Synchronizing Computer Clocks Using Kalman Filters

Judah Levine
Time and Frequency Division and JILA
National Institute of Standards and Technology and the University of Colorado
Boulder, Colorado 80305
jlevine@boulder.nist.gov

Abstract -- I have used the Kalman Filter algorithm to improve the link between the Internet Time Servers operated by the National Institute of Standards and Technology (NIST) and the primary atomic clock ensemble in Boulder, which realizes UTC(NIST) and which is used as the reference for the time servers. The Kalman algorithm is better able to separate the contributions of multiple noise sources such as the fluctuations in the asymmetry of the channel delay and the statistical fluctuations in the clock used as the internal reference for the time server. This improved separation has made it possible to compensate to some extent for the lower-quality telephone circuits that are often used as the links for synchronizing the time servers to the atomic clock ensemble.

INTRODUCTION

The National Institute of Standards and Technology (NIST) currently operates 35 public network time servers that are located at 21 different sites in the United States. The servers provide time over the public Internet in a number of different formats. All of the servers are synchronized to UTC(NIST). The time servers are linked to the atomic clock ensemble in Boulder, Colorado by the use of a hard-wired connection for the systems that are located at the NIST Boulder laboratories and by dial-up telephone lines, which implement the ACTS protocol [1] which will be described below, for the other systems that are located at remote locations.

THE ORIGINAL ACTS SYSTEM

The ACTS system transmits time using standard dial-up telephone lines and modems. It is a two-way protocol, in which the one-way transmission delay between the server and the user is modeled as one-half of the round-trip value. The message from the server to the client contains a time stamp and an on-time marker character, which the client echoes back to the server with as little delay as possible. The internal system latency is on the order of microseconds, whereas the channel delay is typically tens of milliseconds, so that the system latency is small enough to be ignored. (Both the server and the client have operating system software that has been modified to guarantee that the system latency will be not larger than 15 microseconds even when the client is handling many requests for time from the network.) The round-trip delay is measured by the server as the time that has elapsed from when it transmitted the on-time marker to when it receives the echo from the remote system. The one-way delay is estimated as one-half of this value, and the next on-time marker is advanced using this one-way delay estimate so that it will arrive at the client on-time. The server inserts this one-way delay estimate into each message, so that the client can see the advance that was used. When the client receives the on-time marker, it uses the time stamp of that message to compute the difference between its clock and the clock on the server. This process is repeated every second until the telephone connection is broken. The client can break the connection at any time, and the server will automatically break the connection after 40 s if the client has not done so before this.

There are two important aspects to the design of the ACTS system. The first is that the advance that is applied to any on-time marker is derived from the previous round-trip measurement, so that fluctuations in the delay with a period close to the one-second interval between messages will not be handled correctly. The second aspect, which has more subtle consequences, is that the delay is measured by the server and not by the client. Neither of these was important when ACTS was first designed in 1988, but changes in the telephone system since that time have made both of them significant.

The advance used by the ACTS servers will be accurate and unbiased on the average if the asymmetry in the delay is close to 0 (so that the one-way delay is accurately estimated as one-half of the round-trip value) and if both the delay itself and its asymmetry are well characterized as white noise processes, so that the advance that is used by the server is an accurate estimate of the one-way delay on the average. When these conditions are satisfied, an average of the time differences between the client and the server measured using consecutive ACTS transmissions converges to the true time difference between the two systems. The standard-deviation of the mean is improved by the square root of the number of measurements that are used to compute it, and it was relatively easy to realize sub-millisecond timing accuracy using telephone calls that lasted 15 or 20 seconds.

THE EFFECT OF NEWER TELEPHONE CIRCUITS

The original ACTS servers had no way of evaluating the accuracy of the one-way delay estimate, since they could not estimate the asymmetry of the delay. If we define a symmetry parameter, k, where k= 0.5 indicates that the inbound and outbound delays are exactly equal, then the error in the time-difference measurement, Δt, resulting from an asymmetry in the measured round-trip delay, D, is given by,

$$\Delta t = (k - 0.5)D. \quad (1)$$

The limiting values k= 0 and k= 1 indicate that the inbound or outbound delays (with respect to the server), respectively, are negligibly small compared to the delay in the opposite direction. That is, when k= 0 the transit time from the server to the client dominates the round trip value, so that the clock on the client appears slow.

