NIST Special Publication 250-59

NIST Computer Time Services: Internet Time Service (ITS), Automated Computer Time Service (ACTS), and time.gov Web Sites

Judah Levine
Michael A. Lombardi
Andrew N. Novick

Time and Frequency Division
Physics Laboratory
National Institute of Standards and Technology
325 Broadway
Boulder, Colorado 80305

May 2002

U.S. Department of Commerce
Donald L. Evans, Secretary

Technology Administration
Phillip J. Bond, Under Secretary for Technology

National Institute of Standards and Technology
Arden L. Bement, Jr., Director

National Institute of Standards and Technology Special Publication 250-59
Natl. Inst. Stand. Technol. Spec. Publ. 250-59, 77 pages (May 2002)
CODEN: NSPUE2

CONTENTS

Chapter 1 - Internet Time Service (ITS)

The National Institute of Standards and Technology (NIST) provides its Internet Time Service (ITS) to the general public. The service distributes Coordinated Universal Time (UTC) kept at NIST and called UTC(NIST), which is an official source of time both in the United States and internationally. The ITS is an important part of the nation's timekeeping infrastructure. Primarily used for the synchronization of computer clocks, the service currently (2001) handles hundreds of millions of timing requests per day. This chapter describes the ITS in detail. It provides a physical and technical description of the service, and explains how NIST provides and maintains the ITS for the American public.

I. Physical Description of Internet Time Service (ITS)

The Internet Time service is based on commercial UNIX workstations whose clocks are synchronized to UTC(NIST) by means of periodic dial-up telephone connections to the Automated Computer Time Service (ACTS) system described in Chapter 2. Some servers have an additional external reference clock to improve the reliability and short-term stability of their transmitted time. This section provides a physical description of the service.

A. Introduction and History

The first Internet time servers provided by NIST went online in 1993. From the beginning, these servers were designed to respond to requests in any of three standard Internet time protocols: the character-based *Daytime Protocol* [1] (similar to the ACTS format described in Chapter 2), the *Time Protocol* [2], which provides the time as a binary number of seconds since 1900, and the *Network Time Protocol* (NTP) [3, 4], a binary protocol that is more complex and powerful than the other two.

Many ITS client programs were developed at about the same time and were distributed via the Internet using the File Transfer Protocol (FTP) and similar methods. NIST distributed client software utilizing the Daytime Protocol for popular operating systems such as MS-DOS, Windows, UNIX, and VAX/VMS; the same client platforms that used the earlier ACTS system. However, both the Time and NTP formats were existing network standards, and from the very beginning, many ITS users synchronized their computer clocks using third-party software. The most widely used NTP client is the version distributed by David Mills of the University of Delaware; many clients of the Time Protocol use software developed by Novell and other commercial network operating system vendors.

Since it was first begun, the ITS has grown significantly in terms of the number of servers, the number of time requests received per day, and the sophistication and versatility of the client software. At this writing (November 2001), there are 14 servers in operation, and the system receives well over 300 million time requests per day in the three supported formats. An additional server, located at the NIST Boulder laboratories, is used for software testing, and two additional servers have been installed in Tokyo, Japan. The number of time requests is growing at a rate of about 8 % per month compounded; this rate of increase has characterized the service since it began and it shows no sign of decreasing.

The growth rate of the ITS is not completely constant. External events sometimes cause large, sudden increases in traffic. When these large increases in usage occur, many new users become permanent users. For example, on January 1, 2000, the day of the Y2K rollover, the ITS handled more than 238 million timing requests, or what was then about seven times the usual load. While most of that increase in the load was a transient effect, some of it became permanent.

A graph detailing ITS growth is shown in Figure 1. Note that NTP timing requests now account for more than 90 % of the total number of timing requests, and that usage of the Daytime and Time protocols has essentially leveled out. It is also worth noting that the number of NTP requests far exceeds the number of actual computers using the ITS, since a typical NTP client might access the service three times per hour, and some clients access it even more often.

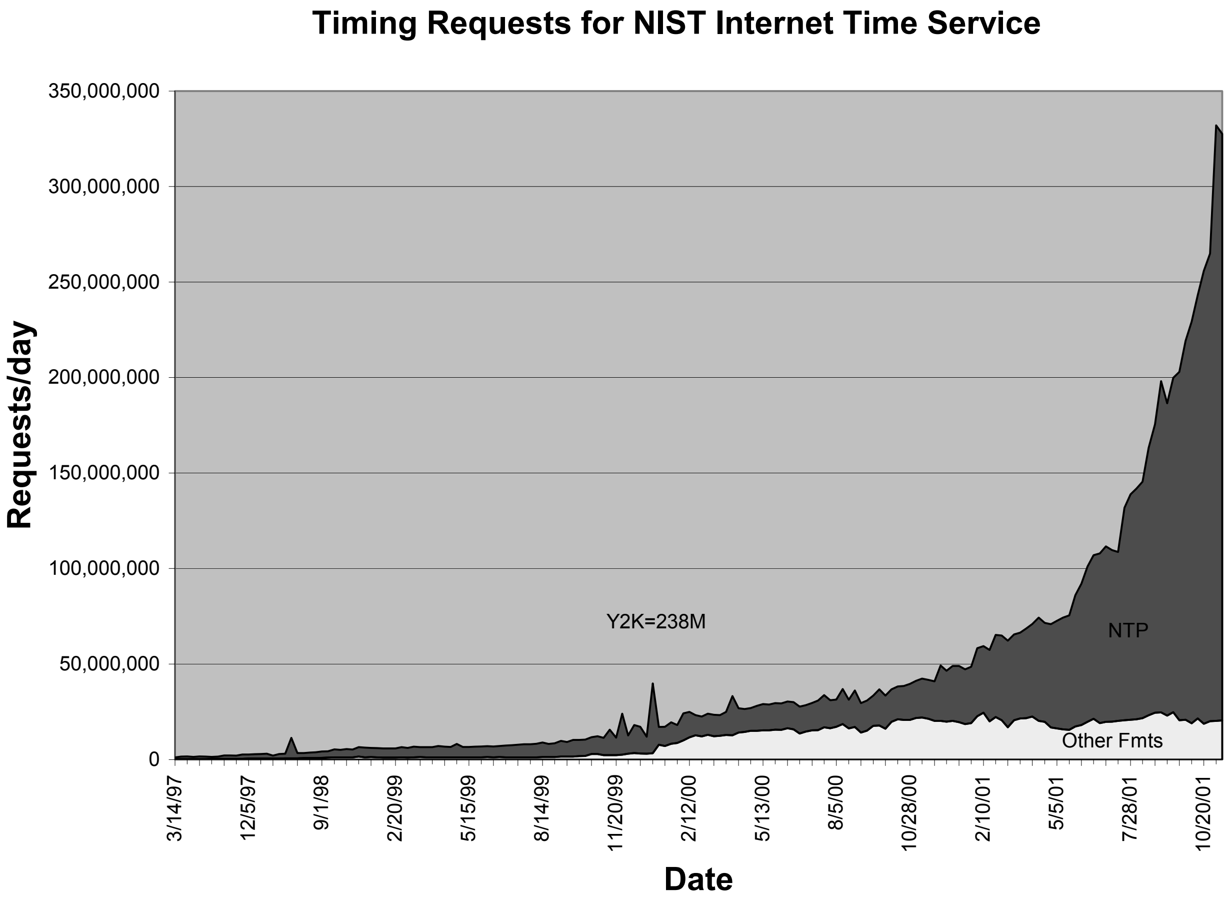


Figure 1. Growth of the Internet Time Service based on number of timing requests.

