Metrologia 45 (2008) S12–S22 doi:10.1088/0026-1394/45/6/S03

Improvements to the NIST network time protocol servers

Judah Levine

Time and Frequency Division, National Institute of Standards and Technology, Boulder, CO 80305, USA

E-mail: Jlevine@boulder.nist.gov

Received 25 June 2008
Published 5 December 2008
Online at stacks.iop.org/Met/45/S12

Abstract
The National Institute of Standards and Technology (NIST) operates 22 network time servers at various locations. These servers respond to requests for time in a number of different formats and provide time stamps that are directly traceable to the NIST atomic clock ensemble in Boulder. The link between the servers at locations outside of the NIST Boulder Laboratories and the atomic clock ensemble is provided by the Automated Computer Time Service (ACTS) system, which has a direct connection to the clock ensemble and which transmits time information over dial-up telephone lines with a two-way protocol to measure the transmission delay. I will discuss improvements to the ACTS servers and to the time servers themselves. These improvements have resulted in an improvement of almost an order of magnitude in the performance of the system.

(Some figures in this article are in colour only in the electronic version)

1. Introduction

The National Institute of Standards and Technology (NIST) currently operates 22 public network time servers that are located at many different sites in the United States [1]. The servers have a direct, hardwired connection to the NIST atomic clock ensemble in Boulder, Colorado, and therefore operate at stratum 1. The connection to the clock ensemble is realized by means of the Automated Computer Time Service (ACTS). The hardware that supports the ACTS system is directly connected to the atomic clock ensemble in Boulder. The ACTS system transmits time over voice-grade dial-up telephone connections and standard modems. Both the ACTS system that provides the time signals and the network servers that receive them have been improved, and I will discuss these improvements in the following text. The improvements have resulted in an increase in the timing accuracy and time stability of the servers. This improvement has been used to increase the accuracy of the service with the same calibration interval and has resulted in an improvement of almost an order of magnitude in the performance of the servers. It is also possible [2] to design the synchronization algorithm to make an explicit trade-off between the accuracy of the process and the interval between calibrations that is required to realize this accuracy.

2. General description of the services

The ACTS service was first offered in 1988 and was widely used at that time to set the time on both personal computers and larger central systems. The service was based on dial-up connections over the public telephone network. The time signals were transmitted using standard modems, and no special hardware was required. The usage of the service declined somewhat in the following years as connections to the Internet became more generally available. More recently, the service has seen a renaissance among general users since it is not affected by many of the problems and malicious activities that have become more common on the public Internet.

When the NIST started offering Internet-based time services in the early 1990s, the ACTS service was a natural choice for synchronizing the Internet time servers that were not located at the NIST facility, since the ACTS system provided secure connections whose accuracy was consistent with the requirements of an Internet service and whose messages were directly traceable to the atomic clock ensemble maintained at NIST in Boulder. Although the ACTS service continues to be used by the general public for many different applications, one of its most demanding uses is to synchronize the systems used to support the NIST network time service. Recent improvements in the network time servers have mandated corresponding improvements to the ACTS system. This report

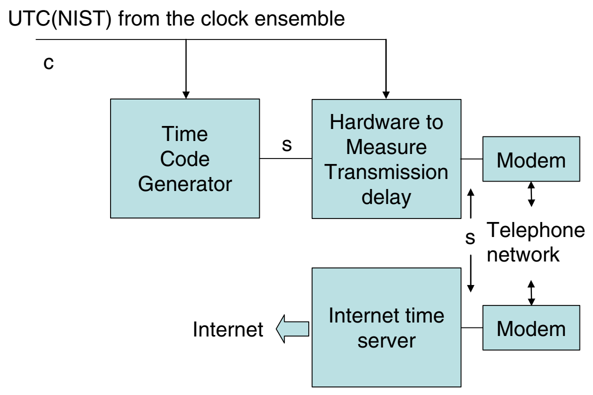


Figure 1. A block diagram showing how UTC(NIST) is transmitted to a typical Internet time server using the ACTS system as the transport method. The connections indicated by 'c' are hardwired connections that transmit signals at 1 Hz and 5 MHz from the NIST clock ensemble to the ACTS system. The connections indicated by 's' are data connections between serial ports communicating using the RS-232 signal format at a constant speed of 9600 bits s^{-1}. The telephone connection is realized using standard voice-grade dial-up lines.

describes both of these improvements and the tests that I performed to evaluate the performance of the system.

Figure 1 shows the general configuration of the improved ACTS system, which transmits the time signals, and the Internet time server which receives them and uses them to discipline its internal clock. In this paper I will focus on the performance of the combination, recognizing that the ACTS service is also used for other purposes.

3. The ACTS protocol

The ACTS protocol [3] is based on the usual two-way algorithm—the time it takes a message to travel from the server to the client is modelled as one-half of the measured round-trip delay. This estimate of the one-way delay is used to correct the time stamp. As with all two-way methods, the accuracy depends on the symmetry of the path delay and not on its magnitude [4].

In this paper I will describe improvements to the hardware that is used to support the ACTS service at NIST. The two changes that have had the greatest impact on the quality of the time service are that (1) a more sophisticated and more accurate hardware-based method is now used to measure the round-trip path delay and (2) the time at which the on-time marker is transmitted is controlled more accurately by the new system. This tighter control has improved both the accuracy and the stability of the ACTS system. I will describe both of these changes in greater detail below. The protocol and the message format have not been changed, however, so that users do not see these improvements directly, and software that received time messages from the original servers can use the newer ones without modification. The following discussion summarizes the operation of the ACTS system.

Once a telephone connection has been established between the client system whose clock is being calibrated and the ACTS server in Boulder, the ACTS server begins sending time messages once per second. (The client system does not have to solicit these messages—the server starts sending them automatically when it detects a new connection.) The message contains several fields in ASCII text, including the Coordinated Universal Time (UTC) time as realized by the NIST time scale, advance notice for leap seconds, advance notice for the transitions to and from daylight saving time (based on the US model) and the advance time in milliseconds when the on-time marker was transmitted relative to the time tag in the same message. As we will discuss in the following text, this advance is intended to correct for the travel time of the message from the server to the client. All of the calculations needed to support the two-way algorithm are performed by the server. (This is in contrast to other protocols, such as the Network Time Protocol, where the calculations are done by the client or two-way satellite time transfer, where both stations contribute to the calculation.)

