An Auto-Regressive Moving-Average Time Scale Algorithm (ARMA) for Synchronizing Networked Clocks

Judah Levine
Time and Frequency Division and JILA
NIST and the University of Colorado
Boulder, Colorado

BIOGRAPHY

Judah Levine is a Fellow of the National Institute of Standards and Technology (NIST) and is also a Fellow of JILA, an institute operated jointly by NIST and the University of Colorado. He received his Ph.D. in physics from New York University in 1966. He is a member of the IEEE and a Fellow of the American Physical Society. Dr. Levine designed and implemented the time scales AT1 and UTC(NIST), which provide the reference signals for all of the NIST time and frequency services. In addition, he designed and built the servers that support the Automated Computer Time Service (ACTS) and the Internet Time Service (ITS), which provide time and frequency information to users in a number of different digital formats. The NIST ITS servers can be accessed through client software built-in to a number of different computer operating systems and receives more than 15 billion timing requests per day. Dr. Levine was the recipient of the PTTI Distinguished Service Award in 2009, the Presidential Rank Award in 2011, and the IEEE International Frequency Control Symposium Rabi Award in 2013.

ABSTRACT

I will report on a study of the usefulness of ARMA time scale algorithms to synchronize clocks on a digital network. The algorithm acquires periodic time differences between a local system clock and a remote time server by means of any of the standard message formats such as the format used by the Network Time Protocol. It models the current time difference as a linear combination of previous time states plus additive noise and uses the model to adjust the local system clock. The algorithm is more flexible than the traditional methods, which are based on physical parameters such as frequency and frequency drift. The ARMA model has a finite impulse response and is therefore able to cope with the non-stationary outliers that characterize the fluctuations in the message delay on a wide-area network. I will compare this method with the frequency lock loop (FLL) algorithm that is currently used to synchronize the time servers operated by NIST. Both methods take advantage of the free-running stability of the clock in the local system, which facilitates the detection of outliers without the need to query multiple remote servers in most situations. Either method is generally more efficient than the phase-lock loop process that is widely used in network synchronization applications.

INTRODUCTION

The National Institute of Standards and Technology currently operates an ensemble of time servers that respond to requests for time in a number of standard formats. The servers receive approximately 150,000 requests per second; approximately 95% of these requests are for time in the Network Time Protocol (NTP) format.[1] This format estimates the transmission delay between the client and the server as one-half of the measured round-trip delay. The accuracy of this estimate depends on the validity of the assumption that the delay is symmetric – that the inbound and outbound delays are equal. This assumption is valid for local networks or those that have only a small number of network elements such as routers and switches. Although it may be reasonably accurate for wide-area networks on the average, there are often large deviations from a symmetric delay, and these deviations compromise the accuracy of the time synchronization process.

SYNCHRONIZATION STATISTICS

The initial design of the algorithm that was used to synchronize the NIST time servers was based on statistical characterizations of the stability of the clock in the client system and the variation in the network delay. The model treated these parameters as stationary, with well-defined variances that could be characterized with the machinery of the Allan variance.[2] The parameters of the synchronization algorithm were chosen so that the remote clock seen through the network channel had a smaller variance than the clock in the local system.[3] Fig. 1 shows a typical example of this type of analysis.

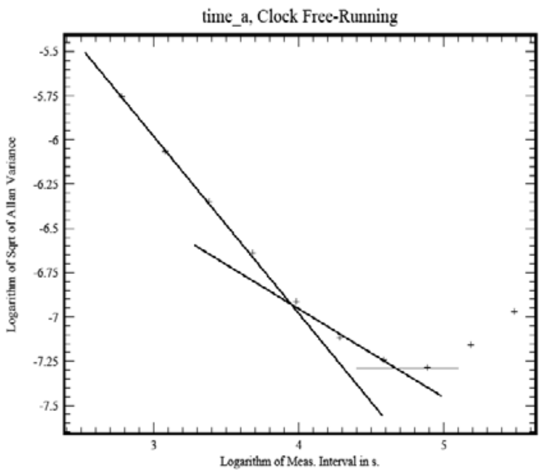

Fig. 1. The two-sample Allan deviation (square root of the Allan Variance) of the free-running stability of the clock in a typical general-purpose computer. The points are the measured time-difference values and the straight lines help to identify the underlying noise types.