B. How ITS is Used by its Customers

It is difficult to characterize hundreds of millions of daily time requests in any simple way. We do not know most of our customers; we find out why they use the service only when they contact us requesting information or assistance. Thus, our description of users is probably incomplete. Nevertheless, we can identify some general classes of users:

- E-commerce and e-business users who synchronize the clocks of systems used to time stamp commercial and financial transactions. This class of users also includes commercial applications similar to the function of a Notary Public or to the Registered Mail and Return Receipt Requested services provided by the United States Postal Service.

- Users who depend on time synchronization in support of authentication and validation. Many "smart card" challenge-response systems depend on time synchronization between the server and the client that is requesting service. This class of users includes some virtual private networks and equivalent public-network configurations that depend on passwords or session keys that are either derived directly from the local clock, or else change periodically at intervals determined by the local system time.

- Users of distributed database systems who must guarantee the correct time order of transactions that might be received by a central server some time after they were actually entered into the system.

- Distributed data-logging systems, where the clock on each remote node is used to time stamp an event as received at that node. These time stamps must be consistent and reliable so that data collected at multiple sites can be combined and compared.

- Special purpose time stamp hardware, such as employee time clocks, parking ticket systems, and so on. Many mechanical clocks used to time stamp documents use either ITS or ACTS signals to establish traceability to NIST.

- Individual personal computer users who want a reliable time source in their home or office for informal reference.

C. ITS Facilities

The ITS is composed of multiple computers configured as *time servers*, which respond to time requests from users. These time servers are available for public access 24 hours per day, 7 days per week. They reside in a number of different cities, but all are controlled from the NIST laboratories in Boulder, Colorado. Table 1 lists each time server, its location, and its Internet Protocol (IP) address. The most current version of this table is maintained at:

http://tf.nist.gov/service/time-servers.html

Table 1. The NIST Internet Time Servers.

Address	ID #	IP Address	Facility	Location
time-a.nist.gov	1	129.6.15.28	NIST	Gaithersburg, Maryland
time-b.nist.gov	2	129.6.15.29	NIST	Gaithersburg, Maryland
time-.timefreq.bldrdoc.gov	3	132.163.4.101	NIST	Boulder, Colorado
time-b.timefreq.bldrdoc.gov	4	132.163.4.102	NIST	Boulder, Colorado
time-c.timefreq.bldrdoc.gov	5	132.163.4.103	NIST	Boulder, Colorado
time.nist.gov	6	192.43.244.18	National Center for Atmospheric Research (NCAR)	Boulder, Colorado
time-nw.nist.gov	7	131.107.1.10	Microsoft Corporation	Redmond, Washington
nist1.dc.certifiedtime.com	8	216.200.93.8	Abovenet	Vienna, Virginia
utcnist.colorado.edu	9	128.138.140.44	University of Colorado	Boulder, Colorado
nist1.datum.com	10	63.149.208.50	Datum/Bancomm	San Jose, California
nist1-aol-va.truetime.com	11	205.188.185.33	America On-Line	Virginia
nist1-aol-sj-truetime.com	12	207.200.81.113	America On-Line	San Jose, California
nist1.ny.glassey.com	13	208.184.49.9	Abovenet	New York City
nist1.sj.glassey.com	14	207.126.103.204	Abovenet	San Jose, California
time-d.timefreq.bldrdoc.gov	15	132.163.4.104	NIST (test system)	Boulder, Colorado
	16	64.56.180.41		Tokyo, Japan (being tested)
	17	64.56.180.42		Tokyo, Japan (being tested)

The time servers are located in environmentally controlled rooms or laboratories. At the NIST laboratories, backup power from uninterruptible power supplies (UPS) can run the hardware for about 20 minutes. The UPS devices are connected to a generator that starts automatically when the power fails. The generator is normally running within 60 seconds after a power failure, and can support the entire load indefinitely. The time servers located outside of NIST all use some form of backup power, either a local UPS, a generator, or a combination of both.

1. Configuration of ITS Time Servers

Each time server is configured with an analog modem that connects to a dedicated telephone line. The time servers periodically dial the ACTS system located in Boulder, Colorado (Chapter 2) and synchronize their internal clocks. The modems are configured as outbound only devices; any incoming calls to the modem are ignored.

The time servers located at the NIST facility in Boulder (IP addresses 132.163.4.x) also have a direct connection to a 1 Hz signal from the NIST primary time and frequency standard, UTC(NIST). These 1 Hz pulses are used to phase lock the internal system clock, and improve the short-term stability of the system between calls to the ACTS servers. They have little effect on the long-term accuracy of the time servers, since the same 1 Hz pulses are used to synchronize the ACTS servers themselves.

Some of the time servers located outside of NIST are installed with ancillary rubidium standards. These standards improve the short-term stability of the time signals and provide a limited “hold-over” capability if the servers cannot call ACTS. These ancillary standards have little or no impact on the accuracy of the time stamps.

All time servers are equipped with standard Ethernet hardware that enables them to receive and respond to Internet requests using the standard Internet Protocol (IP) suite.

D. Organizational Control of ITS

The ITS is managed, controlled, and maintained by the Network Time Services Group of the NIST Time and Frequency Division. The Group Leader of the Network Time Services Group reports to the Division Chief of the Time and Frequency Division, who in turn reports to the Director of the NIST Physics Laboratory. The Physics Laboratory is one of the seven operational units of NIST, and NIST is an agency of the United States Department of Commerce.

Support for the ITS is also provided by the Services Group of the Time and Frequency Division, who assists with customer support; and by the Information Technology Laboratory (ITL) of NIST, who assists with Internet and local area network issues.

The servers at remote locations are supported to some extent by the operations staff at the host location. This support is generally limited to assistance with hardware repairs, as needed.

