $\times 10^{-7}$</td>
<td>29</td>
</tr>
<tr>
<td>1.0 $\times 10^{-8}$</td>
<td>33</td>
</tr>
<tr>
<td>1.0 $\times 10^{-9}$</td>
<td>38</td>
</tr>
<tr>
<td>1.0 $\times 10^{-10}$</td>
<td>42</td>
</tr>
<tr>
<td>1.0 $\times 10^{-11}$</td>
<td>47</td>
</tr>
<tr>
<td>10.0</td>
<td>10</td>
</tr>
<tr>
<td>3.0</td>
<td>10</td>
</tr>
<tr>
<td>2.0</td>
<td>10</td>
</tr>
<tr>
<td>1.0</td>
<td>38</td>
</tr>
<tr>
<td>0.75</td>
<td>68</td>
</tr>
<tr>
<td>0.50</td>
<td>152</td>
</tr>
</tbody>
</table>

Numerical examples that provide an idea of the performance of algorithm PCS. Section IV-B compares the performance of algorithm PCS with other clock synchronization algorithms from the literature.

**Illustrative Performance Data:** The values used in this section for the message delay statistics are based on data reported in [3]. The data reported in [3] are from an experiment involving the measurement of 5000 round-trip message delays, between two lightweight processes running on two IBM 4381 processors. The processors were connected via a channel-to-channel local area network. The measured maximum and minimum delays in this experiment were 93.17 ms and 4.22 ms, the average delay was 4.91 ms, and the median delay was 4.48 ms. In addition, 95% of all observed delays were shorter than 5.2 ms.

For the purpose of estimating the variance of the message delays from these data, the distribution of these delays is approximated by a Gaussian distribution. This approximation may be justified as follows. A message delay is itself the sum of several independent random delays [9]:

1. **Send time:** The time required to assemble and send a synchronization message, which depends, for example, on the computational load on the system and page faults.
2. **Access time:** The medium access time of the sender, which depends on the medium access strategy, the number of nodes trying to access the medium, and the number of messages already queued.
3. **Propagation delay:** The propagation delay of the message, which depends on communication network conditions.
4. **Receive time:** The time required to process the message at the receiving node, which depends on context switching and scheduling delays.

Therefore, from the central limit theorem, the distribution of the message delay can be expected to be roughly Gaussian in nature (a conjecture that is supported by the roughly bell-like shape of the distribution as reported in [3]). Note that the Gaussian approximation used here is strictly for the purpose of obtaining a reasonable value for the variance to illustrate the performance of the algorithm; none of the results presented in the paper are based on this approximation.

If the distribution of the message delay is approximated by a Gaussian distribution, the standard deviation of the message delay may be estimated as 0.184 ms for the given data. This estimate makes use of the fact that the deviation from the median for 95% of the cases is less than 0.36 ms. If the probability that a Gaussian random variable differs from its median by more than 0.36 is 0.05, then the standard deviation of the variable is 0.184, from standard formulae. The examples here use a more conservative value of 0.2 for the standard deviation.

Assume a drift rate of $6 \mu s/s$ and a probability of invalidity of $p = 1 \times 10^{-9}$. By Result 6 in Section IV-A, a maximum clock skew $\tau_{\text{max}}$ of 2 ms can be guaranteed by choosing $\epsilon_{\text{max}} = 0.6$ ms, and $(R_{\text{Synch}} + d_{\text{max}} - d_{\text{min}}) \rho = 0.4$ ms. This gives a resynchronization interval $R_{\text{Synch}} < 67$ s, and by Result 4 of Section IV-A (or by Table II), $n = 10$ messages, assuming a Gaussian cutoff $n_d$ of 10.

If a less tight $\tau_{\text{max}}$ of 4 ms is desired, then the resynchronization interval can be longer. For example, a resynchronization interval of 200 s can be achieved by choosing $(R_{\text{Synch}} + d_{\text{max}} - d_{\text{min}}) \rho = 1.2$ ms and $\epsilon_{\text{max}} = 0.8$ ms. Note that the number of messages required is 10 in both examples, because we have assumed that $n_d = 10$. However, a more realistic estimate of $n_d$ would be much smaller, because the message delays have a distribution that is approximately Gaussian. A smaller number of messages would actually be sufficient in these examples.

A caveat is needed about the values derived for the number of messages. It follows from Result 4 of Section IV-A that because $n_d \propto \left(\frac{\Delta_d}{\epsilon_{\text{max}}}\right)^2$, the number of messages is sensitive to the value assumed for $\Delta_d$, when $\epsilon_{\text{max}}$ is of the same order as or smaller than $\Delta_d$; thus, for example, if the standard deviation of end-to-end delays were actually 0.6 ms, instead of the 0.2 ms that we obtained by using the Gaussian approximation, then the number of synchronization messages in the first example would be much larger than the value derived above (38 instead of 10).

### C. Comparison of PCS with Other Algorithms

**Deterministic Algorithms:** The $\tau_{\text{max}}$ values of 2 ms and 4 ms in the examples in the previous section are better than the best $\tau_{\text{max}}$ of $\frac{1}{2}(d_{\text{max}} - d_{\text{min}}) \approx 50$ ms achievable with

1Note that the data reported in [3] are round-trip delays. Hence, they have to be scaled down by half here.
the deterministic clock synchronization algorithms described in [6], [8], [9], [12]–[14], [17], [18], which have to conform to the theoretical limit established in [15]. Because of the exponential nature of the dependence of the probability of invalidity on the number of messages, it should be possible to reduce the probability of invalidity substantially by increasing the number of messages slightly. This is illustrated in Table II. Note that \( p \) can be reduced by an order of magnitude by adding just about five more messages.

