Time Synchronization Using the Internet

Judah Levine

***Abstract*—This paper discusses a new algorithm for synchronizing the clocks of networked computers using messages transmitted over the network itself. The design is based on a statistical model of the clock and the network, and uses this model to define the parameters of a frequency-lock loop which is used to discipline the local oscillator. The design was tested by synchronizing a standard workstation to a time server located 1200 km away; the time offset between the clock synchronized in this way and UTC is 2 ms rms. This analysis also can be used to design algorithms that provide lower accuracy at lower cost.**

I. Introduction

This paper discusses synchronizing the clocks of computers using messages transmitted over a packet network such as the Internet. Such algorithms are useful because the network infrastructure is often already installed and available so that it can be used for time synchronization with little or no additional cost.

The current algorithm, which we call "interlock," is a generalization of the "lockclock" algorithm which was discussed in a previous paper [1]. It is similar to "lockclock" in that it uses the rms clock error as a measure of performance and the average time between calibrations as a measure of cost. Either algorithm can be used to study the trade-off between cost and performance (as these quantities have been defined) or to compare the performance of different methods whose costs are the same.

The performance of "lockclock" depends on calibration messages transmitted over relatively stable dial-up voice-grade telephone lines from a telephone time server (such as the Automated Computer Time Service operated by NIST). The current algorithm can function with messages transmitted over packet networks (such as the Internet) where the transmission delay is less stable and predictable. The two algorithms can provide comparable performance in spite of this, although "interlock" may require more frequent calibrations than "lockclock" for the same level of performance.

The "interlock" procedure uses messages transmitted in network time protocol (NTP) format [2] and can be used in a heterogeneous environment with other NTP-synchronized systems. It does not depend on the details of the message format, however, and other message formats could be substituted with only minor modifications.

Manuscript received June 11, 1997; accepted August 28, 1997. This work was supported in part by the National Science Foundation through grants NCR-9115055 and NCR-9416663 to the University of Colorado.

The author is with JILA, NIST, and University of Colorado, Boulder, CO 80309 (e-mail: jlevine@utcnist.colorado.edu).

A number of other performance metrics have been discussed in the literature; some try to provide a deterministic, guaranteed upper bound to the error in the time of the local clock [3], [4], and others use delay and dispersion data from several servers to select the synchronization source [2]. These methods have clear advantages in some cases—using several servers is probably the surest way of detecting a glitch and identifying its origin, but it will be more expensive than using data from a single server. If glitches are relatively infrequent events, using multiple servers improves the worst-case performance but may have little impact on the average. As will be discussed below, deterministic upper bounds are often not as iron-clad as they first appear to be and may not be achievable at all in some configurations.

A complete procedure based on either the "lockclock" or "interlock" algorithms is comprised of two parts: a component that estimates the performance of the local clock based on external calibration information and a component that uses this information to adjust the local clock. The algorithms characterize the local clock in terms of its time difference and rate offset with respect to the external reference; both of these parameters are then used to adjust the local clock.

Time differences larger than about 0.5 s (which are normally encountered only during a cold-start or in the case of an error) are corrected in a single step by changing the value in the clock register; smaller ones are realized by temporarily adjusting the effective frequency of the oscillator by about $\pm 1\%$ of its nominal value until the time adjustment has been amortized. The estimated rate offset modifies the effective frequency of the clock oscillator by changing the size of the software tick—the value added to the clock register on each hardware interrupt. A correction for a rate offset, therefore, is equivalent to a periodic series of small time adjustments. The following discussion is limited to how the parameters of the local clock are estimated when the Internet is used as the channel to the calibration source. These parameters are then used to steer the local clock as I described in [1].

The design of the algorithm is based on a model with three components: a clock model, a model of the measurement process, and a model of the delay in the network. Each of these models is further subdivided into deterministic and stochastic components. (A deterministic parameter has a well-defined value that may evolve slowly with time; a stochastic parameter can only be specified using a statistical measure such as a variance or a spectral density.)

II. The Clock Model

The clock in a computer usually consists of two parts: an oscillator that generates periodic interrupts and a software driver that counts these interrupts in a register. Simple systems simply increment the register by one on each interrupt, while more sophisticated systems add the nominal period of the oscillator to the register on each "tick." In either case, the register measures elapsed time from some system-defined origin. In some systems, the hardware also provides a way of interpolating between clock ticks to improve the resolution (but not the accuracy) of the system time. It is also possible to interpolate between clock ticks using software timing loops, although this method is less accurate because it is affected by variations in the system load.

In almost all systems the oscillator hardware is free-running and is not under program control. Software processes of the type to be discussed can adjust the time of the clock by changing the value in the system register; in some systems it is also possible to adjust the effective rate of the clock by changing the value that is added to the register on each increment. In the following discussion, a reference to a computer clock always refers to the clock register maintained by the system, possibly combined with a method of interpolating between ticks using either the system hardware or a software timing loop. Furthermore, all time messages and clock comparisons are made using UTC; conversions to and from the local time zone (including a correction for daylight saving time, if necessary) are made by other processes and are outside of the scope of this discussion.

A clock with two deterministic parameters is characterized using a time offset, which specifies the difference between its time at some epoch and UTC, and a rate offset, which specifies how this time offset evolves. If x_k and y_k specify the time offset and rate offset at some epoch t_k, then these parameters can be used to predict the time at the next epoch, t_{k+1} using:

$$\hat{x}_{k+1} = x_k + y_k \Delta t, \tag{1}$$

where

$$\Delta t = t_{k+1} - t_k. \tag{2}$$

Many quartz-crystal oscillators (which are the type almost always used in computer clocks) also have significant frequency aging—that is the frequency changes nearly linearly with time. However, adding aging to (1):

$$\hat{x}_{k+1} = x_k + y_k \Delta t + 0.5 d_k (\Delta t)^2, \tag{3}$$

where

$$y_{k+1} = y_k + d_k \Delta t \tag{4}$$

often does not improve the accuracy of the model in predicting x_{k+1}. The problem is that the aging parameter itself must be estimated from the measured values of x, and this estimate usually is very uncertain because of the presence of large stochastic frequency fluctuations that mask the aging. For this reason, the deterministic frequency aging and stochastic frequency modulation are estimated together, and the combination is treated as producing a stochastic variation in y.

In addition to these deterministic parameters, the frequency of the oscillator fluctuates stochastically. These fluctuations can be characterized as a "white" process for relatively short averaging times, but the spectrum of these fluctuations exhibits a divergence as $1/f$ (or faster) at lower Fourier frequencies (i.e., longer averaging times).