II. Technical Description of Internet Time Service

This section describes the technical operation of the ITS, including a discussion of the hardware used, and the client and server side software.

A. Technical Description of Hardware

The ITS is built entirely from commercially available hardware that has not been modified by NIST. This section describes the specifications of the hardware.

1. Computer Systems

The time servers are manufactured by Compaq (formerly Digital Equipment). The model used is the Alphaserver DS10, or an equivalent model or upgrade. A typical configuration includes a 466 MHz CPU, 512 megabytes of RAM, a 9 gigabyte hard drive, two serial ports, a network interface, a display, mouse, and keyboard. Although these servers are similar to those made by other manufacturers, their internal clocks are well suited to time service applications because of their inherent frequency stability at periods of up to a few thousand seconds.

2. Modems

Each time server includes an external modem that is interfaced via a serial (RS-232) port. The modem connects to a standard telephone circuit with a RJ-11 jack. Most systems use U.S. Robotics modems, but other brands are also used. The modems were chosen for the stability of the internal equalizing circuitry, which makes them well suited for time-service applications.

The telephone connection is restricted to outbound-only calls. This restriction is often realized in the telephone hardware by placing the line on a portion of the local switch that cannot be reached by outsiders. The server will not accept incoming telephone calls, even if the telephone rings. This is realized by disabling the data terminal ready (DTR) line to the modem so that the modem cannot answer an incoming call.

3. Network Connection

The network connection uses standard cables and hardware, and is usually implemented using twisted-pair cables to auto negotiation 10/100 MHz hubs. The details vary at each site. All of the sites have some kind of firewall or proxy server, and the time server is always located on the "outside" of that system, since it must be visible to the public. The server is never part of the internal network of the site, so even a complete compromise of the server does not compromise the security of the network behind the firewall. The details of each network interface are approved by the local security and network administrators and are beyond the scope of this document.

4. External Reference Oscillator

Some ITS time servers are connected to an additional oscillator that improves both the short-term stability of the time service and its reliability in the case of a communications failure or problem with the ACTS system. Three different types of oscillators are used:

- A cesium oscillator that is always on-time and on-frequency (servers 3, 4, 5, 15).

- A rubidium oscillator that is not on time but has a frequency aging rate of less than 10^{-11} per month and a short-term frequency stability of better than 8×10^{-12} for averaging times of 10^4 s or less (servers 1, 2, 6, 9).

- A stabilized quartz oscillator that has short-term stability of 2×10^{-9} for averaging times up to 1000 s (servers 11, 12).

When an external reference oscillator is used, its 1 pulse per second (pps) signal is connected to the server using one of the status lines of the same serial port used to interface to the modem. The connection is made with a coaxial cable, and no special interface is used or required.

The rising edge of the 1 pps signal serves as the on-time marker. This edge must be compatible with TTL specifications and must have a rise time of less than 5 μs. The width of the pulse is not critical as long as it is wider than about 30 μs. Most commercial oscillators easily meet these requirements.

B. Technical Description of Server Software

The server software consists of the operating system; the software developed at NIST that handles time requests and steers the clock oscillator, and utilities used for file transfer and diagnostics.

1. Operating System

The operating system is based on Tru64 UNIX version 4.0F. The operating system's kernel was modified at NIST to improve its usefulness for time service applications. The principles of the modifications are described below, but the details are not public since they involve source code proprietary to Compaq. The modifications include:

- Automatic handling of leap seconds and leap second notification within the kernel. This support provides for automatic adjustment of the internal kernel clock at the instant of a leap second and for the maintenance of the flags that transmit leap second information to users.

- Automatic notification of the daylight saving time transition date. This feature was added using an algorithm that is based on the calendar date and incorporates the United States rules for the transition dates. Users outside of the United States use other methods to determine these transition dates; Microsoft Windows, for example, defines them using a location-based entry in the system registry.

- Watch dog timers in the kernel are used to detect a failure of the time server software. These timers are interfaced to the server software and are used to declare the system 'unhealthy' in case of a problem. They are also interfaced to the alarm network, which can alert NIST personnel via a numeric pager or via the Internet.

- High resolution (10^{-12}) frequency steering of the kernel time. This is an essential feature of the synchronization algorithm, since it makes it possible to realize the full stability of the system with only a modest number of telephone calls to ACTS.

- Support for transmitting International Atomic Time (TAI) and UTC simultaneously for users who need continuity of time intervals across leap seconds. This capability was developed in collaboration with David Mills of the University of Delaware. [5]

2. Time Server Software

The time servers interface to the outside world using several daemons. A *daemon* (pronounced DEE-muhn) is a program that runs continuously and exists for the purpose of handling periodic service requests that a server expects to receive. The daemon program forwards the requests to other programs (or processes) as appropriate.

The ITS time daemons conform to the specifications listed in the Request for Comment (RFC) documents that define all Internet protocols and conventions. The RFC documents that define the time daemons are listed in Table 2. A short description of each time code format is given below in Section II.C.

Table 2. Time Protocols and Conformance Documents.

Protocol	Conformance Document	UDP/IP Port for Request	Percentage of total ITS requests
Daytime	RFC-867	13	8 %
Time	RFC-868	37	1 %
Network Time	RFC-1305	123	91 %

The time servers continually listen and respond to requests on the ports listed in Table 2, and return a time code in the requested format. The interaction between the server and the client systems is defined and explained in the RFC document. The time servers also support software distribution using FTP.

a) Standard Daemons

Several of the standard daemons supplied with the operating system have been completely rewritten to improve their security or their usefulness for time service applications. These include:

- The *daytime* and *time* daemons, which provide time in formats compatible with RFC 867 and 868. The time daemon is supported for backward compatibility, but is not recommended for new users because its format does not support robust error detection or correction. It is still used by some proprietary network systems.

- The *NTP* daemon, which provides time in a format compatible with RFC-1305. Our version is based on NTP release 4.0.99i; the modifications include additional support for auditing and changes to the interaction between the NTP daemon and the kernel.

- The *finger* daemon, which is used to provide status information on our systems. Our version of the daemon provides a quick snapshot of the performance of the most critical parts of the algorithm. Unlike the standard finger daemon, it accepts no input and cannot be used to find out anything about the system beyond the default status report.

- The *exec*, *snmp* and *rpc* daemons, which are used to check for port scans and possible denial of service attacks.

- The *sendmail* configuration, which has been modified to address potential security vulnerabilities that are present in the standard. These modifications prohibit certain denial of service attacks and use of the time servers to propagate "spam" messages.

b) Clock Synchronization Daemons

The clock synchronization daemons are independent of those daemons that provide the time codes. Their function is to steer the internal clock oscillator of the time server so that it agrees as closely as possible with UTC(NIST). The daemons rely on telephone calls made to the ACTS service in Boulder as their reference for the on-time pulse and time-of-day message. These telephone calls are made at varying intervals (typically every few hours) and last about 15 seconds.