The measured round-trip delay on analog voice circuits was typically about 0.08 s and the delay asymmetry was typically less than 1%, so that the timing accuracy was of order 0.001 s. These values were realized using the same brand of modem on both ends of the connection and by using a signaling speed of 9600 baud, which has the minimum asymmetry for the modems that we used.[2]

Both the asymmetry and the delay are larger on newer telephone circuits, which are often implemented as a combination of conventional analog and digital packet-switched technologies. This combination is increasingly common even on local subscriber loops – the circuit that links the end user to the local telephone exchange. These circuits often have round-trip delays as larger as 0.25 s, with a varying delay asymmetry that can reach 5% or even more. From eq. 1, the timing error in this case exceeds 0.01 s – about a factor of 10 poorer than the original design. As a practical matter, many users of the Internet Time Services do not need even this level of accuracy and the degraded service may still be adequate for their needs. Nevertheless, I have designed software to try and ameliorate this degradation as much as possible, and I describe the initial tests of these improvements in this paper.

A NEW ACTS CLIENT

In the original ACTS design, the client only had to echo the on-time marker back to the server with negligible delay – all the real work of the protocol was done by the server. However, the client can measure the apparent time difference between the time of the local clock and the time stamp in the ACTS message every second – information that is not available to the server. In order to make use of this information, we must construct a model of the clock in the client system. We use the typical iterative model, in which the time difference at the current epoch, t, is estimated based on a clock model computed at a previous time t-τ. The parameters of the model are the time difference as a function of epoch, x, measured in s and the dimensionless frequency offset, y. The parameters ξ and η are the stochastic noise contributions to the measurement process and the frequency of the local clock, respectively. We assume that the measurement noise, ξ, is approximately stationary white phase noise, so that it does not depend on t or τ. The frequency noise, η, is typically estimated using the Allan deviation for an averaging time of τ. We assume that the Allan deviation is stationary as so does not depend on t.

$$x(t) = x(t-\tau) + y(t-\tau)\tau + \xi \quad (2)$$

$$y(t) = y(t-\tau) + \eta(\tau) \quad (3)$$

Although clock models often include a frequency aging parameter, *d*, whose units are s^{-1}, this parameter is not very useful in this application for several reasons. The frequency aging parameter would add terms $0.5d\tau^2$ to eq. 2 and $d\tau$ to eq. 3. In both cases, the contribution of the aging parameter is masked by the stochastic contributions to the corresponding estimates, so that it is difficult to compute a robust estimate of the aging parameter unless relatively large values of τ are used. Unfortunately, the aging parameter is usually not a constant over these longer averaging times, since the frequency of the oscillator is usually affected in a quasi-random manner by temperature fluctuations and other local environmental perturbations.

The quartz-crystal oscillators used in computer systems typically have $y= 2\times10^{-5}$, $\eta(1)= 10^{-7}$, and $\eta(30)= 1.8\times10^{-8}$. (These parameters characterize the oscillator as seen through the operating system software, and therefore include system latency and jitter. It is almost always impossible to measure the characteristics of the actual "bare" oscillator itself, which is probably considerably more stable than these values of η for almost all averaging times.) If we take 30 s as the maximum duration of a typical connection between a client system and the ACTS server, the stochastic frequency variations contribute about 0.1 μs to the estimates of consecutive time differences spaced 1 s apart, and about 0.5 μs to the dispersion of the time differences over the entire connection. In contrast, the measurement noise, ξ, is typically at least 80 μs even on a very good telephone connection, so that the contribution of the stochastic frequency fluctuations plays no role over the duration of a telephone call. The measurement noise is larger than the system latency because of the quasi-synchronous operation of the message interchange over the telephone line between the two modems. Therefore, the measured time dispersion of the clock in the client system over the duration of the telephone connection can be modeled as white noise superimposed on a simple linear variation. The linear variation can be ignored relative to the measurement noise for short telephone connections lasting less than about 4 s, but must be considered for the 30 s connections that we use for the synchronization of the time servers.

DETAILS OF THE MEASUREMENT ALGORITHM

The client system connects to the ACTS server and receives *N* time messages. Each message contains a time stamp and the advance applied to the on-time marker of that transmission. As described above, the advance parameter was actually computed by the server using the previous measurement that it made of the round-trip delay. The client measures the time difference between its clock and the time stamp of each of the messages, and also records the associated advance parameter. These values are T_i and A_i, respectively.