**Algorithm CRI:** The maximum clock skews achievable with algorithm PCS are comparable to those achievable with the probabilistic clock synchronization algorithm CRI [3]. To guarantee a maximum clock skew of 2 ms with a probability of invalidity of \( 1 \times 10^{-9} \), algorithm CRI requires about four messages on the average, to be sent every 67 s. Algorithm PCS requires 10 messages to be sent every 67 s to realize the same specifications. To guarantee a maximum clock skew of 4 ms with the same probability of invalidity, algorithm CRI requires about 2.1 messages on the average, to be sent every 231 s. Algorithm PCS requires 10 messages to be sent every 200 s. As pointed out earlier, the number of messages required by algorithm PCS will be smaller if a more realistic value is used for the Gaussian cutoff \( n_g \).

In a distributed system based on a broadcast network such as Ethernet, with algorithm PCS, the master node need broadcast only a single set of synchronization messages per resynchronization interval. With algorithm CRI, each slave node is individually responsible for maintaining synchronization with the master. Hence, algorithm CRI requires as many sets of synchronization messages as there are slaves. Thus, the number of messages required by algorithm CRI will have to be multiplied by a factor \( k \) for a system with \( k \) slaves. However, algorithm CRI has a different kind of advantage to offer. CRI is fail-safe, i.e., a slave knows when it has lost synchronization [4]. With algorithm PCS, it is possible to reduce the probability of losing synchronization to a very small value. However, a slave has no way of knowing that it has lost synchronization at the time it loses it.

**V. AUXILIARY MECHANISMS TO SUPPORT PCS**

The assumptions underlying algorithm PCS and the algorithm itself were discussed in the preceding sections. In order to use algorithm PCS in systems that do not conform to these assumptions, auxiliary mechanisms that provide support for realizing these assumptions will be required. This section discusses these mechanisms.

**A. Determining Message Delay Statistics**

Algorithm PCS assumes that estimates of the expected value and standard deviation of the message delay are available. In many cases, determining the statistics of the message delay is a challenging problem in its own right.

If message delays in the system are isotropic, i.e., the distribution of the delay from node \( A \) to node \( B \) is identical to that from node \( B \) to node \( A \), determination of the distribution is simple. In this case, unidirectional message delays can be determined by measuring round-trip message delays and dividing them by a factor of 2.

If message delays are anisotropic, other approaches will have to be used. One approach is to use an external mechanism for clock synchronization between the master and slave node when the delay statistics are being measured. The master node time-stamps messages that it sends out; the slave node computes the message delays based on the time-stamps and the times of receipt of the messages. An example of an external synchronization mechanism is the use of Universal Coordinated Time (UTC) receivers to synchronize the master and slave clocks with the time signals broadcast by the National Institute of Science and Technology (NIST).

A second approach that merits further investigation is the "bootstrap approach." The bootstrap approach makes use of algorithm PCS itself to synchronize the master and slave clocks during the statistics measurement phase. First, the algorithm operates with rough estimates of the message delay statistics (obtained, for example, by measuring round-trip message delays). Message delays are then measured as in the first approach, by using algorithm PCS instead of UTC receivers, to synchronize the master and slave clocks. The original estimates of the delays used by PCS are then replaced by the newly measured values, and the measurements are made once again. This cycle of successive refinement is continued until a satisfactory degree of convergence is achieved.

**B. Dealing with Dynamic Systems**

If the system involved is a static or a steady-state system, i.e., the message delay statistics do not change with time, algorithm PCS can be directly used. If message delay statistics can change with time, a special process at the master node can be dedicated to keeping track of the current values of the delay statistics. Such a scheme would be suitable for systems in which the delay statistics change slowly.

Alternatively, the dependence of the algorithm on the current values of the message delay statistics may be eliminated. One approach is to use worst-case values for the statistics, instead of the current values. The statistics of the delay enter the analysis of the algorithm in two ways. First, the expected value of \( \epsilon \), which determines the smallest acceptable value of \( \epsilon_{\text{max}} \) (see Result 5 in Section IV), is dependent on the estimate of the expected value of the delay. Second, the variance of \( \epsilon \), which determines the minimum number of messages required to guarantee a specified skew and probability of invalidity, is dependent on the estimate of the variance of the delay. By using worst-case values for the expected value and variance of the delay, we merely increase the smallest value of \( \epsilon_{\text{max}} \) that can be guaranteed and the number of synchronization messages required to provide this guarantee.

The dependence of the algorithm on the average message delay can be completely eliminated by making a simple modification to the time transmission protocol (TTP). When node \( S \) receives a time-stamped message msg\(_{t} \) from node \( M \), it not only records the time of receipt of the message but also returns the message to \( M \) immediately. On receipt of the returned message, \( M \) records the time of receipt and
computes the round-trip time $r_i$ of the message. At the end of the transmission period, $M$ sends the average

$$\bar{r}(n) = \frac{1}{n} \sum_{i=1}^{n} r_i$$

of these round-trip delays to $S$. On receipt of this information, $S$ estimates the time on $M$’s clock as follows:

$$T_{est} = R_m - \bar{r}(n) + \bar{T}(n) + \frac{1}{2} \bar{r}(n).$$

In a broadcast network, only one slave has to be involved in this dynamic computation of the average. Once the master has computed $\bar{r}(n)$ on the basis of the messages returned by this slave, it can broadcast the value to the other slaves. It can be shown that the transmission error $\epsilon$ is given, in this case, by the following:

$$\epsilon = \frac{\hat{\rho}}{2} - \frac{1}{n} \sum_{i=1}^{n} d_i + \frac{1}{2n} \sum_{i=1}^{n} r_i,$$

where $d_i$ and $r_i$ denote the unidirectional and round-trip delays of the $i$th synchronization message, $n$ is the number of messages and $\hat{\rho}$ is the drift rate. The expected value of the transmission error is then given by the following:

$$E[\epsilon] = \frac{\hat{\rho} \tau}{2},$$

assuming that message delays are isotropic. Note that the dependence on the expected value of the delay has been eliminated. The variance of the transmission error now depends on the variance of the difference between the upward and downward components of a round-trip delay. The number of messages required to guarantee a specified probability of invalidity can be computed by using a worst-case estimate of this variance.