If an external oscillator is available, these daemons *evaluate* the server performance using both comparisons with the ACTS messages and internal statistical estimators of reliability. This evaluation is crucial for the rubidium and quartz oscillators described in Section II.A.4, since the output pulse from these oscillators is not on time, and might not be on-frequency. This evaluation is less important if a cesium oscillator is available as an external reference. However, it is still performed since it provides additional robustness to the error detection algorithms. The daemon code is the same on all servers; the configuration (the type of external reference oscillator, for example) is specified in a file.

Three clock synchronization daemons have been published in the literature; *lockclock* [6], *interlock* [7], and *autolock* [8]. The algorithms used by these daemons are the heart of the Internet Time Service. Their performance helps determine the measurement uncertainty, and their reliability establishes continuous traceability to UTC(NIST). The *lockclock* algorithm is the primary algorithm used by the ITS servers. Some features of *interlock* and *autolock* have been implemented in the server software; other features of these algorithms are used by client software, and we expect to add more features in the future.

1. Lockclock Algorithm

The *lockclock* algorithm synchronizes time server clocks to UTC(NIST) with an uncertainty of about 1 ms (1σ). The algorithm models the performance of the server clock, and steers it to agree with UTC using a frequency locked loop (FLL). It makes periodic time corrections based on a statistical evaluation of the server clock oscillator frequency. The server clock oscillator is repeatedly calibrated by calling ACTS using ordinary telephone circuits. The uncertainty of the server time is limited primarily by the stability of the clock oscillator, but is also limited by the resolution of the server clock and the operating system latency.

The Deterministic Portion of the Clock Model - The model begins by measuring the time offset of the server clock with respect to UTC(NIST). The measurement is made at the epoch t_k, and returns the time difference, x_k, between the server clock and UTC(NIST). A positive value of x_k signifies that the server clock is fast. The dimensionless frequency offset of the server clock with respect to UTC during the interval between t_{k-1} and t_k is given by y_k, and the filtered frequency estimate at epoch t_k (using the method to be described below) is $\overline{y}_k$. The algorithm drives x_k to zero by estimating the optimum value for the filtered frequency $\overline{y}_k$ and uses this frequency to periodically correct the server clock's time.

The time offset measurements are approximately equally spaced in time by the calibration interval τ_0 measured by the server clock. ACTS is called when τ_0 has elapsed. The actual interval between t_{k-1} and t_k is τ_k; it is computed from the ACTS messages and is larger than τ_0 by about 30 or 40 s; or by the amount of time needed to complete the ACTS call and to perform several housekeeping chores.

We predict the time offset at t_k using the average frequency offset estimated in the previous measurement cycle:

$$\hat{x}_k = x_{k-1} + \overline{y}_{k-1}\tau_k \ , \tag{1}$$

where $\hat{x}_k$ is the predicted time difference. The prediction equation contains only linear terms, and neglects terms such as oscillator frequency aging. Ignoring aging is valid if the calibration interval is a few thousand seconds or less; it may be inadequate for longer intervals.

If the difference between the predicted and measured time offsets is not too large (to be quantified below), we use this time difference to modify our estimate of the average oscillator frequency. The current frequency estimate, y_k, computed using the first-difference of the measured time offsets

$$y_k = \frac{x_k - x_{k-1}}{\tau_k} , \tag{2}$$

is combined with the previous estimate of the filtered frequency to form an updated estimate:

$$\overline{y}_k = \frac{\overline{y_{k-1}} + G\, y_k}{1+G} \ , \tag{3}$$

where G is computed from the statistical parameters of the server clock. This average frequency is the primary output of the model; it is used to schedule periodic time adjustments to the server clock. These adjustments are scheduled often enough that the clock is not allowed to drift off time by more than the measured uncertainty in the time offset measurements. Since this measured uncertainty is about 0.5 ms, the nominal interval between clock adjustments during the kth time interval is

$$T_k = \frac{0.0005}{\overline{y}_k} \ . \tag{4}$$

To simplify the algorithm, the adjustment interval is modified so that an even number of adjustments occurs during the calibration interval τ_0. The magnitude of the adjustment is recalculated to the nearest microsecond based on the modified adjustment interval.

The complete algorithm has two loops: an "outer" loop activated every τ_0 seconds to update the parameters of the model, and an "inner" loop activated every T_k seconds to adjust the server's clock. Additional time adjustments are scheduled if the time measurement x_k is large or if a leap second or similar time change is imminent. The purpose of the algorithm is to drive x_k to zero so that $\hat{x}_k$ is identically zero for all k. The consequences of this are discussed below.

The Stochastic Portion of the Clock Model - The ACTS calibrations measure the frequency of the server clock oscillator. The measurement certainty is limited by ACTS noise, by measurement noise, and by noise from the server clock oscillator.

After dialing ACTS, the time server measures the time offset between its local clock and UTC(NIST) using three consecutive time codes spaced 1 s apart. Three readings are used to detect outliers using majority voting. The server clock is read each time the ACTS on-time marker is received, and the server software interpolates between ticks of the server clock with a tight loop whose execution speed is measured (interpolation is not necessary if the hardware and software have sufficient resolution). The time dispersion of the server clock is negligible during this 3 s interval, so the measurement reveals the stability of the telephone delay and the processing latency in the operating system. A typical stability is 0.4 ms, and the average dispersion over the last six calibrations is used to test the current measurements (the exact number of calibrations used to compute the average is not critical). If the scatter (max – min) among the three measurements from the current calibration is less than three times the average variation, the average of the three measurements is used. If only two measurements meet the criteria, the outlier is rejected and the average of the two is accepted. If no two values agree then all measurements are rejected and another calibration begins after a short delay. In all cases, the measurement epoch is defined as the midpoint of the averaged data. This logic adapts to slow changes in the reciprocity of the telephone circuits or in the operating system latency, and always rejects sudden time changes as an error.

Noise from the ACTS time offset measurements can be characterized as white phase noise, which means that the measurement uncertainty could be reduced by taking and averaging more samples. However, in practice ACTS should be called only often enough so that ACTS noise is not the dominant source of uncertainty. To determine how often ACTS should be called, note that the algorithm estimates the frequency of the local oscillator using the first difference of the phase measurements (equation 2). Since fluctuations in the two phase measurements are not correlated, the uncertainty in the frequency estimate is $\sqrt{2}$ times the measurement uncertainty, divided by the time interval between the measurements. If we know that the frequency offset of the server clock is 1×10^{-5} (a time offset of about 1 s per day), and if we want to limit ACTS noise to 10 % of the overall uncertainty, then the ACTS calibration interval should be about 1000 s. The actual calibration interval is determined by the actual performance of the server's clock.