Unfortunately, stochastic fluctuations in the oscillator frequency can be observed only through their effect on the time. Because the clock integrates these stochastic frequency fluctuations, the spectral density of the resulting time fluctuations is almost never white. In other words, the time fluctuations produced by frequency noise can never be characterized by a mean and a standard deviation; although these parameters always will exist in a formal sense, their values will not necessarily correspond to intuitive expectations. In particular, when the performance is dominated by frequency noise (even *white* frequency noise) the rms prediction error of the time-difference will not be improved by averaging repeated observations of x_k.

III. The Measurement Process

The measurement process involves comparing the time of the clock with the time transmitted over the network. There is a jitter associated with this measurement that is comprised of hardware and software components. Both of these components have mean values that are characteristic of the system. The overall mean delay is usually on the order of microseconds in a well-designed system—a value that is small compared to the error budget of the synchronization process. (Strictly speaking, there are also delays in the clock hardware itself; any fluctuations in this delay appear as phase noise.) In addition, there are fluctuations in the measurement delay that are due to variations in the software latency, the instantaneous load, and similar factors. Assume that the factors that drive these fluctuations vary rapidly with time, so that the variations in these delays are largely uncorrelated with each other even for measurements made only a few seconds apart. This jitter about the mean, therefore, is a random variable with a reasonably well-defined mean and standard deviation; this jitter is called white phase noise, by analogy with the analogous fluctuations in hardware clocks. In fact, the jitter will probably have a distribution that is not completely symmetric about its mean, system-specific value: the distribution is likely to have a relatively sharp cut-off at values smaller than the mean, and larger values are likely to be more numerous than the symmetrical distribution function would predict because of short-term increases in the system load or the software latency. In fact, this problem tends to be more serious in the delay through the network;

a strategy for addressing this problem is discussed below.

These fluctuations exact a price from any client process that uses the clock to time-tag an event. Because a time-tag involves a single reading of the clock, a knowledge of the distribution of an ensemble of such measurements cannot be exploited. However, this knowledge does have an important consequence for prediction applications: it suggests that the average of N rapid-fire measurements will have a standard deviation that is smaller than that of a single measurement by a factor of $1/\sqrt{(N-1)}$. This improvement depends, of course, on the assumption that the measurements are dominated by white phase noise—in other words, the deterministic parameters of the other components of the model must not change during the course of the N measurements, and the contributions of any non-white noise source must be small.

IV. The Network Delay

The network delay enters directly into the measurement of the time difference. It is usually estimated from the measurements themselves using the usual round-trip method. The client machine requests a time-packet at time t_1; the request arrives at the time server at time t_2; the server responds at time t_3 and the response is received at time t_4. The round-trip delay due to the network path is:

$$\Delta = (t_4 - t_1) - (t_3 - t_2). \tag{5}$$

The first bracket measures the total time that has elapsed from when the request was sent until the reply was received as measured by the clock in the client. The second is the processing delay in the server as measured by its clock. If the one-way outbound delay is d, the time difference between the client and the server is:

$$x = (t_1 + d) - t_2. \tag{6}$$

If the path delay is symmetrical, then $d = \Delta/2$, and

$$x = \frac{(t_1 - t_2) + (t_4 - t_3)}{2}. \tag{7}$$

If the path delay is not exactly symmetrical, this estimate will be wrong by an amount proportional to the asymmetry. If the actual outbound delay is given by $d = k\Delta$, where $0 < k < 1$, then the estimate above is wrong by:

$$\varepsilon = (k - 0.5)\Delta, \tag{8}$$

which depends both on the path delay and on its asymmetry. Paths with short delays are clearly advantageous.

Networks are usually configured so that the inbound and outbound paths are symmetrical, although there is no physical reason why this must be true. There is no way of detecting such a static asymmetry using only timing information transmitted over the network itself. Therefore, it can be assumed that either the static configuration is symmetrical or else that any static asymmetry is calibrated using some external means. In either case, it is unlikely that ε has a normal distribution about this mean value so that

$$\sigma^2 \sim \langle \varepsilon^2 \rangle - \langle \varepsilon \rangle^2 \tag{9}$$

will not have any simple interpretation analogous to the variance or standard deviation of a traditional distribution. The problem is not in Δ, because we measure it for each transmission—it is in the degree of asymmetry, specified by $(k - 0.5)$, and the fact that this asymmetry varies from one transmission to the next one. Another way of describing the problem is that the distribution function for the asymmetry cannot be deduced from the magnitude of the variance.

The situation is more favorable if we consider the distribution of the mean of a group of closely spaced calibration messages. If the spacing between the messages in the group is close enough so that the deterministic parameters of the clock model are essentially constant, while at the same time being far enough apart so that the variations in the network delays between consecutive messages are independent of each other, then the central limit theorem guarantees that the distribution of these group-means will have a normal distribution, independent of the distribution of the individual messages [5]. A typical computer clock will take at least several seconds to gain or lose 1 ms, while a typical packet network will have processed tens of thousands of messages in the same time interval. The requirement, therefore, is easily satisfied, even if the consecutive requests are separated by only 100 ms. Furthermore, the Lindeberg condition [6], which, loosely speaking, requires that the time-differences be finite and bounded, is automatically satisfied by the time-out constraints that are incorporated into the design of all networks.

Once we have computed the mean of the group of measurements, it is possible to look for gross outliers within the group by examining the distribution of the individual measurements about the mean. There is no robust interpretation of the standard deviation in this case, because the distribution of the measurements is not a normal one. If the true asymmetry varies randomly between the two asymptotic values $k \approx 0$ and $k \approx 1$, then the mean value is not a bad estimate of the actual time difference, but the situation is less clear when there are a few outliers, with all of the other values clustering about a single value. We generally choose to reject the outliers in this case if they differ from the mean by more than 3 standard deviations of the points that remain after they have been rejected. (Alternatively, we have used the difference between the outliers and the median in the same manner, because the median is less sensitive than the mean to the presence of outliers.) We reject the entire group of measurements if more than 5% of the measurements are rejected using this procedure—either our notion of the standard deviation is too optimistic or the asymmetry is too variable to make a robust estimate of the time difference.

V. Deterministic Guarantees

If the clock on our client system must be synchronized to a server with a maximum time offset of E, and if we can find a server such that $\varepsilon < E$ (at least for some reasonable fraction of the messages), then we can be certain that our requirement has been met by simply using any one of those messages to set our clock. If we can find a server such that this requirement can only be satisfied by averaging N messages, then we can assert a somewhat weaker guarantee that is now limited by the validity of the assumption that the averaging is meaningful [3], [4].

