# SYNCHRONIZING COMPUTER CLOCKS USING A LOCAL AREA NETWORK

Judah Levine
Time and Frequency Division
and
Joint Institute for Laboratory Astrophysics
National Institute of Standards and Technology
Boulder, Colorado 80303

***Abstract***

***We have completed the first tests of a method to synchronize the clocks of networked computers to the NIST time scale. The method uses a server computer to disseminate the time to other clients on the same local–area network. The server is synchronized to NIST using the ACTS protocol over a dial–up telephone line. The software in both the server and the clients constructs a statistical model of the performance of the local clock and of the calibration channel, and the parameters of this model are used to adjust the time of the local clock and the interval between calibration requests in a statistically optimum way. The algorithm maximizes the time between calibrations while at the same time keeping the time of the local clock correct within a specified tolerance. The method can be extended to synchronize computers linked over wide–area networks, and an experiment to test the performance of the algorithms over such networks is being planned.***

## Introduction

In 1988, the National Institute of Standards and Technology (NIST, formerly the National Bureau of Standards) introduced the Automated Computer Time Service. This service is a dial–up telephone service designed to provide digital access to the NIST time scale at accuracies approaching 1 ms. The service transmits time information at either 300 bits/s or 1200 bits/s. All of the lines support both transmission formats using standard modems. The data are transmitted using standard ASCII characters with 7 data bits, space parity and 1 stop bit. The current telephone number for the service is (303) 494–4774 (not toll–free). The transmitted data include the day number, the civil date and time and flags that give advance notification for the insertion of leap seconds and for the transition to and from Daylight Saving Time. See Levine et al. (1989) for more details.

In this paper we report on preliminary tests of a system to synchronize the time of networked computers that is based on this NIST service. Our method uses server computers to disseminate the time on the computer network. The servers in turn are synchronized using periodic calibrations from the NIST time scale. One of the most important features of the system is the statistical model that is incorporated into the software in both the server and client machines. These models dynamically adjust the interval between calibrations based on the desired synchronization accuracy, the jitter in the calibration link, the cost of each calibration and other factors. They make optimum use of the calibration data and optimize the performance of the network based on the specified accuracy requirements.

## The Network Servers

The network servers are general-purpose computers and use a standard commercial multi–processing operating system. They are linked to a local–area network using standard hardware and also have an external modem that is connected to a voice–grade telephone circuit. The interface between the computer and the modem uses one of the RS–232 serial ports of the system. The time of the server is advanced automatically in response to hardware interrupts generated by an internal crystal–controlled oscillator. The interval between interrupts is 10 ms for our hardware, but periods ranging from 1 ms to 50 ms are often found in hardware from other suppliers. In addition to these hardware interrupts, the time of the server is adjusted by the software as described below. These adjustments are made to the software–maintained time registers only; the actual oscillator cannot be directly controlled.

The server software consists of two parts: the process that sets and maintains the time of the local clock and the process that transmits the time to other machines on the network. These two processes are independent, although they share access to a global set of parameters that define the state of the server. Both of the processes are normally activated as part of the start–up of the server. The process that controls the local clock runs as a dæmon in the background, while the transmitter is activated whenever a calibration request is received from a client machine. Neither process uses a significant fraction of the resources of a modest–sized workstation.

The clock control dæmon is divided into three sub–processes:

1. A process that measures the time difference between the server and NIST. The comparison is made using the dial–up telephone line and the ACTS protocol. The telephone connection time is typically 15 s. The accuracy of this calibration is about 10 ms and the repeatability is about 0.3 ms. This repeatability is achieved by careful design of the process, especially the algorithm to interpolate between the ticks of the local clock. This algorithm can reliably resolve intervals on the order of 1% of the period of the local clock (less than 1 ms on most systems).

2. A process that uses the time difference data to construct a model of the local oscillator. The principal parameters of the model are:

   | | |
   |---|---|
   | $y$ | Clock Rate in seconds of deviation/second |
   | $D$ | Change in Clock Rate seconds/second$^2$ |
   | $\Delta$ | Interval between external calibrations |
   | $\epsilon$ | Clock Noise |
   | $\eta$ | Channel noise |
   | $E_m$ | Maximum time error |

   The model dynamically adjusts the first three parameters, ($y$, $D$ and $\Delta$), to optimize the performance of the clock in the presence of the two noise sources, $\epsilon$ and $\eta$. Specifically, the model attempts to maximize the interval between calibrations, $\Delta$, while at the same time keeping the local time correct within the bounds specified by $E_m$.

