TIME SYNCHRONIZATION USING THE INTERNET

Kenneth W. Monington
JILA, University of Colorado
and
Judah Levine
JILA, NIST and University of Colorado
Boulder, Colorado 80309

Abstract

We will discuss 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. We have tested the design 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. Our analysis can also be used to design algorithms that provide lower accuracy at lower cost.

Introduction

In this paper we discuss 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.

This work is a generalization of the "lockclock" algorithm, which we discussed in a previous paper [1]. That algorithm uses messages transmitted over dial-up voice-grade telephone lines from a telephone time server (such as the Automated Computer Time Service operated by NIST) to synchronize a client system. The current algorithm, which we call "interlock," performs about as well as "lockclock" but does not require access to a telephone line. Instead, it uses calibration information that is transmitted over a packet network such as the Internet. It is compatible with existing network standards and message formats, and can be used in a heterogeneous environment in conjunction with other algorithms (such as NTP, the network time protocol [2]).

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, while a stochastic parameter can only be specified using a statistical measure such as a variance or a spectral density.)

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. The register measures elapsed time from some system-defined origin. In almost all systems the oscillator hardware is free-running and is not under program control. Software processes of the type we will discuss 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 our discussion.

We characterize a clock with two deterministic parameters: 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

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

where

$$\Delta t = t_{k+1} - t_k. \quad (2)$$

Although quartz-crystal oscillators (which are the type almost always used in computer clocks) usually have significant frequency aging, it is difficult to estimate it in the presence of white frequency noise and we therefore do not include it in the model.

In addition to these deterministic parameters, the frequency of the oscillator fluctuates stochastically. These fluctuations can be characterized as a "white" Gaussian 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, we can observe the oscillator frequency only through its effect on the time. The integration of the frequency implied by eq. 1 means that the spectral density of the time fluctuations resulting from almost any kind of frequency noise is not white. In other words, the time fluctuations produced by frequency noise can never be characterized by a mean and a standard deviation; while these parameters will always exist in a formal sense, their values will not necessarily correspond to our 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.

The Measurement Process

The measurement process involves comparing the time of the clock with the time transmitted over the network. The delay in making these measurements 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. This delay has a jitter due to variations in system load and interrupt latency. We will assume that the factors that drive these fluctuations vary rapidly with time, so that the variations in consecutive delays are largely uncorrelated with each other even for measurements made only a few seconds apart. This jitter about the mean is therefore a random variable with a reasonably well-defined standard deviation; we call this jitter white phase noise, by analogy with the analogous fluctuations in hardware clocks.

These fluctuations exact a price from any client process that uses the clock to time-tag an event. Since a time-tag involves a single reading of the clock, the fact that the distribution of an ensemble of such measurements would be approximately Gaussian cannot be exploited. However, this Gaussian distribution 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.

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 we receive the response at time t_4. The round-trip delay due to the network path is

$$\Delta = (t_4 - t_1) - (t_3 - t_2). \quad (3)$$

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

$$x = (t_1 + d) - t_2. \quad (4)$$

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

$$x = \frac{(t_1 - t_2) + (t_4 - t_3)}{2}. \quad (5)$$

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

$$\varepsilon = (k - 0.5)\Delta, \tag{6}$$

which depends both on the path delay and on its asymmetry.

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. We will therefore assume 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

$$\sigma^2 \sim \left\langle \varepsilon^2 \right\rangle - \left\langle \varepsilon \right\rangle^2 \tag{7}$$

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.

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. [3] 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 is therefore easily satisfied, even if the consecutive requests are separated by only 100 ms. Furthermore, the Lindeberg condition [4] 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 definition of the standard deviation in this case, since 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.

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 equation 1 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 we are considering. 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.

The Allan variance, [5] 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 is therefore crucial to designing an optimum synchronization procedure. In addition, the spectral density of the noise is

often an important indication of its source. See ref. 1 for more details.