The message ends with an on-time marker, which is initially set to the ASCII character '$*$'. The protocol is designed so that the epoch specified in the message has arrived when the stop bit of the on-time marker is received by the client system, and the server advances the transmission time of this character by the estimated one-way travel time across the network so as to realize this condition. The server uses a default advance of 145 ms, and this default advance will continue to be used if the user does not participate in the two-way algorithm by echoing the on-time marker back to the server as discussed in the following text. The server does not consider the lack of an echo to be an error but rather an indication that the client is prepared to accept the default advance as sufficiently accurate for its needs.

To ensure that the echo will be received before the maximum time limit is reached (see below) even when the true round-trip delay is very long, the default advance is set near its maximum value. (This can happen with some modems and noisy telephone lines, which require low-speed connections.) In more typical cases, the actual one-way delay is closer to 75 ms, so that the on-time marker will arrive early by up to 70 ms of the correct time when the default advance is used. Since the modems at both the server and the client make significant contributions to the path delay, the variation in the delay from one telephone connection to the next one is often less than 1 ms peak to peak even when the default advance is used, so that an application that uses the ACTS service for a frequency calibration may see no advantage in echoing the on-time marker back to the server.

The stability of the delay from one call to the next is illustrated in figure 2, which shows the delay measured by the ACTS server on three consecutive calls spaced a few minutes apart. In this test the remote modem calls the ACTS server over a standard dial-up telephone line and is switched into loopback mode as soon as the modems complete the connection by connecting a jumper between the received data output and the transmitted data input. The configuration is shown in figure 3, with the wire marked 'LB' connected. (See the figure caption for additional details of the test configuration.) Although the initial delay transients are different, the measured delays in each call from the fifth transmission to the end of the call have mean values of 81.8 ms, 81.9 ms and 81.75 ms. For all three

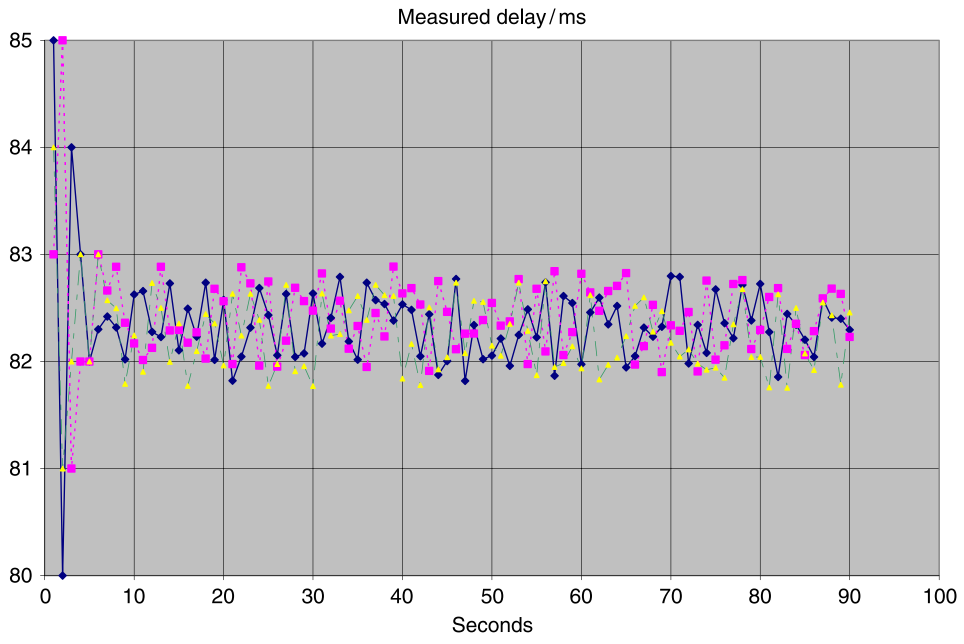


Figure 2. The one-way delay over a dial-up telephone connection measured by the ACTS server between the server and a remote modem. The remote modem is used to call the ACTS server. As soon as the call is established, the remote modem is connected in loopback mode by connecting the received data output line to the transmitted data input line at the RS-232 connector as shown in figure 3 with the 'LB' wire connected.

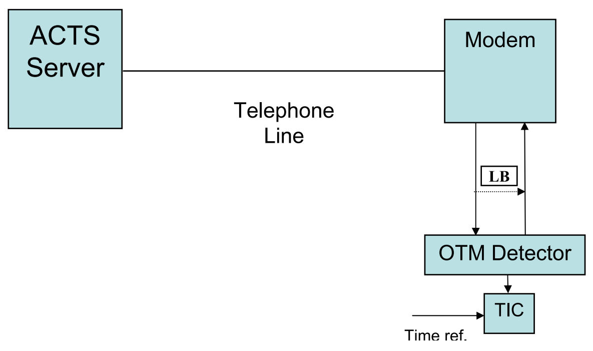


Figure 3. The hardware configuration used to test the ACTS servers. The remote modem dials the ACTS server to establish the connection. The output of the modem is connected to a simple circuit that monitors the received characters for an on-time marker (OTM) character. When the circuit detects either of these characters ('$*$' or '#'), it echoes it back to the server and emits a pulse on the output line. The time of the pulse is compared with a local reference by the use of a standard time interval counter (TIC). The local reference clock is derived from UTC(NIST), so that the data are a measure of both accuracy and stability. The dotted arrow marked 'LB' shows the loopback configuration discussed in the text and used for the data in figure 2. The circuit that monitors the characters is not connected when the loopback tests are being performed.

calls, the delays can be well characterized as white phase noise. Figure 4 shows the time deviation (TDEV) for these data, starting with the fifth point once the delay has become stable.

In order to measure the round-trip path delay, the client system must echo the on-time marker back to the server as soon as it is received, and the protocol assumes that the delay in doing so is small enough to be ignored. If the client simply echoes the entire message back to the server, the server will ignore all of the characters in the time message except for the on-time marker. Therefore, a client system can implement the protocol by configuring its receiver to be in loopback mode, which immediately echoes everything that has been received back to the sender. (This method will work, but it may not be optimum if the receiving system is very busy[1]). Since the point at which the on-time marker is echoed is the effective reference plane for the two-way measurement algorithm, it is important that the system accounts for any delay in processing the time stamp after that reference point, since the algorithm that measures the path delay will not compensate for it. It is generally not possible to measure this extra delay, and the next-best strategy is to make it small enough that it can be ignored.