This strategy may not be usable in general. There may be no server satisfying this requirement for a given client, and providing enough servers to do so for every potential client may be prohibitively expensive, especially as the Internet becomes more crowded and its delays become more variable and less predictable.

The real limitation of this method is that the guarantee is only valid at the instant of synchronization. Even assuming that we estimate and remove the deterministic frequency offset of the clock, the guarantee issued at the instant of synchronization becomes weaker with time because of stochastic frequency noise. The Allan deviation can estimate the rms magnitude of these frequency fluctuations, but this estimate is only statistical and does not come with any guarantees whatsoever about the maximum frequency offset and its associated time error.

We can reduce the time dispersion of this frequency noise by more frequent calibrations. Eventually, we will reach a limit given by ε—the noise in the measurement process itself. Effectively, we are not using the local clock at all in this limit—the price for removing the time dispersion due to its frequency noise is to remove its frequency stability as well.

VI. Separation of the Variance

Any method of clock synchronization usually has only one type of observation to work with: the series x_k, giving the time differences at consecutive epochs between the clock we are trying to control and some distant server. It is very important that we separate the contributions to the variance of this time series arising from the different components which we identified above. The reason is that we must not adjust the clock because of noise in the measurement process that did not arise from the clock in the first place. While both (1) and elementary considerations might suggest that, if the time of the clock is wrong, then its frequency must be wrong as well, this is not true in the environment being considered. Both the network and the measurement process make contributions to the time difference and its variance. In other words, it is possible to observe a time error that is not due to the frequency offset of the clock—indeed it may have nothing to do with the clock at all, and correcting the clock for this error (either in time or in frequency) will simply make matters worse. (This conclusion is true no matter how we choose to correct the clock.)

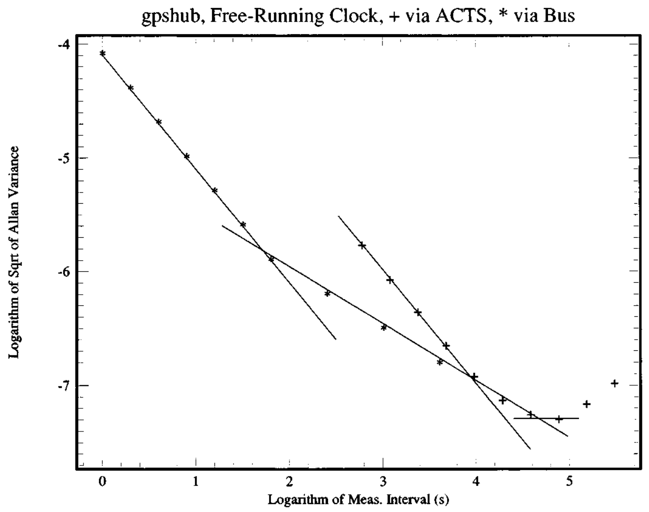


Fig. 1. The Allan deviation of a computer clock measured using the ACTS dial-up system (+) and via a direct connection to the computer bus (*). The straight lines have slopes of −1, −0.5, and 0 to illustrate the various noise types. The relationship between slope and noise type is explained in [1] and [4].

The Allan variance [7], usually denoted by $\sigma_y^2(\tau)$, is a time-domain analysis tool which is very useful in characterizing the spectral density of a time series. This characterization is very important because there are statistically optimum measurement strategies for each type of noise, and knowing the noise type, therefore, is crucial to designing an optimum synchronization procedure. In addition, the spectral density of the noise is often an important indication of its source (see [1] for more details).

The power of the Allan variance is illustrated by the measurements shown in Fig. 1. We have monitored the time of a computer named *gpshub* using two techniques: the "*" show the square root of the Allan variance (often called the Allan deviation) when the time of the clock is compared to the time of a cesium frequency standard using an interface connected directly to the computer bus. The "+" data shows the Allan deviation for the same clock when the time differences are measured using periodic dial-up connections to the NIST Automated Computer Time Service. (The lines on the figure are to assist in the discussion and are not otherwise significant.) In both cases the computer clock is not adjusted. We also did not use any of the outlier-rejection schemes discussed above.

For both measurement techniques, the initial slope is −1, showing that both are limited by white and/or flicker phase noise at short times. The magnitudes are quite different in the two cases. The directly connected hardware device has a near white phase noise level of about 80 μs, and the link to the ACTS system has a noise level of about 1 ms.[1] I am not looking at the oscillator in either case. This

[1]Here and in what follows, I use $\sigma_y(\tau)\tau$ to estimate the phase noise corresponding to an Allan deviation whose magnitude is $\sigma_y(\tau)$. If the spectrum is pure white phase noise, this estimate is too big; a more exact estimate would be $\sigma_y(\tau)\tau/\sqrt{3}$. My estimate is more likely to be appropriate for real-world systems which often have contributions from both white and flicker processes.

is measurement noise; it arises from the jitter in the measurement process and has nothing to do with the clock. The measurements made using the ACTS system have additional jitter because of the hardware and software that are required to receive data using a modem and a serial line, and the directly connected device is limited primarily by the much smaller latency in processing an interrupt request to read a hardware device on the bus.

Because this is near white measurement noise, it would be a mistake to use these data to adjust the clock; as we mentioned above, this is time jitter with no corresponding frequency variations. The uncertainty of the time difference can be decreased by averaging consecutive readings because we are sampling a near white random process. The limit to this improvement comes at about 100 s for the measurements made via the bus device and at about 3000 s for the ACTS comparisons. The improvement in either case would be proportional to the square root of the number of measurements in the average, but this improvement is only available to a process that can benefit from an average time offset. Furthermore the cost of adding more data to the average increases linearly with the number of points so that the cost/benefit ratio becomes increasingly unfavorable as the uncertainty is decreased.

The Allan variance is not sensitive to deterministic rate offsets, because it is computed using the second-difference of the time-difference measurements. However, a rate offset between the client and the server will result in a deterministic trend in the time-difference measurements. This trend will introduce a bias into the averaging procedure if it is not removed beforehand. Because the frequency offset between the client and the server may not be known very accurately, the averaging procedure may need to be limited in practice to intervals where the effect of its uncertainty on the time predictions is not significant.