The algorithm models each of the advance values as the sum of a constant value that is a characteristic of that telephone connection and an additional stochastic value that has some unknown statistical distribution. We explicitly *do not* assume that the distribution of the advance values about the constant value has any particular form. However, we do assume that the variations in the advance have a mean of 0. That is, the mean of the advance values over the entire telephone connection is an unbiased estimate of the true transmission delay. (This assumption is basic to all two-way methods. There is no way of detecting a static bias in the asymmetry.) Thus, the algorithm assumes that the true advance for this telephone connection is given by

$$\overline{A} = \frac{\sum_{i=1}^{N} A_i}{N} \tag{4}$$

and that the additional stochastic contribution to the advance in each message is given by

$$a_i = A_i - \overline{A} \quad . \tag{5}$$

We model the measured time differences (by the use of eq. 2) with a constant time offset, a linear time variation due to a deterministic, constant frequency offset and a measurement noise parameter, which we assume to be stationary. We assume that the measurement noise has a random distribution with a mean of 0 that can be characterized as white phase noise. Therefore, the measured time differences should be accurately modeled using a least-squares straight line, and the measurements should scatter about that line with a random distribution that is characterized by the measurement noise of the process. We ascribe statistically significant deviations from the straight line assumption (relative to the estimate of the measurement noise) as due to an error in the advance parameter for that measurement. The error can be due either to a change in the asymmetry of the telephone connection or to a rapid change in the actual round-trip delay between one-second measurements. Our experience is that rapid changes in the round-trip delay are much less likely after the first few seconds of the connection, and changes in the asymmetry are much more common. See fig. 2 of [2], which shows that the advance varies by less than 1 ms P-P after the first few seconds of the connection.

I can illustrate the basis of the method with a simple example. The client system connects to the ACTS server and receives *N* messages, each of which has a time tag, an advance parameter and an on-time marker. When the on-time marker is received, the client measures the time difference between the time tag and the time of its clock. It also records the advance value that was used.

The first step of the analysis is to fit the time differences to a least-squares line and examine the residuals. In the simplest case, the RMS magnitude of the residuals is of order 1 ms, and the maximum deviation is not greater than 3 ms. The time difference is taken as the intercept of the fitted line (with the appropriate time tag), the frequency is the slope of the line, and the algorithm continues with the next step as described below.

If any residual exceeds the threshold of 3× the RMS value, the algorithm examines the advance values to see if the residual could be due to an error in the advance calculation. The algorithm calculates the mean advance and the deviation of each individual advance from that mean as in eq. 4 and 5. The time difference associated with each message is corrected by the difference between the advance used for that message and the average of all of the advances. This operation models statistically large time differences as due to errors in the advance calculation by the server – most often a result of a change in the asymmetry of the connection, which the server cannot detect. The sign of the correction depends on the direction of the asymmetry – whether the inbound or outbound paths were affected, as shown in eq. 1. Therefore, the algorithm must investigate both possibilities and choose the one that reduces the magnitude of the residuals. In most cases, this calculation reduces the scatter in the time differences, confirming the assumption that fluctuations in the advance are really the cause of the apparent fluctuations in the time differences. Any time difference that still exceeds 3× the standard deviation of the residuals of a least squares straight-line fit to the modified time differences is taken to be an error due to some other unknown reason. It is dropped from the estimate and the least-squares line is re-calculated. When no further outliers are detected, the time difference and frequency offset are estimated as the intercept and slope, respectively, of the fitted line, and the algorithm continues with the next step. If more than 5 points are discarded in this process, the estimate does not proceed. Either the RMS deviation of the model is too optimistic for the real data or the system has failed. The algorithm waits a short time and tries again; the time server is declared unhealthy as a precaution, since the time difference calculation did not complete successfully. The time server will continue to be set to an unhealthy state as long as the problem continues. This condition will trigger an operator alarm.

