An Algorithm to Synchronize the Time of a Computer to Universal Time

Judah Levine

Abstract—**I describe an algorithm that synchronizes the time of a computer clock to UTC with an uncertainty due to all causes of about 1 ms RMS. The method uses periodic calibration data obtained via dial-up telephone access to the NIST automated computer time service. The interval between calibrations can be chosen to provide optimum time accuracy or reduced accuracy at reduced cost based on a preliminary evaluation of the statistical performance of the clock. The computer can serve as a primary network time server or can be used stand-alone whenever precise time-stamps are required.**

I. Introduction

IN THIS paper I describe an algorithm that I have used to synchronize the time of computer clocks to UTC with an uncertainty of about 1 ms RMS. The algorithm combines periodic time corrections based on a statistical evaluation of the frequency of the clock oscillator with calibration data obtained by calling the NIST automated computer time service (ACTS) [1] using ordinary direct-dial telephone circuits.

Computers synchronized in this way can be used in data-logging applications and wherever accurate digital time-stamps are required; they can also serve as primary time servers on a network, transmitting time to clients using any of the established time protocols such as NTP [2]. The ACTS transmissions provide advance notice of leap seconds and of the transitions to and from daylight saving time and this information can also be automatically incorporated into the time messages sent to client machines.

I have implemented this algorithm on several different types of computers that are connected to local-area networks and to the Internet. All of the machines are currently in use as primary time servers. The stability of the server time is limited primarily by the quality of the clock oscillator in all cases, although some additional limitations are imposed by the operating system. I discuss these limitations in more detail below.

II. The Deterministic Portion of the Clock Model

In what follows, t_k is an epoch at which I have measured the time of the machine, and x_k is the measured time difference at that epoch between the local clock and UTC. A positive value of x signifies that the local clock is fast. The effective rate of the clock with respect to UTC during the interval between t_{k-1} and t_k is given by y_k, and the filtered rate estimate at epoch t_k (using the method to be described below) is $\overline{y}_k$. The rates are dimensionless (seconds/second). The algorithm drives x_k to zero by estimating the optimum value for the filtered frequency $\overline{y}_k$ and uses this frequency to schedule periodic corrections to the time of the local clock.

The time-difference measurements are approximately equally spaced in epoch; this approximate interval is τ_0. It is measured by the local clock, and a telephone call to the ACTS system is initiated when it has elapsed. The actual interval between t_{k-1} and t_k is τ_k; it is computed from the ACTS messages and is larger than τ_0 by about 30 or 40 s—the amount of time needed to complete the call to the ACTS time server and to perform several housekeeping chores.

I predict the time difference at t_k using the average rate offset estimated in the previous measurement cycle

$$\hat{X}_k = X_{k-1} + \overline{y}_{k-1}\tau_k \tag{1}$$

where $\hat{X}_k$is the predicted time difference. This prediction equation contains only terms that are linear in the time interval; I neglect any deterministic frequency aging during the prediction interval. This is valid if the prediction interval is a few thousand seconds or less; it may be inadequate for intervals that are much longer than this.

If the difference between the predicted and measured time differences is not too large (to be quantified below), I use this time difference to modify the estimate of the average frequency of the oscillator. The current frequency estimate, y_k, computed using the first-difference of the measured time differences

$$y_k = \frac{X_k - X_{k-1}}{\tau_k} \tag{2}$$

is combined with the previous estimate of the filtered frequency to form an updated estimate

$$\overline{y_k} = \frac{\overline{y_{k-1}} + Gy_k}{1+G} \tag{3}$$

where G is computed from the statistical parameters of the clock as is described in the next section.

This average frequency is the primary output of the model; it is used to schedule periodic incremental time adjustments to the local clock. These adjustments are scheduled frequently enough so that the clock is not allowed to drift off time by more than the measured jitter in consecutive time-difference measurements made during a single calibration cycle. Since

Manuscript received December 23, 1993; revised July 12, 1994; approved by IEEE/ACM Transactions on Networking Editor I. Cidon. This work was supported in part by Grant NCR-9 115 055 from the National Science Foundation through the University of Colorado.

The author is with the Joint Institute for Laboratory Astrophysics, National Institute of Standards and Technology, University of Colorado, Boulder, CO 80309 USA (e-mail: Judah.Levine@Colorado.edu).

IEEE Log Number 9408393.

this uncertainty is about 0.5 ms, the nominal interval between clock adjustments during the k-th time interval is

$$T_k = \frac{0.0005}{\overline{y_k}}. \quad (4)$$

This nominal time interval is adjusted so that there are an exact integral number of adjustment intervals in the calibration interval τ_0. The calibration interval τ_0 can then be measured by simply counting the number of adjustments that have been performed. The actual adjustment is then recalculated to the nearest microsecond as the product of this new interval and the estimate of the filtered frequency. If, for example, $\overline{y}_k = 1.15 \times 10^{-5}$, (4) gives $t_k = 43s$. If $\tau_0 = 3000s$, t_k would be adjusted to 40 s so that there would be exactly 75 adjustment intervals in a calibration interval. An adjustment of 460 μs would be applied every 40 s, and a calibration cycle would be initiated after the 75th adjustment had been performed.

The complete algorithm is composed of two loops: an "outer" loop activated every τ_0 s to update the parameters of the model, and an "inner" loop that is activated every t_k s to make a small adjustment to the clock. The method outlined in the previous paragraph simplifies the design of these two loops by making the two periods commensurate. Additional time adjustments will be scheduled if the time measurement x_k is large or if a leap second or similar time change is imminent.