Even if these measurements are used to predict the future performance of the clock, the uncertainty of a single time-tag will remain unchanged at the value specified by the white phase noise level of the measurement process. A prediction procedure cannot improve on this limit because the fact that the slope of the Allan deviation as a function of averaging time on a log-log plot is -1 implies that $\sigma_y(\tau)\tau$ is a constant. This is the essence of a process with a normal distribution: the previous data are of no help in predicting the magnitude of the next observation.

Adjustments to the clock made using measurements acquired in the near white phase-noise domain will degrade the stability of the clock on the average because these adjustments convert measurement noise into frequency noise. There are only two strategies that can improve matters in this domain: either the measurement noise of a single time-tag must be decreased or the process that uses the time-tags must be designed so that it can benefit from averaging a number of them.

In both data sets, the slope changes to -0.5 at longer averaging times showing a transition to white frequency noise. The time differences can no longer be characterized as a random variable with a white spectrum. Now it is the frequency that has this distribution. We can continue to improve our knowledge by averaging, but we must now average the frequency (i.e., the first difference of the time measurements) rather than the time measurements themselves.

We have made a transition from a domain in which measurement noise dominates the spectral density to one where clock frequency noise plays the major role. The transitions occur at different values for the two measurement schemes, but the two data sets lie on almost exactly the same line in this regime. In fact, the two sets of points become essentially indistinguishable for measurement intervals longer than about 4000 s, and only one data set is shown after that point for clarity. This is to be expected. Once we are looking at the frequency stability of the local oscillator, the method that we use to evaluate it must not matter once the noise of the method falls below the inherent noise of the oscillator hardware itself.

Another way of describing this is that the cesium reference and the bus interface improved the variance only when the averaging time between measurements was short enough that it is measurement noise that counts. The performance of the clock for averaging times longer than about 4000 s is not improved at all because it is dominated by the frequency noise of the oscillator itself—the spectral density of the calibration process has been pretty much "forgotten." Conversely, using a reference whose measurement noise was larger than that of the ACTS system would not degrade the long-period performance of the computer clock until the contribution of the phase-noise to the variance was larger than that of the oscillator frequency noise for all averaging times.

If we operate our synchronization loop in the white-frequency domain, then the optimum strategy would be to acquire x_k data as fast as possible, average these data until we reach the transition to the white-frequency domain, then average the first-difference of these averages to reduce the uncertainty of the frequency estimate. This process could continue until the end of the white-frequency domain, where it is no longer optimum. For the system whose Allan deviation is shown in Fig. 1, the upper end of the white-frequency domain is at an averaging interval of about 12 hours.

In contrast to the white phase noise domain, where the prediction error for a single measurement is independent of the averaging time, the prediction error for time differences in this domain is proportional to $\tau^{0.5}$—there is now an explicit trade-off between the time accuracy of a single time-tag and the cost of obtaining the synchronization data. The cost of averaging increases linearly with the number of points that are averaged, whereas the uncertainty decreases only as the square root of this number. Incremental improvements in the accuracy of a single time-tag therefore become more and more expensive.

Just as correcting the time of a clock is not optimum in the domain where the spectrum of its fluctuations is dominated by near white phase noise, correcting its frequency is equally inappropriate in the domain where that spectrum is dominated by white frequency noise. As above, the prediction of the future performance of a parameter whose fluctuations are given by a normal distribution cannot be improved by a linear combination of previous observations. The optimum strategy is to average the frequency observations until their spectral density can no longer be characterized by a normal distribution. This is usually done with a recursive filter that realizes an exponential response in the time domain. [1], (3). The time constant of this filter is set to the point at which the noise spectrum is no longer dominated by white frequency noise. This time constant is independent of the rate at which data are acquired or the phase noise of the measurement process; it is determined only by the characteristics of the oscillator itself.

VII. The Network

It is not difficult in principle to incorporate the effect of the network delay using this framework. The first step is to understand the spectrum of its fluctuations. As pointed out above, the distribution of the means of groups of relatively closely spaced messages will have a normal distribution even though the distribution of the individual messages is almost never Gaussian. A series of experiments was conducted to verify that this is true.

Packets were sent in the network time protocol format between two time servers whose clocks were both independently synchronized to UTC. For each packet, we computed the time-difference between the two systems using (7). This process was repeated 25 times with requests spaced one second apart; we averaged these results to form a single time-difference point. This entire process was then repeated every 5 minutes for several days. These data are not used to discipline either clock; that is accomplished by procedures that are outside of the scope of the experiment.

As discussed above, these averages should have a normal distribution with a mean value of 0 (because the clocks at both ends are synchronized) and a variance that is determined by the characteristics of the network delay. The Allan deviations of these measurements are shown in Fig. 2.

The bottom set of points (with the symbol "+") show the results of a loop-back test. (A straight line with a slope of −1 is shown superimposed on these data for reference.) The source and destination machines are the same for this experiment, so that the fluctuations in the clock itself are irrelevant. The data are characterized by near white phase noise over the entire time domain; the magnitude is about 5 μs, which is an estimate of the fluctuations in the delay in the machine itself (the actual loop-back delay is a function of the load on the system and was about 240 μs in these tests). Strictly speaking, the near white phase noise in the loop-back experiments is a measure of the fluctuations in the asymmetry of the measurement delay, because we correct for the delay itself using the round-trip method described above. This effective asymmetry could arise from a rapid change in the processing delay between the time the message was sent and when it was received.

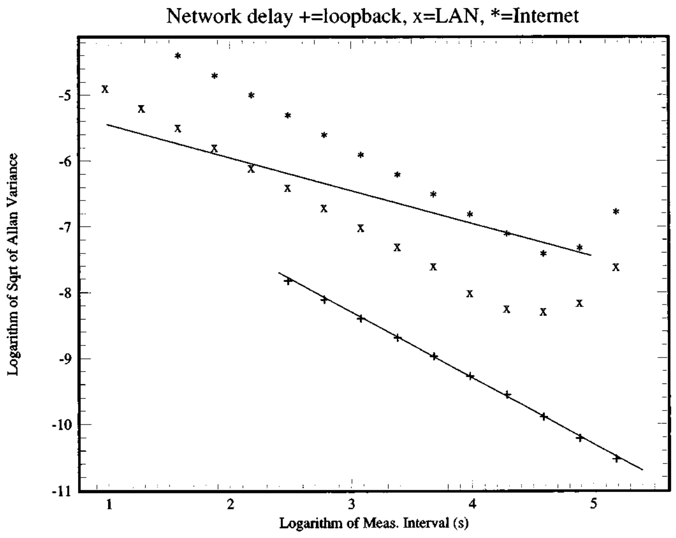


Fig. 2. The Allan deviation of the network delay between two machines that are independently synchronized to UTC. In the plot with "+" symbols, the packets are looped back to the sending machine; the plot with "x" symbols are for two machines on the same local area network and the plot with the "*" symbols are for two machines on the Internet and separated by 1200 km. The line of slope −1 is shown for reference and the line of slope −0.5 is copied from Fig 1.