C. Dealing with Faults

Algorithm PCS assumes that processes and communication links are free from faults. This section sketches schemes to handle performance, omission, and master crash failures.

Performance and omission failures result in messages being delayed or lost. One of two approaches may be used to deal with delayed or lost messages. The first approach is to transmit an additional $n_x$ time-stamped messages over the number of messages normally required by the algorithm. The number $n_x$ is chosen such that the probability that more than $n_x$ of the transmitted messages are delayed or lost is smaller than a desired bound. The second approach is for a slave to estimate the master’s time by using whatever number of messages it has received by the estimation deadline. This approach will result in an increased probability of losing synchronization when messages are delayed or lost, because fewer messages than required are used to estimate the master’s time.

Master crashes may be handled using adaptations of schemes suggested for this purpose in [3]. Two such schemes are proposed in [2]. The first scheme is based on passive redundancy. In this scheme, a designated backup node takes over the master’s functions when the master crashes. The second scheme is based on fault masking through active redundancy. In this scheme, multiple masters, each of which is synchronized to a common external reference of time, execute concurrently.

VI. Conclusion

A new probabilistic clock synchronization algorithm PCS has been presented. The algorithm is based on a time transmission protocol TTP that can achieve very small transmission errors. TTP involves the transmission of a sequence of time-stamped messages and a time estimation procedure. The estimation procedure works by filtering out random variations in synchronization message delays through a process of averaging. Algorithm PCS is derived from TTP by incorporating TTP into a master-slave structure.

Algorithm PCS, being a probabilistic algorithm, is not subject to the bounds on the clock skew imposed by the theoretical limit derived in [14]. Therefore, algorithm PCS performs much better than deterministic clock synchronization algorithms that are constrained to obey this limit. The examples presented demonstrated that algorithm PCS is capable of guaranteeing a maximum clock skew of the order of a few milliseconds. In comparison, the smallest maximum clock skew that deterministic algorithms could guarantee was about 50 ms. However, though deterministic algorithms provide an absolute guarantee, algorithm PCS can provide only a probabilistic guarantee. There is a nonzero probability that the guarantee provided by algorithm PCS will fail to hold. This probability of invalidity may, however, be made extremely small by transmitting a sufficient number of synchronization messages. The examples have demonstrated that it is possible to achieve a probability of invalidity as low as $1 \times 10^{-9}$, by transmitting just 10 messages. The examples also showed that the performance of algorithm PCS is comparable to that of algorithm CRI, another probabilistic clock synchronization algorithm proposed in the literature.

An expression has been derived to relate the minimum number of synchronization messages required, the probability of invalidity, and the maximum clock skew guaranteed by the algorithm. This expression is used to convert service specifications for the algorithm into the parameters of the algorithm. A bound that makes explicit the exponential nature of the dependence of the probability of invalidity on the number of synchronization messages has also been derived.

There are a number of aspects of algorithm PCS that can be improved through further research. Unlike most other work in this area, which focuses mainly on fault tolerance, the focus of this work has been on achieving a clock skew that is smaller than previous bounds. Improvement of the fault tolerance of the algorithm, by identifying new schemes to handle various kinds of failures, is one area of research. Another area of research is to identify how the time transmission protocol proposed in this paper can be combined with the algorithms in [6], [8], [9], [12]–[14], [17], [18] to avoid the theoretical limit established in [14]. Finally, it would be interesting to identify other kinds of probabilistic algorithms, and to determine the
theoretical and practical bounds on the clock skew that can be guaranteed by probabilistic clock synchronization algorithms.

APPENDIX

A. Detailed Analysis of Algorithm PCS

Section IV listed several results and provided intuitive justifications for them. This section proves the results rigorously.

The three most important variables of algorithm PCS are the clock skew, the probability of invalidity, and the number of synchronization messages. This section contains proofs for various results and lemmas relevant to the clock skew at resynchronization (which is also referred to as the transmission error). Various useful bounds on the probability of invalidity are derived in the next section. In the last section of this Appendix, an important result relating the number of synchronization messages, the probability of invalidity, and the maximum clock skew at resynchronization is proved.

Analysis of the Clock Skew at Resynchronization:

**Lemma 1:** Let $d_i$ and $\delta_i$, respectively, denote the end-to-end delay of the $i$th message and the skew between the clocks of slave $S$ and the master $M$ when the $i$th message is received at $S$. The transmission error $\epsilon$ is given by the following:

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

where $\Delta \delta_n \triangleq \delta_n - \bar{\delta}(n)$, $\Delta \bar{d}(n) \triangleq \bar{d}(n) - \bar{d}$, $\bar{d}(n) = \frac{1}{n} \sum_{i=1}^{n} d_i$, $\bar{d}(n) = \frac{1}{n} \sum_{i=1}^{n} d_i$, and $n$ is the total number of messages transmitted.