Although the underlying deterministic model of the clock does not change once the adjustment algorithm is enabled, the detailed implementation is different. The goal of the adjustment loop is to drive x_k to zero so that $\hat{X}_k$ is identically zero for all k. The consequences of this are discussed below.

III. The Stochastic Portion of the Clock Model

The primary function of the ACTS calibration data is to provide a robust estimation of the frequency of the local clock oscillator. Two noise processes will affect the estimate:

A. Noise in the Time-Difference Measurements Using ACTS

The ACTS hardware at NIST dynamically estimates the delay in the telephone circuit and the modems between NIST and the user. The one-way delay is estimated as one-half of the round-trip delay between the two end-points. This round-trip delay is measured by the transmitter each time the user echoes the received on-time marker back to NIST, and subsequent transmissions are advanced so that they will arrive on-time at the user's modem. The advance used for each transmission is the average of the four most recent one-way delay measurements. We have tested the performance of the ACTS system using both local telephone circuits in Boulder, Colorado and long-distance connections between Boulder and radio station WWVH in Kauai, Hawaii [3]. The on-time marker was received within 0.4 ms of UTC in all cases; the RMS scatter during a single telephone connection varied somewhat from call to call but was never greater than 90 μs.

The receiving software measures the time difference between the local clock and UTC using three consecutive ACTS messages spaced 1 s apart; I use three readings to detect outliers using majority voting. The time of the computer clock is read each time the ACTS on-time marker is received, and the program interpolates between ticks of the local clock using a tight loop whose execution speed is measured during each calibration cycle. (This interpolation routine can be disabled if the hardware and software environment support an equivalent capability.) The frequency offset of the local clock will produce a negligible time dispersion during this 3-s interval so that the observed dispersion (calculated as max-min) is a direct measure of the stability of the telephone delay and the processing latency in the system. A typical value for this dispersion is 0.4 ms, and the average dispersion over the last 6 calibration cycles is used to test the current measurements. (The exact number of cycles used to compute the average is not critical.) If the scatter (max-min) among the 3 measurements in the current calibration cycle is less than three times this average variation, the average of the three is used; if only two of the three satisfy this criterion, the outlier is rejected and the average of the two is accepted; if no two values agree then the entire measurement cycle is rejected and is reinitiated after a short delay. In all cases, the epoch of the measurement is set to the mid-point of the data that have been averaged. This logic adapts to slow changes in the reciprocity of the telephone circuit or in the latency of the system, but will reject a sudden change as an error.

The dispersion of the time-difference measurements can be well characterized as white phase noise, which means that the accuracy of the measurements could be improved by averaging additional short-term data beyond the three values used in the current implementation. (Short term in this context means an interval that is short enough that I can neglect the time dispersion in consecutive measurements due to the offset frequency of the local oscillator.) This is not done because this variation is not the dominant contributor to the total prediction error budget, so that the increase in telephone charges would not produce a corresponding increase in accuracy.

The algorithm estimates the frequency of the local oscillator using the first difference of the phase measurements (2). Since the fluctuations in the two phase measurements are not correlated, the uncertainty in the frequency estimate is $\sqrt{2}$ times the measurement uncertainty divided by the time interval between the measurements. If the frequency offset of the local oscillator is on the order of 10^{-5} (about 1 s/day), and if the noise in the time difference measurements is to contribute no more than 10% to the error budget of the frequency determination, then the minimum interval between calibration cycles is on the order of 1000 s.

B. Noise in the Clock Oscillator Frequency

Stochastic fluctuations in the frequency of the clock oscillator are the dominant limit to the accuracy of the prediction algorithm. The underlying physics of these fluctuations can often be identified by examining how the power spectral density of these fluctuations varies with Fourier frequency. This description is particularly simple for most oscillators [4] because the dependence on Fourier frequency can be

approximated by a sum of five power-law noise processes

$$S_y(f) = \sum_{\alpha=-2}^{\alpha=2} h_\alpha f^\alpha . \quad (5)$$

In this equation, $S_y(f)$ is the power spectral density at Fourier frequency f and the parameters h_α are the constant coefficients of its polynomial expansion. Each term in the expansion corresponds to a physical process:

$\alpha = -2$ Random Walk Frequency Modulation;
$\alpha = -1$ Flicker Frequency Modulation;
$\alpha = 0$ White Frequency Modulation;
$\alpha = +1$ Flicker Phase Modulation;
$\alpha = +2$ White Phase Modulation.

For most oscillators, this equation can be simplified even further by noting that there are usually broad regions of Fourier frequency space in which the spectral density can be adequately modeled using only a single term, so that I can speak of a Fourier regime in which the system is dominated by white frequency modulation, etc. This characterization is particularly useful because there is a statistically optimum strategy for dealing with each noise process [4].

The Allan variance, usually denoted by $\sigma_y^2(\tau)$, is a time-domain measure of the noise of an oscillator [4]. It is proportional to the time-average value of the square of the first differences of consecutive frequency estimates. It is most useful for this discussion because its variation with sampling time τ can also be approximated by a polynomial expansion in τ

$$\sigma_y^2(\tau) = \sum_{\mu=-1}^{\mu=2} a_\mu \tau^\mu . \quad (6)$$

If the expansion of the spectral density in (5) can be approximated by a single term, then the expansion of (6) becomes a single term as well. If $\alpha \leq 1$, the exponents of the single surviving terms in each expansion are related by $\alpha = -\mu - 1$ so that determining μ determines α and allows us to identify the underlying physical process that dominates the stochastic variation. This identification provides a guide to the optimum strategy for dealing with the oscillator in the corresponding time and Fourier-frequency domains. (If $\alpha > 1$, the one-to-one relationship between α and μ can be preserved by computing the "modified" Allan variance as described in the literature [4].) In particular, averaging a quantity will only be an optimum strategy when its underlying noise spectrum is white, and these sorts of analyses delineate the time-domain regime in which such averaging converges to a statistically robust estimate of the underlying physical parameter and is therefore appropriate. This correspondence between time- and frequency-domain representations is exploited throughout the remainder of the discussion.