The top two plots on Fig. 2 show the Allan deviation of the measurements between two time-servers on the same local area network (shown by the symbol "x") and two machines on the Internet separated by about 1200 km (shown with the symbol "∗"). The mean time difference in these experiments was typically about 1.5 ms—a value consistent with the jitter in the independent "lockclock" processes used to synchronize the servers themselves. All of the data are clearly limited by near white phase noise at short periods. The magnitude of the noise increases with the length of the network path, but its character does not change until we reach periods longer than about 6 hours where large, correlated fluctuations in the network delay become important. (Such correlated fluctuations violate the conditions needed for the validity of the central limit theorem.)

Superimposed on these upper two plots is a line with a slope of −0.5 copied from Fig. 1. As discussed above, that line represented an estimate of the magnitude of the frequency noise in the clock oscillator of machine *gpshub*. The parameters of an algorithm to synchronize the clock of this machine can be deduced from the figure by comparing the noise associated with the network with the frequency noise of the clock itself.

The goal of the synchronization procedure is to improve the stability and accuracy of the local clock. Therefore, adjustments of its time or frequency are limited to those portions of the spectral domain where the calibration data adds information, that is, where its noise spectrum is better than that of the free-running local oscillator. The spectral density of the noise of the local clock is almost always less than the noise of the time server when seen through a noisy channel and measurement process, so that some kind of averaging of the calibration messages is usually necessary to realize an optimum design. The data of Fig. 2 can be used to design a quantitative algorithm based on this general principle.

If we plan to synchronize *gpshub* using a time server that is connected to the same local area network, for example, then the variance of the calibration signals will be given by the data shown by "x" on the plot. The variance of these calibration signals falls below the variance due to the frequency noise of the clock for averaging times longer than about 150 s. This is the minimum time-constant that we should use in correcting the time of the clock so as to not degrade its inherent stability with network noise. It is also the maximum time over which the time-differences are dominated by near white phase noise and so could be improved by averaging them. The Allan deviation results shown on the plot were obtained by averaging 25 consecutive time-difference measurements spaced 1 s apart, so that these data are well within the white phase-noise regime where averaging them is justified. Because the cross-over point is a function of network load and other factors, we could estimate the frequency of the local clock using consecutive groups of time-differences spaced no less than about 180 s apart to be sure that we are well into the domain where the spectrum is dominated by white frequency noise. These frequency estimates would be averaged using a time-constant of about 14000 s—the upper-end of the domain in which white frequency noise dominates the spectrum as seen from Fig. 1. This average frequency could then be used to correct the clock using the method described in [1]. The exact values of these parameters would be determined based on a trade-off between how well we want to synchronize the clock and how much we are willing to pay in terms of computer cycles and network bandwidth. This trade-off enters at two points: how many consecutive time requests we choose to average to build each mean time-difference, and how long we wait between each group once we are in the white frequency noise regime of the clock. Using the choices above, the minimum uncertainty in the clock synchronization would be about $\sqrt{2}\sigma_y(180)180 \approx 200\ \mu$s, assuming, of course, that the time of the server was known this well.

This is an impressive performance, but it is pretty expensive; it requires groups of 25 time packets every 180 s or about 1 packet every 7 s on the average. Note, however, that replacing this procedure with the "simpler" one of requesting a single packet every 7 s will not necessarily result in the same level of performance, even though the average cost is the same in both cases. The uneven spacing that was described is an important concept—the members of each group are close enough together that these intervals are in the domain where white phase noise is the dominant problem so that averaging them is appropriate. At the same time the separation between groups is chosen so that it is in the domain where white frequency noise dominates the spectrum so that the difference between consecutive means is optimum for estimating the frequency offset of the local clock. Neither of these requirements would be adequately addressed using single requests at equally spaced intervals.

The same analysis would determine the parameters for a synchronization loop using the distant server on the Internet. These data are much noisier and so we would have to average for a much longer time; the noise of the network-based time-difference measurements does not fall below the noise of the clock itself until an averaging time of almost 5.3 h, and this would set the minimum time-constant for the time clock-correction loop. This value is about the same as the upper limit for the domain in which white frequency noise dominates the spectrum. The performance of this loop would not be as good as the first one, of course, because the loop operates almost completely in the white-frequency noise domain, and the prediction error there increases as $\tau^{0.5}$. In other words, the network noise is so high that the clock would be essentially free running for averaging times less than about 5 hours.

Matters could be improved by averaging more than 25 time-differences in each group. The spectral density of the time-difference measurements would only improve as the square root of this number, so that significant improvements become expensive pretty quickly. Assuming 25 time-differences were kept in each group, the best that could be done would be to estimate the frequency using measurements about every 5.3 hours (which is the earliest time that the measurement noise due to the network fluctuations becomes small enough that we can begin to see the fluctuations in the clock oscillator itself). The prediction error using this time-constant would be about $\sqrt{2}\sigma_y(19000)19000 \approx 2$ ms. This is comparable to the uncertainty achieved with "lockclock," which is not surprising. As pointed out above, the source of synchronization does not matter very much once the fluctuations are dominated by the frequency noise of the local oscillator. Note that this performance is just about at the edge of what can be achieved with this clock hardware, because the cross-over point at which the network noise drops below the time dispersion due to the frequency noise of the clock is very close to the upper end of the region in which the clock noise is still white frequency noise. If the network noise was greater or the frequency stability of the clock oscillator was worse, we could still go through the motions of averaging, but the mean value no longer has the nice properties expected from a normal distribution, and the prediction variance degrades accordingly.

In summary, the analysis using the Allan deviation can be used to evaluate the trade-off between the synchronization accuracy obtained and the cost of realizing it. As the noise in the channel used to transmit calibration data gets worse, we must compensate by adding more measurements to each group of time-differences and by increasing the time interval between groups so that the equivalent frequency-measurement noise of the channel drops below the inherent frequency noise of the clock. The local clock is better than its calibration source (as seen through the network) for times shorter than this cross-over, so that correcting it in this domain (either in time or in frequency) is a mistake. Because averaging improves matters only by a factor that is proportional to the square root of the number of measurements, improvements realized by including more time differences in each group become expensive very quickly. Conversely, the cost/benefit ratio becomes much more favorable if we can settle for poorer time synchronization (in an rms sense). Using 3 time-differences in each group instead of 25 and increasing the interval between measurements from 5.3 to 12 hours would decrease the cost by a factor of about 20, while still providing synchronization with an uncertainty of better than 30 ms rms.