When the client system receives the on-time marker and echoes it back to the ACTS server, it also captures the time of its own clock. The protocol depends on the fact that this capture is done following the receipt of the on-time marker with a delay that is small enough to be ignored. If the server has accurately modelled the one-way delay, then the on-time marker arrives at the time specified by the message text, and the client system simply subtracts the time of its local clock from the received epoch. No additional processing is required. The server indicates that it has measured the delay by changing the on-time marker character from '$*$' to '#.' If the server cannot measure a consistent delay, then the on-time marker does not change, and this is a signal that the delay measurement is not being performed accurately and that the default advance is being used.

The improvements to the network time service have focused on improving both ends of this two-way exchange

[1] If there are any characters in the output buffer, then the on-time marker will be added to the queue and its transmission can be delayed by a variable amount until the buffer empties. A better strategy is to minimize the number of characters echoed so that the output buffer is always empty when the on-time marker is sent to the output driver for transmission.

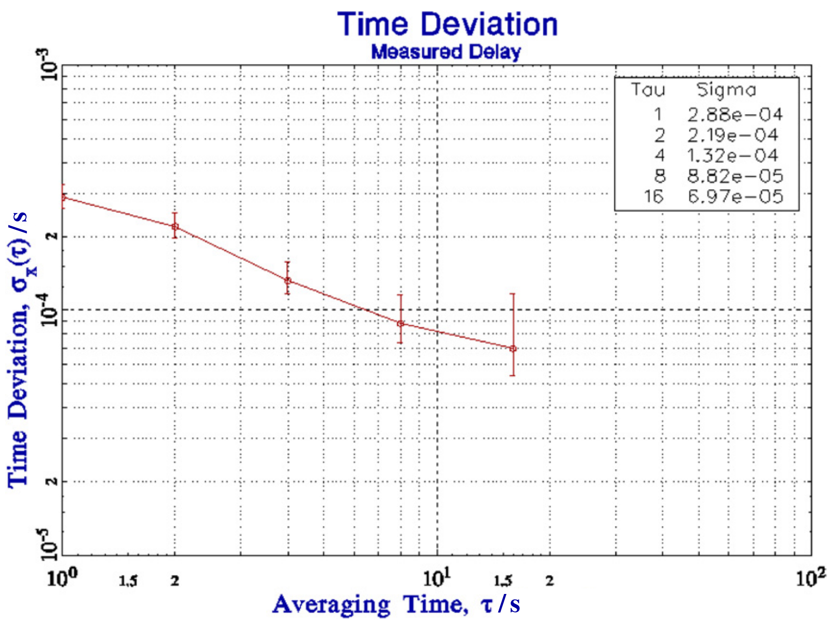


Figure 4. The TDEV of one of the data sets shown in figure 2, starting from the fifth data point after the delay has stabilized and continuing for the rest of the data.

between the Internet time servers whose clocks are being synchronized and the ACTS systems that are used for this purpose.

4. The ACTS time messages

The messages transmitted by the ACTS system provide the epoch in two independent formats, and a lack of consistency between these two formats is an indication of a transmission error. A series of messages are shown in the following text.

```
National Institute of Standards and Technology

Telephone Time Service, Generator 4B

Enter the question mark character for HELP

                         D L

MJD   YR MO DA HH MM SS ST S UT1 msADV           <OTM>

54630 08-06-13 15:46:36 50 0 +.3 145.0 UTC(NIST)    *

54630 08-06-13 15:46:37 50 0 +.3 079.7 UTC(NIST)    *

54630 08-06-13 15:46:38 50 0 +.3 078.7 UTC(NIST)    *

54630 08-06-13 15:46:39 50 0 +.3 080.9 UTC(NIST)    *

54630 08-06-13 15:46:40 50 0 +.3 079.3 UTC(NIST)    *

54630 08-06-13 15:46:41 50 0 +.3 080.1 UTC(NIST)    #

54630 08-06-13 15:46:42 50 0 +.3 080.0 UTC(NIST)    #

54630 08-06-13 15:46:43 50 0 +.3 079.8 UTC(NIST)    #

54630 08-06-13 15:46:44 50 0 +.3 080.4 UTC(NIST)    #
```

The first parameters give the date in a redundant format. The presence of the Modified Julian Day (MJD) number serves to remove the ambiguity in the two-digit year, and the two formats of the date provide a consistency check on the transmissions. The remaining data are not used if this consistency check fails.

The remaining data in the message contain the flag for daylight saving time, DST (where 50 indicates that daylight saving time is in effect based on the US transition dates), the advance notice for leap seconds, LS (where a value of 0 indicates that no leap second is pending), the UT1 correction in units of seconds with a resolution of 0.1 s and the advance used for the on-time marker corresponding to this message. Note the initial value of 145 ms, and the convergence to a stable value after several iterations, after which the on-time marker changes from '$*$' to '#'. The details of the format for the daylight saving time flag have been described previously, and an explanation is also available in the help message, which can be received by entering a question mark at any time. The calculation of the DST value was changed in 2007 to match the new US transition dates that were effective at that time, but the definition was not changed. The DST flag is not used for the Internet time services, since all processing is done in UTC. The leap second flag is used by the Internet clients to set the corresponding flag in the transmitted messages. The UT1 correction is not used. The advance value is used for testing and debugging, but is ignored in normal operations, since it has been used by the server to calculate the transmission time of the on-time marker.

The clocks on the time servers have been keeping time in units of seconds and fractions since 1 January 1970, so that it is a simple matter to compare the local second with the received epoch by the use of the MJD alone. If $T^{\mathrm{s}}_{\mathrm{sys}}$ is the seconds portion of the time of the clock on the client when the on-time marker is received, and if MJD, *h*, *m* and *s* are the Modified Julian Day number, the hours, minutes and seconds, respectively, in the message, then the integral seconds portion of the time difference (local client − ACTS server) is given by

$$\Delta t_s = T^{s}_{sys} - 86\,400(\mathrm{MJD} - 40\,587) - 3600h - 60m - s, \quad (1)$$

where 40 587 is the MJD corresponding to 1 January 1970. This calculation is done in integer arithmetic to prevent round-off or truncation. When the clocks are synchronized, the result of this calculation is either 0 or −1. The fraction of a second portion of the time of the client, $T^{\mu s}_{sys}$, must be added to this value to give the final time difference. This fraction has units of microseconds on most systems, so that the time difference (local clock—ACTS) is given by

$$\Delta t = 10^6 \Delta t_s + T^{\mu s}_{sts}. \quad (2)$$

This calculation is also done in integer arithmetic and the calculations are divided into two parts as shown to prevent loss of significance (or integer overflow) in the intermediate results.