Fig. 1 (to be discussed in more detail later) shows the square root of the Allan variance (often called the Allan deviation) as a function of averaging time for a typical oscillator, and the three lines drawn through the points identify three domains in which different noise processes dominate the spectrum. Using the relationship between μ and α, I can identify the processes that dominate the noise spectrum in three different domains.

The white noise in the measurement process (white phase noise) discussed in the previous section tends to be important at the shortest periods, with the fluctuations in the frequency of the oscillator itself becoming important at the longer periods. Starting at about 10^4 s, the fluctuations in the oscillator frequency are initially independent of Fourier frequency (white frequency noise), but the power spectral density eventually starts to increase as the Fourier frequency decreases, or, equivalently, as the averaging time is made longer. The dependence on Fourier frequency gradually changes to $1/f$ (flicker frequency noise) and then to $1/f^2$ (random-walk frequency noise). As discussed in [4], these characterizations are important because there is an optimum measurement strategy for each noise type.

White frequency fluctuations are the "best" from a prediction point of view, because an underlying mean frequency exists, and the fractional fluctuations about this mean can be made smaller with increased averaging. The increased averaging is most easily implemented by decreasing τ_0 and therefore G in (3) as discussed below. The error in predicting the time in this region increases only as the square root of the time interval between calibrations. The performance of the algorithm can therefore be improved (with a corresponding increase in telephone charges) by shortening the interval between calibrations until the floor set by the white phase noise discussed above is reached or until some other process, such as jitter in the interrupt latency, becomes important. Conversely, the performance of the time prediction algorithm will deteriorate as the interval between calibrations is made longer. The cost will decrease linearly with the time interval and the prediction error will increase as the square root of this quantity.

This favorable dependence of the prediction error on the square root of the calibration interval will not hold true at longer periods as the nonwhite fluctuations in the oscillator frequency become important. A mean frequency no longer exists in this situation, and increased averaging does not improve (and eventually degrades) the accuracy of the frequency estimate. In other words, if G in (3) is made too small, the frequency estimated by this equation does not respond to the slow changes in mean clock frequency that characterize a nonwhite spectral distribution. This consideration sets a lower bound to G which must be enforced independent of the value τ_0. It is important to remember that while G and τ_0 are related quantities, they are determined from different considerations: G by the point at which the frequency deviations become nonwhite and τ_0 by considerations of the trade-off between calibration cost and time accuracy.

Once the spectral density of frequency fluctuations becomes nonwhite the uncertainty in the time prediction begins to grow linearly with the prediction interval, and will eventually grow faster than linearly as the prediction interval is made still longer. If they were secular or deterministic, these frequency changes might be modeled by adding an aging parameter to the prediction equation. Unfortunately, this is generally not the case, and the best that can be done is to compute an average frequency that gradually "forgets" older data with a time-constant that is roughly equal to the time at which the

frequency fluctuations deviate from a white spectrum. This is another way of describing the purpose of the parameter G in (3). If T_{nw} is the measurement interval at which the frequency fluctuations begin to deviate from a white spectrum, then

$$G \approx \frac{\tau_0}{T_{nw}} . \tag{7}$$

The value of T_{nw} is determined by inspecting the square root of the two-sample Allan variance of the free-running clock oscillator and looking for the time interval at which the slope begins to increase from -0.5 (indicating white frequency noise) toward 0 (which is characteristic of flicker frequency noise). A typical value for T_{nw} is 12 000 s; combining this result with the previous discussion limits τ_0 to the approximate range 1000–12 000 s for optimum performance—the performance will not improve at shorter intervals because of the white phase noise in the measurement process and will begin to degrade at longer intervals because of stochastic nonwhite frequency fluctuations. The corresponding values of G range from about 0.08 to 1.

I have found experimentally that using $\tau_0 = 3000s$ ($G \approx 0.25$) provides a good balance between prediction accuracy and telephone cost; the resulting prediction loop has an error of about 1 ms RMS. Using a smaller value for τ_0 would improve the prediction error in principle, although this improvement would not always be realized in practice because of interrupt latency and similar issues to be discussed below.

IV. Time and Frequency Steps—The Reset Algorithm

The prediction error is the difference between x_k and $\hat{X}_k$. The ensemble of these errors will have a white distribution about a mean of zero if the model described above is an accurate representation of the performance of the oscillator and if τ_0 has been properly chosen. The standard deviation is a function of the noise sources discussed above.

Although it is very unlikely, the prediction error in any cycle can differ from zero by many standard deviations without violating the statistical model. Nevertheless, it is usually better to treat these unlikely events as nonstatistical glitches rather than as very-low-probability conforming events. This choice is justified primarily by experience both with computer clocks and with other types of systems which suffer from the same kinds of problems. The simplest assumption is that a time step has occurred in the local clock (due to a dropped interrupt, for example), and the optimum strategy is to remove the time step by a special adjustment and to not allow the frequency estimator to be modified by this spurious measurement.

