Implementing a Probabilistic Clock Synchronization Algorithm

Article *in* Real-Time Systems · August 1999

DOI: 10.1023/A:1007900902186 · Source: CiteSeer

Augusto Ciuffoletti
Università di Pisa

Real Time Systems, 13, 25–46 (1997)

Implementing a Probabilistic Clock Synchronization Algorithm [*]

GIANLUIGI ALARI gigi@info.ucl.ac.be
Departement d'Ingegnerie Informatique, Université Catholique de Louvain
Place St. Barbe, 2, B-1348 Louvain La Neuve, Belgique

AUGUSTO CIUFFOLETTI augusto@di.unipi.it
Dipartimento di Informatica, Università degli Studi di Pisa
Corso Italia n. 40, 56125 Pisa - Italy

Abstract. In this paper we present a new probabilistic clock synchronization algorithm, its prototype implementation and experimental results. The algorithm follows the client-server programming paradigm and is designed to work in a departmental environment with few servers and a number of clients connected through an arbitrary network topology.

At the core of the algorithm is a remote clock reading method that mitigates the negative effects of message delay uncertainty. The implementation proves the effectiveness of this approach and corroborates the theoretical speculations.

Keywords: clock synchronization, probabilistic algorithms, distributed systems, distributed algorithms, local area networks, TCP/IP

1. Introduction

In a distributed system there are strong reasons to keep the clocks of the units as synchronized as possible since sharply synchronized clocks can ease the development of several distributed applications and services such as real time control, performance evaluation, distributed simulations, transaction processing, data recovery, atomic broadcast, group membership and many others (Lamport, 1984).

However, the task of keeping the clocks synchronized is a hard one, since:

- *variable clock drift*
  A hardware clock doesn't measure the time at the desired speed of $1 sec/sec$ but, even if fault-free and initially synchronized with a standard time reference, it tends to drift away from it. The drift rate itself varies during the lifetime of the clock, and according with external conditions in an unpredictable way;

- *message delay uncertainty*
  The messages that convey timing information are exposed to UN-foreseeable delays. This delay uncertainty is due to a number of different factors such as network load, queuing delays, I/O throughput, protocol stack implementation and others.

[*] This work has been carried out with the financial contribution of the Europen Commission in the framework of the Human Capital Mobility Program, CABERNET Excellence Network under contract number ERBCHBGCT 93 04 58

Therefore, even perfectly synchronized clocks tend to diverge and cannot be exactly resynchronized.

The primary goal of a distributed synchronization service is to keep all non-faulty clocks synchronized within δ time units; δ is called the *synchronization precision*, and is a key parameter to evaluate the performance of a synchronization service (Lamport, 1984).

In the literature, we can identify two different approaches to implement a distributed clock synchronization. An *internal* clock synchronization algorithm keeps each node's clock synchronized within δ time units while an *external* synchronization algorithm keeps each node's clock synchronized within δ time units from an external time reference such as *UTC* (Universal Time Coordinates) or *GPS* (Global Positioning System).

A number of deterministic clock synchronization algorithms have been published (Cristian, 1986, Kopetz and Ochsenreiter, 1987, Lamport, 1985, Srikanth and Toueg, 1987). Most of them are structured around periodic rounds of broadcast communication and address fault tolerance aspects; for a survey see (Schneider, 1986).

An important result (Lundelius and Lynch, 1984) fixes the upper bound $\delta = (max - min)/(1 - 1/N)$ on the synchronization precision for deterministic algorithms executing in a distributed system with N nodes. Here min and max are the minimum and maximum network delay and the quantity $(max - min)$ gives an indication of the network delay uncertainty.

According to these results, in networks with unbounded message delay, even if we assume that all system nodes are fault-free, it is impossible to achieve clock synchronization through a deterministic algorithm.

In order to overcome this limitation, probabilistic clock synchronization algorithms have been proposed (Arvnid, 1994, Cristian, 1989, Olson and Shin, 1991). These algorithms are probabilistic since there is a non null probability p_{fail} that the precision of the system clocks will exceed δ time units but p_{fail} can be determined or at least bounded (Arvnid, 1994).

Relaxing the deterministic constraint, probabilistic algorithms are able to keep a given synchronization precision even in systems where there is no upper bound on message delays, with a probability of failure that is comparable with the reliability figures of the processing units (Arvnid, 1994, Cristian, 1989).

This work presents a probabilistic clock synchronization algorithm which adopts the remote clock reading method of (Alari and Ciuffoletti, 1992) and a client-server approach similar to the one in (Cristian, 1989). Unlike the one in (Alari and Ciuffoletti, 1992) this algorithm tolerates transient faults provided that they do not occur during synchronization periods. Transient faults during the synchronization periods prevent the correct behavior of the algorithm but only during those periods; therefore correct synchronization will be re-established during the first fault-free synchronization period. The effectiveness of the algorithm is proved by some results obtained through a prototype implementation.

The rest of the paper is organized as follows. In Section 2 we introduce our model of the system. In Section 3 we present and analyze the remote reading method adopted by our algorithm that is presented in Section 4; Section 5 contains the description of the implementation and the results of the experiments and Section 6 concludes the paper with some final consideration.

2. The Distributed Time Service

A distributed system consists of a set of nodes, interconnected by a communication network, exchanging information only by means of messages. Each node N_i is equipped with a hardware clock HC_i of quartz accuracy.

A hardware clock implements a monotonic increasing function of time $HC(t)$ and is characterized by ρ, the maximum clock drift rate from real time. We define a predicate $Correct$ that defines the requirements a hardware clock must comply with over the time interval $[t_1, t_2]$:

$$Correct(HC, [t_1, t_2]) \overset{def}{\Leftrightarrow} (1-\rho) \leq \frac{(HC(t_2) - HC(t_1))}{(t_2 - t_1)} \leq (1+\rho)$$

Hardware clocks can only be read but not modified by software modules; we introduce *logical clocks* that implement adjustable clocks. A logical clock $LC(t)$ is a function of the underlying hardware clock $HC(t)$ (and hence of time t), and is implemented by a simple data structure. For example in (Cristian, 1989) a logical clock LC is a pair of real values (A, B) such that $LC(t) = (1 + A) * HC(t) + B$: the rate of LC changes according with the value of A while B is an offset value; the logical clock $LC(t) = (0, 0)$ is equivalent to $HC(t)$.