In the more general case, we use the standard Kalman formalism to estimate average values of each of the parameters over the duration of the telephone connection. In other words, the Kalman algorithm partitions the variance of the time differences measured during a single telephone connection as

due to fluctuations in the advance parameter, which have some unknown statistical distribution with a mean of 0 and a linear time variation with a slope and intercept that are constant during the connection. This calculation is considerably more complicated than the simpler algorithm described above, but it is more general and able to handle more complex noise types. For example, we have experimented with adding a diurnal term to the Kalman model, which could be helpful in estimating the effect of the nearly-diurnal fluctuations in ambient temperature on the frequency of the oscillator in the client. This term must be incorporated into the covariance matrix, since we do not know the amplitude or phase of the admittance to the temperature. This process is often described as "state vector augmentation." [3]

When the computation is completed, we have an estimate of the time offset of the system clock, the average frequency over the time of the connection, and the measurement noise parameter, ξ, which is essentially the RMS value of the residuals of the computation.

We cannot determine the stochastic frequency variation, η, in this way, since its contribution to the time differences is too small over the duration of the telephone call. For averaging times less than a few hours, the frequency fluctuations are well modeled as white frequency modulation. Therefore, we estimate the frequency of the oscillator by

$$y(t) = \frac{y(t-\tau) + T_y \dfrac{x(t) - x(t-\tau)}{\tau}}{1 + T_y} \quad . \tag{6}$$

The first term in the numerator on the right side is the estimate of the frequency on the previous measurement cycle, and the second term is the current average frequency estimated as the evolution of the time difference over the interval since the last measurement cycle. When the time server is operating in steady state, the control software drives the time difference to zero on every measurement cycle, so that $x(t-\tau)= 0$ in this mode. The estimate of the current frequency is then simply the time difference that has accumulated divided by the interval between measurments. The weighting factor T_y implements an exponential filter with a time constant derived from the assumption that the frequency variations can be characterized as white frequency modulation. For most of the computers that we have tested, this assumption is valid for averaging times up to about 12 000 s, so that

$$T_y = \frac{\tau}{12000} \quad . \tag{7}$$

For averaging times shorter than τ, the value of η, the stochastic variation in the frequency of the local clock oscillator is measured using special-purpose hardware that connects directly to the computer bus.

The value of τ, the time interval between measurement cycles is determined as a compromise among several competing considerations. If the interval between measurements is too short, it is difficult to estimate the deterministic frequency, y, in the presence of the measurement noise, ξ and the stochastic fluctuations in the estimate of the delay. Using a conservative value of 1 ms for the noise term, the interval between measurements should be at least

$$\tau \geq \frac{0.001}{2 \times 10^{-5}} = 50\,s \; . \tag{7}$$

In the limit of a very short measurement interval, the measurement noise dominates the calculation, and the performance is dominated by the characteristics of the noisy channel. The local clock is more stable than the remote clock seen through the noisy channel, and the inherent frequency stability of the local clock oscillator is not being exploited in an optimum way. On the other hand, the interval cannot be made too long without violating the assumption that the evolution of the time difference over the measurement interval satisfies the model equations. The algorithm must also consider the possibility of non-statistical glitches in the clock oscillator, and decreasing the interval between measurements would decrease the impact of these glitches because they would be detected (and removed) more rapidly. Finally, the cost of the telephone connections varies as $1/\tau$. Using the interval between calibrations based on the minimum value in eq. 7 would be too expensive and therefore impractical for this reason.

The value of τ was initially about 3 000 s – 4 000 s, but this value is generally too large for some of the poorer telephone connections, and a value of 2 000 s is often needed to realize an acceptable level of timing accuracy at the server of no worse than 5 ms P-P.

DISCUSSION AND CONCLUSIONS

The algorithm I have developed has made it possible to continue to operate the NIST Internet time servers with the telephone circuits that are becoming increasingly common both for the local connection to the telephone central office and for long-distance links. Maintaining a network of time servers synchronized in this way is important for two reasons. In the first place, the time service provided by these systems is independent of satellite signals in general and the global positioning satellites in particular, so that the systems are not compromised by a denial of service attack on satellite signals. In addition, the servers do not need external antennas and can be installed at any location that has telephone connections. This is a significant advantage, since many of our systems are installed in underground or windowless, secure locations with no access to the outside. On the hand, the servers depend on periodic long-distance connections to the ACTS serves in the NIST Boulder laboratories, and this is an expense that would

not be needed for a system that was synchronized using a satellite timing receiver.

As with all two-way methods, the algorithm cannot detect a static asymmetry in the measurement of the round-trip delay of the telephone circuit. We can estimate the magnitude of this effect by comparing the times of different servers to each other, but these comparisons introduce possible asymmetries in the delay of the Internet path between them. As a practical matter, the asymmetries of the Internet paths are typically at least as large or even larger than the values for the telephone system, so that the results of these comparisons are not definitive.