If the frequency has suddenly changed on the other hand, removing the time step will not help, since the changed frequency will result in a time step on the next calibration cycle as well. (Modeling the initial glitch as a pure time-step has in fact delayed the application of the proper fix by an additional calibration cycle.) The algorithm tries to repair a second consecutive time-step by modeling it as a frequency step. The second time step will be removed using a special adjustment as above, but the average frequency also will be modified using (2) and (3). This process will be allowed to continue for 1/G calibration cycles, since that is the time constant for the new frequency to be incorporated into the steering equation. If a time-step is again detected after 1/G cycles the algorithm assumes that another time-step has occurred, and the entire process is repeated. Our experience is that this process converges after one or two cycles unless the hardware has failed in which case it never converges at all.

The reset behavior is triggered if the prediction error in any cycle exceeds three times the standard deviation of the most recent cycles so that a slow change in the performance of the system will be automatically accommodated without causing resets. This threshold is somewhat arbitrary and it has been chosen as a reasonable compromise based on experience. Typical computer oscillators will trigger the reset logic about 1% of the time (i.e., about once per week for a calibration interval of about one hour). The reset frequency is not a measure of clock quality since the reset trigger is relative to the dynamically-determined standard deviation rather than to some absolute threshold – poorer clocks will have poorer performance but will not have more resets because the dynamically-determined standard deviation will be higher.

A fundamental assumption of this reset logic is that a large prediction error indicates a local problem rather than a problem with ACTS or with the telephone network. This is a fundamental difference from the clock model in the network time protocol [2], for example, which is prepared to consider the possibility that its received time data are wrong.

A more subtle version of this problem will arise when the power spectral density of the noise in the measurement process is substantially larger than the spectral density of the noise of the clock itself. This is almost always true of the Internet for calibration intervals of a few hours, for example, so that the performance of a good local clock at these periods may be better than the calibration source as seen through the noisy transmission channel. If these calibration data are nevertheless applied to steer the oscillator, its performance at periods of a few hours will be poorer than its free-running performance would have been, and an optimum lock algorithm would have to be designed with these considerations in mind. It is important to emphasize that simply averaging consecutive measurements in this case is unlikely to be the optimum choice, since the power spectrum of the Internet measurements is almost certainly not white at periods of a few hours. As a result, while the adjustment algorithm and the time-difference measurement system are formally independent "black boxes," their designs are coupled through the assumptions each component makes about the underlying noise processes.

V. Clock Resolution and System Latency

Most computers keep time by counting relatively infrequent "ticks"—interrupts that are generated periodically by an internal quartz-crystal oscillator (or perhaps by the zero crossings of the input ac power). The interval between ticks is generally on the order of milliseconds. Some hardware environments provide a much finer time resolution, and some software environments make this finer resolution available to a user task. It is also possible to interpolate between ticks

using calibrated tight loops if the high resolution hardware is not available, but such methods cannot be extended to arbitrary precision because of fluctuations in the system load. My experience is that it is possible to interpolate reliably to only about 5% of the tick interval.

Time signals received through any input channel will be degraded by the time needed to transmit the information between the input port and the higher-level process. The accuracy of time-difference measurements made using ACTS transmissions have a special problem because those transmissions are received through one of the serial ports of the system, and the serial-port driver is particularly complex. Some systems split the serial port driver into two parts—an "outer" part that copies characters from the input hardware to a memory buffer and is interrupt driven and an "inner" part that completes the processing of the received characters and places them in the user buffer. This inner part often runs as a scheduled task rather than as an interrupt-driven one. Since the scheduler is usually activated as the result of a timer interrupt, there can be substantial jitter between the time a character is received by the hardware and the time it is available to the algorithm. This problem is particularly serious if x_k is small, since then the ACTS on-time marker character will arrive almost simultaneously with a clock interrupt and small fluctuations in the arrival time may produce jitter of a full tick in the arrival times as measured by the algorithm software. In some implementations, the inner portion of the driver runs only every few ticks so as to process characters in bunches (in the name of efficiency), and this obviously makes the jitter in the arrival time that much worse. These problems will add white noise to the time measurements if they result in jitter that is less than a few milliseconds. The jitter will result in a systematic offset as well if it exceeds this value, because the ACTS system may drop out of the mode in which it advances the on-time markers to correct for the measured delay to a mode in which it locks the advance at a constant value of 45 ms independent of the true delay. The resulting systematic error in the delay measurement may be tens of milliseconds.

I can address the race condition in which the local tick and the on-time marker arrive almost simultaneously by adding a fixed time offset to the lock algorithm so that the local time is offset by one-half of a tick from the time in the ACTS message. This replaces the race condition by a systematic offset on the order of one-half of a tick. While this is not an optimum solution, the processing delay is likely to be stable enough to keep the ACTS system running in its measured-delay mode. There is no good solution to the multi- tick latency problem; I have modified the serial-line drivers of one vendor to reduce the magnitude of this problem, but such modifications require access to the source code.

VI. Initial Start-up

The first step in running the algorithm on a new machine is to estimate the performance of the clock oscillator by running the algorithm software in a calibration-only mode which periodically measures x_k but does not uses these data to adjust the local clock. Typical calibration experiments consist of making 10 consecutive time difference measurements 1 s apart to estimate the measurement noise itself followed by 100 time differences spaced about 1 h apart to measure the two sample Allan variance as a function of time lag out to lags of 30 or 40 h. (The maximum lag is limited to one-third of the length of the data set for statistical stability.) This is usually sufficient to show the point at which white frequency noise ceases to be the dominant noise process. All of the parameters of the algorithm can be estimated from these data using the procedures that I have outlined above. This analysis is usually necessary only once since the parameters of the oscillator change very slowly with time.