As already stated in the previous section, the goal of a distributed time service is to keep the values of all logical clocks $LC(t)$ synchronized within δ time units:

$$Correct(LC_i, t) \overset{def}{\Leftrightarrow} \forall j, Correct(LC_j, t), -\delta \leq LC_i(t) - LC_j(t) \leq \delta$$

With respect to the time service, we will partition the system nodes into two disjoint sets: *Time Service Servers* and *Time Service Clients*. Every node i in *Time Clients (Servers)* runs a *Client (Server) Synchronization Task* P_i. The set of all the synchronization tasks cooperate to implement a distributed time service by exchanging timing information over the network and acting on their logical clocks. Any other task may access this service through a set of time service interface calls.

Thereafter, since clock drift ρ is much smaller than one, we will neglect terms of the order of ρ^2 or higher in expressions whose value is comparable with the unit or larger. Such approximation in polynomial expansions justifies $(1+\rho)^{-1} = (1-\rho)$, and $(1-\rho)^{-1} = (1+\rho)$).

3. Reading a remote clock

In order to achieve clock synchronization we have to provide each node with the capability of remotely reading the clock of another node of the system; the more precise this method is, the better the algorithm will perform. In this section we briefly introduce the remote reading method of (Alari and Ciuffoletti, 1992), and we analyze its efficiency.

To read the remote clock of node S, the synchronization task P_C running on client node C sends at real time t_0 a message to the synchronization task P_S on server node S. Let the message be delivered to P_S at real time t_r; the receiver immediately sends a reply message containing the value $LC_S(t_r)$. At real time $t_1 > t_0$, P_C receives the server reply (see Figure 1).

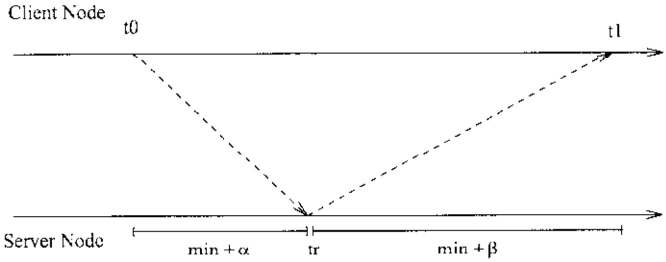

Figure 1. Remote clock reading rule

Let $T_0 = LC_C(t_0), T_1 = LC_C(t_1), 2D = T_1 - T_0$, and let min be the minimum point to point message delay between P_S and P_C the following holds (Alari and Ciuffoletti, 1992):

LEMMA 1 *If* $|LC_C(t_0) - LC_S(t_0)| \leq \delta_0$ *and the hardware clocks of the nodes C and S are correct then the value* $LC_S(t_1)$ *of the server's clock when the client receives the reply containing the clock value* $LC_S(t_r)$ *is within the interval:*

$$[LC_S(t_r) + (min + \beta_{min})(1 - \rho), LC_S(t_r) + (min + \beta_{MAX})(1 + \rho)] \tag{1}$$

where

$$\begin{aligned} \beta_{min} &= max(2D(1-\rho) - 2min - \alpha_{MAX}, 0) & (2)\\ \beta_{MAX} &= 2D(1+\rho) - 2min - \alpha_{min} & (3)\\ \alpha_{min} &= max((LC_S(t_r) - (T_0 + \delta_0))(1-\rho) - min, 0) & (4)\\ \alpha_{MAX} &= (LC_S(t_r) - (T_0 - \delta_0))(1+\rho) - min & (5) \end{aligned}$$

The proof follows directly from the one in (Cristian, 1989) for a similar statement.

The following theorem is a direct consequence of the above lemma and indicates that, when the client receives the reply, it can compute an approximation of the clock of the server and bound the precision of the approximation.

THEOREM 1 *Under the hypotheses and the notation of Lemma 1, at time* t_1, *when* P_C *receives the reply from* P_S, P_C *can compute an approximation* $LC_S^C(t_1)$ *of the clock of the server at that time, and bound the difference between the approximation and the real value of the clock of the server with the upper bound* ψ *computed as follows:*

$$\begin{aligned} \psi &= \frac{(\beta_{MAX} - \beta_{min}) + \rho * (\beta_{MAX} - \beta_{min})}{2} & (6)\\ LC_S^C(t_1) &= LC_S(t_r) + min + \beta_{min} + \psi & (7) \end{aligned}$$

Here ψ is nothing else than the half of the length of 1 while $LC_S^C(t_1)$ is the midpoint of the same interval.

In order to evaluate the effectiveness of our method with respect to the one proposed in (Cristian, 1989) we compare Equations 6 and 7 to the corresponding ones derived according to the rules in (Cristian, 1989):

$$\begin{aligned} \psi' &= D(1+2\rho) - min && (8) \\ LC_S'^C(t_1) &= LC_S(t_r) + D(1+2\rho) - min * \rho && (9) \end{aligned}$$

Thereafter, for the sake of clarity, we ignore also terms of the form $D*\rho$: since the figures used in the formulas have a magnitude comparable with that of D, a term $D*\rho$ can be neglected.

The following statement says that the clock precision obtained using our clock reading rule compares favorably with that obtained using Cristian's rule:

COROLLARY 1 *Using the notation introduced in Theorem 1, if*

$$LC_S(t_r) \notin [T_0 + D - |D - (min + \delta_0)|, T_0 + D + |D - (min + \delta_0)|]$$

it holds that

$$\psi < D - min$$

and

$$\psi = min(\frac{\alpha_{MAX}}{2}, \frac{\beta_{MAX}}{2})$$

The previous corollary identifies the cases when our reading rule performs better than that defined in (Cristian, 1989), and gives the corresponding error estimates. To complete the comparison we need a result stating that our reading is not worse than the other in any other case:

COROLLARY 2 *Using the notation introduced in Theorem 1, if*

$$LC_S(t_r) \in [T_0 + D - |D - (min + \delta_0)|, T_0 + D + |D - (min + \delta_0)|]$$

it holds that

$$\psi = min(D - min, \delta_0)$$

The above Corollary states that, when the hypothesis is satisfied, the reading method either does not improve the present knowledge of server's clock, in that case $\psi = \delta_0$, or returns an approximation of that value which is the same that would have been obtained with the application of Cristian's rule, and in that case $\psi = D - min$. In either case, the reading accuracy is at least $D - min$.

From the above two corollaries we can derive an important conclusion stated by the following:

COROLLARY 3 *If a client reads a server clock following the reading rule of Theorem 1 the error in approximating the server clock at real time t_1 is not worse than the current approximation of the clock precision* ($\psi \leq \delta_0$).