This is the easy part of the protocol. The hard part is ensuring that the advance of on-time marker accurately reflects the path delay and that the client system receives it and processes the associated time tag with negligible delay. The following discussion will explain how these requirements have been addressed.

5. The ACTS servers

The hardware used to support the ACTS servers is divided into two parts: a modem and a measurement module for each telephone line that is being used to support the protocol and a single computer that generates the time code and handles the housekeeping for a number of telephone lines. In the current implementation, a single computer handles either four or eight telephone lines. (The number of serial lines is set by the hardware used to interface them to the computer, and some of the older ACTS computers could also support six lines.)

The improvements to the ACTS system are primarily a result of the separation of the hardware that measures the path delay and the computer that generates the time codes. These two functions were combined in a single system in the previous version.

Using a general-purpose computer as an ACTS server has the important advantages that the hardware is a standard off-the-shelf item and the software can be developed and tested using widely available software tools. This advantage is offset by the facts that there can be significant jitter in the receipt of the on-time marker echoed by the remote user and it is generally impossible to control the time at which a character is actually transmitted to the remote user by the serial output device. Both of these problems tend to become more serious as the load on the system increases, and these problems are only marginally improved by using a faster system, since the process is primarily limited by the input/output speed of the serial connections. These two problems resulted in a noise floor of the original ACTS system of 1 ms to 2 ms RMS, with a spectrum that was consistent with white phase noise for averaging times up to the duration of the telephone connection. This value was adequate for that time, but it has now become an important limit in the overall performance of the time service.

The system we describe here has addressed these limitations by using the general-purpose computer to generate the time code, since the time code is rather complicated but changes only every second. The code is also highly redundant from second to second so that only a small part need be re-calculated almost all the time. The time-critical aspects of the protocol are handled by a group of special-purpose hardware modules that are dedicated to this function. Each of these modules handles the delay measurement for only a single telephone line. The construction of these modules is simplified by the use of cross-compilers that support programming the integrated circuits using a high-level language.

6. The common computer

The computer is synchronized to UTC by means of 1 Hz pulses from the atomic clock ensemble. The pulses are interfaced to the system through one of the interrupt lines on a serial port that is dedicated to this function. (It is also possible to use 1 Hz pulses from any other precision timing source, such as a GPS receiver or a local atomic standard, and this alternative input can be used if the signal from the clock ensemble becomes unavailable for any reason.) The current configuration uses the ring indicator (RI) status line for this purpose, but any other status line that can be read by the driver can be used. The interrupt service handler for the serial port has been modified so that the jitter in processing these pulses is less than 10 µs. (The resolution of the system clock is 1 µs.) However, the computer is not used for any precision timing, and this is not a critical specification. The UTC epochs (that is, the names to be associated with the 1 Hz ticks) can also be derived from any source, including a local GPS receiver or a network time server. The epoch can even be manually set by the operator, since it need be set only once during a cold-start and need be accurate only to the nearest second.

I will not discuss the routine housekeeping tasks that are needed to support the algorithm, and I will concentrate on the generation and transmission of the on-time markers, which are the aspects that define the performance of the system. These housekeeping tasks include monitoring the status of each serial port to detect new connections and disconnections, monitoring the duration of a call, providing a visual display of the state of the system and sending a help message if it is requested by the remote system.

When the computer receives a 1 Hz tick on the serial port indicating that a new second has started, it proceeds to generate the new time code. The epoch in the time code corresponds to a time that is 1 s in the future, since it will be associated with an on-time marker that will be transmitted near the end of the current second and received 1 s later. Most of the parameters in the time code can change only at the first second of a new day, and it takes about 450 µs to generate the full new code at that time. (This new-day time code is actually generated at 23 : 59 : 59 of the previous day as described above. If a positive leap second is in progress at that point, the system transmits the correct time code of 23 : 59 : 60 and generates the time code for the next day during the leap second. The software also supports negative leap seconds, and the time code for the new day would be generated at 23 : 59 : 58 in that case. However, negative leap seconds will never happen with the current definition of the length of the SI second.) At other epochs the time to generate the new time code is about 37 µs (the increase in the computation time due to a rollover of the minute or the hour is too small to measure). However, the measurement hardware has control of the modem at that point (see below), and the computer remains idle. If there are no active connections, then nothing happens until the next second, when a new time code is generated and the loop repeats. (A time code is generated even if there are no active connections so that the time code on the operator's display will remain correct.)

If any serial port has an active connection, the computer sends the new time code on that line starting at 250 ms into the second. The time code consists of 51 characters (starting with carriage return, line feed, the MJD value and ending with the space just before the on-time marker). The slowest supported telephone line speed is 1200 baud, so that the time code takes no more than 425 ms to transmit, and the transmission has finished no later than 675 ms after the start of the second. (The modems that handle the connections are configured to negotiate the telephone line speed automatically when the call is first received. However, the communication line between the computer and the external hardware runs at a fixed speed of 9600 baud, so that neither the computer nor the external measurement hardware need to know the telephone line speed that was negotiated. As we will show below, the auto-negotiation process is not optimum, and the system would provide somewhat better accuracy if the line speed were also fixed at 9600 baud.) The computer hands control of the serial port over to the external measurement hardware at that point. If nothing else happens, the computer takes back the control of the line at 250 ms into the next second, and the process repeats as long as the connection is active. The computer will disconnect the caller after 40 time codes have been sent (by turning off Data Terminal Ready, which forces the modem to hang up and reset itself), unless the remote end hangs up first, which is detected when the Carrier Detect status line from the modem switches to FALSE. The system returns to an idle state at that point and waits for the next call on that line.