Subsequent cold starts reestimate the average frequency of the oscillator by letting it free run for three calibration cycles. If the prediction error during the restart procedure is less than 3 times the value expected from the free-running calibration tests, then the algorithm advances to the locked mode and begins operating as described above. If this is not the case the cold-start procedure re-estimates the parameters and advances to the locked mode if this effort is successful. Otherwise the algorithm will remain in the start-up procedure indefinitely. This is probably indicative of a hardware failure.

VII. Implementation of the Algorithm—Details and Examples

The underlying deterministic model of the algorithm does not change when the computer time is being adjusted, but the details are different. I consider three important effects:

1) Since the time of the local clock is being continuously adjusted in accordance with (4), the periodic measurements with respect to ACTS do not see a free-running clock, but rather one whose apparent frequency during the interval between two calibrations has been changed by $\overline{y}_k$.
2) Since the goal of the algorithm is to drive x_k to zero, $\hat{X}_k = 0$ for all k; to the extent that the algorithm is successful, it is also true that $x_k = x_{k-1} = 0$ as well.
3) If x_{k-1} was large, then a time step was sensed during the previous calibration cycle, and this time step was removed by a special time adjustment. The current prediction therefore should not use the measured value of x_{k-1} but a value of 0.

As a result of these considerations, the implementation of the algorithm is modified to include time steering as well as frequency steering. In addition to the periodic adjustments at intervals of t_k s, there is an adjustment of magnitude x_k after each calibration cycle. The periodic adjustments correct the average frequency of the local clock and the discrete time adjustments remove the short-term fluctuations in the frequency and keep it on time. As a result, $x_{k-1} = 0$. (The only difference between normal operation and what happens when a time-step is detected is that the frequency update loop is bypassed on a reset.) Equation (1) becomes

$$\overline{y_{k-1}}\tau_k = 0 \tag{8}$$

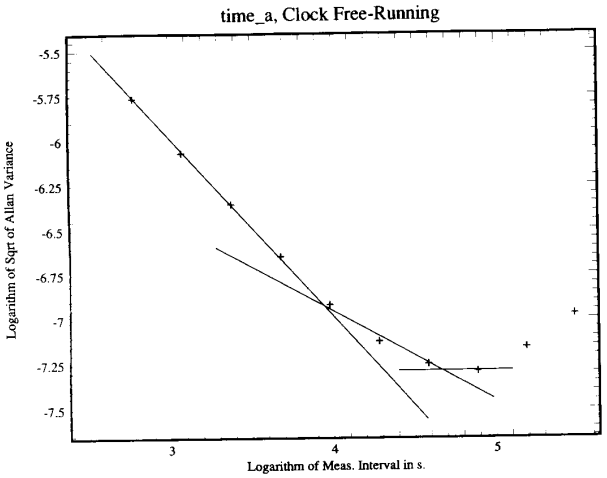

Fig. 1. The two-sample Allan deviation for time_a when its clock is free-running. The reference lines drawn through the points have slopes of –1, –0.5 and 0, showing the domains where the fluctuations in the oscillator can be characterized as white phase noise, white frequency noise, and flicker frequency noise, respectively.

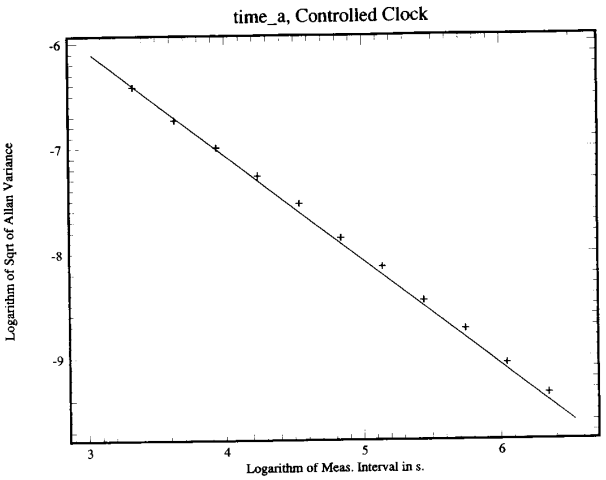

Fig. 2. The two-sample Allan deviation for time_a when its clock is being controlled by the algorithm described in this paper. The reference line drawn through the points has a slope of –1, which is characteristic of white phase noise.

and (2) is replaced by

$$y_k = \overline{y_{k-1}} + \frac{X_k}{\tau_k}. \tag{9}$$

Equations (3) and (4) are not altered.

I can illustrate the algorithm by showing its performance on two computers. The first is a machine whose name is time_a. The first step in implementing the algorithm is evaluating the free-running performance of the clock oscillator. Consecutive time differences spaced 1 s apart exhibited a scatter of 0.5 ms peak to peak. The time of time_a with respect to ACTS was then measured every 600 s, and the square root of the two-sample Allan variance of these data are shown in Fig. 1. Reference lines with slopes of –1, –0.5, and 0 are drawn through the points, and the transition zones between different noise types are clearly visible.

Although I always use these plots to determine the operating point that produces the best time performance, the plot can also be used to estimate the operating point that would produce a lower operating cost. The last point in the figure shows that the frequency jitter is about 1×10^{-7} at a measurement interval of 307 200 s. If that interval was used as τ_0, the frequency jitter in the oscillator would result in a time jitter of about 0.03 s. This is about 30 times worse than the performance I have achieved, but the telephone charges have been reduced by about a factor of 100 (from one call every 3000 s to one every 300 000 s).