The method I have described here cannot cope with every type of asymmetry. At one of our sites, the asymmetry had a bimodal character. That is, the asymmetry was constant for some period and then abruptly changed to a different value. The change in the asymmetry resulted in time steps of order 20 ms with an irregular period of a few hours. I had no basis for eliminating these steps, since I didn't know the actual time offset of the client system. Both of the time offsets satisfied the usual statistical tests within a single telephone connection, and the asymmetry was constant over several telephone connections, suggesting that the routing of the telephone calls changed periodically in an unpredictable manner.

The problem of a bi-model asymmetry is not very common on telephone connections – we have only one example on 35 connections. However, a slowly varying asymmetry is common, and it also arises quite often in time synchronization applications that use the Internet as the communications medium. A nearly bimodal asymmetry on the Internet can arise for a system, such as a web server, that often has very different inbound and outbound network activity. The "Huff-'n-Puff" [4] filter is an attempt to deal with both of these cases. The procedure assumes that the minimum delay will also have the minimum asymmetry, which is often true for web servers and similar systems because the asymmetry is a strong function of the load on the client or on the server and not a fundamental characteristic of the intervening path itself, which is assumed to have better symmetry. We do not have a good idea of the source of the varying asymmetry in our configuration, and it is less clear that assuming that the minimum delay also has the minimum asymmetry is appropriate. Nevertheless, there could be an advantage to using only the time differences with the minimum delay in the computation of the estimate of the time of the client. From eq. 1, the time error due to any asymmetry is bounded by one-half of the round-trip delay, so that a smaller delay will have a smaller time error, even if the asymmetry fraction is the same. However, even the shortest delay is many tens of milliseconds, so that this limit is not very useful, since the goal of the time service is to synchronize the servers with an uncertainty of 1 ms RMS. Unfortunately, we also do not have an independent estimate of the true time difference between the two systems, so that we have no basis for choosing one time difference over the other one.

One way to address this asymmetry problem would be to improve the frequency stability of the oscillator in the time server so that much longer averaging times could be supported. The easiest way to do this would be to add an external atomic frequency standard or even a high-quality quartz oscillator and then lock the computer oscillator to the external device. Some of our time servers have external rubidium oscillators interfaced to the system through an interrupt line on the serial port for this purpose. Both the frequency offset and the frequency stability of the clock oscillator are improved by several orders of magnitude using this technique.

When an external oscillator is present, the time-varying asymmetry (and similar slowly-varying perturbations) can be averaged for much longer periods – at least 1 day and probably longer than this, since the frequency stability of the enhanced local oscillator will now support these longer intervals between measurements. Since the variations in the asymmetry are bounded, it can always be divided into two components: a static value and a bounded variation about that value that has a mean of 0 for sufficiently long averaging times. A long averaging time would attenuate the time variations in the asymmetry and convert a static asymmetry into a time offset. This time offset would have a slow variation in time if the static value of the asymmetry was not stationary. (This would be true if the asymmetry had a random-walk spectrum, for example, since such a spectrum has no robust mean value.) The time offset could be estimated through comparisons with the NIST time servers that are located at NIST facilities and are synchronized by direct connection to the NIST clock ensemble. These directly-connected time servers would serve as the constraints on the ensemble of network servers, so that the ensemble would provide a stable source of time messages that would be less affected by the asymmetry of any single network path. To the extent that the network paths are disjoint, an ensemble of network paths might also have an overall asymmetry smaller than the asymmetry of any one of them.

REFERENCES

[1] J. Levine, M. Weiss, D. D. Davis, D. W. Allan and D. B. Sullivan, "The NIST Automated Computer Time Service," J. Res. NIST, vol. 94, pages 311-321, 1989.

[2] Judah Levine, "Improvements to the NIST network time servers," Metrologia, vol. 45, pages S12-S22, 2008.

[3] Arthur Gelb, Editor, "Applied Optimal Estimation," 1974, Cambridge, Massachusetts, The M.I.T. Press. See pages 78-82.

[4] David L. Mills, “Computer Network Time Synchronization,” Boca Raton, Florida, CRC Press, 2006. See pages 54-55 and 103-104.