The dominant limit to the uncertainty is stochastic fluctuations in the frequency of the server clock's oscillator. These fluctuations are often characterized in terms of how their power spectral density varies with frequency, or, equivalently, how their Allan deviation [9] varies with averaging time. Figure 2 shows the square root of the Allan deviation as a function of averaging time for a typical server clock oscillator, and the three lines drawn through the points identify three domains in which different noise processes dominate the spectrum.

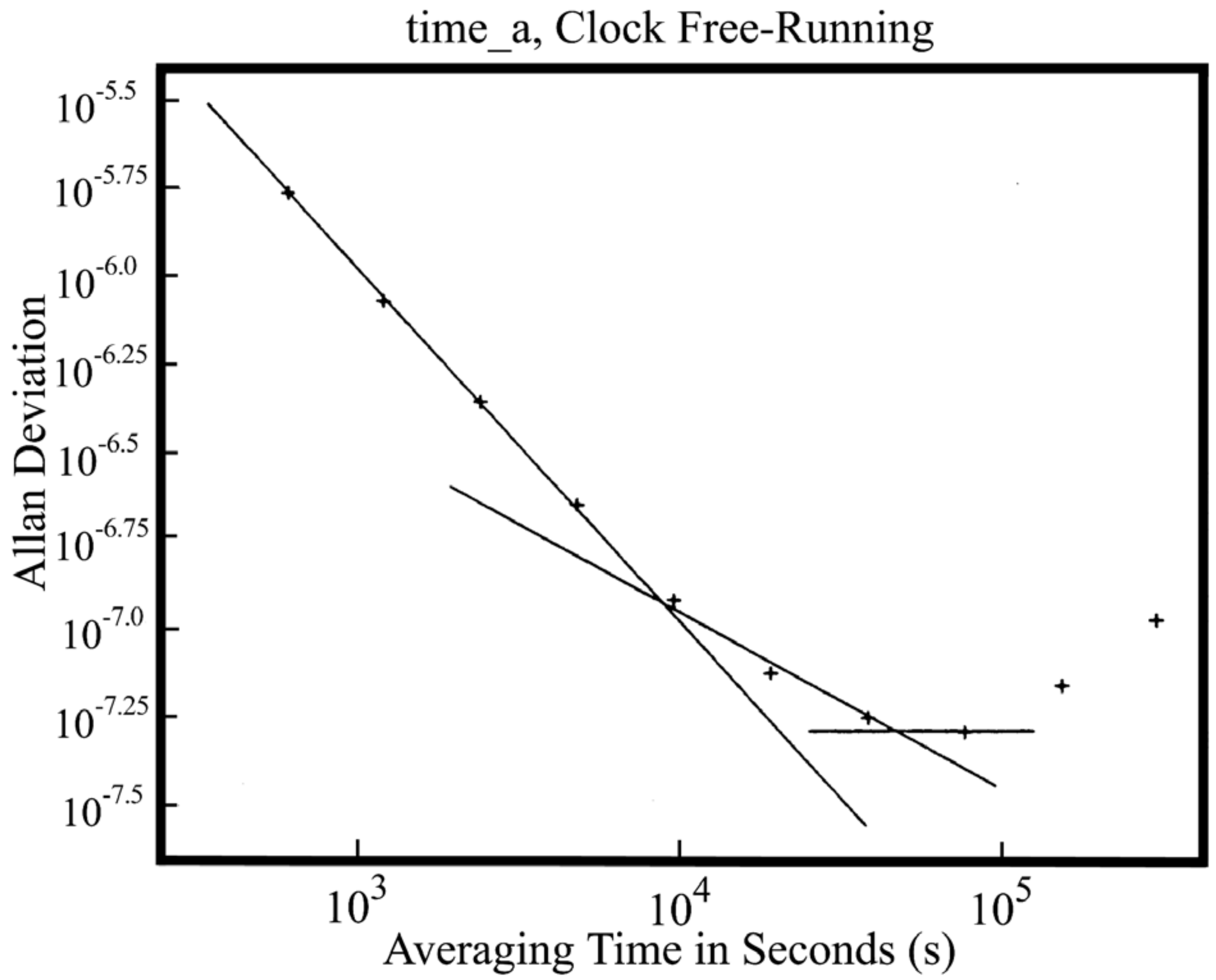


Figure 2. The two-sample Allan deviation for the free running server clock on time_a. The reference lines drawn through the points have slopes of –1, –0.5, and 0, showing the domains where the fluctuations in the oscillator can respectively be characterized as white phase noise, white frequency noise, and flicker frequency noise.

The white phase noise in the measurement tends to be most important at the shortest averaging periods. Oscillator frequency fluctuations become more important at longer averaging periods. Oscillator frequency fluctuations are initially independent of Fourier frequency (white frequency noise), but the power spectral density eventually starts to increase as the Fourier frequency decreases, or, equivalently, as the averaging time is made longer. The dependence on Fourier frequency gradually changes to 1/f (flicker frequency noise) and then to $1/f^2$ (random-walk frequency noise). As discussed in reference [9], these characterizations are important because there is an optimum averaging strategy for reducing each type of noise.

White frequency fluctuations are "best" from a prediction point of view, because an underlying mean frequency exists, and the fractional fluctuations about this mean can be reduced with more averaging. The prediction error increases only as the square root of the calibration interval. Shortening the calibration interval improves the predictions until the white phase noise floor is reached, or until another noise source (such as interrupt latency) becomes dominant. Lengthening the calibration interval makes the predictions worse, but only by a factor of the square root of the calibration interval.

This favorable dependence of the prediction error on the square root of the calibration interval does not hold true at longer periods. Long calibration intervals allow non-white noise processes in the oscillator frequency to become important, and more averaging will no longer improve the uncertainty. The prediction error will begin to grow linearly with the calibration interval, and will eventually grow faster than linearly. If the frequency changes were secular or deterministic, they might be modeled by adding an aging parameter to the prediction equation. However, this is generally not the case. In practice, the best that can be done is to compute an average frequency that gradually "forgets" older data with a time constant that roughly equals the time at which the frequency fluctuations deviate from a white spectrum. This is the purpose of the parameter G in equation (3). If T_{nw} is the calibration interval when the frequency fluctuations begin to deviate from a white spectrum, then

$$G \approx \frac{\tau_0}{T_{nw}} \; . \qquad \textbf{(5)}$$

The value of T_{nw} is determined by inspecting the Allan deviation of the free running clock oscillator and looking for the averaging period when the slope begins to increase from –0.5 (white frequency noise) towards 0 (flicker frequency noise). A typical value for T_{nw} is 12000 s; combining this result with the previous discussion limits τ_0 to a range of about 1000 s to 12000 s for optimum performance. The performance will not improve at shorter calibration intervals because of white phase noise from the ACTS measurement. It will begin to degrade at longer intervals because of non-white noise processes (such as aging) in the oscillator frequency. The corresponding values of G range from about 0.08 to 1.