The reader understands that the role played by the reading rule in a clock synchronization algorithm is that of obtaining a new clock value whose distance from the current server's clock is smaller than that of the current clock value; the result of a clock reading operation that does not satisfy this basic requirement, expressed by the inequality $\psi < \delta_0$, is completely useless. Therefore we say that when $\psi < \delta_0$ the clock reading attempt has been *useful*. Otherwise, and from Corollary 3 we know that in that case $\psi = \delta_0$, the clock reading attempt has been *useless*, and the result can be simply discarded.

The above discussion indicates the algorithm that should be followed in order to compute the value of ψ:

```
function read-rule($\delta_0$: time; $lc$: logical-clock): time;
begin
$T_0$ := read-clock($lc$);
read-remote-clock(ServerIP,$t_r$);
$D$ := $\frac{(\text{read-clock}(lc) - T_0)}{2}$;
$\alpha_{max} := t_r - T_0 + \delta_0 - min$;
$\alpha_{min} := max((t_r - T_0 - \delta_0 - min), 0)$;
$\beta_{max} := 2 * D - 2 * min - \alpha_{min}$;
$\beta_{min} := max((2 * D - 2 * min - \alpha_{MAX}), 0)$;
$x$ := $|D - (min + \delta_0)|$;
$l_1$ := $T_0 + D - x$;
$l_2$ := $T_0 + D + x$;
if (($t_r < l_1$) or ($t_r > l_2$)) then
        # useful clock reading, improved precision
        $\psi := min(\frac{\beta_{max}}{2}, \frac{\alpha_{max}}{2})$;
else
        if $((D - min) <= \delta_0)$ then
                # useful clock reading, precision as in (Cristian, 1989)
                $\psi$ := $D - min$;
        else
                # useless clock reading
                $\psi$ := $\delta_0$;
set-clock($lc$, $t_r + min + \beta_{min} + \psi$);
return($\psi$);
```

In order to design a probabilistic clock synchronization algorithm we need to know the distribution of the reading precision. To this purpose, we introduce the stochastic variable $\overline{\psi}$ and analyze the distribution

$$P(\overline{\psi} \leq \psi_0)$$

as a result of the distribution of other stochastic variables that may be reasonably assumed as known. We introduce two auxiliary variables that have a relevant intuitive meaning and whose distribution can be easily obtained by sampling or analytically. Let δ_{real} the real difference between the clock of the server and that of the client

$\alpha_s = \alpha - \delta_{real}$ the apparent delay of the request message; it is the time that the server could estimate if the client's message were time-stamped with client's clock value;

$\beta_c = \beta + \delta_{real}$ the apparent delay of the reply message, as might be observed by the client;

Using these variables, the Corollary 1 can be rewritten as follows

COROLLARY 4 *Let $x_1 = min(\alpha_s, \beta_c)$, and $x_2 = max(\alpha_s, \beta_c)$. If $x_1 < \delta_0$ then $\psi = x_1 + min(\delta_0, x_2)$ otherwise the reading operation is useless.*

We recall that we are now evaluating an analytical expression for the probability of having a reading precision better than a certain threshold: none of the variables mentioned in the previous expression is in fact computed by the client or the server of the clock synchronization service.

The probability that the client reaches rapport with the server is given by the following compound probability, that uses the stochastic variables $\overline{x_1}$ and $\overline{x_2}$ associated with x_1 and x_2:

THEOREM 2 *Using the notation introduced in Corollary 4 we have that, for any value $\psi_0 < \delta_0$, the probability distribution of the reading precision ψ is given by:*

$$P(\overline{\psi} \leq \psi_0) = p(\overline{x_1} < \delta_0) * p(\overline{x_1} < 2 * \psi_0 - min(\delta_0, \overline{x_2}))$$

which depends on the distribution of the communication delays x_1 and x_2, and on the precision of the current clock, δ_0. In the sequel we will indicate the density and distribution functions of the reading precision as p_{δ_0} and P_{δ_0} respectively, to show their dependency from the value of the current clock precision.

To show the impact of Corollary 1, we have simulated an environment similar to that illustrated in (Cristian, 1989), and compared the results of the two reading rules by plotting the distributions of the reading precisions obtained using Cristian's reading rule, and the reading rule presented in this paper. Figure 2 shows the distribution of the offset delays α and β, that have been generated using a $Weibull(1.3, 0.2)$ distribution, that model communication delays. In Figure 3 the reading precision distributions are plotted for $\delta_0 = 0.25\, msecs$, showing that our reading rule improves significantly the performance of Cristian's one.

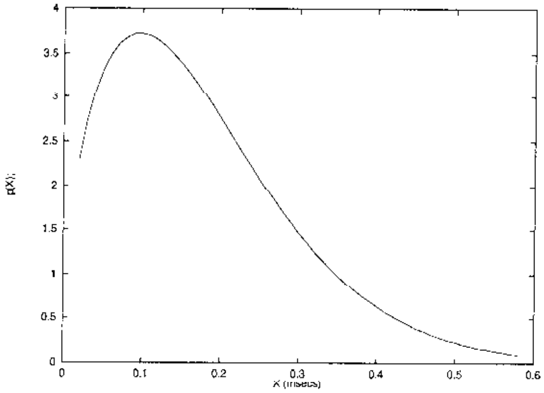

Figure 2. Distribution of the offset of the communication delay from min. X axis: communication delay exceeding min $(msecs)$. Y axis: probability density of the event X

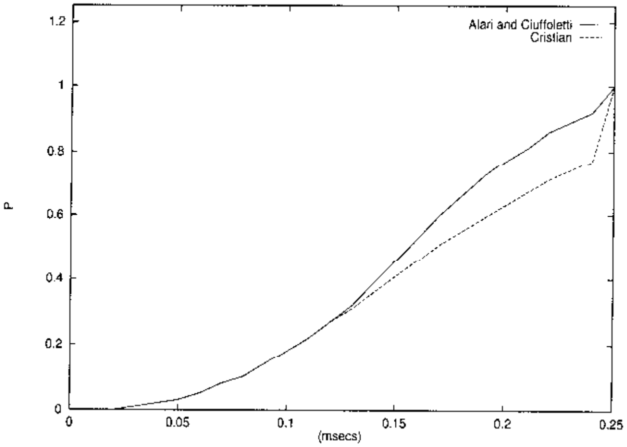

Figure 3. Comparison between success frequencies of the reading rules when $\delta_0 = 0.25\, msecs$ (each point 100000 experiments). X axis: ψ_0 $(msecs)$. Y axis: probability that $\psi < \psi_0$

4. The probabilistic algorithm

The basic idea of the probabilistic clock synchronization algorithm, as introduced in (Cristian, 1989), is that each client repeatedly reads the server's clock, trying to *reach contact* with it by reading its value with a precision better than a predefined ϵ; the protocol used to read the remote clock has been illustrated in the previous section. The client is not allowed to fail un-definitely: after a certain timeout elapses from the last contact, the client's clock is considered faulty and the client is considered unable to carry out time critical tasks. The timeout depends on the minimal synchronization precision that allows an acceptable coordination of the time critical system activities.