The figure shows a log-log display of the two-sample Allan deviation for the free-running clock oscillator in a typical general-purpose computer system. The points in the figure were computed by using an external atomic clock to generate a system interrupt every second and then reading the system time on every interrupt. This procedure evaluates the software clock as it is seen by a user process. The physical clock oscillator is not observable. The latency of the measurement process itself was tested and found to be on the order of a few microseconds, which is negligible in this configuration. The three straight lines in the figure help to identify the noise processes that are important for different averaging time domains. For averaging times, τ, less than about 10^4 s, the Allan deviation, $\sigma_y(\tau)$, is approximately $0.0011/\tau$; it is approximately $1.1\times10^{-5}/\sqrt{\tau}$ for averaging times between 10^4 s and 5×10^4 s and approximately 5×10^{-8} for averaging times between 5×10^4 s and 1 day.

The second step in the design of a synchronization algorithm is to estimate the statistical characteristics of the data channel back to the reference clock. For the NIST time servers, the link between the time server and the NIST clock ensemble could be characterized as white phase noise with an Allan variance of approximately $2\times10^{-3}/\tau$. This variance is dominated by the noise in the channel delay and the local measurement process, and is essentially independent of the characteristics of the remote reference time system. The two noise contributions will be equal when

$$\frac{2\times10^{-3}}{\tau}=\frac{1.1\times10^{-5}}{\sqrt{\tau}} \qquad (1)$$
$$\tau \sim 33000\ s$$

and the remote reference clock seen through the communications channel will be more stable for all averaging times greater than this value. Conversely, the free-running stability of the local clock oscillator is more stable than the data received over the communications channel from the remote system at shorter periods.

Since the link between a time server and the NIST clock ensemble is characterized by white phase noise, it is possible to reduce the variance of these data by averaging consecutive one-second measurements. For example, if the algorithm uses the average of five consecutive messages, the standard deviation of the received data is improved by $\sqrt{5}$, and the cross-over time is decreased by a factor of 5. This is a specific example of the more general principle that the interval between calibration messages need not be the same as the interval between adjustments of the local clock.[4] The optimum values for these two parameters are determined by different considerations.

LIMITATIONS OF THE METHOD

The algorithm described in the previous section worked very well for many years, and was able to synchronize the clock in a time server with an uncertainty of less than 1 ms for averaging times on the order of the averaging time in eq. 1. However, it depended on the assumption that the statistics could be modeled as stationary processes based on physical parameters such as time and frequency with corresponding noise contributions. Furthermore, the models were relatively easy to implement because only one noise source dominated the variance at any averaging time.

The most serious challenge to this method is the assumption that the statistics are stationary. Unfortunately, this is no longer true in many situations. For example, fig. 2 shows the time differences measured every 60 s between two systems that are linked by a connection over the public Internet. Both systems are synchronized to UTC, so that the expected time differences should all be zero.

The average round-trip network delay between the two systems was about 200 ms. The algorithm models the one-way delay as one-half of this value, or 100 ms. If the actual delay on any cycle was completely asymmetric, the model estimate will be wrong by 100 ms. However, the measured round-trip delays associated with the spikes are much larger – often approaching 0.5 s in magnitude. These large delays are also very asymmetric, and introduce timing errors of up to 0.25 s. If there were only a small number of these spikes, it would be practical to ignore any measurement that had an unusually large value for the measured delay. (David Mills names this strategy the “huff-n-puff’ filter in ref. 1) The very large number of these problem measurements makes this strategy problematic.