For a typical server clock, using τ_0 = 3000 s and G ≈ 0.25 provides a good balance between performance and telephone cost, and results in an uncertainty of about 1 ms (1σ). Using a smaller value for τ_0 reduces the uncertainty in principle, but perhaps not in practice, due to interrupt latency and similar issues discussed below.

Clock Resolution and System Latency – Most computers keep time by counting "ticks" or interrupts that are periodically generated by a quartz crystal oscillator. The interval between ticks is generally measured in milliseconds. Some systems have a higher clock frequency and time resolution, and some operating systems make this high resolution available to a user task. If high resolution clock hardware is not available, it is possible to interpolate between ticks using calibrated software loops, but this method is usually limited by fluctuations in the system load. Our experience is that it is possible to interpolate reliably to only about 5 % of the tick interval. Therefore, using software routines to increase the time resolution adds to the overall uncertainty.

Latency, or delays introduced by the operating system and hardware, also contribute to the uncertainty. Signals received through any input channel are delayed by the time needed to transmit the information between the input port and the higher level process. This affects the uncertainty of the ACTS calibrations since the ACTS signals are received through a serial port, and the serial port driver is particularly complex. Some systems split the serial port driver into two parts; an “outer” part that is interrupt driven and that copies characters from the input hardware to a memory buffer, and an “inner” part that processes the received characters and places them in the user buffer. This inner part often runs as a scheduled task rather than as an interrupt driven task. Since the scheduler is usually activated as the result of a timer interrupt, there can be substantial jitter between the time a character is received by the hardware and the time it is available to the algorithm. This problem is particularly serious if x_k is small, since then the ACTS on-time marker will arrive almost simultaneously with a clock interrupt and small fluctuations in the arrival time may produce jitter of one full tick in the arrival times as measured by the algorithm. In some implementations, the inner portion of the driver runs only every few ticks and processes characters in bunches (in the name of efficiency), and this makes the jitter in the arrival time even worse. Normally, the jitter due to latency as an amplitude of a few milliseconds or less and can be characterized as white phase noise. The algorithm was designed to reduce the latency problem as much as possible, but there is no perfect solution.

Implementation of the Algorithm: Details and Examples - The underlying deterministic model of the algorithm does not change when the computer time is being adjusted, but the details are different. We consider three important effects:

- Since the time of the local clock is continuously adjusted in accordance with equation (4), the periodic measurements with respect to ACTS do not see a free-running clock, but rather one whose apparent frequency during the calibration interval has changed by $\overline{y}_k$.

- Since the goal of the algorithm is to drive x_k to zero, $\widehat{X}_k = 0$ for all k; to the extent that the algorithm is successful, it is also true that $x_k = x_{k-1} = 0$ as well.

- If x_{k-1} was large, then a time step was sensed during the previous calibration cycle, and this time step was removed by a special time adjustment. The current prediction therefore should not use the measured value of x_{k-1} but a value of 0 instead.

As a result of these considerations, the original algorithm was modified to include time steering as well as frequency steering. In addition to the periodic adjustments at intervals of T_k seconds, there is an adjustment of magnitude x_k after each calibration. The periodic adjustments correct the average frequency of the server clock, and the discrete time adjustments remove the short-term fluctuations in the frequency and keep the clock on time. As a result, $x_{k-1} = 0$ and equation (1) becomes

$$\overline{y_{k-1}}\,\tau_k = 0 \ , \qquad \textbf{(6)}$$

and equation (2) is replaced by

$$y_k = \overline{y_{k-1}} + \frac{x_k}{\tau_k} \ . \qquad \textbf{(7)}$$

Equations (3) and (4) are not altered.

The *lockclock* algorithm was tested on the time server named time_a. The first step was to evaluate the free-running performance of the server clock oscillator. Consecutive time differences spaced 1 s apart exhibited a scatter of 0.5 ms peak-to-peak. The time offset of time_a with respect to ACTS was measured every 600 s, and the Allan deviation of the collected data are shown in Figure 2. Reference lines with slopes of −1, −0.5, and 0 are drawn through the points, and the transition zones between different noise types are clearly visible.

The plot in Figure 2 can help determine the calibration interval that produces the lowest time uncertainty, as well as the calibration interval the produces the lowest operating cost. The last point in Figure 2 shows that the frequency stability is about 1×10^{-7} at a measurement interval of 307 200 s. If we used that interval as τ_0, the frequency uncertainty in the oscillator would result in a time uncertainty of about 3 ms, or about 30 times worse than the 0.1 ms performance we have achieved. However, the telephone charges have been reduced by a factor of 100 (from one call every 3000 s to one every 300 000 s).

After running the algorithm on time_a for about 110 days, the x_k values for this period have a mean of 0.1 ms and a standard deviation of 0.16 ms. An Allan deviation plot is shown in Figure 3. For comparison, a reference line with a slope of −1 has been drawn on the plot, and the measurements exhibit this slope over the entire range. This suggests that the residuals of the locking algorithm are pure white phase noise, which is the optimum performance that can be realized with the existing hardware and measurement method. The frequency noise that normally dominates the performance of the free-running oscillator at longer periods has been totally removed. A Fourier transform of the x_k values shows that the data are not quite white, and that there is a weak peak near 1 cycle/day with an amplitude of 0.1 ms. This is probably the response of the oscillator to temperature (and possibly to input voltage) changes, and running the system in an environment with better temperature stability (the temperature in the computer room varies by ±2° C) would probably improve the server clock performance [7].