In order to succeed in reaching contact with the server with a given probability, usually higher than that associated to the single attempt, the clock synchronization algorithm will start a series of at most K clock reading attempts hopefully concluded by a contact before the timeout expires. The designer is in charge of deciding how many attempts have to be available before that time, in order to guarantee that the client will be able to synchronize with the server before the last attempt with a probability higher than that required by the reliability associated with the time synchronization service.

Note that, since each attempt is considered as an independent experiment, two successive attempts should be separated by a period of time W sufficient to ensure their statistic independence on the possible causes preventing a rapport such as network load peaks, system performance degradations and many other transient events: therefore, if we envision at most K attempts, the clock synchronization algorithm will start operating $K * W$ time units before the timeout expires. Precise details on the timing of the algorithm can be found in (Cristian, 1989).

Let us analyze how the clock synchronization algorithm operates in our case. The clock synchronization starts by creating a temporary logical clock TLC_C identical to LC_C; then it sets $\delta_0 = \infty$. Until the next rapport, all the time information needed to execute the protocol at the client side is measured according to TLC_C. At each successful attempt of reading the server's clock (i.e., each time the reading error ψ is lower than δ_0) both TLC_C and δ_0 are updated by the following rules:

$$\delta_0 := \psi + 2\rho * W \tag{10}$$

$$TLC_C(t_1) := t_r + min + \psi \tag{11}$$

otherwise only δ_0 will be updated by:

$$\delta_0 := \delta_0 + 2\rho * W \tag{12}$$

The rules stated above are valid under the assumption that both agents have identical drift rates (ρ). The generalization to the case where they are characterized by different drift rates is straightforward.

At rapport (i.e. whenever $\psi \leq \epsilon$), P_C adjusts the value of LC_C using TLC_C, and computes the time when the next clock synchronization operation will be invoked as a function of ρ, K, W, ψ and of the overall synchronization precision δ. On the other hand, if after K consecutive reading attempts P_C does not reach contact with the server, then it signals the exceptional event to upper layer software levels.

The following program unfolds the previous description of client's activity:

```
clock-copy($LC$,$TLC$);
# $\delta_0$ is initially undefined
$\delta_0$ := $\infty$;
forever
    begin
    $count$ := 0;
    $\delta_0$ := read-rule($\delta_0$, $TLC$);
    while $(count < K)$ and $(\delta_0 > \epsilon)$ do
        begin
        sleep($W$);
        $\delta_0$ := $\delta_0 + 2 * \rho * W$;
        $count$ := $count + 1$;
        $\delta_0$ := read-rule($\delta_0$, $TLC$);
        end;
    if ($count = K$) and ($\delta_0 > \epsilon$) then
        raise-exception(synch-lost);
    else
        begin
        clock-copy($TLC$,$LC$);
        sleep(delay-to-next-attempt);
        $\delta_0$ := $\delta_0 + 2 * \rho *$ delay-to-next-attempt;
        end
    end;
```

As mentioned above, a key role is played by K: in order to evaluate its value as a function of the communication delay distribution, of the required precision and reliability, we introduce a Markov process that describes the precision of TLC_C after the n-th clock reading attempt:

$$p(\psi_n) = p(\psi_n|\psi_1)$$

and, since at the first attempt we have no knowledge of the server's clock, that attempt is surely useful and returns a :

$$p(\psi_1) = p(D - min)$$

that, in our example, corresponds to the Weibull density plotted in Figure 2. Then, using the Chapman-Kolgomorov theorem:

$$p(\psi_n) = \int_{-\infty}^{\infty} p(\psi_n|\psi_{n-1}) * p(\psi_{n-1}|\psi_1)\, d\psi_{n-1}$$

Since the value of ψ at the i-th attempt is used, incremented by $2 * \rho * W$, as the value of δ_0 in the next attempt, we can compute the transition probability $p(\psi_n|\psi_{n-1})$ as:

$$p(\psi_n|\psi_{n-1}) = p_{\delta_0}(\psi)$$

where $\delta_0 = \psi_{n-1} + 2 * \rho * W$ and $\psi = \psi_n$: the probability p_{δ_0} has been discussed in the previous section, and its distribution has been plotted in Figure 3. Therefore we can write:

$$p(\psi_n) = \int_{-\infty}^{\infty} p_{(\psi_{n-1}+2*\rho*W)}(\psi_n) * p(\psi_{n-1}) d\psi_{n-1}$$

which can be computed numerically by recursion on n.

Once computed the Markov process, the value of K will be chosen so that ($\overline{\psi}_K$ indicates the stochastic variable associated with ψ_K):

$$P(\overline{\psi}_K < \epsilon) > 1 - p_{fail}$$

4.1. Comparison with other related algorithms

The main difference with respect to (Cristian, 1989) is in the new clock reading rule: we have proved that the improvement introduced by our rule may be significant and favorably reflects on the overhead introduced by the clock synchronization algorithm.

However, we cannot assess that this result is optimal: as formally defined in (Patt-Shamir and Rajsbaum, 1994) and (Herzberg, Attya, and Rajsbaum, 1993), in order to be optimal, a reading rule should address the whole past history (the *view* of the client at t_1, following the terminology of the authors): but in both Cristian's and our reading clock rule only the information conveyed by the last round trip is used.

With respect to Cristian's reading rule, the one presented in this paper exploits further information, available at client side with no additional cost: the estimated accuracy of the clock of the client. The reader may observe that this figure makes available to the client some information dependent on past synchronization attempts: in essence the present estimated accuracy contains *part* of the information contained in the view of the client at time t_1. This formally justifies the fact that we can improve over the precision of Cristian's reading rule, which is based on a more restricted amount of information.

On the other hand, the protocol presented in this paper shares several features with Cristian's one: it does not require an upper bound for the communication delay, and tolerates transient faults occurring in the lapse between two successive synchronizations.