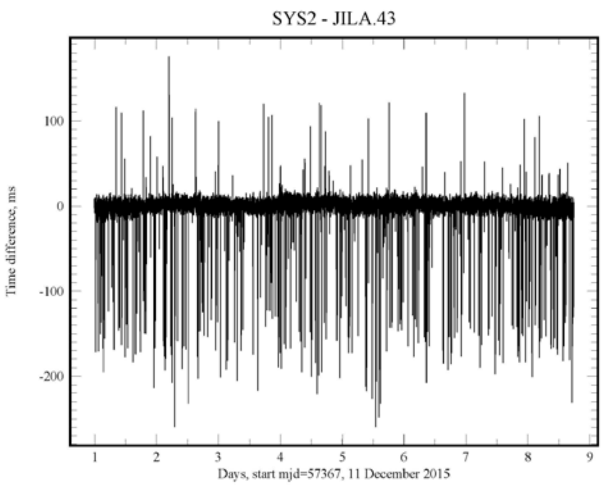

Fig. 2. The time differences between two systems that are both synchronized to UTC. The time differences are measured by means of messages transmitted over the Internet in the Network Time Protocol format, which estimates the one-way delay as one-half of the measured round-trip value. The number of negative spikes is much larger than the number of positive ones, indicating that the apparent delay for a message traveling from JILA.43 to SYS2 was often much larger than the apparent delay in the opposite direction. The analysis is not sensitive to the sign of a spike.

The formal standard deviation of the data in the figure is only 26 ms, but it is clear that the observed variation is not stationary and would not be well characterized by this parameter or by an Allan-variance estimate of the stability. The problem is not the magnitude of the round-trip delay, since this is measured on every message exchange. Rather, it is the very large and rapidly varying asymmetry of the delay, which violates the fundamental assumption of the two-way protocol that the one-way delay is well characterized as one-half of the measured round-trip value.

Since the data shown in fig. 2 cannot be characterized by means of the time and frequency noise parameters, I have studied the usefulness of other models that are not derived from the physical parameters of time and frequency that are normally used to characterize clocks and oscillators.

AUTO-REGRESSIVE MODELS

An auto-regressive model characterizes the state of a clock by using only a single parameter – the time difference between the clock and a reference device. The time difference at epoch t_j, denoted by x_j, is modeled as a linear combination of previous values of the state:

$$x_j = \sum_{i=1}^{i\_arm} a_i x_{j-i} \tag{2}$$

where the a_i are the coefficients of the auto-regressive model. The time difference measured at epoch t_j is X_j, and the next step in a standard analysis would be to adjust the auto-regressive coefficients so as to minimize the RMS time difference, X_j-x_j, averaged over j.[5,6,7] The number of auto-regressive coefficients could also be varied.

This relatively simple method is not adequate for these data because there is a significant probability that the measured time difference is a spike. In addition to introducing a time error when the bad measurement is received, a spike that is not detected will cause more trouble because it will be incorporated into the auto-regressive model and will introduce additional time errors into subsequent data. (This problem is not unique to auto-regressive analyses. All time scale algorithms have some sort of “reset” procedure to prevent a bad measurement from corrupting the state parameters of a clock.)

After some experimentation, the method we have developed detects a spike as a deviation from a running-average of the variance of the time-difference data. This method works well if the spikes are not too close together, so that the running-average variance is an accurate measure of the underlying measurements. The spike detector is triggered when a time-difference measurement exceeds three times the magnitude of the running-average variance. This criterion has no difficulty detecting the large spikes in the figure, but it accepts data that would probably be rejected by eye. There is no algorithmic fix for this problem because the data are not well-characterized by a stationary variance.

Since the time-difference measurements are acquired every minute, our initial assumption is that the underlying time differences are well-characterized by white phase noise, and the initial values of the auto-regressive coefficients effectively estimate the current time difference as the average of the previous measurements after the spikes have been removed as described above. (In other words, the initial values of all of the five auto-regressive coefficients are equal to 0.2.) When a spike is detected, the algorithm effectively goes into a hold-over mode, in which it replaces the current measured time difference with the auto-regressive estimate of the time state. It treats continuous repetitive "spikes" as a true time step of the local clock, so that a true time step is recognized after some delay. The algorithm shares the weakness of most other algorithms in that it is difficult to detect a frequency step of the local clock oscillator unless the step is large enough to produce a large time dispersion. Such frequency steps are unusual, and are generally an indication of a hardware failure.