7. The measurement hardware

If the computer has transmitted a time code on any serial line, then it enables the corresponding measurement module for that line starting at 700 ms after the start of the second. On the first cycle of a connection (or on any cycle if no on-time marker has been received on this line during the previous cycle), the measurement module sends the default character '$*$' at 855 ms after the start of the second (an advance of 145 ms with respect to the next second) and it immediately starts listening for an echo, since the echo might be received in the current second if the round-trip delay is less than 145 ms.

If the module receives an on-time marker from the remote end before 150 ms into the next second, it considers the echo to be valid. It uses the elapsed time since it sent the on-time marker as the round-trip delay and sets one-half of this value as the advance for the on-time marker to be sent on the next second. The module will regain control of the serial line in the next second after the computer has sent the time code and it will use the advance just computed to decide when to send the on-time marker. The goal is to have equality between the advance value and the time after the 1 Hz tick at which the echo is received. After four successful measurement cycles, the on-time marker is changed to '#' to show that it is being measured and is stable. It transmits the computed delay to the computer through the serial port so that the computer can insert the value into the next time code.

If the module does not receive an on-time marker within 150 ms, then the module returns a time-out status back to the computer. It resets the on-time marker to '$*$' and sets the advance to the default value for subsequent transmissions. The operation continues until the computer stops enabling the line, at which time the module resets its configuration to be ready for the next call.

The advance used for every on-time marker is based on a round-trip measurement made during the previous second, so that there could be some advantage, in principle, in having the client correct the time difference measured with any on-time marker with the difference between the delay in the time message that precedes it with the delay in the message that follows it. This method turns out to be essentially equivalent to averaging the time differences themselves, since both are well characterized as white phase noise over the duration of a telephone call. (The implicit requirement that the signal path and the clocks be well behaved during the message exchange is a common feature of all two-way protocols [4].)

Each module is synchronized from the same 1 Hz pulses that synchronize the computer. In addition, the module uses an external oscillator (5 MHz or 10 MHz) to drive the counters that measure the delay. The module has no need for epoch information.

The modules are implemented using commercial programmable logic arrays. The receipt of the on-time marker from the remote system and the transmission time of the on-time marker to the remote user is controlled by a clock that is synchronized to the external 1 Hz timing pulses. The internal timing control has a resolution of 25 µs, which is an improvement of about a factor of 10 relative to the previous implementation of the system. The external reference signals that are used to synchronize the hardware are derived from the NIST atomic clock ensemble and have stabilities and accuracies several orders of magnitude better than this value.

8. Performance testing

I tested the performance of the ACTS servers with the configuration shown in figure 3. The modem dials the ACTS server. When the connection is established, an external circuit, connected to the output data line from the modem, monitors the received characters. When an on-time marker is received (either '$*$' or '#'), the circuit echoes the on-time marker back through the transmit line of the modem. It also emits an

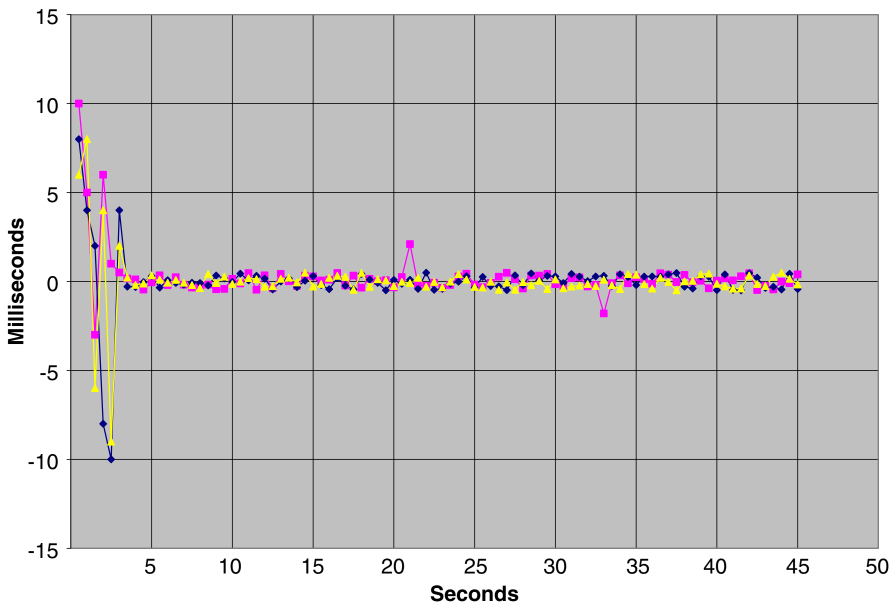
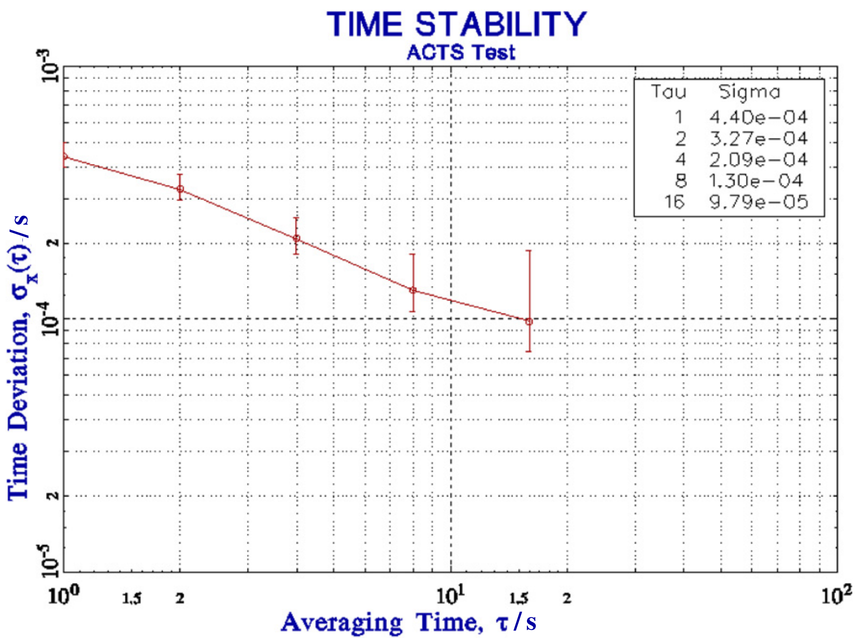


Figure 5. The time errors on three consecutive calls spaced a few minutes apart measured by the test configuration shown in figure 3. The ACTS server switches to measured delay mode and changes the on-time marker to a '#' after the fourth time code. The reference for the time interval counter is UTC(NIST), so these data are a measure of accuracy as well as stability.