This last feature is the most significant improvement with respect to our previous proposal, presented in (Alari and Ciuffoletti, 1992), and is obtained by assuming that the algorithm restarts each time with a “zero knowledge” hypothesis. This option introduces a kind of self-stabilizing behavior (Dijkstra, 1974): the client gets finally synchronized regardless of the initial value of the clock LC_C. Therefore the client is able to resume a correct synchronization as soon as its functionality is restored, without an explicit recovery activity. This characteristic greatly simplifies the design of complex systems, improving their reliability.

A recent direction in research (see (Dolev and Welch, 1995)) indicates that self-stabilization can be fruitfully applied to synchronization algorithms. The interested reader can find the original definition of self-stabilization in (Dijkstra, 1974), and an algorithm for *external* clock synchronization that exhibits some self-stabilizing aspects in (Ciuffoletti, 1994).

Finally, another improvement with respect to (Alari and Ciuffoletti, 1992) is that now δ_0 is not any more related to the overall service precision δ widening the applicability of the new algorithm to services where δ is big compared to message delays.

5. Implementation

5.1. Software and Hardware environment

In this section we present the result of two experiments running on the network infrastructure consisting of two $10Mb/s$ thin Ethernet segments interconnected by a routing host and depicted in Fig. 4; hosts are *Sun* workstation whose clock resolution is of the order of microseconds. In the first experiment the time clients and the time server were hosted on the same Ethernet segment while in the second one they run on hosts located on two different segments.

Client and server software is written in the C programming language (Kernighan and Ritchie, 1988) and the communication protocol used to send messages over the network is UDP, the connection-less service offered by the TCP/IP protocol suite (Comer, 1988), (Stevens, 1990). We decided to implement the service over a connection-less protocol for two main reasons:

- connection-less protocols are lighter than connection oriented ones, so they introduce less overhead and a small message latency
- loss of messages does not cause problems because it can be modeled as an extremely long network delay; this delay will be reflected in an unsuccessful remote clock reading attempt.

We programmed two time clients: one implementing Cristian's algorithm and the other the algorithm described in Section 4. Time clients were run on the same host and started at the same time so that the performance measurements of our time service and of Cristian's one were taken under approximately the same network and host load; this implies that the differences in the obtained results are essentially due to the algorithms implemented by the two time clients.

Experiments run during normal network activity and all refer to 1000 clock reading attempts. Once fixed the value of δ to two and three milliseconds for the single Ethernet segment and for the routed environment respectively, we then sampled network delays and appropriately set the remaining algorithm parameters to have an overall service failure

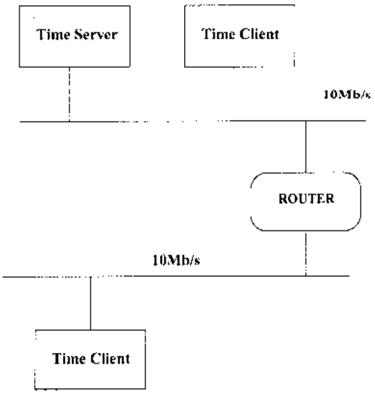

Figure 4. Experiments setup

probability p_{fail} of the order of 10^{-6}. The maximum reading error was therefore set so that a single reading attempt with $\delta_0 = \infty$ (i.e. a reading attempt such that $\psi = D - min$) had probability of reaching contact $p_{rc} = 0.35$ and the maximum number of consecutive reading attempts was $K = 30$. The time between two consecutive attempts was set to $W = 2sec$ and the client drift rate with respect to the server clock was found to be bounded by $\rho = 3E - 5sec/sec$.

With this setting, in the worst case assumption that we never improve with respect to the algorithm of (Cristian, 1989), we get $p_{fail} = (1 - p_{rc})^K < 2.5E-6$ that respects the above probability requirements. In reality, following the arguments of Section 3 and Section 4, we proved that under realistic assumptions on message delays the probability of service disruption of our algorithm is less then the one of Cristian's.

5.2. Results

Tables 1 and 2 show reading error figures with respect to message delays experienced during the experiments, the number of rapport obtained, mean DNA (Delay to the Next Attempt timeout) and a measure of the communication cost in terms of messages per second obtained using the formula $\frac{NumOfMessages}{Duration}$ where $NumOfMessages$ is the total number of messages exchanged between the client and the server processes and $Duration$ is the duration in seconds of the experiment [1].

The cost figure quantifies the benefits obtained by the application of the protocol presented in this paper: due to the increased clock reading precision (see columns 1-2 in the tables), we obtain more successful attempts and a longer period between two successive synchronization periods (see columns 3-4 in the tables) and consequently less message exchanges are performed during the same time period.

It is also important to analyze the different behavior of the two reading methods with respect to another key protocol parameter, min. In both the experiments the value assigned

Table 1. Delays and reading errors (in μsec), rapports, DNA (in sec) and communication cost (in msg/sec) for the single subnet experiment; $min = 800\mu$sec and $\delta = 2000\mu$sec

Algorithm	mean delay	mean error	rapports	mean DNA	cost
Cristian	1594.1	794.1	252	26.70	0.246
Alari Ciuffoletti	1481.6	526.7	403	31.71	0.143

Table 2. Delays and reading errors (in μsec), rapports, DNA (in sec) and communication cost (in msg/sec) for the two subnet experiment; $min = 1100\mu$sec and $\delta = 3000\mu$sec

Algorithm	mean delay	mean error	rapports	mean DNA	cost
Cristian	1849.4	749.4	269	28.03	0.222
Alari Ciuffoletti	1940.9	643.2	361	29.79	0.166

to min was about 100 μsec smaller with respect to the real minimum delay experienced during the experiments. Again, our reading method was able to overcome this fact because the discrepancy between the experienced minimum delay value and min was *absorbed* by β_{min} and β_{max} approximations. On the contrary, with the clock reading method of (Cristian, 1989), the difference between min and the effective point to point client server minimum delay is directly reflected in the reading error (see Equation 8).

The numerical results of our experiments presented in the above tables is visually rendered by the following Figures plotting the frequencies of two relevant parameters common to both the probabilistic algorithms: the DNA period and the number of attempts between two consecutive contacts.