Fig. 3 shows the output time differences predicted by a 5-state auto-regressive model applied to the data in fig. 2. The five auto-regressive parameters were initially set equal to 0.2, and the algorithm adjusted them dynamically by about ±10%. The data in the figure can be characterized approximately as white phase noise, with a mean of 0.022 ms and a standard deviation of 0.22 ms. The value of the mean is much small than the standard deviation and is therefore consistent with a mean of zero.

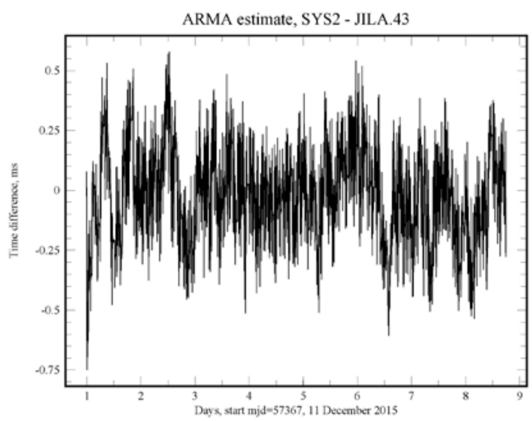

Fig. 3. The time differences between SYS2 and JILA.43 estimated by the auto-regressive process as described in the text. The standard deviation of the data in the figure is 0.22 ms, which is consistent with a mean of zero, confirming the assumption that both systems are independently synchronized to UTC.

UNSYNCHRONIZED CLOCK TEST

The clocks at both ends of the network link were independently synchronized to UTC in the experiment described in the previous section. Therefore, the auto-regressive model did not contain a term to estimate a possible frequency offset in the measured time differences.

In a second experiment, we relaxed this condition, and used time-difference data between a system whose clock was free-running and a remote time server synchronized to UTC. We used the ARMA estimator to calculate the correction that we would have applied to the time of the local clock and we then analyzed the residuals of that simulation.

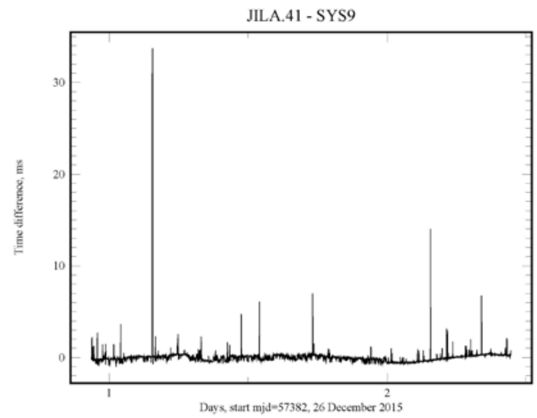

Fig. 4. The time differences, measured every minute, between the free-running clock in system JILA.41 and the remote system SYS9, which was synchronized to UTC. The JILA.41 system had an initial time offset of -270 s and a rate offset of -1.92 s/day, with respect to the remote system, and a straight line with these parameters has been removed from the data in the plot for clarity.

Fig.4 shows the time differences, measured every 60 s, between a local system with a free-running clock and a remote time server. A constant frequency offset of -1.96 s/day and an initial time offset of -270 s have been subtracted from the data in the display. These offsets were removed only for the display, and were not removed in the analysis.

If we analyze these data with a 5-point auto-regressive model, the frequency offset of -1.96 s/day produces a time dispersion of approximately -1.4 ms per minute. It is difficult to compute a robust estimate of this frequency offset if only the 5 points of the auto-regressive algorithm are used because the time dispersion due to the frequency offset is not sufficiently larger than the background white phase noise. Therefore, a more sophisticated analysis must be developed in which the auto-regressive equation provides an estimate of the time offset at any epoch, and the frequency offset is estimated by combining a number of auto-regressive time difference estimates separated by 1200 s, or four 5-point one-minute measurement groups. (It is possible in principle to modify the auto-regressive coefficients to estimate the frequency offset as well as the time offset by using an auto-regressive *integrated* algorithm[8]. These models proved to be unstable because they did not provide an adequate separation between noise in the time-differences and the time dispersion due to frequency noise. I will address this point in the conclusions.)

Fig. 5 shows the time differences estimated by using the modified auto-regressive model described in the previous paragraph.