3. A process that uses the model parameters to continuously correct the local clock. These corrections are implemented as 1 ms time steps in the appropriate direction. The interval between these corrections is at least 10 ms, so that the steering does not exceed 10% of the clock rate.

## The Clock Model

The free–running performance of the server clock is modeled by

$$\begin{aligned} x(t_{j+1}) &= x(t_j) + y \times (t_{j+1} - t_j) \\ &\quad +0.5 \times D \times (t_{j+1} - t_j)^2 + \epsilon. \end{aligned} \tag{1}$$

This expression relates the time difference, $x$, between the server and NIST at time $t_{j+1}$ to the difference at some earlier time $t_j$. Note that $\Delta = t_{j+1} - t_j$, and that $\Delta$ is not constant but is adjusted by the algorithm. The server constructs initial estimates for $y$ and $D$ by several successive calibrations separated by a time interval of between 1 and 2 hours. This interval is chosen long enough so that the channel noise $\eta$ will not make a significant contribution to the rate and short enough so that $D$ can be neglected and the clock noise $\epsilon$ can be modeled as a random process (even though all oscillators have a noise spectrum that shows increased variance at longer periods).

If the data are statistically consistent, then the server changes to a phase–lock mode. The time of the local clock is set (usually by slewing it at the maximum permitted frequency). When the local clock is on time, the rate estimate is used to apply periodic corrections to it as described above.

Subsequent calibrations are used to refine the parameters of the model. The calibration interval is gradually lengthened until the rms error of the model approaches the value of $E_m$. For any combination of oscillator and calibration channel, there is generally an optimum calibration interval and a corresponding minimum time error, $E_m$: the performance at shorter intervals is degraded because the random component of the measurement noise degrades the rate estimates, and the performance using longer intervals is degraded because the low–frequency components of the oscillator noise spectrum are not adequately predicted by the model. This optimum interval is usually about 24 hours for the crystal oscillators that are normally used in computer clocks, and the corresponding minimum time error is about 1 ms.

One important aspect of the algorithm is the reset logic. Most oscillators exhibit occasional glitches that are much larger than the standard deviation of their performance. Similar effects are often found in the performance of the calibration channel. It is important that these effects be recognized as unusual and that they not be allowed to corrupt the statistical model of the oscillator. If the error of a calibration exceeds twice the running average standard deviation for the previous day, then a reset algorithm is executed. The algorithm first repeats the calibration. If the two calibrations are statistically different, a channel error is assumed. If the two calibrations agree then either a time or frequency step is assumed. If the difference between the next two calibrations agrees with the long–term trend, then the problem is a time step only; if the subsequent calibrations are consistent with a new trend then a frequency step (including a possible time step) has occurred. If the subsequent calibrations cannot be modeled as a combination of a time step and a rate step, then a hardware failure is indicated. The algorithm attempts to re–initialize itself in this case; if that attempt fails then an unrecoverable error is signaled.

## Server Test Results

We have conducted several tests to demonstrate the capabilities of the method. Fig. 1 show the free–running performance of the clock in node TILT. The time of the computer is compared periodically with the NIST time scale using a dial–up connection and the ACTS protocol. The effect of the clock rate $y$, dominates the performance at this level.

The rate of the clock is then estimated using the first 4 hours of data, and the performance of the clock for the remainder of the month is predicted using (1). The subsequent difference measurements normally would be used to modify $y$, $D$ and $\Delta$, but this update loop was disabled for these tests.

Fig. 2 shows the residuals between the data of Fig. 1 and the predictions of the model. The short–term variance is 0.8 ms, (8% of a 10 ms tick) and the spectrum appears white. The performance of the ACTS system is considerably better than this (Allan et al., 1990), and we are probably being limited by jitter in both the modem equalizers and the interrupt latency of the server. This interrupt latency could be reduced by moving some of the code into the device driver for the RS–232 port, but we have not done this at this time. The longer–period fluctuations are due to changes in the rate of the clock oscillator; the relatively small diurnal changes probably result from temperature fluctuations while the longer term effect can be broadly characterized by a random–walk in frequency that is undoubtedly due to aging of the oscillator crystal. It is important to note, however, that the server does not deviate from the time predicted by the model by more than $\pm 20$ ms for the entire observation period. If this level of performance was satisfactory, the server could run for at least 1 month with no additional external calibrations once the rate had been estimated.