The algorithm on time_a has been running for about 110 days; the x_k values for this period have a mean of 0.1 ms and a standard deviation of 0.16 ms. The square root of the two-sample Allan variance of these data are shown in Fig. 2. A reference line with a slope of –1 has been drawn through the data for comparison, and the data exhibit this slope over the entire range, suggesting that the residuals of the locking algorithm are pure white phase noise, which is the optimum performance that can be realized using the existing hardware and measurement method. Note especially that the noise which dominates the performance of the free-running oscillator at longer periods has been totally removed. A Fourier transform of the x_k values shows that the data are not quite white, and that there is a weak peak near 1 cycle/day with an amplitude of 0.1 ms. This is presumably the response of the oscillator to temperature (and possibly to input voltage), and running the system in an environment that had better temperature stability (the temperature in the computer room varies by $\pm 2°C$) would probably improve the performance of the oscillator.

The second machine is named baldwin. It suffered initially from the latency problems in the serial line driver that I discussed above, and I eventually had to modify the driver to obtain acceptable performance. The free-running performance of its clock oscillator is shown in Fig. 3. Reference lines with slopes of –1 and –0.5 are added to the points for reference. The performance at short-term is roughly similar to that of time_a, but its performance at longer periods is substantially worse. A power spectrum of its time differences shows a distinct peak near 1 cycle/day which is presumably driven by temperature with additional peaks at periods of several hours of unknown origin. A formal analysis of these data lead us to use roughly the same parameters on baldwin as I used on time_a but the performance will not be as good because the underlying hardware is noisier. This is confirmed by Fig. 4, which shows the square root of the Allan variance of the oscillator when the time is being adjusted by the algorithm. The RMS noise is about four times worse than time_a.

VIII. Comparison with NTP

I have conducted several experiments to compare the performance of this algorithm with NTP. Each of these experiments used two machines that were physically close together on the same network. The first experiment used two machines named baldwin and pogo. The time of baldwin was controlled by

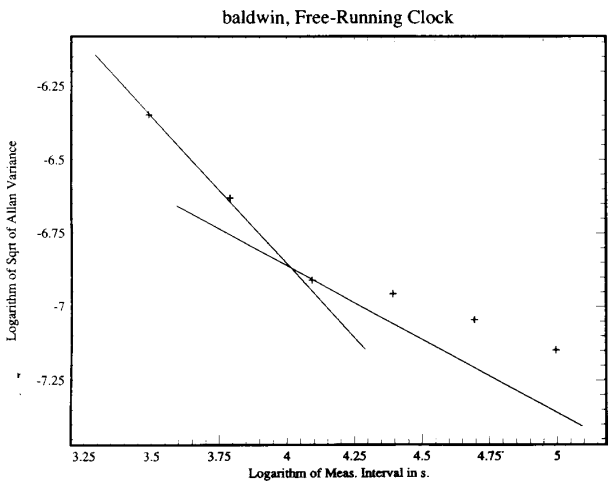

Fig. 3. The two-sample Allan deviation for baldwin when its clock is free-running. The reference lines drawn through the points have slopes of −1 and −0.5, showing the domains where the fluctuations in the oscillator can be characterized as white phase noise, and white frequency noise, respectively. Note the excess noise at intermediate periods.

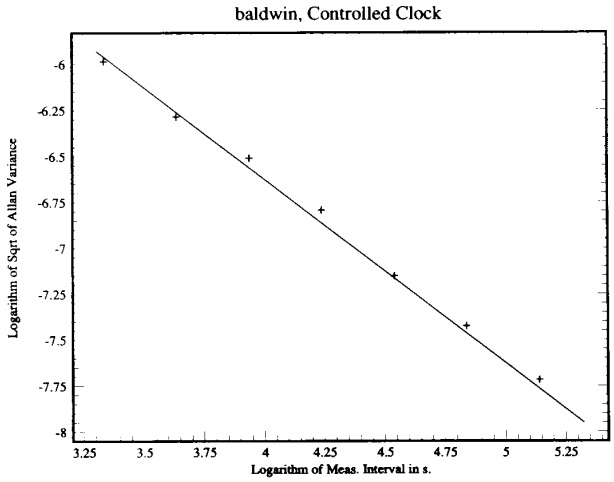

Fig. 4. The two-sample Allan deviation for baldwin when its clock is being controlled by the algorithm described in this paper. The reference line drawn through the points has a slope of −1, which is characteristic of white phase noise.

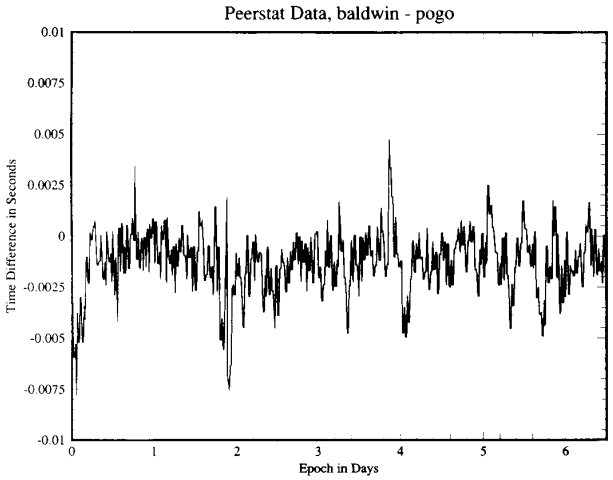

Fig. 5. The time difference between baldwin and pogo measured by NTP on pogo.