**Proof:** The $i$th 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 $\delta_i$ time units. Hence, we get the following:

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

and

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

The latter equation can be rewritten as follows:

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

The $n$th 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 $n$th 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 subtracting the skew $\delta_n$ between $S$ and $M$ at the time of arrival of the $n$th message at $S$, from the time of arrival. Hence, we get the following:

$$T_{act} = T_n + d_n = R_n - \delta_n. \tag{7}$$

The transmission error defined as $\epsilon = T_{act} - T_{act}$ is then given by (1) and (7) as follows:

$$\epsilon = T_{act} - T_{act}$$

$$= (R_n - \bar{R}(n) + \bar{T}(n) + \bar{d}) - (R_n - \delta_n)$$

$$= \bar{d} + \delta_n - (\bar{d}(n) + \delta(n))(\text{ from } (6))$$

$$= \Delta \delta_n - \Delta \bar{d}(n) \tag{8}$$

**Lemma 2:** Let $\delta_0$ denote the skew between the clocks of slave $S$ and the master $M$ at the start of transmission of the first message; let $\delta_i^r$ denote the increase in skew between the times of transmission of the first and the $i$th messages; and let $\delta_i^d$ denote the increase in skew during the time it takes $msg_i$ to reach $S$ after it has been transmitted (i.e., the increase in skew during the message delay $d_i$ of $msg_i$). The transmission error $\epsilon$ can be approximated as follows:

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

where $\Delta \delta_n = \delta_n^r - \bar{\delta}(n), \bar{\delta}(n) = \frac{1}{n} \sum_{i=1}^{n} \delta_i^r, \Delta \bar{d}(n)$ is as defined in Lemma 1, and $n$ is the total number of messages transmitted.

**Proof:** The quantity $\Delta \delta_n$ represents the amount by which the clock of $S$ is ahead of the clock of $M$, when the $i$th message arrives at $S$. It can be decomposed into three components, namely, the skew between the clocks of $S$ and $M$ at the start of transmission of the first message ($\delta_0$), the increase in skew between the clocks of $S$ and $M$ between the times of transmission of the first and $i$th messages ($\delta_i^r$), and $\delta_i^d$, the increase in skew during the time it takes the $i$th message to reach $S$ after it has been transmitted ($\delta_i^d$). Thus, $\delta_i = \delta_0 + \delta_i^r + \delta_i^d (1 \leq i \leq n)$. Hence, we get the following:

$$\bar{\delta}(n) = \frac{1}{n} \sum_{i=1}^{n} \delta_i = \delta_0 + \bar{\delta}(n) + \frac{1}{n} \sum_{i=1}^{n} \delta_i^d.$$ Therefore, we get the following:

$$\Delta \delta_n = \delta_n - \bar{\delta}(n) = \delta_n - \bar{\delta}(n) + \frac{1}{n} \sum_{i=1}^{n} \delta_i^d - \frac{1}{n} \sum_{i=1}^{n} \delta_i^d. \tag{9}$$

Each $\delta_i^d$ can be written as $\delta_i^d = \bar{\rho}d_i$, where $\bar{\rho} \leq \rho$ is the actual relative drift rate. Equation (9) can then be rewritten as follows:

$$\Delta \delta_n = \delta_n^r - \bar{\delta}(n) + \rho d_n - \frac{1}{n} \sum_{i=1}^{n} \bar{\rho}d_i. \tag{10}$$

It follows from Lemma 1 that

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

$$= \delta_n^r - \bar{\delta}(n) + \rho d_n - \frac{1}{n} \sum_{i=1}^{n} \bar{\rho}d_i - \frac{1}{n} \sum_{i=1}^{n} d_i + \bar{d}$$

$$= \delta_n^r - \bar{\delta}(n) - \frac{1}{n} \sum_{i=1}^{n} d_i + \bar{d}$$

$$= \Delta \delta_n - \Delta \bar{d}(n).$$

In the third step above, $\bar{\rho}$ and $\bar{\delta}$ were neglected in comparison to $\frac{1}{n}$, by assuming that $n \ll \rho \frac{1}{\bar{\rho}}$ is of the order of $10^9$.

**Q.E.D.**

**Lemma 3:**

$$\Delta \delta_n = \frac{\bar{\rho} \tau}{2},$$

where $\bar{\rho}$ denotes the actual relative drift rate between the clocks of $M$ and $S$ during the transmission period, and $\tau$ is the transmission period.
Proof: The increase in the skew between the clocks of \( M \) and \( S \) between the times of transmission of the first and \( i \)th messages, \( \delta_i \), is equal to \((i - 1)W \dot{\rho} \), the clock drift during this interval. Therefore, \( \delta_i(n) = \frac{1}{n} \sum_{i=1}^{n} \delta_i = \frac{(n - 1)}{2} W \dot{\rho} \), and \( \Delta \delta_i = \delta_i - \delta_i(n) = \frac{(n - 1)}{2} W \dot{\rho} \). The result follows from the definition of \( \tau \). Q.E.D.