Fig. 3 shows the locked performance of node TILT. The parameters of the lock algorithm have been tuned to lock the time to the minimum error $E_m$ consistent with the channel and measurement noise spectra. The optimum calibration interval was about 18 hours, but $\Delta$ was fixed at 1.8 hours and the faster estimates were exponentially averaged to estimate the optimum rate.

## Local Area Network Tests

The network tests were performed using a relatively large local–area network on the campus of the University of Colorado. The network consists of segments of thick–wire cable within buildings with bridges to connect the various buildings together. The maximum distance between buildings is about 3 km.

Two independent servers (named STRAIN and TILT) were locked to the NIST time scale using the methods outlined above. These servers responded to requests from any host on the network by sending the time together with flags providing advance notice of leap seconds and of upcoming transitions to and from Daylight Saving Time. A single character specifying the health of the server was also transmitted.

There are several different versions of the client software. The simplest version simply requests the time from the server and uses the data to set the time of the client if the server is healthy. A more complex client program utilizes the same algorithm as was described above for the server to keep the time of the client within a specified tolerance using periodic calibrations from the server. The client algorithm uses a somewhat different characterization for the channel noise parameter $\eta$ since the client receives its calibration data over a packet network rather than over a single dial–up telephone circuit.

To test the performance of the network software, we used two servers and a single client. The client (named JILACK) periodically requests calibration data from both servers (named STRAIN and TILT) and computes the difference of the differences:

$$\begin{aligned}\delta t_s &= (\text{JILACK} + \epsilon_j - \text{TILT} + \eta_1) - (\text{JILACK} + \epsilon_j - \text{STRAIN} + \eta_2) \\ &\approx (\text{STRAIN} - \text{TILT} + \eta_1 - \eta_2)\end{aligned} \tag{2}$$

since the two requests are so close together in time that the fluctuations of JILACK can be neglected. Here $\epsilon_j$ is the unmodeled clock noise of node JILACK while $\eta_1$ and $\eta_2$ are the network delays between JILACK and the two calibration nodes. The times of nodes STRAIN and TILT are known with respect to the NIST time service since both are locked via periodic calibrations using the ACTS protocol:

$$\begin{aligned}(\text{STRAIN} - \text{ACTS}) &= E_s + \eta_{a1} \qquad \text{where } < E_s >= 0 \text{ and } < E_s^2 >= \sigma_s^2 \\ (\text{TILT} - \text{ACTS}) &= E_t + \eta_{a2} \qquad \text{where } < E_t >= 0 \text{ and } < E_t^2 >= \sigma_t^2\end{aligned}$$

Here $\sigma^2$ is the variance of the clock noise and $\eta_a$ is the noise in the ACTS calibration channel, which is assumed to be the same for both nodes (on the average). Thus the differential network delay can be estimated from

$$(\eta_1 - \eta_2) \approx \delta t_s - (E_s - E_t) + (\eta_{a1} - \eta_{a2}) \tag{3}$$

We found that the differential network delay varies by less than $\pm 1$ ms over the range of network loads we have observed and that $< \delta t_s >= 0 \pm 2$ ms.

## Expansion to Wide Area Networks

There is no difficulty in principle in expanding this system to wide area networks, and we are currently planning a wide–area network test. One practical limitation will be the characterization of the network delay parameter, $\eta$, which is likely to contain time–varying non–stationary components. It is also likely that the network delay will not be reciprocal so that the one–way delay between two points cannot be estimated by one–half of the round–trip time. Under these circumstances, a local–area network synchronized via a single server which is in turn calibrated using ACTS will have certain advantages over the same network synchronized using a wide–area network protocol because the telephone connections between the server and the NIST time scale are much easier to characterize and are much more likely to be stable and reciprocal. The additional cost of the charges for the toll calls is not high — one or two 15 s calls per day would be adequate for almost all applications, and lower–accuracy applications would require correspondingly less frequent calibrations.

## Conclusions

We have designed and tested a method that can be used to synchronize the time of computers on computer networks using periodic calibrations from the NIST time scale. The method makes optimum use of the calibration data by constructing statistical models of the performance of the oscillators in both the server and client machines. The calibration of the servers uses dial–up telephone connections which are highly reciprocal and stable and are therefore much easier to characterize than dissemination methods which depend on a wide–area network for synchronization.

## References

1. Allan, D. W., D. D. Davis, J. Levine, M. A. Weiss, N. Hironaka and D. Okayama, 1990. *New Inexpensive Frequency Calibration Service from NIST*, Proceedings of the Symposium on Frequency Control, in press.