Tau	Sigma
1	4.40e−04
2	3.27e−04
4	2.09e−04
8	1.30e−04
16	9.79e−05

Figure 6. The TDEV of one of the data sets from the previous figure.

output pulse. The time of the output pulse is compared with a separate time reference by means of a standard time interval counter. Since the time reference for the time interval counter is UTC(NIST), these data are a measure of accuracy as well as stability. The results are shown in figure 5, and the TDEVs of these data (starting with the fifth point) are shown in figure 6.

The test results shown here used the same brand of modem at both ends of the connection. The stability was about the same when different brands of modems were used at each end of the connection, but the accuracy varied from brand to brand. Some brands of modems had offsets as large as 6 ms with respect to the 'standard' device, which is simply the one that we chose for the server. I also found some variation among different modems of the same brand, although this variation was less than ±0.8 ms maximum and might be attributed to fluctuations in the telephone connection.

Although the ACTS service was never intended to be a method for calibrating the frequency of a remote device, these data show that it can perform this function in some less-demanding applications. For example, if the user were to average 10 consecutive time differences, then the data of figure 6 would suggest that the variance of the average would be about 100 μs. If the user made a second time comparison 2 h later, the variance in the time measurements would translate

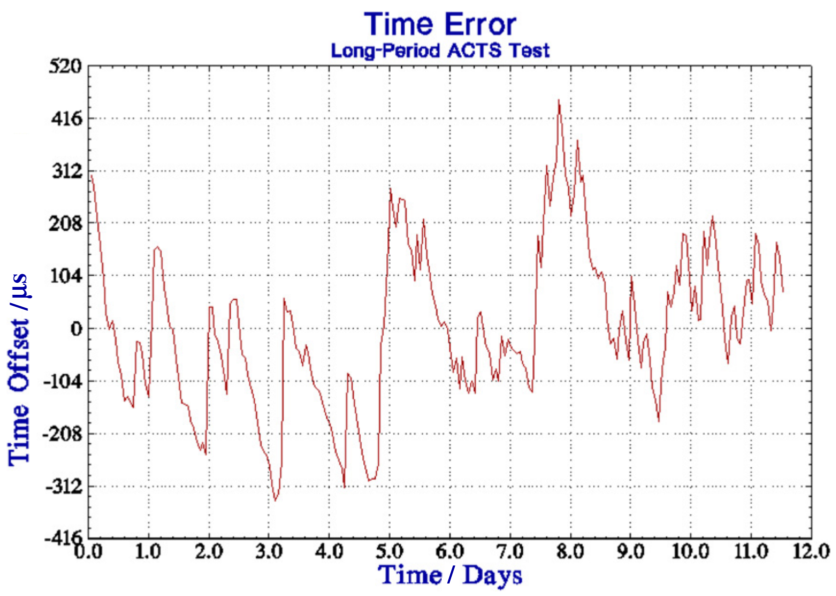


Figure 7. The time errors of the ACTS system. The data are acquired by the test configuration shown in figure 3. The measurement is repeated every hour for 12 days, and the results are shown in this figure.

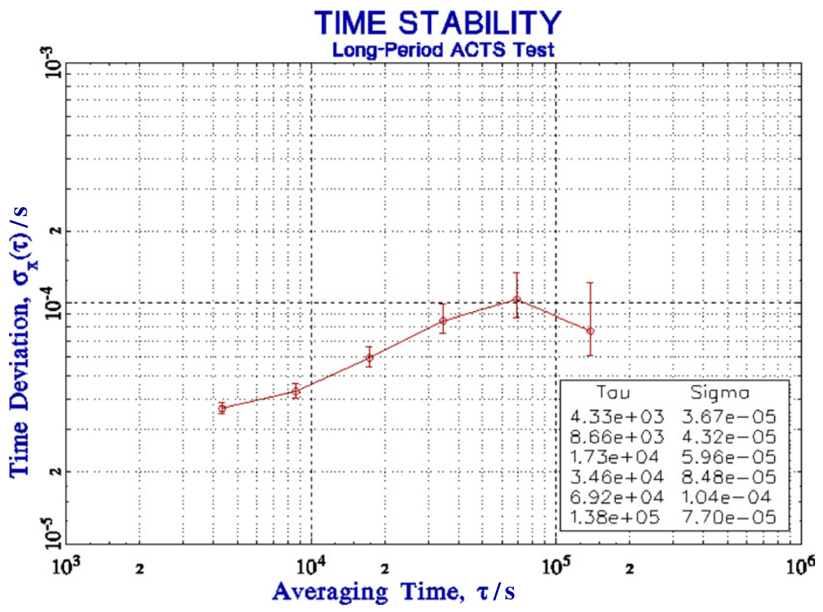


Figure 8. The time deviation (TDEV) of the data shown in figure 6.

into an uncertainty in the frequency estimate of about $\sqrt{2} \times 10^{-4}/7200 = 2 \times 10^{-8}$, which is adequate for calibrating many lower-end quartz-crystal oscillators. (In order to perform such a calibration, the time differences between the 1 Hz pulses from the device under test and the arrival times of consecutive on-time markers would be measured, and the time differences received during each telephone call would be averaged. The details of this comparison would depend on the device being calibrated; one way of doing this has been described elsewhere [3].)

I repeated this test every hour for 12 days to estimate the long-term stability of the ACTS system. The results of this test are shown in figure 7, and the TDEVs of these data are shown in figure 8. The data do not have a statistically significant offset in time or in frequency, but there is a clear diurnal fluctuation, which is probably, in part, due to the diurnal change in the load on the telephone system.

Finally, I have tested the performance of the ACTS server when the telephone line speed is varied from 1200 baud to 19200 baud. The connection between the ACTS server and the modem and between the remote modem and the test hardware of figure 3 run at a constant speed of 9600 baud, and the different speeds are accommodated automatically within the modems. This accommodation is simplified by the fact that the actual number of characters transmitted in either direction is less than the baud rate even at the slowest supported speed, so that questions of buffer overflow need not be considered.