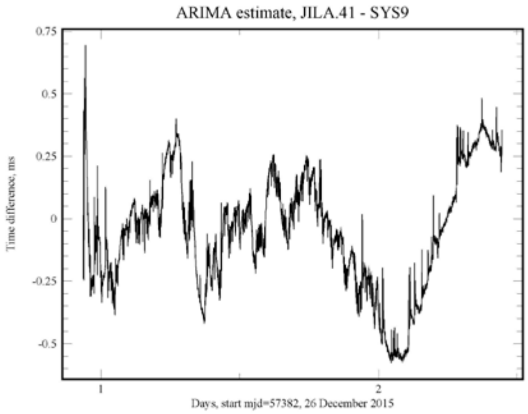

Fig. 5. The residuals of the time adjustments that were calculated by the modified auto-regressive algorithm applied to the input data shown in fig. 4 and the measured time differences, with outliers removed as described in the text.

The data in the figure have a mean of 0.05 ms and a formal standard deviation of 0.24 ms. The mean is not statistically different from zero, and there is no statistically significant residual rate offset. The initial time offset and the rate offset have been modeled correctly, and the residuals are dominated by the white frequency noise of the local clock oscillator.

SUMMARY AND CONCLUSIONS

The auto-regressive model has been applied to time differences between the clocks of two systems that are compared by messages transmitted in the Network Time Protocol Format over the public Internet. The protocol estimates the one-way delay between the two systems as one-half of the measured round-trip value. The accuracy of this estimate is limited by the symmetry of the round-trip delay, and this assumption of a symmetric delay is often not appropriate for a message exchange over the public Internet.

The auto-regressive model detects departures from delay symmetry by comparing each measured time difference with a weighted sum of the previous time states of the local clock. The method replaces the measured time difference with the auto-regressive estimate when a bad measurement is detected.

The method is successful because the one-minute interval between measurements is short enough so that both the deterministic and stochastic frequency of the local clock do not result in a large time dispersion over the elapsed time of the auto-regressive estimate. The white phase noise of the local clock is attenuated by the effective averaging of the auto-regressive process.

On the other hand, the short interval between measurements makes it difficult to obtain a robust estimate of the deterministic frequency offset of the local clock. Although an auto-regressive *integrated* analysis can provide such an estimate in principle, attempts to use this method to estimate the deterministic frequency offset were not successful, and an estimate derived from multiple auto-regressive estimates, separated by a longer time interval, was found to produce more robust estimates of the frequency.

The auto-regressive method provides a useful complement to the methods based on statistics and the Allan variance that were described in the introduction. The auto-regressive method is particularly well suited to time comparisons made by means of messages exchanged over the public Internet, because these messages usually cannot be well characterized by the standard statistical estimators.

REFERENCES

[1] David L. Mills, Computer Network Time Synchronization, Boca Raton, Florida, CRC Press, Second edition, pp. 64 ff.

[2] D. W. Allan, "Time and frequency (time-domain) characterization, estimation, and prediction of precision clocks and oscillators," *IEEE Trans. UFFC*, Vol. 34, pp. 647-654, November 1987.

[3] Judah Levine, An Algorithm for Synchronizing a Clock when the data are received over a network with an unstable delay, IEEE Trans. UFFC, in press (2016).

[4] Judah Levine, The Statistical Model of Atomic Clocks and the design of Time Scales, Rev. Sci. Inst., vol. 83, 012201-28 (2012).

[5] A. Lepek, Clock Prediction and Characterization, Metrologia, Vol. 34, pp. 379-386, 1997.

[6] Lara S. Schmidt, Atomic Clock Models Using Fractionally Integrated Noise Processes, Metrologia, Vol. 40, pp. S305-S311, 2003

[7] L. Schmidt, L. Breakiron and C. Ekstrom, Fractional Difference Pre-Whitening in Atomic Clock Modeling, Proc. 34th Annual Precise Time and Time Interval Meeting, pp 473-480, 2002.

[8] George E. P. Box and Gwilyn M. Jenkins, Time Series Analysis, Forecasting and Control, San Francisco, Holden-Day, 1970, Chapter 4.