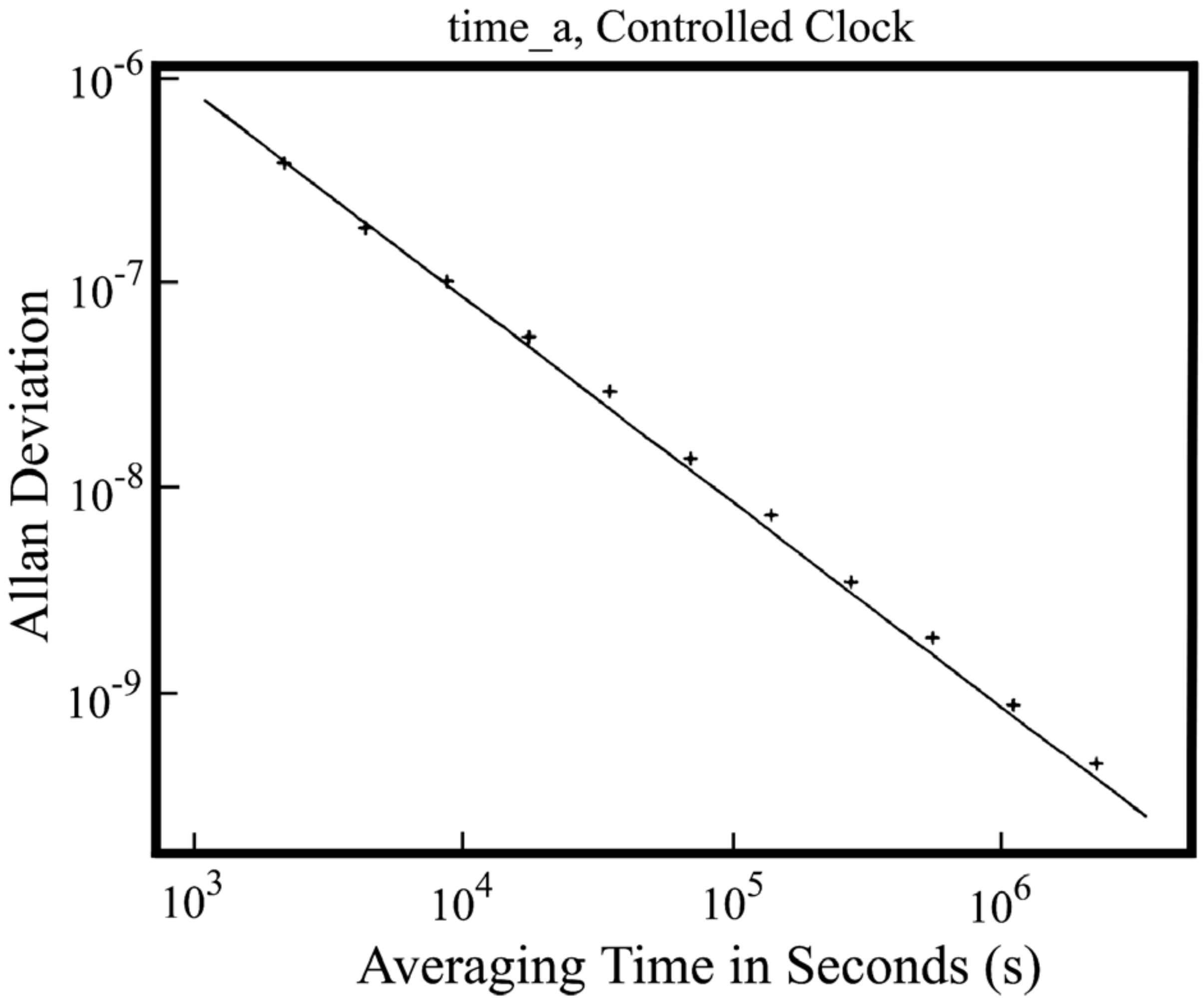


Figure 3. The Allan deviation for the time_a server clock when controlled by the lockclock algorithm. The reference line drawn through the points has a slope of –1, which is characteristic of white phase noise.

2. Interlock Algorithm

Like *lockclock*, the *interlock* algorithm is based on a statistical model of the clock and the network, and uses this model to define the parameters of a FLL that is used to discipline the server clock oscillator. However, *interlock* is able to perform calibrations using time codes transmitted over the Internet, a noisier calibration channel than the telephone lines used by *lockclock* to call ACTS. This makes *interlock* less expensive to use, because the network infrastructure is often already installed and available and can be used for time synchronization without the cost of a telephone call. Even so, *interlock* can provide comparable performance by performing more calibrations. While the performance of *interlock* is acceptable for a time server, it is also well suited for inclusion in client software.

Interlock uses messages transmitted in Network Time Protocol (NTP) format [3, 4] and can be used in an environment with other NTP synchronized systems. It does not depend on the details of the NTP format, however, and could be modified to work with other time code formats. *Interlock* characterizes the server clock in terms of its time difference and frequency offset with respect to the external reference; both parameters are then used to adjust the server clock to agree with UTC(NIST).

The Clock Model – Interlock characterizes a clock with two deterministic parameters: a time offset, which specifies the difference between its time at some epoch and UTC(NIST), and a frequency offset, which specifies how this time offset evolves. If x_k and y_k specify the time offset and frequency offset at some epoch t_k, then these parameters can be used to predict the time at the next epoch, t_{k+1} by using

$$\hat{x}_{k+1} = x_k + y_k \Delta t, \qquad \textbf{(8)}$$

where

$$\Delta t = t_{k+1} - t_k. \qquad \textbf{(9)}$$

Quartz crystal oscillators (such as the server clock oscillator) have significant frequency aging, which means their frequency changes nearly linearly with time. However, adding aging to equation (8):

$$\hat{x}_{k+1} = x_k + y_k \Delta t + 0.5 d_k (\Delta t)^2, \qquad \textbf{(10)}$$

where

$$y_{k+1} = y_k + d_k \Delta t, \qquad \textbf{(11)}$$

often does not improve the uncertainty of the model in predicting x_{k+1}. The problem is that the aging parameter itself must be estimated from the measured values of *x*, and this estimate usually is very uncertain because of large stochastic frequency fluctuations that mask the aging. For this reason, the algorithm estimates deterministic frequency aging and stochastic frequency modulation together, and treats the combination as producing a stochastic variation in *y*.

The Measurement Process - The measurement process involves comparing the server clock time with the time transmitted over the network. The network delay enters directly into this measurement of time offset. It is usually estimated from the measurements themselves using the usual round-trip method. The client (which could also be a time server) requests a time code at time t_1; the request arrives at the reference server at time t_2; the server responds at time t_3, and the client receives the response at time t_4. The round-trip delay due to the network path is

$$\Delta = (t_4 - t_1) - (t_3 - t_2). \qquad \textbf{(12)}$$

The first bracket contains the elapsed time (measured by the client's clock) from when the request was sent until the reply was received. The second bracket is the server processing delay (measured by the server's clock). If the one-way outbound delay is d, the time offset between the client and server is

$$x = (t_1 + d) - t_2. \tag{13}$$

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

$$x = \frac{(t_1 - t_2) + (t_4 - t_3)}{2}. \tag{14}$$

If the path delay is not exactly symmetrical, this estimate will be wrong by an amount proportional to the asymmetry. If the actual outbound delay is given by $d = k\Delta,$ where $0 < k < 1,$ then the estimate above is wrong by

$$\varepsilon = (k - 0.5)\Delta, \tag{15}$$

which depends both on the path delay and on its asymmetry. Paths with short delays are clearly advantageous.