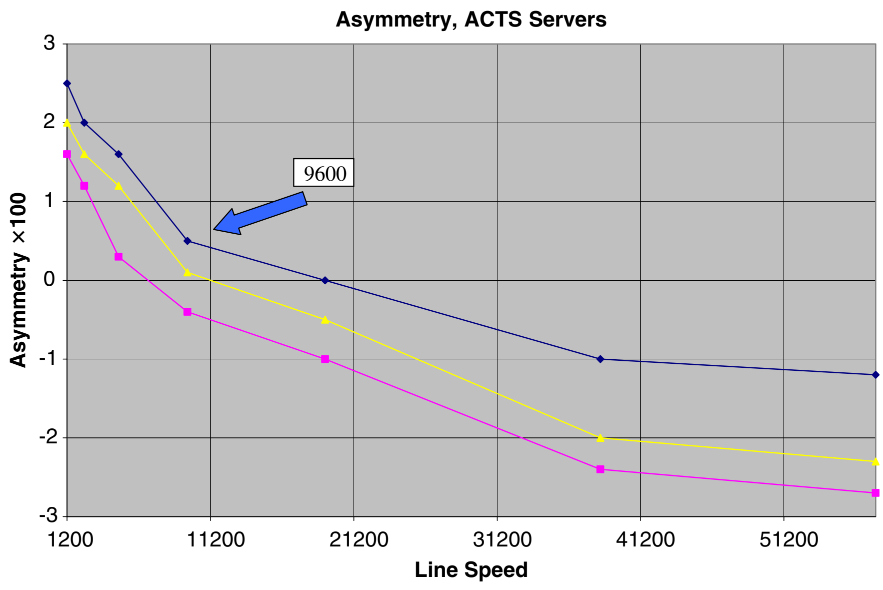


Figure 9. The asymmetry (as defined in the text) in the channel delay measured as a function of the line speed of the telephone connection. The test used the setup of figure 3 and varied the speed of the telephone connection. The connections between the modems and the hardware at each end used a constant speed of 9600 baud. The three traces show the variation in the measured asymmetry over a number of calls spaced a few minutes apart. The arrow shows the measurement when the line speed was the same as the speed used to connect the modems to the external hardware.

I have characterized the time offsets of these data in terms of the effective asymmetry of the communications line (including the modems). The two-way algorithm that is the basis of the ACTS protocol assumes that the inbound and outbound delays are equal in magnitude. An asymmetry of $X\%$ means that the outbound and inbound delays (viewed from the point of view of the server) are $50 + X\%$ and $50 - X\%$, respectively. The resulting time error is $XD/100$, where D is the round-trip delay. In the configuration that I used for testing, an asymmetry of 1% resulted in a time offset of about 1.6 ms, so that the maximum asymmetry I measured at any speed (about $\pm 3\%$) would result in a time error of about 5 ms. This asymmetry is probably mostly a function of the modems, and its magnitude would probably be substantially independent of the length of the telephone line. However, I have not verified this.

The results of this measurement are shown in figure 9. The three traces show the maximum, average and minimum asymmetries measured by means of a series of telephone connections spaced a few hours apart. The asymmetry is the smallest when the telephone line speed is the same as the speed used to connect the modems to the external hardware. These results vary somewhat from one brand of modem to another, but I have not conducted an exhaustive test of this variation. However, in all of the brands that I have tested, the asymmetry is smallest at 9600 baud.

9. The NIST time servers

The NIST time servers respond to requests for time in three different formats: the TIME [5] protocol, the DAYTIME [6] protocol and NTP, the Network Time Protocol [7]. Each of the protocols uses the system clock as its time reference, and the system clock is synchronized to UTC(NIST) by a separate algorithm [8] called LOCKCLOCK. (Implementing the synchronization of the local clock as a separate algorithm rather than as part of the process that responds to requests for time allows the synchronization process to be optimized independently of the daemon processes that respond to time requests.) The LOCKCLOCK algorithm uses periodic telephone calls to the ACTS time service to synchronize the local system clock, and the front-end of the algorithm, which is the interface to the ACTS system, has been improved. These improvements, combined with the new ACTS servers described above, have resulted in a significant improvement in the accuracy and stability of the time servers.

The interface between LOCKCLOCK and ACTS is through a serial port on the time server, and that interface has been improved in the following ways.

1. The echo of the on-time marker has been moved into the interrupt service for the serial port, so that the on-time marker is echoed back to the ACTS system with a delay that is both much smaller and much more stable than in the previous version of the software. As I discussed above, the ACTS protocol assumes that this echo delay is 0.
2. The serial port driver now assigns a time tag to an on-time marker and passes this value to the LOCKCLOCK program. In the previous version, the association of the time stamp and the receipt of the on-time marker was not completed until the on-time marker character had been passed to the application layer.
3. The delay between the receipt of an on-time marker by the hardware and the time when this value is available to one of the server processes has been shortened so that it does not exceed 50 µs in normal operation and is generally about 20 µs RMS. This reduces the portion of the channel delay that is not part of the two-way measurement process.