It is possible to further decrease the cost by further increasing the interval between measurements, but the algorithm may have to be modified should we choose to do so. Once the interval between calibrations exceeds the value for which the frequency of the oscillator is dominated by white frequency noise, averaging consecutive frequency estimates is the wrong thing to do. The optimum estimate for an oscillator dominated by a random-walk of its frequency is the most recent value (with no averaging at all); the frequency aging would have a white spectrum in this case, but it is difficult to exploit this fact using standard hardware. In spite of these problems, using calibration intervals of a day or longer might be interesting for clients whose accuracy requirements were very modest—on the order of 1 s, for example.

VIII. Detecting False-tickers

A false-ticker is a server that is transmitting the wrong time even though its status is shown as healthy. This should never happen, but a client must be prepared for it nevertheless. The simplest strategy for detecting this problem is to use data from several independent time-servers. A false-ticker can be detected by comparing the time of each server with a weighted average of all of the measurements (as is commonly done in time-scale algorithms [8], [9]), or the "best" server can be selected using various criteria [2].

In this work we develop a strategy that uses only a single server and the time of the local clock itself to validate the received data. The reason for this is that a false-ticking server should be a relatively rare event, so that querying several servers on every cycle is unnecessary most of the time. Using only a single server initially, therefore, will decrease the average load on the servers by confining multiple-server requests to those cycles when the preliminary analysis suggests that something has gone wrong.

If it has been τ seconds since our last calibration, then the time dispersion of the local clock is of order $\sqrt{2}\sigma_y(\tau)\tau$, and a time correction that is much bigger than this is suspect—something has changed in the interval since the last calibration cycle. Deciding what "much bigger" means is a matter of probabilities: we can choose to classify small conforming fluctuations as errors by setting the threshold too low or we can include glitches as acceptable data by setting it too high. We have found that a threshold of 3 times the standard deviation of the running mean is a reasonable compromise above which we assume that something has gone wrong. (This threshold implies a false-alarm limit of $< 1\%$ which is small but not negligible. If the interval between calibrations is 1 hour, for example, there will be about one false alarm per week.) If the data pass this test, then we can assume that nothing strange has happened, and we can use the data as part of our update procedure. If, on the other hand, this threshold has been crossed, there are four possibilities:

- The server is broken.
- Our clock has experienced a time-step.
- Our clock has experienced a frequency step.
- There has been a large change in the parameters of the network delay.

We can differentiate among these possibilities by switching to another server and seeing if its data are consistent with the time of our local clock using the procedure outlined above. If yes, then we vote the first server as sick by a vote of 2-1, including ourselves in the decision. If the two servers agree (within the uncertainty of the network noise), then we vote ourselves as sick. We cannot yet distinguish between possibilities the second and third and so we correct our clock based on the time measurements and wait for the next cycle.

If another error of comparable magnitude is found (again, within the network noise) on the next calibration cycle, then we assume that our clock has had a change in its frequency rather than a simple time step. This is a rather rare event and may be a signal that a hardware failure is imminent. We would adjust the frequency in our prediction model in this case. Again, we would wait to see what happened on the next cycle. If the change to the frequency results in agreement between our clock and the time of the server on the next cycle, then the diagnosis of a frequency step is probably correct. If not, then we probably have a hardware problem which may require outside assistance.

If the data from neither server agrees with our time prediction, then either two of the systems are sick or the network delay has become very asymmetric. The current algorithm does nothing in this case; it waits for the next calibration cycle, assuming that doing nothing is the best strategy when the problem cannot be well specified.

In summary, there is no guaranteed way of detecting a false-ticker, unless its time error is much larger than any of the other noise sources. It is generally easier to detect time-steps if we increase the sample rate because this allows us to reduce the measurement noise (by averaging) so that they are easier to see. This strategy contains the usual trade-off between cost and accuracy. Small frequency steps, on the other hand, are much more difficult to detect; at some point they become indistinguishable from the background of stochastic frequency fluctuations. Even a deterministic step in frequency is masked by the white frequency noise of the oscillator in the short term and can be detected only when that noise has been reduced by averaging. The same thing is true for frequency aging. The Allan deviations for the oscillators typically found in computer clocks tend to increase for averaging times longer than about a day (see Fig. 1), and this sets an upper-limit to how much can be achieved by averaging.

IX. Tests and Results

The methods described above have been used to synchronize the clock of a workstation named *strain*, which is located in our laboratory in Boulder, CO. We chose a network time servers located about 1200 km away in Seattle, WA, as a time reference. The one-way delay is about 48 ms and is quite variable; although the delay to another server in Gaithersburg, MD is about 80 ms, that delay is somewhat more stable because the path contains fewer routers.

The clock oscillator in *strain* has a level of white frequency noise that is about a factor of two smaller than that of *gpshub*, which was shown in Fig. 1; the statistics of the network delay are about the same as the "Internet" data of Fig. 2. The synchronization algorithm used groups of 50 messages separated by 3000 s. The time differences in each group were averaged and the resulting averages were examined for outliers as described above. These averages were then used to compute the average frequency offset of the local clock as described in [1]; this average frequency was used to schedule periodic adjustments to the local time. The time adjustments were performed in the usual way by changing the value added to the clock register on each interrupt (i.e., the effective frequency of the oscillator) for a set number of times until the time adjustment had been amortized. The magnitude of these frequency adjustments was limited to less than 1% of the clock rate to provide very small time adjustments, and also to ensure that the clock did not stop or run backwards.

Fig. 3 shows the prediction error, ε_x as a function of time when the algorithm is running in steady-state. This error is $\varepsilon_x(k) = |(x_k - \hat{x}_k)|$, i.e., the difference between the average measured time-difference at some epoch and the value predicted using (1). These data have a mean of 0.92 ms; they are somewhat better than the noise level expected using the "Internet" delay characteristics of Fig. 2 because we increased the size of each group to 50 messages rather than 25.