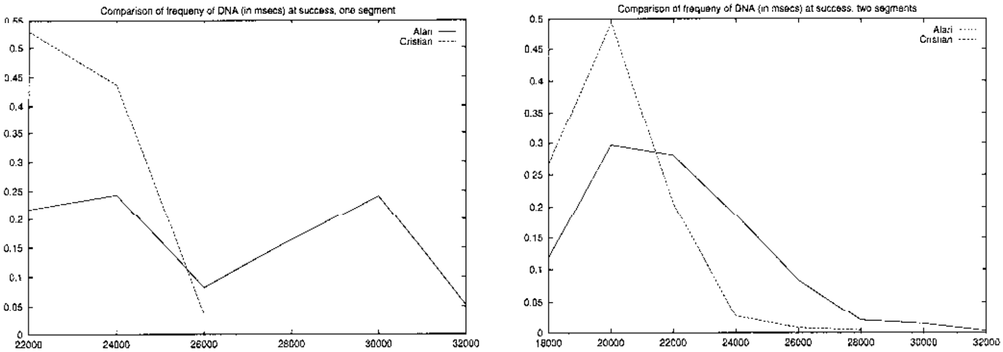

Figure 5. DNA (Delay to Next Attempt) frequencies

In Figure 5 the frequencies of DNA are plotted for the case of the single Ethernet and two Ethernet segments: the improvement of our variant is significant since the values of the DNA period are higher in our case reflecting the fact that the mean remote clock reading error is smaller.

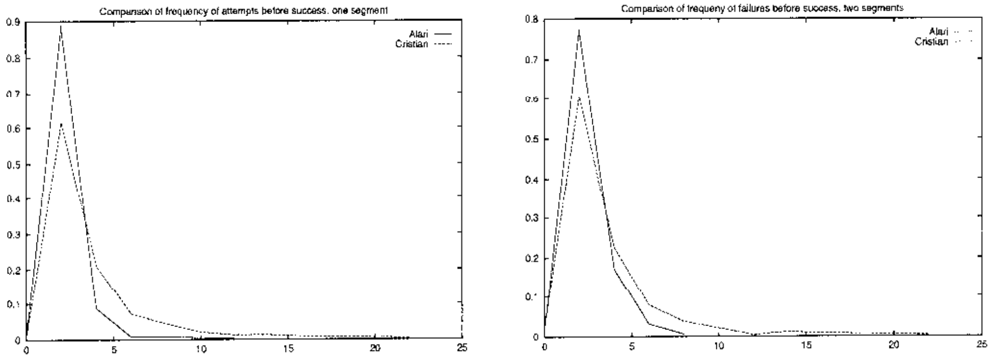

Figure 6. Frequency of attempts to reach contact with the server

In Figure 6 the frequencies of the number of attempts before reaching contact are plotted for the case of the single Ethernet and two Ethernet segments. Also for this significant parameter the new algorithm presents advantages with respect to the algorithm of (Cristian, 1989) since it reaches higher frequencies for smaller values of the number of attempts with respect to the algorithm by Cristian. This indicates that, as studied in Section 4, our time client has a higher probability to have contact with the time server host compared to the one of the time client implementing the probabilistic algorithm described in (Cristian, 1989). So, our protocol permits a time client to obtain more contacts with the time server process and to get a minimal remote clock reading error, usually smaller than the reading error obtained with the probabilistic algorithm of Cristian.

6. Summary

We designed and implemented a probabilistic clock synchronization algorithm. The strength of our implementation resides in the remote reading method adopted; this method allows a sensible reduction of the remote clock reading error by mitigating the negative effect of network delay uncertainty. Experimental results confirm the effectiveness of the algorithm which improves the results in term of communication cost with respect to other probabilistic time services. An implementation of the protocol presented in this paper is available as public domain-software at the following URL `http://www.di.unipi.it/~augusto.html`

Appendix

Proofs

A.1. Proof of Lemma 1

From Figure 1 the following equations are satisfied:

$$\begin{aligned} 2d &= 2min + \alpha + \beta && \text{(A.1)} \\ min + \alpha &= t_r - t_0 && \text{(A.2)} \end{aligned}$$

Since by hypothesis the two clocks at t_0 does not differ by more than δ_0 time units we have:

$$LC_S(t_0) \in [T_0 - \delta_0, T_0 + \delta_0] \qquad \text{(A.3)}$$

From (A.2), (A.3) and the fact that the server drift is bounded by ρ, we get a first approximation of the quantity $min + \alpha$:

$$(LC_S(t_r) - (T_0 + \delta_0))(1 - \rho) \le min + \alpha \le (LC_S(t_r) - (T_0 - \delta_0))(1 + \rho) \text{(A.4)}$$

The interval (A.4) induces the following expressions for α_{min} and α_{MAX}:

$$\begin{aligned} \alpha_{min} &= max((LC_S(t_r) - (T_0 + \delta_0))(1 - \rho) - min, 0) && \text{(A.5)} \\ \alpha_{MAX} &= (LC_S(t_r) - (T_0 - \delta_0))(1 + \rho) - min && \text{(A.6)} \end{aligned}$$

and combining (A.5) and (A.6) with (A.1) we obtain:

$$max(2d - 2min - \alpha_{MAX} \le \beta \le 2d - 2min - \alpha_{min}) \qquad \text{(A.7)}$$

Since, by hypothesis, $2D(1 - \rho) \le 2d \le 2D(1 + \rho)$, from (A.7) we obtain:

$$\begin{aligned} \beta_{min} &= max(2D(1 - \rho) - 2min - \alpha_{MAX}, 0) \\ \beta_{MAX} &= 2D(1 + \rho) - 2min - \alpha_{min} \end{aligned}$$

Finally, from the above bounds on β and the equation $t_1 = t_r + min + \beta$ the following bounds on $LC_S(t_1)$ can be easily obtained:

$$LC_S(t_1) \in [LC_S(t_r) + (min + \beta_{min})(1 - \rho), LC_S(t_r) + (min + \beta_{MAX})(1 + \rho)]$$

that concludes the proof of Lemma 1. □

A.2. Proof of Corollary 1

The corollary states that the value of $LC_S(t_r)$ must be outside an interval centered on the mid point of the interval (as measured by the client) elapsing from the request of clock value and the receipt of the reply. We can turn this requirement in the following inequalities:

$$LC_S(t_r) < T_0 + D - |D - (min + \delta_0)| \tag{A.8}$$
$$LC_S(t_r) > T_0 + D + |D - (min + \delta_0)| \tag{A.9}$$

If either inequality holds, then $\psi < D - min$. From Inequality A.8 we derive

$$LC_S(t_r) - T_0 - D < -(D - (min + \delta_0)) < T_0 + D - LC_S(t_r)$$
$$\begin{cases} LC_S(t_r) - T_0 - \delta_0 - min < 0 \\ 2D - 2min - LC_s(t_r) + T_0 - \delta_0 + min > 0 \end{cases}$$
$$\begin{cases} \alpha_{min} = 0 \\ \alpha_{MAX} < 2D - 2min \end{cases}$$

We have therefore that

$$\begin{aligned} \psi &= \frac{\beta_{MAX} - \beta_{min}}{2} \\ &= \frac{2D - 2min - \alpha_{min} - (2D - 2min - \alpha_{MAX})}{2} \\ &= \frac{\alpha_{MAX}}{2} \\ &< D - min \end{aligned}$$

Since, according with $\alpha_{min} = 0$, $\beta_{MAX} = 2D - 2min$, we also have that

$$\frac{\alpha_{MAX}}{2} < \frac{\beta_{MAX}}{2}$$

which concludes the first part of the proof.