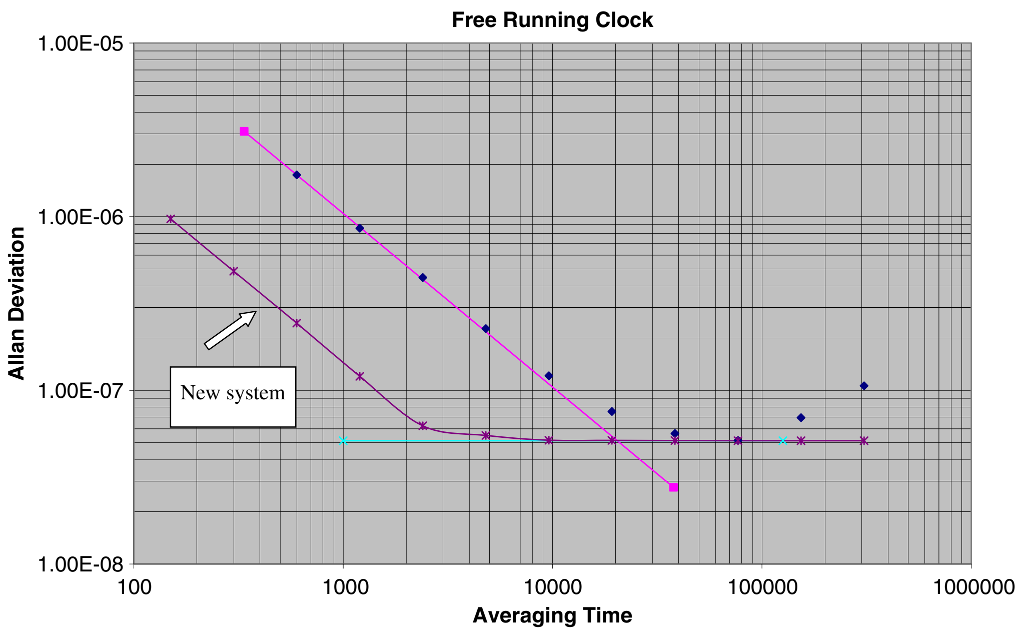


Figure 10. The performance of the system clock used as the reference for the time service, showing the performance with the old and new ACTS interfaces. The flicker floor of the system is determined by the characteristics of the oscillator used in the system clock and has not changed. The straight lines through the points have slopes of -1 and 0 to indicate the noise types that dominate in the different measurement regimes [9]. The measurements based on the old ACTS system are taken from [8], figure 1.

The result of all of these improvements is shown in figure 10. The upper trace is taken from figure 1 of the original LOCKCLOCK description [8], which shows the free-running stability of a system clock measured by means of periodic calls to the ACTS server. The lower trace shows the measurements made with the new system. The improvement is almost a factor of 10 for all averaging times until the flicker floor (due to the oscillator in the computer clock) is reached.

10. Adding an external oscillator

I have added an external rubidium-stabilized oscillator to some of the time servers and I have interfaced the 1 Hz output pulses from these devices to the RI (RING) interrupt line on a dedicated serial port (as I described above for synchronizing the ACTS system.) The frequency stability of this oscillator is better than 5×10^{-11} for averaging times out to several hours, so that its performance is several orders of magnitude better than the internal quartz-crystal oscillator that drives the system clock. The jitter in processing an interrupt on the RI line is about 10 µs, and it is well characterized as white phase noise. If the system clock is compared with the rubidium standard every second, the system clock can be synchronized to within 1 µs, (which is the resolution with which it can be read) in about 100 s, so that the short-term stability of the time of the server is greatly improved.

Although the short-term stability of the server is improved, the time accuracy of the server is not. The epochs of the ticks from the rubidium standard must be calibrated by the use of the ACTS system, and this calibration can be no better in long term than the performance of ACTS itself. (See figure 6.) This calibration is an ongoing process, since the frequency ageing of a rubidium standard may be small, but it is not 0, and the long-term stability of the servers is derived from the ACTS system, which is directly traceable to UTC(NIST). The time constant used to calibrate the output frequency of the rubidium standard varies from one device to another, but is typically about 1 month. Nevertheless, the rubidium standards provide much better hold-over performance (that is, the stability of the time server if the link to ACTS fails) than could be realized by the use of the internal computer oscillator alone, and it has been added to the server primarily for this reason.

11. Summary and conclusions

I have described improvements to servers that support the NIST Automated Computer Time Service (ACTS). The stability is improved by about a factor of 5 compared with the previous version; the accuracy is also improved, although this improvement will depend to some degree on the line speed and the brand of modem that are used. These improvements will be available to any application that uses the ACTS system.

The software that controls the serial-line interface on the Internet time servers that links the servers to the ACTS system has also been improved. The jitter in the delay between when an on-time marker is received and when it is echoed has been reduced by moving the echo function into the interrupt service routine for the serial line. The algorithm that applies the time stamp to the reception time has also been improved in a similar fashion.

Taken together, these improvements have resulted in an improvement of almost an order of magnitude in the performance of the Internet time servers. This improvement is most visible for users of the Internet time service whose network connections have delays that are either small or very symmetric, so that the time they receive from the server is not degraded by fluctuations in the network delay or in its inbound–outbound asymmetry. This same limitation is true for those servers that have external rubidium devices—the increased stability may not be visible to users with network connections that do not have stable, symmetric delays.

The new servers are particularly well suited for use on local area networks that may be isolated from the public Internet for various reasons. Such networks often have a relatively small number of routers and intermediate gateways, so that the network delays are usually stable and symmetric. This type of network configuration allows client systems to realize the full potential of the servers, which provide stratum 1 performance and traceability to UTC(NIST) without the need for an external antenna.

References

[1] The NIST web page contains a list of the current servers. See http://tf.nist.gov/tf-cgi/servers.cgi

[2] Levine J 1998 Time synchronization over the Internet using 'AUTOLOCK' *Proc. IEEE Int. Frequency Control Symp. (Pasadena, CA)* (IEEE Catalog No 98CH36165) pp 241–9

[3] The ACTS protocol and the first servers are described in, Levine J, Weiss M, Davis D D, Allan D W and Sullivan D B 1989 The NIST automated computer time service *J. Res. Natl Inst. Sci. Technol.* **94** 311–21
See also Beehler R E and Lombardi M A 1991 *NIST Time and Frequency Services NIST (Boulder, CO) Special Publication* 432
Levine J, Lombardi M A and Novick A N 2002 *NIST Computer Time Services: Internet Time Service (ITS), Automated Computer Time Service (ACTS)* and time.gov web sites *(Boulder, CO) NIST Special Publication* 250–9

[4] Levine J 2008 A review of time and frequency transfer methods *Metrologia* **45** S162–74

[5] The format of the messages in this protocol is described in Request for Comments (RFC) 868, available on the web at tools.ietf.org/html/868

[6] The format of the messages in this protocol is described in Request for Comments (RFC) 867, available on the web at tools.ietf.org/html/867

[7] The format of the messages in this protocol is described in Request for Comments (RFC) 1305, available on the web at tools.ietf.org/html/1305

[8] Levine J 1995 An algorithm to synchronize the time of a computer to universal time *IEEE/ACM Trans. Networking* **3** 42–50

[9] The significance of the slope is discussed in Levine J 1999 Introduction to time and frequency metrology *Rev. Sci. Instrum.* **70** 2567–95
See also Sullivan D B, Allan D W, Howe D A and Walls F L (ed) 1990 Characterization of clocks and oscillators *NIST Technical Note* 1337, Boulder, CO