From Lemmas 1, 2, and 3, we can write the skew at resynchronization \( \epsilon \) as follows:

\[
\epsilon = \frac{\hat{\rho} \tau}{2} - \frac{1}{n} \sum_{i=1}^{n} d_i + \tilde{d}.
\]

(11)

The skew at resynchronization \( \epsilon \) is the sum of several random variables. Hence, it is itself a random variable. The expected value of \( \epsilon \) is given by the following:

\[
E[\epsilon] = \frac{\hat{\rho} \tau}{2} - E[d_i] + \tilde{d}.
\]

(12)

Clearly,

\[
E[\epsilon] \leq \frac{\hat{\rho} \tau}{2} + \Delta \tilde{d},
\]

(13)

where \( \Delta \tilde{d} \) represents the maximum expected value of the message delay in the time transmission protocol TTP (i.e., \( \Delta \tilde{d} \geq |E[d_i] - \tilde{d}| \)); \( \rho > \hat{\rho} \) is the maximum relative drift rate. Clearly, the following property holds if the transmission period \( \tau \) is short enough (\( \tau \ll \frac{2 \Delta \tilde{d}}{\rho} \)).

Property 1: The magnitude of the average skew at resynchronization \( E[\epsilon] \) is negligible compared to a specified maximum skew at resynchronization \( \epsilon_{\text{max}} \), i.e., \( E[\epsilon] \ll \epsilon_{\text{max}} \), provided that the error in estimating the expected value of the message delay, \( \Delta \tilde{d} \), is negligible compared to \( \epsilon_{\text{max}} \).

The variance of \( \epsilon \) is given by the following:

\[
V[\epsilon] = \frac{1}{n^2} \sum_{i=1}^{n} V[d_i] = \frac{V[d_i]}{n}.
\]

(14)

The above equation makes use of one of the assumptions of the algorithm, that the actual drift rate does not change rapidly and hence can be assumed to be constant over a transmission period.

If \( \sigma_d^2 \) denotes a worst-case (maximum) estimate of the variance of end-to-end delay, (14) reduces to the following:

\[
V[\epsilon] \leq \frac{\sigma_d^2}{n}.
\]

(15)

Thus, the better the estimate of the variance of the end-to-end delay, the smaller is the bound on the variance of the transmission error.

B. Analysis of the Probability of Invalidity

This section contains a detailed analysis of the probability of invalidity. The probability of invalidity may be determined empirically. However, this is an impractical approach involving an extremely large number of trials, because we are seeking probabilities of the order of \( 10^{-9} \). So, a theoretical analysis is necessary. Three theoretical bounds, namely, the Tchebycheff, error function and exponential bounds are derived in this section. The Tchebycheff bound has general applicability, but is a weak bound. The error function and exponential bounds are stronger bounds, which assume that message delays are independent and identically distributed over a transmission period. The error function bound directly leads to an expression relating the number of messages required to the service specifications of the algorithm. The error function bound makes explicit the exponential nature of the relationship between the probability of invalidity and the number of synchronization messages.

Tchebycheff Bound on the Probability of Invalidity: An upper bound on the probability of invalidity may be determined using Tchebycheff’s inequality. Tchebycheff’s inequality states that

\[
P[|\epsilon - \bar{\epsilon}| \geq \Delta \epsilon] \leq \frac{V[\epsilon]}{(\Delta \epsilon)^2},
\]

for any \( \Delta \epsilon > 0 \). Here \( \bar{\epsilon} \) denotes the expected value of the random variable \( \epsilon \), and \( V[\epsilon] \) denotes the variance of \( \epsilon \). The following lemma is useful in determining the probability of invalidity from Tchebycheff’s inequality.

Lemma 4:

\[
P[|\epsilon - \bar{\epsilon}| \geq \epsilon_{\text{max}}] \leq P[|\epsilon - \bar{\epsilon}| \geq \Delta \epsilon_{\text{max}}],
\]

if \( \Delta \epsilon_{\text{max}} > 0 \), where \( \Delta \epsilon_{\text{max}} = \epsilon_{\text{max}} - |\bar{\epsilon}| \).

Proof: This result follows from the observation that regardless of whether \( \bar{\epsilon} \) is positive or negative, the range of values of \( \epsilon \), \([-\Delta \epsilon_{\text{max}} + \bar{\epsilon}, \bar{\epsilon} + \Delta \epsilon_{\text{max}}] \), is a subset of the range of values, \([-|\bar{\epsilon}| - \Delta \epsilon_{\text{max}}, |\bar{\epsilon}| + \Delta \epsilon_{\text{max}}] \). This implies that

\[
P[|\epsilon - \bar{\epsilon}| \geq \Delta \epsilon_{\text{max}}] \leq P[-|\bar{\epsilon}| - \Delta \epsilon_{\text{max}} \leq \epsilon \leq \bar{\epsilon} + \Delta \epsilon_{\text{max}} \geq P[-\Delta \epsilon_{\text{max}} + \bar{\epsilon} \leq \epsilon \leq \bar{\epsilon} + \Delta \epsilon_{\text{max}}],
\]

i.e.,

\[
P[|\epsilon - \bar{\epsilon}| \leq \epsilon_{\text{max}}] \geq P[|\epsilon - \bar{\epsilon}| \leq \Delta \epsilon_{\text{max}}],
\]

or, equivalently,

\[
P[|\epsilon - \bar{\epsilon}| \geq \Delta \epsilon_{\text{max}}] \geq P[|\epsilon - \bar{\epsilon}| \geq \epsilon_{\text{max}}].
\]

Q.E.D

The following bound, referred to as the Tchebycheff bound, follows from Tchebycheff’s inequality and Lemma 4:

\[
P[|\epsilon - \bar{\epsilon}| \geq \epsilon_{\text{max}}] \leq \frac{V[\epsilon]}{(\epsilon_{\text{max}} - |\bar{\epsilon}|)^2}.
\]

(16)

The Tchebycheff bound on the probability of invalidity is proportional to the variance of the transmission error. This bound on the probability of invalidity can be reduced by reducing the variance of the transmission error. However, the corresponding decrease in the bound will be linear.