2. Levine, J., M. Weiss, D. D. Davis, D. W. Allan and D. B. Sullivan, 1989. *The NIST Automated Computer Time Service*, J. Res. of NIST, 94, 311 - 321.

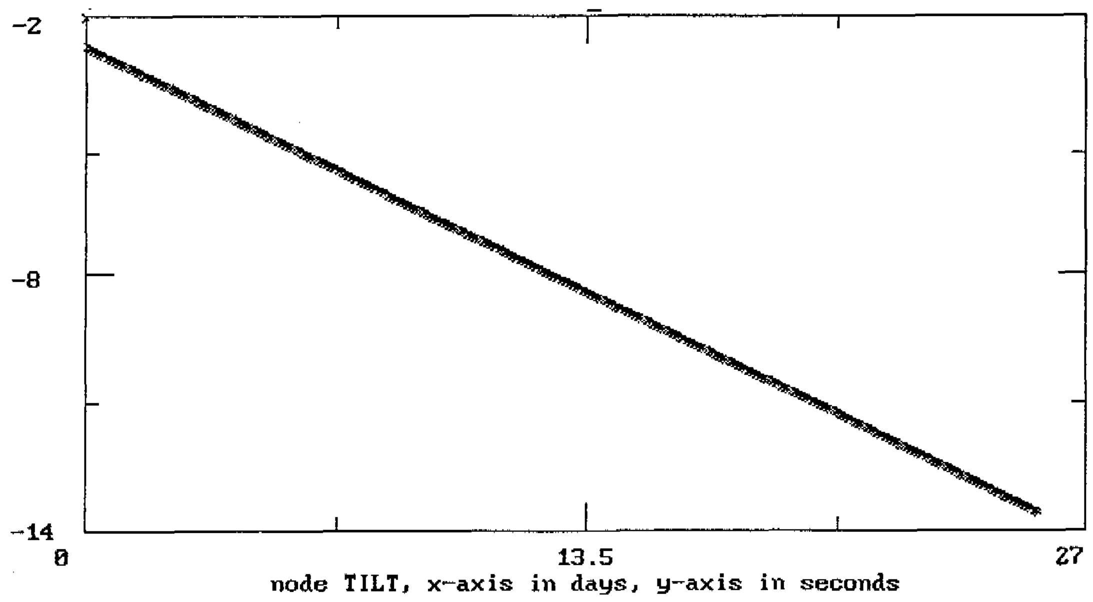

Fig. 1. Time of node TILT - NIST time scale. The time difference is measured using the ACTS time service over a dial-up telephone connection. The slope in these measurements shows the frequency offset of the free-running oscillator, and the deviations of the measurement line from a straight line are mainly due to measurement noise. (The width of the line is due to the size of the symbol used at each point.)