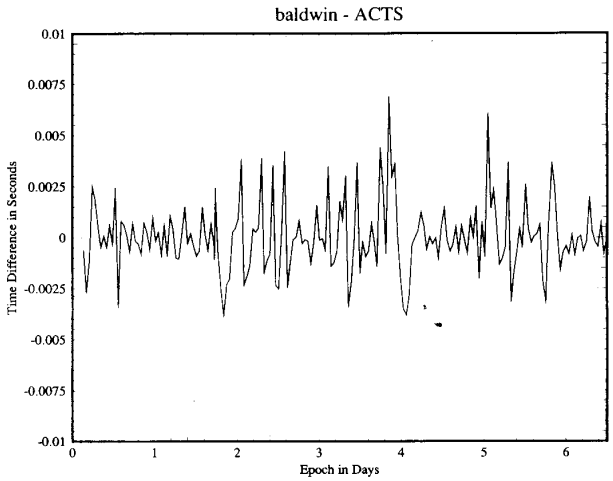

Fig. 6. The time difference between baldwin and ACTS. The limits of this plot are identical to those of the preceding figure.

dialing ACTS periodically and using this algorithm to adjust the clock; the time of pogo was controlled by NTP using a primary radio-clock for the time reference. I assumed that the variations in the time of pogo were small enough so that they could be neglected. The time of baldwin with respect to pogo as estimated by NTP was compared with the time difference measured by calling ACTS. The two time series are shown in Figs. 5 and 6. The overall agreement is very good – the difference is not greater than 2.5 ms anywhere and is generally much less than this value. I repeated this experiment with a second pair of identical machines named time_a and time_b. These machines had better clocks than baldwin and the agreement between the two methods was much closer, as discussed below. (The short-term stability of the clock is important because the two measurements via ACTS and NTP are not simultaneous.)

My final experiment compared the adjustment algorithm that I describe here with the clock model of NTP. The experiment was conducted in three phases using time_b. In the first phase, time_b was locked to a weighted average of the times of time_a (a machine that is very close by and is on the same local network cable) and to pogo (a machine that is about a dozen hops and 1500 km away on the Internet backbone) using NTP operating according to its standard specifications for this multiple-peer configuration. I assume that the path between time_b and pogo is a typical Internet path. In the second phase, time_b was locked via NTP using only time_a as a peer and in the third phase, time_b is locked via periodic calls to ACTS.

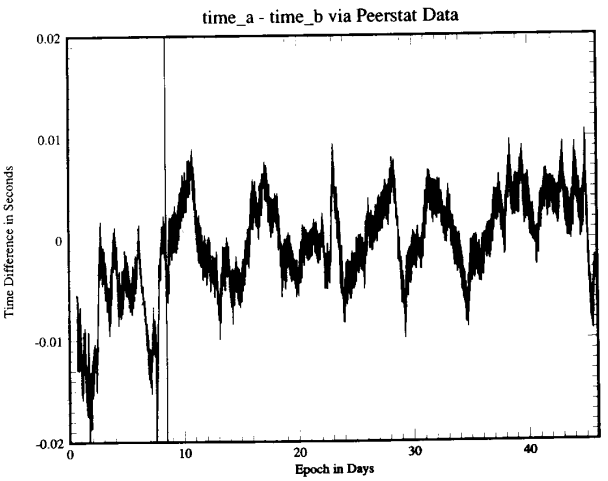

Fig. 7. The time difference between time_a and time_b measured by NTP on time_b. The vertical line near day number 8 shows the transition between phase 1 and phase 2 of the experiment as described in the text.

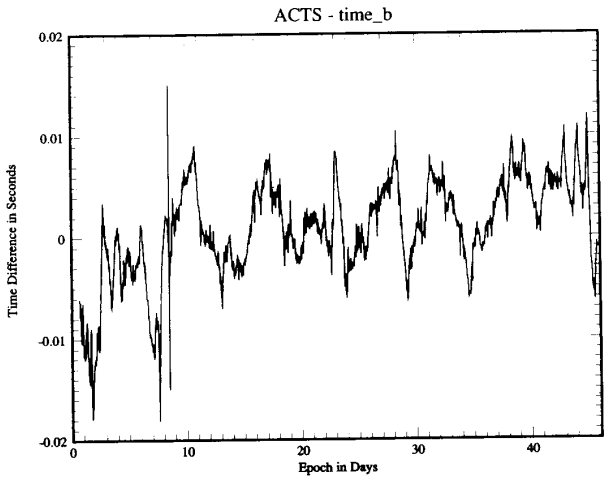

Fig. 8. The time difference between ACTS and time_b. The line near day 8 shows the transition between phase 1 and phase 2 of the experiment as described in the text. The limits of this plot are identical to those of the preceding figure.

Fig. 7 shows the NTP estimates of time_a-time_b for the first two phases of the experiment and Fig. 8 shows the estimates of ACTS-time_b for the same time period. The clock on time_b is better than the one on baldwin and the two plots agree almost perfectly. The vertical line on both plots near day 8.5 shows the transition between phase 1 and phase 2. Removing the peer that was separated by a transcontinental connection produced a change in the mean value of time_b by a few milliseconds and possibly some change in the character of the high frequency fluctuations. Fig. 9 shows the transition between phase 2 and phase 3—the switch between a clock locked using NTP and the same clock controlled using the current algorithm. The switch-over was made near day 27 (first vertical line) and the algorithm switched to its locked mode about a day later (second vertical line).