Gaussian Approximation Bounds: Tchebycheff’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. The Gaussian distribution is of particular interest, because the distribution of \( \epsilon \) tends to become Gaussian for large \( n \), if the message delays are independent and identically distributed for the duration of a transmission period \( \tau \). A transmission period is the interval of time within a resynchronization interval during which synchronization messages are transmitted.
Property 2: If the delays of synchronization messages are independent and identically distributed during a transmission period, the probability distribution of the normalized transmission error \( Z_n \) defined as follows:

\[
Z_n = \frac{\epsilon - \bar{\epsilon}}{\sqrt{V[\epsilon]}},
\]

approaches the standard normal distribution \( N(0,1) \) in limit, as the number of messages \( n \) increases.

Proof: As can be seen from (12), \( \epsilon \) may be rewritten as follows:

\[
\epsilon = \frac{1}{n} \sum_{i=1}^{n} \frac{\bar{r}t}{2} + \bar{d} - d_i,
\]

and can be viewed as the sample mean of a random sample drawn from a distribution with mean

\[
\frac{\bar{r}t}{2} + \bar{d} - E[d_i] = \bar{\epsilon},
\]

and variance \( \frac{V[d_i]}{n} = V[\epsilon] \). The result follows from a popular version of the central limit theorem, which states that the normalized sample mean of a random sample approaches the standard normal distribution in limit as the number of samples increases. Q.E.D.

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

\[
\left| \frac{G_n(Z_n) - \Phi(Z_n)}{\Phi(Z_n)} \right| < \xi,
\]

where \( \xi > 0 \) can be made arbitrarily small by choosing \( n_g \) large enough. As mentioned in Section IV, \( n_g \), the minimum value of \( n \) for \( N(0,1) \) to approximate \( G_n(Z_n) \) with an accuracy \( \xi \), is referred to as the Gaussian cutoff. If \( d \) itself is 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 has to be determined by experiment. In a simulation study of the relationship between \( n_g \) and \( \xi \), in which a uniform distribution for \( d \) was assumed, it was 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 to approximate the distribution of \( \epsilon \) by a Gaussian distribution with good accuracy, even when the distribution of \( d \) is uniform (i.e., not close to Gaussian). But, as conjectured in Section IV-B, the distribution of \( d \) itself is approximately Gaussian. Hence, a smaller \( n_g \) should suffice.

Error Function Bound:

Lemma 5:

\[
P[|\epsilon| \geq \epsilon_{\max}] \leq \text{erfc} \left( \frac{\Delta \epsilon_{\max}}{\sqrt{2V[\epsilon]}} \right),
\]

if \( \Delta \epsilon_{\max} > 0 \). Here \( \epsilon \) is the clock skew at resynchronization, \( \epsilon_{\max} \) is the specified maximum clock skew at resynchronization, \( \bar{\epsilon} \) is the expected value of \( \epsilon \), \( V[\epsilon] \) is the variance of \( \epsilon \), \( \Delta \epsilon_{\max} = \epsilon_{\max} - |\bar{\epsilon}| \), and \( \text{erfc}(u) \) is the complementary error function. The complementary error function is defined as \( \text{erfc}(u) \triangleq 1 - \text{erf}(u) \), where \( \text{erf}(u) \triangleq \frac{2}{\sqrt{\pi}} \int_{0}^{u} e^{-t^2} dt \).

Proof: For a normally distributed random variable \( X \), with distribution \( N(0,1) \), the probability that \( X \) lies within \( \omega \) of its mean \( (\omega=0) \) is given by the following:

\[
P[|X| \leq \omega] = \int_{-\omega}^{\omega} \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}x^2} dx
\]

\[
= 2 \int_{0}^{\omega} \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}x^2} dx
\]

\[
= 2 \int_{0}^{\omega} \frac{\sqrt{\pi}}{\sqrt{2}} e^{-y^2} dy
\]

\[
= \text{erf} \left( \frac{\omega}{\sqrt{2}} \right).
\]

Therefore,

\[
P[|X| \geq \omega] = 1 - \text{erf} \left( \frac{\omega}{\sqrt{2}} \right) = \text{erfc} \left( \frac{\omega}{\sqrt{2}} \right) \quad (17)
\]

For \( n > n_g \), \( G_n(Z_n) \) (where \( Z_n \) is defined in Property 2 as \( Z_n \triangleq \frac{\epsilon - \bar{\epsilon}}{\sqrt{V[\epsilon]}} \)) can be approximated by \( N(0,1) \). Hence, by (17),

\[
P[|\epsilon| \geq \epsilon_{\max}] = 1 - \text{erf} \left( \frac{\Delta \epsilon_{\max}}{\sqrt{2V[\epsilon]}} \right).
\]

By Lemma 4 and (18), we have the following:

\[
P[|\epsilon| \geq \epsilon_{\max}] \leq P[|\epsilon - \bar{\epsilon}| \geq \Delta \epsilon_{\max}] = \text{erfc} \left( \frac{\Delta \epsilon_{\max}}{\sqrt{2V[\epsilon]}} \right) \quad \text{Q.E.D.}
\]

The upper bound on the probability of invalidity derived above is referred to as the error function bound. The error function bound depends on the variance of the clock skew at resynchronization \( \epsilon \), through the complementary error function. As discussed in the main body of the paper, the complementary error function is extremely sensitive to its argument. Thus, by decreasing the variance of \( \epsilon \) by a small amount, the probability of invalidity can be reduced by orders of magnitude.