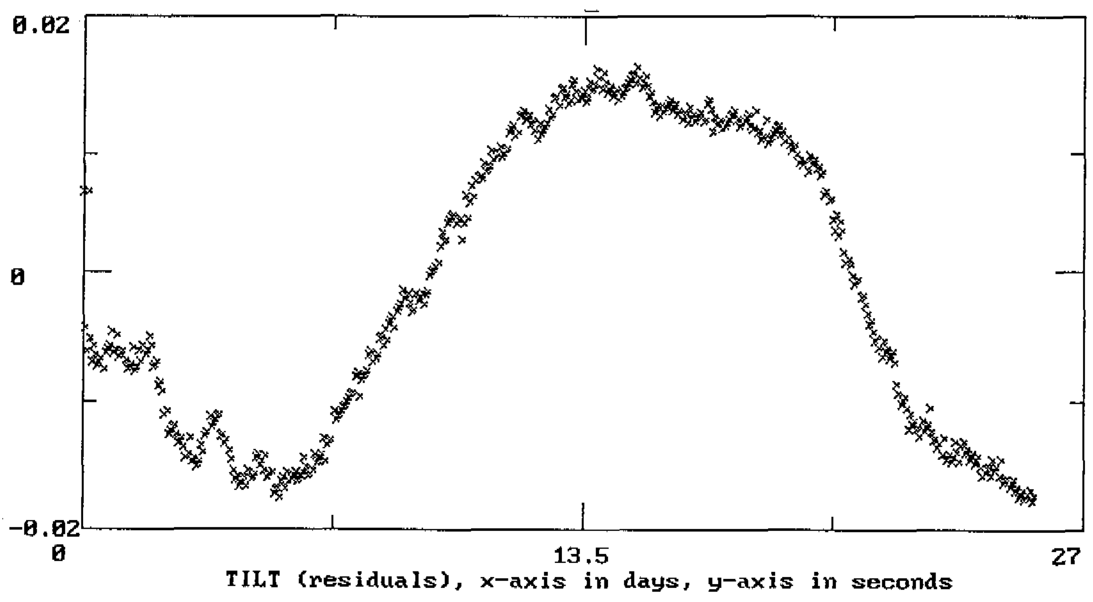

Fig. 2. Difference between the data in Fig. 1 and the predictions of the clock model. The rate of the oscillator was estimated from the first 4 hours of the data. The model parameters were locked at that point and the residuals between the modeled clock and NIST are shown in the figure.

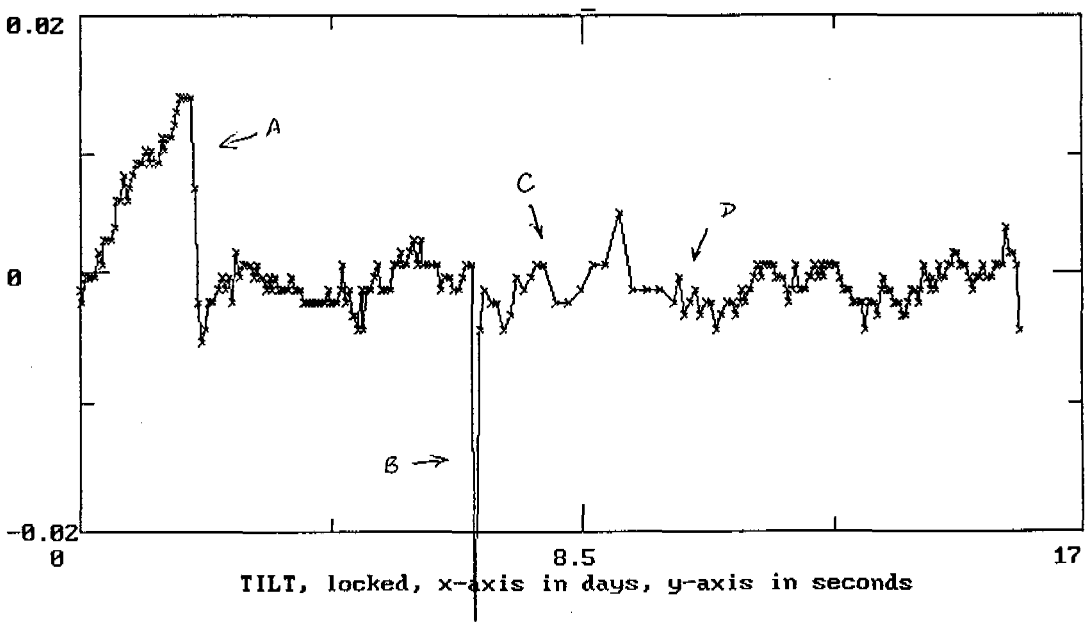

Fig. 3. Time of node TILT - NIST. The time of node tilt is adjusted in increments of ±1 ms. The interval between these software adjustments is derived from the model equation (1). The parameters of the model are updated periodically using calibrations via the ACTS time service over a dial-up telephone connection. Note the change from the parameter estimation mode to the locked mode at point A, the glitch at point B that was caused by the re-boot of the server following a short power failure. Note also the automatic adjustment of the interval between calibrations at points C and D.

## QUESTIONS AND ANSWERS

**Dr. Winkler, U. S. Naval Observatory:** I miss, in your discussion, the substantial difference in timing in the computers depending on the operating system. In the PC, which has an increment of time in the timing registers of 16 milliseconds, usually, and the actual processor runs at 5 or 8 or 12 megaHertz, but the registers are not updated faster than every 16 milliseconds. Compare that to the OS-2 or the Unix systems where the actual time interval kept is in seconds, and the real-time system such as the HP system 1000. They are all completely different. It has been our experience that it is better not to fool around with the computer time internally, but to do everything externally. Make the time measurements externally and use the interupts to read the external time. I wonder how the different operating systems will affect that.

**Mr. Allan:** I am sure that they will. We have done experiments on Micro-Vaxes and on a Sun system. The same technique would work on a PC, just at a higher level.