From the Inequality A.9 we derive

$$T_0 + D - LC_S(t_r) < D - (min + \delta_0) < LC_S(t_r) - T_0 - D$$
$$\begin{cases} 2D - 2min - LC_s(t_r) + T_0 - \delta_0 + min < 0 \\ LC_S(t_r) - T_0 - \delta_0 - min > 0 \end{cases}$$
$$\begin{cases} \beta_{min} = 0 \\ \alpha_{min} > 0 \end{cases}$$

We have therefore that

$$\begin{aligned} \psi &= \frac{(\beta_{MAX} - \beta_{min})}{2} \\ &= \frac{\beta_{MAX}}{2} \\ &= \frac{2D - 2min - \alpha_{min}}{2} \\ &< D - min \end{aligned}$$

Since, according with $\beta_{min} = 0$, $\alpha_{MAX} > 2D - 2min$, we also have that:

$$\frac{\beta_{MAX}}{2} < \frac{\alpha_{MAX}}{2} \tag{A.10}$$

that concludes the proof of Corollary 1. □

A.3. Proof of Corollary 2

We want to prove that $\psi = min(D - min, \delta_0)$ when

$$-|(D - (min + \delta_0))| \leq LC_S(t_r) - (T_0 + D) \leq |D - (min + \delta_0)|$$

The inequality splits into two different cases that have to be considered separately. If

$$D - (min + \delta_0) \leq 0$$

which is the case of a short round-trip it follows that

$$\begin{array}{l} D - (min + \delta_0) \leq LC_S(t_r) - (T_0 + D) \leq -(D - (min + \delta_0)) \\ \begin{cases} 2D - 2min - LC_s(t_r) + T_0 - \delta_0 + min \leq 0 \\ LC_S(t_r) - T_0 - \delta_0 - min \leq 0 \end{cases} \\ \begin{cases} \beta_{min} = 0 \\ \alpha_{min} = 0 \end{cases} \end{array}$$

We have therefore that

$$\begin{aligned} \psi &= \frac{(\beta_{MAX} - \beta_{min})}{2} \\ &= \frac{\beta_{MAX}}{2} \\ &= \frac{2D - 2min - \alpha_{min}}{2} \\ &= D - min \end{aligned}$$

that concludes the first part of the proof.

On the other hand, if

$$D - (min + \delta_0) > 0$$

which is the case of a long round-trip it follows that

$$\begin{array}{l} \delta_0 - (D - min) \leq LC_S(t_r) - (T_0 + D) \leq (D - min) - \delta_0 \\ \begin{cases} LC_S(t_r) - T_0 - \delta_0 - min \geq 0 \\ 2D - 2min - LC_s(t_r) + T_0 - \delta_0 + min \geq 0 \end{cases} \\ \begin{cases} \alpha_{min} \geq 0 \\ \alpha_{MAX} \leq 2D - 2min \end{cases} \end{array}$$

We have therefore that

$$\begin{aligned}\psi &= \frac{(\beta_{MAX} - \beta_{min})}{2}\\ &= \frac{2D - 2min - \alpha_{min} - (2D - 2min - \alpha_{MAX})}{2}\\ &= \frac{\alpha_{MAX} - \alpha_{min}}{2}\\ &= \delta_0\end{aligned}$$

and this concludes the second part of the proof. Summarizing, under the hypotheses of the corollary, we have that

$$\begin{aligned}&\delta_0 \geq D - min \;\Rightarrow\; \psi = D - min\\ &\delta_0 < D - min \;\Rightarrow\; \psi = \delta_0\end{aligned}$$

which is the same that $\psi = min(D - min, \delta_0)$. □

A.4. Proof of Corollary 3

We prove the corollary by dividing the domain of the possible reading scenarios into two sub-domains; exactly the ones corresponding to the Corollaries 1 and 2. For each of them we show that is impossible for ψ to respect the error equation of the appropriated corollary and at the same time $\psi > \delta_0$.

For the domain of Corollary 1 the error equation is $\psi = min(\frac{\alpha_{MAX}}{2}, \psi = \frac{\beta_{MAX}}{2})$. Let us first suppose that $\psi = \frac{\alpha_{MAX}}{2}$ and that, following the proof of Corollary 1, $LC_S(t_r) < T_0 + D - |D - (min + \delta_0)|$. Let $\psi > \delta_0$ then

$$\begin{aligned}&\psi > \delta_0\\ \Rightarrow\; &\frac{\alpha_{MAX}}{2} > \delta_0\\ \Rightarrow\; &\frac{LC_S(t_r) - T_0 + \delta_0 - min}{2} > \delta_0\\ \Rightarrow\; &LC_S(t_r) > T_0 + min + \delta_0\end{aligned} \tag{A.11}$$

We now analyze both the cases of $LC_S(t_r) < T_0 + D - |D - (min + \delta_0)|$:

i) $D - (min + \delta_0) < 0$

This means that $LC_S(t_r) < T_0 + 2D - min - \delta_0$ and by Equation A.11 we have $D - (min + \delta_0) > 0$ that is incompatible with the case at hand;

ii) $D - (min + \delta_0) > 0$

This means that $LC_S(t_r) < T_0 + min + \delta_0$ and it is in contrast with equation (A.11).

The same simple reasoning shows that $\psi = \frac{\beta_{MAX}}{2} > \delta_0$ is impossible if $LC_S(t_r) > T_0 + D + |D - (min + \delta_0)|$ concluding the proof for the sub-domain of Corollary 1.