Exponential Bound: The following bound, referred to as the exponential bound, shows the exponential nature of the dependence of the probability of invalidity on the variance of the transmission error.

Lemma 6:

\[
P[|\epsilon| \geq \epsilon_{\max}] \leq \exp \left( \frac{-\Delta \epsilon_{\max}^2}{2V[\epsilon]} \right) \quad \text{if } \Delta \epsilon_{\max} > 0,
\]

where \( \epsilon \) is the clock skew at resynchronization, \( \epsilon_{\max} \) is the specified maximum clock skew at resynchronization, \( V[\epsilon] \) is the variance of \( \epsilon \), and \( \Delta \epsilon_{\max} = \epsilon_{\max} - |\bar{\epsilon}| \).
Proof: Let \( \sigma = \sqrt{\epsilon} \). By Property 2, \( Z_n = \frac{\epsilon - \bar{\epsilon}}{\sqrt{\epsilon}} \) is distributed \( N(0, 1) \). By Lemma 4,

\[
\begin{align*}
P[|\epsilon| \geq \epsilon_{\text{max}}] &\leq P[|\epsilon - \bar{\epsilon}| \geq \Delta \epsilon_{\text{max}}] \\
&= P\left[|Z_n| \geq \frac{\Delta \epsilon_{\text{max}}}{\sqrt{\epsilon}}\right] \\
&= \frac{2}{\sqrt{2\pi}} \int_{\Delta \epsilon_{\text{max}} / \sqrt{\epsilon}}^{\infty} e^{-y^2} dy \\
&= \frac{2}{\pi} \int_{\Delta \epsilon_{\text{max}} / \sqrt{\epsilon}}^{\infty} e^{-y^2} dy
\end{align*}
\] (19)

Let

\[
Q(z) \triangleq \int_{z}^{\infty} e^{-y^2} dy
\]

A bound can be derived [19] for \( Q(z) \) as follows. For \( z > 0 \),

\[
Q(z) = \int_{z}^{\infty} e^{-y^2} dy
\]

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

\[
Q(z) < \frac{1}{2} e^{-z^2}
\] (21)

From (19), (20), and (21), it follows that

\[
P[|\epsilon| \geq \epsilon_{\text{max}}] \leq \sqrt{\frac{2}{\pi}} Q\left(\frac{\Delta \epsilon_{\text{max}}}{\sigma \sqrt{\epsilon}}\right) < \sqrt{\frac{2}{\pi}} \frac{e^{-n(\Delta \epsilon_{\text{max}})^2}}{\Delta \epsilon_{\text{max}}^2}.
\]

The result follows by substituting \( \sigma = \sqrt{\epsilon} \). Q.E.D.

C. Analysis of the Number of Messages

In this section, the bound derived in the above section is used to compute the number of synchronization messages required from the service specifications for the algorithm, namely, the maximum clock skew and the probability of invalidity. It is also shown that the probability of invalidity decreases exponentially or better with the number of synchronization messages.

**Theorem 1:** The minimum number of messages required to guarantee a maximum skew at resynchronization of \( \epsilon_{\text{max}} \) with a probability of invalidity of \( p \), is given by \( n_{\text{min}} = \max(n_y, n_e) \), where

\[
n_e = \frac{2\sigma_d^2(\text{erfc}^{-1}(p))^2}{(\Delta \epsilon_{\text{max}})^2},
\] (22)

provided that \( \Delta \epsilon_{\text{max}} > 0 \). Here \( \sigma_d \) is the standard deviation of the message delay, \( n_y \) is the Gaussian cutoff, \( \Delta \epsilon_{\text{max}} = \epsilon_{\text{max}} - |\bar{\epsilon}|, \bar{\epsilon} \) is the expected value of \( \epsilon \), and \( \epsilon \) is the clock skew at resynchronization.

Proof: By Lemma 5,

\[
P[|\epsilon| > \epsilon_{\text{max}}] \leq \text{erfc}\left(\frac{\Delta \epsilon_{\text{max}}}{\sqrt{2\epsilon}}\right).
\]

To guarantee a maximum skew of \( \epsilon_{\text{max}} \) with a probability of invalidity \( p \), we must have \( P[|\epsilon| > \epsilon_{\text{max}}] \leq p \), i.e.,

\[
\text{erfc}\left(\frac{\Delta \epsilon_{\text{max}}}{\sqrt{2\epsilon}}\right) \leq p,
\]

i.e.,

\[
\text{erfc}\left(\frac{\sqrt{n_e \epsilon_{\text{max}}}}{2\sigma_d}\right) \leq p
\]

which when solved for \( n \) yields the \( n_e \) of (22). Since \( n > n_y \) for the Gaussian approximation to be accurate, \( n_{\text{min}} = \max(n_y, n_e) \). Q.E.D.

Note that whenever Property 1 holds, \( \bar{\epsilon} \) can be neglected in comparison to \( \epsilon_{\text{max}} \). In this case, the minimum number of synchronization messages required can be determined in a straightforward manner by replacing \( \Delta \epsilon_{\text{max}} \) in (22) with \( \epsilon_{\text{max}} \).

**Property 3:** For a given \( \epsilon_{\text{max}}(\epsilon_{\text{max}} > |\bar{\epsilon}|) \), 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 \( \alpha \) is a nonincreasing function of \( n \), and \( \beta > 0 \) is independent of \( n \), i.e.,

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

whenever Property 1 holds true.