Fig. 4 shows the time of the clock on *strain* with respect to the local ACTS system while that clock is being synchronized using the time server in Seattle as a reference. Both the noise and the accuracy of the ACTS comparison are about 1 ms, so that differences on this order are not statistically significant. The Allan deviation of these data are shown in Fig. 5, with the usual straight line with a slope of -1 for comparison. The rms time error is about 2 ms for all averaging times, but the large 10 ms spikes make an appreciable contribution to the variance. In addition, there is some indication in both figures of a long-period fluctuation. This is probably due to a combination of long-period fluctuations in the network asymmetry (perhaps having a 7-day period) and changes in the ambient temperature of the client machine (which will affect the frequency of the quartz-crystal oscillator).

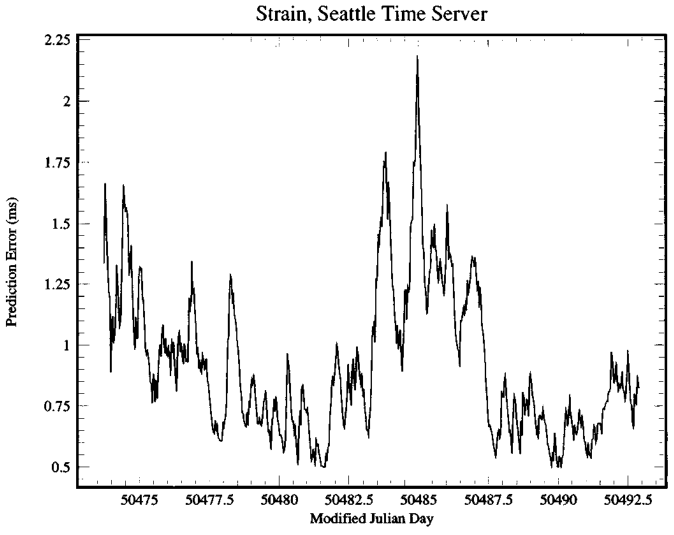


Fig. 3. The rms prediction error in ms when the clock on strain is synchronized to the server in Seattle. The long-period structure may be due to slow changes in the ambient temperature of our laboratory or to long-period fluctuations in the network delay.

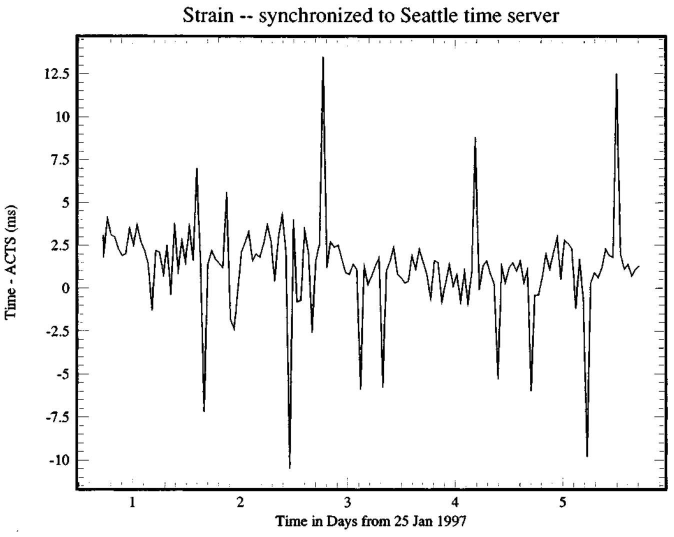


Fig. 4. The time offset of the clock on strain with respect to UTC as measured using the NIST ACTS time signals. The ACTS comparisons are characterized by white phase noise of about 1 ms, so that fluctuations of that order or smaller are not significant.

X. Accuracy/Cost Tradeoff

The procedures I have discussed can be used to investigate the tradeoff between synchronization accuracy and the cost of achieving it. These considerations are especially important for users who do not need millisecond-level accuracy, because even a moderate relaxation in the accuracy requirement results in a significant decrease in the cost of achieving it.

The cost of synchronizing a client clock is roughly pro-

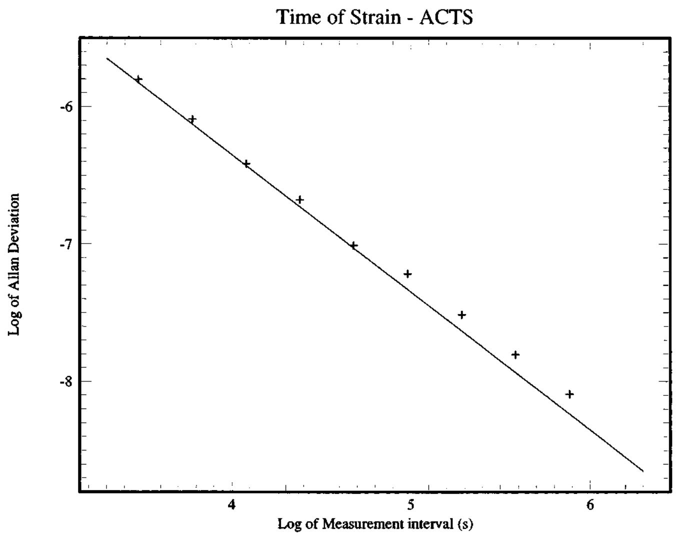


Fig. 5. The Allan deviation of the data shown in Fig. 4. The straight line has a slope of −1. The points deviate from this straight line starting at a period of about 1 day, which is caused by the long-period fluctuations that are visible in the data set.

portional to the average number of network packets/s that must be exchanged with the server. This number, in turn, is driven by several considerations: the number of rapid-fire messages that are averaged to reduce the measurement noise and the network asymmetry; the interval between groups of messages that is chosen to estimate the frequency noise in the local oscillator; and the importance of detecting a failure either in the local clock or in one of the servers. Deciding how to balance the cost and the accuracy involves estimating the importance of these factors.

If the server selected is on a local-area network or is on the Internet but is not too far away, then the impact of any network asymmetry is likely to be small. Local-area networks tend to be symmetric, and the asymmetry in a short path is bounded by the round-trip delay (8). The number of rapid-fire message could be reduced in each group in this case, and this reduction would have only a small impact on the overall accuracy. Groups of 3 messages are about right for this case; this size provides some protection against glitches, but is a factor of almost 20 less expensive than the groups of 50 that used in our experiments. Groups larger than about 50 are not practical, because the standard deviation of the mean improves only as the square root of this number. The goal is to use the minimum number of messages so that the mean of the group has a normal distribution; using a value smaller than this will make it difficult to protect the client against time-difference outliers due to large network asymmetries. Using a group that is larger than necessary increases the cost without significantly improving the performance. At least in principle, it would be possible to adjust the number of messages in each group dynamically by comparing a desired performance level with a real-time estimate of the variance in the network delay.