Concerning the sub-domain of Corollary 2 the error equation is $\psi = min(D - min, \delta_0)$ that is manifestly incompatible with $\psi > \delta_0$; this concludes the proof of Corollary 3. □

A.5. Proof of Corollary 4

Note that the following inequalities hold:

$$\begin{array}{l} D - (min + \delta_0) = \frac{\alpha_s + \beta_c}{2} - \delta_0 \\ LC_s(t_r) - T_0 - D = \frac{\alpha_s - \beta_c}{2} \end{array}$$

Using the above, the Inequality A.8, that refers to the case where $\psi = \frac{\alpha_{MAX}}{2}$ is rewritten as:

$$\alpha_s < \delta_0 < \beta_c$$

In order to have a non empty interval, we also need $\beta_c > \alpha_s$. The Inequality A.9, that refers to the case where $\psi = \frac{\beta_{MAX}}{2}$ is rewritten as:

$$\beta_c < \delta_0 < \alpha_s$$

In order to have a non empty interval, we also need $\beta_c < \alpha_s$.

Concerning the Corollary 2, the only case where the clock reading is significant is when $D - (min + \delta_0) < 0$. In terms of the new variables:

$$\delta_0 > \frac{\alpha_s + \beta_c}{2} \tag{A.12}$$

In that case we have that the reading precision is $D - min$ when,

$$(D - min) - \delta_0 \leq LC_S(t_r) - (T_0 + D) \leq -((D - min) - \delta_0)$$

that can be rewritten as:

$$\delta_0 > max(\beta + \delta_{real}, \alpha - \delta_{real})$$

Which also implies the validity of Equation A.12 Summarizing, we conclude that:

- if $\alpha_s < \beta_c$ then
  - if $\delta_0 \in [\alpha_s, \beta_c]$ then $\psi = \frac{\alpha_s + \delta_0}{2}$
  - if $\delta_0 > \beta_c$ then $\psi = \frac{\alpha_s + \beta_c}{2}$

- if $\alpha_s > \beta_c$ then
  - if $\delta_0 \in [\beta_c, \alpha_s]$ then $\psi = \frac{\beta_c - \delta_0}{2}$
  - if $\delta_0 > \alpha_s$ then $\psi = \frac{\beta_c + \alpha_s}{2}$

Now let $x_1 = min(\alpha_s, \beta_c)$ $x_2 = max(\alpha_s, \beta_c)$ we can rewrite the above assert as:

- if $\delta_0 \in [x_1, x_2]$ then $\psi = \frac{x_1 + \delta_0}{2}$
- if $\delta_0 > x_2$ then $\psi = \frac{x_1 + x_2}{2}$

and if $\delta_0 > x_1$ then

- if $\delta_0 < x_2$ then $\psi = \frac{x_1 + \delta_0}{2}$
- if $\delta_0 > x_2$ then $\psi = \frac{x_1 + x_2}{2}$

and finally if $\delta_0 > x_1$ then $\psi = \frac{x_1 + min(\delta_0, x_2)}{2}$

The proof that when $x_1 \geq \delta_0$ the reading attempt is useless is straightforward. □

A.6. Proof of the theorems

The proof of Theorem 1 and Theorem 2 is immediate from Lemma 1 and Corollary 4 respectively. □

Notes

1. $NumOfMessages = 2 \times NumOfAttempts$ (i.e. the number of messages per attempt times the total number of attempts of the experiment) and $Duration = (NumOfAttempts - NumOfDNA) \times W + NumOfRapports \times MeanDNA$ (i.e. the time spent in unsuccessful attempts plus the number of successful attempts times the mean value of DNA experienced during the experiment).

References

Alari, B. and Ciuffoletti, A. Improving the probabilistic clock synchronization algorithm. In *Proceedings of the 18th Euromicro Conference*, Paris, 1992.

Arvind, K. Probabilistic clock synchronization in distributed systems. *IEEE Transaction on Parallel and Distributed Systems*, 5(5):474–487, 1994.

Ciuffoletti, A. Using simple diffusion to synchronize the clocks in a distributed system. In *Proc. 14th International Conference on Distributed Computing Systems*, pages 484–491, Poznan (Poland), June 1994.

Comer, D. *Internetworking with TCP/IP: principles, protocols and architectures*. Prentice Hall, Englewood Cliffs, NJ, 1988.

Cristian, F. Clock synchronization in the presence of omission and performance faults, and processor joins. In *Proceedings of the 16th Fault Tolerant Computing Symposium*, Vienna, 1986.

Cristian, F. Probabilistic clock synchronization. *Distributed Computing*, (3):146–158, 1989.

Dijkstra, E.W. Self-stabilizing systems in spite of distributed control. *Communications of the ACM*, 17(11):643–644, 1974.

Dolev, S. and Welch, J.L. Self-stabilizing clock synchronization in the presence of byzantyne faults. Second Workshop on Self-Stabilizing Systems, University of Nevada, Las Vegas. Technical Report TR 86-735, 1995.

Herzberg, A., Attya, H. and Rajsbaum, S. Optimal clock synchronization under different delay assumptions. In *12th ACM Symposium on Principles of Distributed Computing*, Ithaca, NY, 1993.

Kernighan, B.W. and Ritchie, D.M. *The C programming language, second edition*. Prentice Hall, Englewood Cliffs, NJ, 1988.

Kopetz, H. and Ochsenreiter, W. Clock synchronization in distributed real-time systems. *IEEE Transactions on Computers*, c-36(8):933–940, 1987.

Lamport, L. Using time instead of timeout for fault-tolerant distributed systems. *ACM Journal of Programming Languages and Systems*, 6(2):254–280, 1984.

Lamport, L. Synchronizing clocks in the presence of faults. *Journal of the ACM*, 32(1):52–78, 1985.

Lundelius, J. and Lynch, N.A. An upper and lower bound for clock synchronization. *Information Control*, 62:190–204, 1984.

Olson, A. and Shin, K. Probabilistic clock synchronization in large distributed systems. In *Proceedings of the 11th International Conference on Distributed Computing Systems*, Arlington, Texas, 1991.

Patt-Shamir, B. and Rajsbaum, S. A theory of clock synchronization. In *ACM Symposium on Theory of Computing*, Montreal, Quebec, Canada, 1994.

Schneider, F.B. A paradigm for reliable clock synchronization. Technical Report TR 86-735, Cornell University, Ithaca, NY, 1986.

Srikanth, T.K. and Toueg, S. Optimal clock synchronization. *Journal of the ACM*, 34(3):626–645, 1987.

Stevens, W.R. *UNIX network programming*. Prentice Hall, Englewood Cliffs, NJ., 1990.