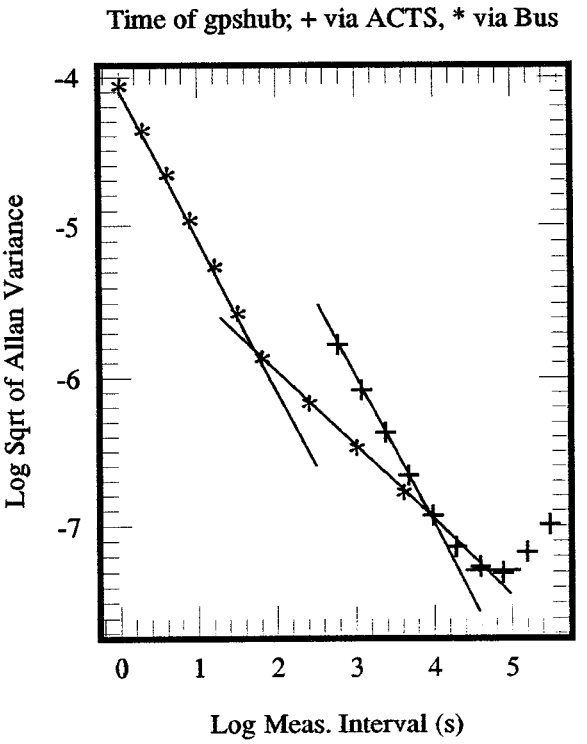

Fig. 1. Allan deviation measured using the ACTS dial-up system (+) and via a direct connection to the computer bus (*). The straight lines have slopes of –1, -0.5 and 0 to illustrate the various noise types.

The 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 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 we discussed above.

For both measurement techniques, the initial slope is –1, showing that both are limited by white phase noise at short times. The magnitudes are quite different in the two cases. The directly-connected hardware device has a white phase noise level of about 80 μs, while the link to the ACTS system has a noise level of about 1 ms. We are not looking at the oscillator in either case. This 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, while 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.

Since this is 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. We can decrease our uncertainty of the time difference by averaging consecutive readings since we are sampling a Gaussian 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, since 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 and must therefore be removed before the average is computed.

Note that 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 the measurements acquired in the 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 Gaussian random variable. 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. The transitions to white frequency noise occur at different values for the two measurement schemes, but the two data sets lie on almost exactly the same line after it. This is to be expected. Once we are limited by the frequency stability of the local oscillator (and not by the measurement process), the noise in the measurement process must not matter anymore, and any measurement process that can achieve this goal will result in the same longer-term performance.

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 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. (Ref. 1, equation 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.

The Network

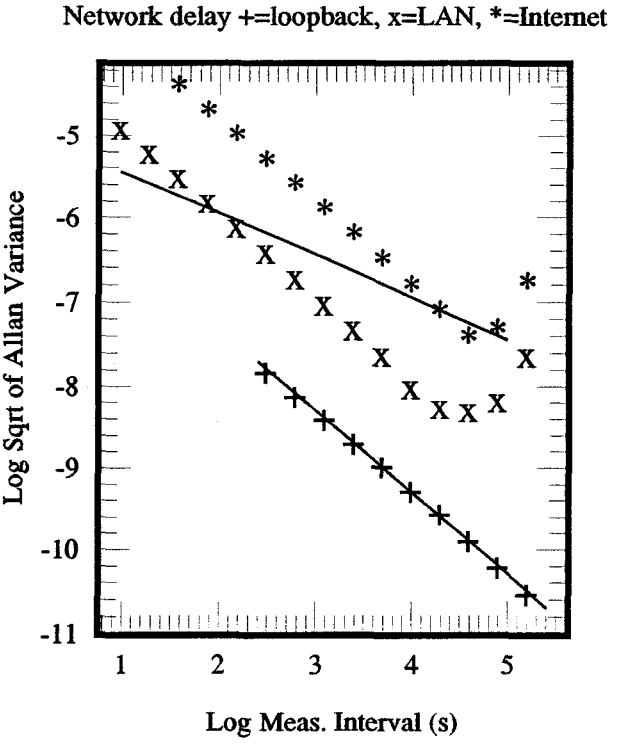

Fig. 2. The Allan deviation of the network delay; loop-back to the sending machine (+); machines on same local area network (x); machines on the Internet, 1200 km apart (+). Line of slope –1 for reference and the line of slope –0.5 is copied from fig 1.