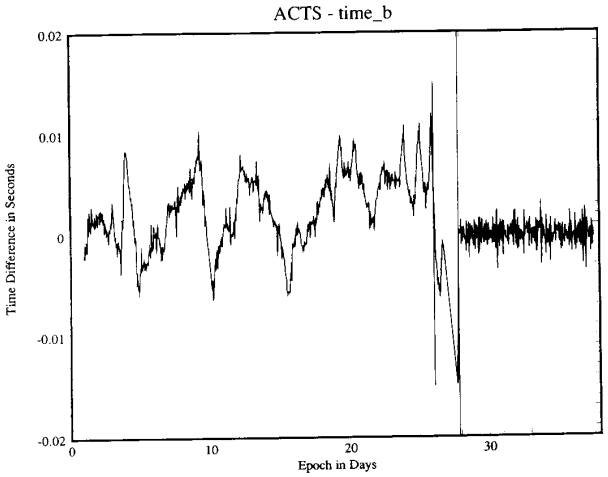

Fig. 9. The time difference between ACTS and time_b. This figure is a continuation of Fig. 8; the origin of this figure is at day 20 of the Fig. 8. The first line near day 27 shows the transition between phase 2 and phase 3 of the experiment as described in the text, and the second, longer line shows the point at which the local algorithm switched from estimating the parameters of the local clock to adjusting its time.

The current algorithm produces much quieter performance than NTP, especially at intermediate periods. I think this is probably due to the advantages of using ACTS, which does not suffer from the intermediate-period fluctuations in delay that characterize a typical Internet path. The short-term performance is somewhat worse than NTP, and I think this is due to the fact that the nominal calibration interval of 3000 s is significantly longer than the corresponding NTP parameter. Neither the Internet nor the ACTS system can have fluctuations at the very longest periods so that the performance of the two algorithms become very similar at periods of a few weeks or longer.

The algorithm I have described in based on a frequency-lock approach, and this is fundamentally different from the phase-lock approach that is the basis of the clock model used in NTP. While I do not think that either algorithm is fundamentally "better" or more powerful in an a-priori sense, the frequency lock approach is better suited to the noise characteristics of the crystal oscillators that are usually used in computers and is therefore able to provide better (or at least comparable) performance with much less frequent calibrations.

The performance of any algorithm will always be limited by the underlying noise in the clock and the calibration channel, and a statistically optimum algorithm must be able to separate the two contributions to the noise budget in order to get the best performance. A frequency-lock algorithm based on ACTS has a distinct advantage over NTP here. The noise process in the ACTS channel is dominated by white phase noise which is most important at high Fourier frequencies, while the noise process in the clock is dominated by flicker and random-walk frequency-modulations, which become more important at lower Fourier frequencies. The contributions of the clock and the channel to the noise budget can be cleanly separated based on Fourier frequency. A simplistic way of saying this

is that fluctuations with high Fourier frequencies are ascribed to channel while the low Fourier frequencies are assigned to the clock.

The variances of the clock and the channel do not separate so cleanly in the NTP environment. The channel noise tends to increase at lower Fourier frequencies, which is just where the clock needs more help because its noise spectrum is no longer white either. To complicate the problem, simple averaging is significantly less than optimum from a statistical point of view at these Fourier frequencies, since neither the conventional mean nor the conventional variance are well-defined for processes dominated by flicker or random-walk effects. it is difficult to say whether a frequency-lock loop would be better in this environment. The decision would depend on which parameter (time or frequency or perhaps something else) is closest to white and can therefore be safely averaged to yield a statistically robust estimate of the correction to be applied to the local clock.

IX. Summary and Conclusions

I have designed a new algorithm for controlling the time of a computer using periodic dial-up connections to the ACTS system at NIST. The algorithm can provide time performance that is limited primarily by the quality of the clock oscillator, although the latency in the serial-line driver can be a problem in some systems. Machines that are synchronized in this way can serve as primary network time servers, transmitting time to clients in NTP or in any other application where accurate and traceable time is needed. The machines need not be connected to any network; only a modem and a standard telephone line are required. The algorithm is not tied to any particular hardware or software environment, and I have implemented it on several commonly-available systems with no fundamental changes. The performance in all of the cases was comparable to the results I have presented here.

Acknowledgment

I am extremely grateful to D. Mills of the University of Delaware for his many kindnesses, for his advice and for allowing us to use his computers baldwin and pogo for part of this work. I am also grateful to E. Perkins of the University of Delaware for modifying the kernel on baldwin.

References

[1] J. Levine, M. Weiss, D. D. Davis, D. W. Allan, and D. B. Sullivan, "The NIST automated computer time service," in *J. Res. Nat. Inst. Standards, Technol.*, vol. 94, pp. 311–321, 1989.

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

[3] D. W. Allan *et al.*, "New inexpensive frequency calibration service from NIST," in *Proc. Symp. Frequency Contr.* (Baltimore, MD), June, 1990, pp. 107–116.

[4] D. B. Sullivan, D. W. Allan, D. A. Howe, and F. L. Walls, Eds., "Characterization of clocks and oscillators," Nat. Inst. Standards, Technol., Tech. Note 1337, Mar. 1990.

Judah Levine was born in New York City in 1940. He received the B.A. degree from Yeshiva College in 1960, and the Ph.D. degree in physics from New York University, New York, NY, in 1966.

He is currently a Fellow of the Joint Institute for Laboratory Astrophysics at the University of Colorado, Boulder. He has been with the Time and Frequency Division of NIST since 1969 and presently holds the rank of Physicist. His research interests include the design and implementation of algorithms to define the national time scales TA(NIST) and UTC(NIST). He has also been involved in developing new techniques for disseminating time and frequency information using telephone and computer networks and GPS satellites.

Dr. Levine is a Fellow of the American Physical Society and a Member of the American Association of Physics Teachers and the American Geophysical Union. He is an affiliate member of the IEEE.