Proof: By Lemma 6,

\[
P[|\epsilon| > \epsilon_{\text{max}}] \leq \sqrt{\frac{2}{\pi}} Q\left(\frac{\Delta \epsilon_{\text{max}}}{\sigma \sqrt{\epsilon}}\right) < \sqrt{\frac{2}{\pi}} \frac{e^{-n(\epsilon_{\text{max}})^2}}{\epsilon_{\text{max}}^2}.
\]

The result follows by substituting \( \sigma = \sqrt{\epsilon} \). Q.E.D.

In the above equation, \( \bar{\epsilon} \) was neglected in comparison to \( \epsilon_{\text{max}} \), using Property 1. Q.E.D.

Note that the coefficient \( \alpha \) is a decreasing function of \( n \left(\frac{\text{erfc}^{-1}(p)}{\sqrt{\epsilon}}\right) \). Thus, the coefficient is also responsible, although weakly, for the decrease in the probability of invalidity with increasing \( n \).

D. Analysis of the Skew Between Any Two Clocks

**Property 4:** Algorithm PCS can guarantee (with an associated probability of invalidity) an upper bound, \( \tau_{\text{max}} = 2(\epsilon_{\text{max}} + (R_{\text{synch}} + d_{\text{max}} - d_{\text{min}})\rho) \), on the skew between any two clocks in the system.

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

$$\gamma_{m,s}^{\text{max}} = \epsilon_{\text{max}} + \rho (R_{\text{synch}} + d_{\text{max}} - d_{\text{min}}).$$

The maximum skew $\gamma_{\text{max}}$ between any two clocks in the system is given by $2\gamma_{m,s}^{\text{max}}$:

$$\gamma_{\text{max}} = 2\gamma_{m,s}^{\text{max}} = 2(\epsilon_{\text{max}} + \rho (R_{\text{synch}} + d_{\text{max}} - d_{\text{min}}))$$

Q.E.D.

The above result assumes that the distribution of the 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.

B. Notation

- $A \triangleq B$ $A$ is defined as $B$.
- $A \approx B$ $A$ is of the form $B$.
- $d$ end-to-end message delay.
- $d_{\text{e}}$ end-to-end delay of the $i$th synchronization message.
- $d_{\text{max}}$ maximum message delay.
- $d_{\text{min}}$ minimum message delay.
- $d(n)$ average $d_i$ over $n$ synchronization messages.
- $\delta_0$ clock skew between $S$ and $M$ at the start of transmission of the first message.
- $\delta_i$ clock skew between $S$ and $M$ when the $i$th message is received by $S$.
- $\delta_{i+1}^d$ increase in the clock skew between $S$ and $M$ between the times of transmission of the first and $i$th messages.
- $\delta_{i+1}^d$ increase in the clock skew between the message delay $d_i$ of the $i$th message.
- $\delta(n)$ average skew between the clocks of $S$ and $M$ over $n$ message receipts.
- $\delta_n$ average increase in skew prior to transmission.
- $\Delta d$ deviation of $d(n)$ from $d$.
- $\Delta T(n)$ deviation of $T(n)$ from $T$.
- $\epsilon$ base of the natural logarithms.
- $\text{erf}(x)$ error function.
- $\text{erfc}(x)$ complementary error function, $1 - \text{erf}(x)$.
- $E[X]$ expected value of random variable $X$.
- $\epsilon_{\text{trans}}$ transmission (delay estimation) error.
- $\epsilon_{\text{trans}}$ expected value of transmission error.
- $\epsilon_{\text{max}}$ desired maximum transmission error.
- $G_n$ probability distribution of the transmission error.
- $\gamma_{\text{max}}$ guaranteed maximum skew between any two clocks in the system.
- $\gamma_{m,s}^{\text{max}}$ maximum clock skew between the master and a slave clock.
- $M$ sender, master.
- $n$ number of synchronization messages.
- $n_g$ Gaussian cutoff.
- $n_e$ number of extra messages needed to compensate for delayed messages.
- $N$ number of nodes in the system.
- $N(m, \kappa^2)$ standard normal distribution with mean $m$ and variance $\kappa^2$.
- $p$ probability of invalidity.
- $P[\text{event}]$ probability of the event occurring.
- $\Phi(x)$ Gaussian probability distribution function corresponding to $N(\bar{x}, \sigma^2)$.
- $Q(x)$ defined in Lemma 6.
- $r_i$ round-trip delay of a synchronization message.
- $\bar{r}(n)$ average round trip delay (over $n$ messages).
- $R_i$ time of receipt of the $i$th message according to the receiver’s clock.
- $\bar{R}(n)$ average of the message receipt times.
- $R_{\text{synch}}$ resynchronization interval.
- $\rho$ maximum relative clock drift rate.
- $\hat{\rho}$ actual relative clock drift rate.
- $S$ receiver, slave.
- $\sigma_d$ standard deviation of the message delay.
- $T_{\text{act}}$ estimate of time on slave’s clock.
- $T_{\text{est}}$ estimate of time on master’s clock.
- $\bar{T}_i$ time stamped on the $i$th message.
- $\bar{T}(n)$ average of the times stamped on the $n$ messages received.
- $\tau$ transmission period.
- $U$ half time-out period in algorithm CR1.
- $V[X]$ variance of random variable $X$.
- $W$ separation between successive messages.
- $\bar{X}$ expected value of the random variable $X$.
- $\xi$ error bound.

ACKNOWLEDGMENT

I would like to thank K. Ramanritham, J. Stankovic, F. Cristian, T. Philips (IBM), and B.R. Badrinath for their comments, suggestions, and encouragement. I am thankful to the referees for their valuable suggestions and corrections. I am also thankful to the Associate Editor for his various suggestions for improving the presentation of the material.

REFERENCES