As we 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. We conducted a series of experiments to verify that this is true; the results are presented in fig. 2.

We sent packets 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 equation 5. We repeated this process 25 times with requests spaced one second apart and averaged these results to form a single time-difference point. We then repeated this entire process 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. The Allan deviations of these measurements are shown in fig. 2.

The bottom set of points (with the symbol "+") is from a loop-back test, so that the fluctuations in the clock itself are irrelevant. (A straight line with a slope of –1 is shown superimposed on these data for reference.) The data are characterized by white phase noise over the entire time domain; the magnitude is about 240 μs, which represents the fluctuations in the symmetry and delay in the machine itself.

The next two plots 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 "*"). All of the data are clearly limited by 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 we 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.

We limit our adjustments of the clock 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 local clock itself. 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.

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 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. Since 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 ref. 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 $180\sigma_y(180) \approx 200\mu s$, 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.

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. The performance of this loop would not be as good as the first one, of course, because the prediction error increases as $\tau^{0.5}$ in the white frequency noise domain. Since the network noise is so high, the clock would be essentially free running for averaging times less than about 5 h.

We could improve matters 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 we kept 25 time-differences in each group, the best we could do would be to estimate the frequency using measurements about every 5.3 h (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 $19000\sigma_y(19000) \approx 1.5$ ms. This is comparable to the uncertainty that we achieved with "lockclock", which is not surprising. As we 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 that we expect 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 we can obtain 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. Since 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 h to 12 h would decrease the cost by a factor of about 20, while still providing synchronization with an uncertainty of better than 30 ms rms.

Detecting False-tickers

A "false-ticker" is a time server whose status is shown as healthy but is nevertheless transmitting the wrong time. This "should never happen," but a client must be prepared for it. The simplest strategy for detecting this problem is to get a second opinion from a different time-server, and then a third opinion to allow a decision by majority voting. Clearly, this can get pretty expensive. It potentially doubles (or maybe triples) the load on every server.

Another strategy is to use time of the local clock itself to validate the received data. If it has been τ seconds since our last calibration, then the time dispersion of the local clock is of order $\sigma_y(\tau)\tau$, 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. 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:

1. The server is broken.
2. Our clock has experienced a time-step.
3. Our clock has experienced a frequency step.
4. 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 we 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 2 and 3 and so we correct our clock based on the time measurements and wait for the next cycle.

If we find another error of comparable magnitude (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, and then 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 our diagnosis of a frequency step is probably correct. If not, then we probably have a hardware problem which may require outside assistance.

Finally, 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 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. They are masked by the white frequency noise of the oscillator in the short term and can only be detected 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.

Tests and Results

We have used the methods described above to synchronize the clock of a workstation named *strain,* which is located in our laboratory in Boulder, Colorado. We chose a network time server located about 1200 km away in Seattle, Washington as a time reference. The one-way delay is about 48 ms and is quite variable.

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 adjustment was limited to less than one percent of the clock rate to provide very small time adjustments, and also to ensure that the clock did not stop or run backwards.

Using this procedure, the rms prediction error (the difference between the average measured time-difference at some epoch and the value predicted using equation 1) is 0.92 ms – somewhat better than the noise level we would expect using the "Internet" delay characteristics of fig. 2 because we increased the size of each group to 50 messages rather than 25.