The effect of adjusting the interval between groups can be evaluated more quantitatively. The highest synchronization accuracy will be realized when this interval is set to the start of the domain in which the Allan deviation is dominated by the frequency noise of the local clock. The synchronization error for any other time interval τ is approximately $\sqrt{2}\sigma_y(\tau)\tau$, which can be calculated from the free-running Allan deviation data shown in Fig. 1. Using an interval of 3.5 days (the right-most point in Fig. 1) between calibrations, for example, would result in a synchronization error of about 40 ms rms. The Allan deviation of an oscillator is determined by its design, so that these parameters are likely to be the same for all machines of a given type.

If a synchronization error of 40 ms rms is adequate, then measuring the asymmetry of the network delay may be less of a problem in some cases. If the server and the client are on the same local-area network, the fluctuations in the asymmetry are usually substantially less than 40 ms, so that this level of synchronization could be achieved by exchanging an average of less than 1 message/day with the server (1 group containing 3 packets every 3.5 days). The maximum synchronization error would almost always be less than 120 ms (3 σ), which is likely to be adequate for most applications.

The final consideration in choosing the interval between calibration messages is the desire to detect a problem as quickly as possible. Using a time interval of 3.5 days between calibrations, for example, means that a glitch in the local clock will persist (on the average) for 1.75 days before being detected. It is very difficult to estimate the probability of finding a large time-step, because, by definition, they are glitches that do not conform to the statistical measures discussed. Nevertheless, the cost of such a rare event might be high enough to drive the interval between messages to a value significantly smaller than that which would be required to realize a given synchronization uncertainty.

XI. Conclusions

The considerations that govern the design of algorithms used to synchronize computer clocks using messages transmitted over a packet network such as the Internet have been discussed. This method was used to design an algorithm for synchronizing the clock of a standard workstation using a server about 1200 km away. Both the client and the server are connected to local area networks, which are in turn connected to the Internet using standard methods and hardware. The client system operated in a standard office environment with no special control of the ambient temperature or any other environmental parameter.

The communication between the client and the server used the standard protocols and format of the network time protocol (NTP), but the remainder of the algorithm is completely different. Only one server for this study was used, although procedures for querying a second server when the first one seems to be broken were outlined. The procedures for adjusting the time of the local clock are the same as described previously in [1].

The performance of the algorithm is comparable to that obtained with the "lockclock" algorithm or with the network time protocol using a directly connected radio clock. This algorithm, therefore, can provide near "stratum-1" performance using only network data.

Although "lockclock" and "interlock" share a common design philosophy, there are important differences between them. These differences can be traced to the method of receiving time calibrations: "lockclock" depends on ACTS while "interlock" can work with data transmitted over the noisier Internet.

The dial-up telephone lines that are used to connect a machine using "lockclock" to the NIST ACTS system have delays that are stable and highly symmetric. In addition, the NIST server hardware measures and automatically corrects for the transmission delay. The result is that the time-differences estimated using the ACTS data are characterized by almost pure white phase noise for all averaging times. Furthermore, the level of these fluctuations is on the order of milliseconds, so that the ACTS data can be used "as is" with little or no averaging or analysis. This greatly simplifies the algorithm because there is no need for the cumbersome separation-of-variance machinery. Furthermore, the probability that the ACTS system will transmit the wrong time is small enough that the "lockclock" algorithm need not make provision for testing this possibility. This simplifies the procedure and improves the performance because an estimate based on ACTS data is not limited by the statistical uncertainties that characterize the separation of variance procedures.

In addition, the procedures discussed provide a quantitative way of evaluating the trade-off between synchronization accuracy and cost. As has been shown, the cost/benefit ratio is not linear; an increase in accuracy by a factor N requires an increase in cost proportional to N^2. A moderate relaxation in the synchronization accuracy, therefore, can result in substantial savings both in the network bandwidth that is required to realize it and in the number of public servers that must be supported.

Acknowledgments

Kenneth Monington assisted in the early phases of the measurements.

References

[1] J. Levine, "An algorithm to synchronize the time of a computer to universal time," *IEEE/ACM Trans. Networking*, vol. 3, pp. 42–50, 1995.

[2] D. L. Mills, "Network time protocol (version 3); specification, implementation and analysis," in DARPA Network Working Group Rep. RFC-1305, Newark, Delaware, Univ. Delaware, 1992.

[3] K. Arvind, "Probabilistic clock synchronization in distributed systems," *IEEE Trans. Parallel Distrib. Syst.*, vol. 5, pp. 474–487, 1994.

[4] F. Cristian, "A probabilistic approach to distributed clock synchronization," *Distrib. Comput.*, vol. 3, pp. 146–158, 1989.

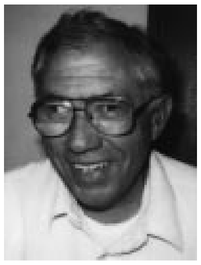
[5] J. R. Taylor, *An Introduction to Error Analysis*. Mill Valley, California: Univ. Sci. Books, 1982, pp. 197–199.

[6] C. W. Gardiner, *Handbook of Stochastic Methods*. New York: Springer Verlag, 1990, pp. 37–39.

[7] D. B. Sullivan, D. W. Allan, D. A. Howe, and F. L. Walls, Eds., *Characterization of Clocks and Oscillators*. Boulder, Colorado. NIST Technical Note 1337, 1990.

[8] P. Tavella and C. Thomas, "Comparative study of time-scale algorithms," *Metrologia*, vol. 28, pp. 57–63, 1991.

[9] D. W. Allan, J. E. Gray, and H. Machlan, *The National Bureau of Standards Atomic Time Scale: Generation, Stability, Accuracy and Accessibility*. NBS Monograph 140. Boulder, Colorado: National Bureau of Standards, 1972, pp. 205–230.



Judah Levine was born in New York City in 1940. He received a Ph.D. degree in physics from New York University in 1966. He is currently a physicist in the Time and Frequency Division of the National Institute of Standards and Technology in Boulder, Colorado, and is also a Fellow of the Joint Institute for Laboratory Astrophysics, which is operated jointly by NIST and the University of Colorado at Boulder. He has worked for about 25 years on time scales and on methods for distributing time and frequency information.

Dr. Levine is a Fellow of the American Physical Society and is a member of the American Geophysical Union and the IEEE Computer Society.