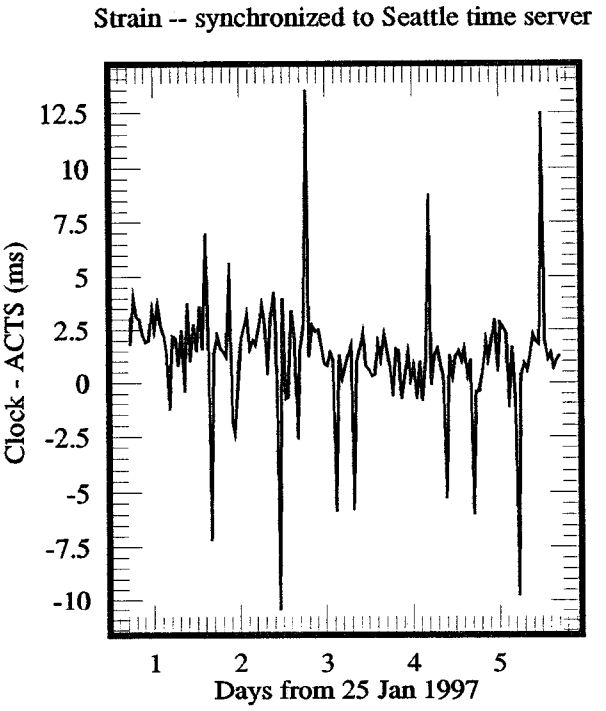

Fig. 3. The time of strain - UTC measured using the NIST ACTS system. The noise in the ACTS comparisons is about 1 ms; fluctuations of that order or smaller are not significant.

We evaluated the accuracy of the procedure by comparing the time of strain to UTC using periodic calls to the NIST ACTS system (fig. 3). The noise and the accuracy of the ACTS comparison are about 1 ms, so that differences on this order or less are not statistically significant. The Allan deviation of these data are shown in fig. 4, 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. Residual fluctuations in the network asymmetry that are not completely removed by the averaging also contribute.

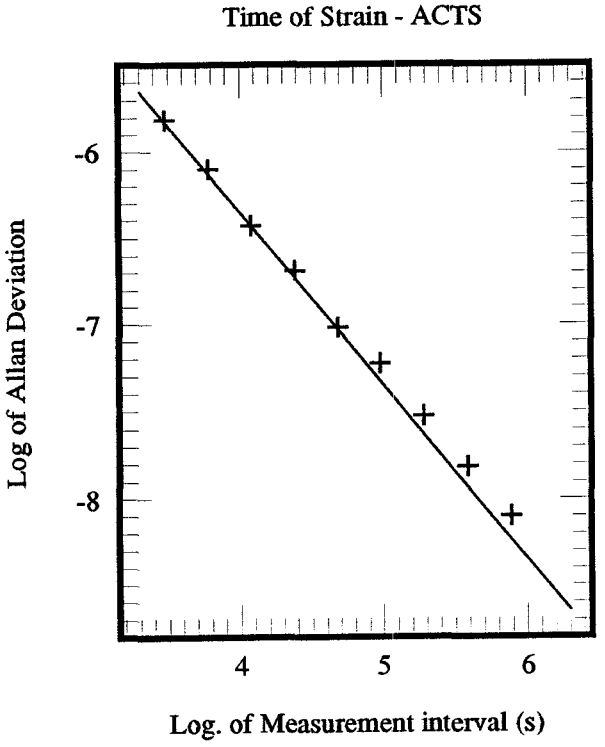

Fig. 4. The Allan deviation of the data shown in fig. 3. The straight line has a slope of –1. The points deviate from this straight line starting at a period of about 1day, due to the long-period fluctuations that are visible in the data set.

Conclusions

We have discussed the considerations that govern the design of algorithms used to synchronize computer clocks using messages transmitted over a packet network such as the Internet. We have used this method 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. We used only one server for this study, although we outlined procedures for querying a second server when the first one seems to be broken. The procedures for adjusting the time of the local clock are the same as we described previously in reference 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 can therefore provide near "stratum-1" performance using only network data.

In addition, the procedures we have discussed provide a quantitative way of evaluating the trade-off between synchronization accuracy and cost. As we have shown, the cost/benefit ratio is not linear – an increase in accuracy by a factor N requires and increase in cost proportional to N^2. A moderate relaxation in the synchronization accuracy can therefore 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.

References

[1] Judah Levine, "An Algorithm to Synchronize the Time of a Computer to Universal Time," *IEEE/ACM Trans. on Networking,* Vol. 3, pp. 42-50, 1995.

[2] D. L. Mills, "Network Time Protocol (version 3); Specification, Implementation and Analysis," DARPA Network Working Group Report RFC-1305, Univ. Delaware, 1992.

[3] John R. Taylor, "An Introduction to Error Analysis," Mill Valley, California: University Science Books, 1982, Ch. 10, pp. 197-199.

[4] C. W. Gardiner, "Handbook of Stochastic Methods," New York: Springer Verlag, 1990, Ch. 2, pp. 37-39.

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