Networks are usually configured so that the inbound and outbound paths are symmetrical, although there is no physical reason why this *must* be true. There is no way to detect asymmetry using only timing information transmitted over the network itself, but analyzing a group of time offset measurements can usually help determine whether asymmetry exists. The algorithm assumes that the network paths are symmetrical, but performs a series of tests that reject measurements if asymmetry is found.

Increasing the number of calibrations can reduce the frequency noise of the server clock. Eventually, we will reach a noise floor of ε; or the noise of the measurement process itself. We are not using the server clock at all when we reach this limit; the price for removing the time dispersion due to its frequency noise is to remove the frequency stability of the server clock as well.

Tests and Results – Interlock was used to synchronize the clock of a workstation named *strain,* located at NIST in Boulder, Colorado. A time server located about 1200 km away in Seattle, Washington served as the reference. The one-way delay is about 48 ms and is quite variable, since a number of routers exist along the network path between Boulder and Seattle.

The synchronization algorithm used groups of 50 time codes separated by 3000 s. The time differences in each group were averaged and the outliers were removed. These averages were used to compute the average frequency offset of the *strain* clock oscillator, and the frequency offset was used to periodically adjust the *strain* clock time. The time adjustments were performed by changing the value added to the clock register on each interrupt (i.e., the effective frequency of the oscillator) for a set number of times until the time adjustment had been amortized. The magnitude of these frequency adjustments was limited to less than 1 % of the clock frequency to provide very small time adjustments, and to ensure that the clock did not stop or run backwards.

Figure 4 shows the prediction error, εx as a function of time when the *interlock* algorithm is running. This error is $\varepsilon_x(k) = |(x_k - \hat{x}_k)|$, i.e., the difference between the average measured time-difference at some epoch and the value predicted using equation (8). These data have a mean of 0.92 ms.

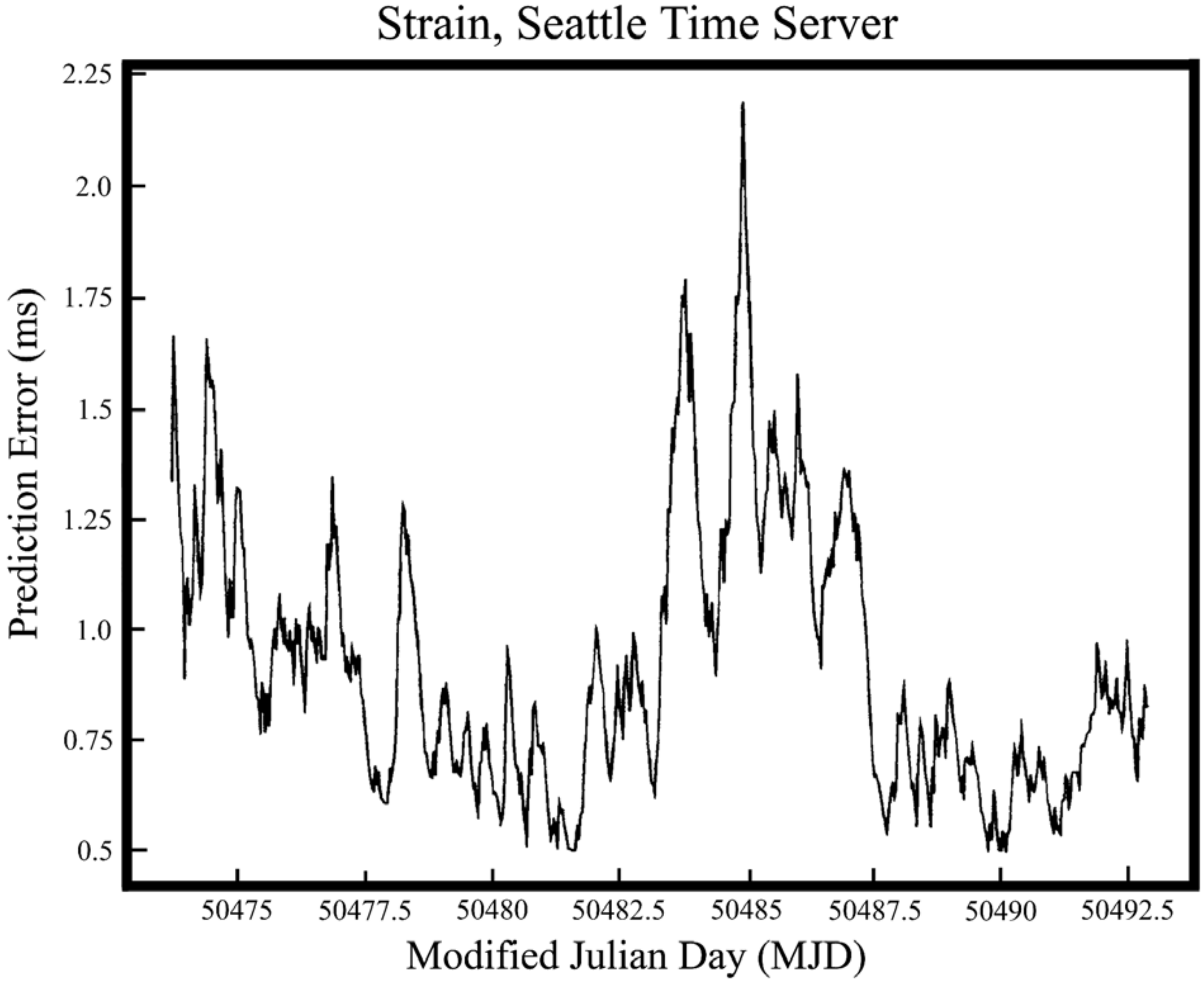


Figure 4. The uncertainty of the prediction (milliseconds) when the *strain* clock is synchronized to the server in Seattle. The long-period structure may be due to slow changes in the ambient temperature of the laboratory or to long-period fluctuations in the network delay.

Figure 5 shows the time of the *strain* clock with respect to ACTS, while *strain* is being synchronized to the Seattle time server. Both the noise and the accuracy of the ACTS comparison are about 1 ms, so that differences on this order are not statistically significant. The timing uncertainty is about 2 ms (1σ) for all averaging times, made appreciably worse by the large 10 ms spikes. In addition, there is some indication in both Figures 4 and 5 of a long period fluctuation. This is probably due to a combination of long period fluctuations in the network asymmetry (perhaps having a 7 day period) and the effect of temperature changes on the *strain* server clock.

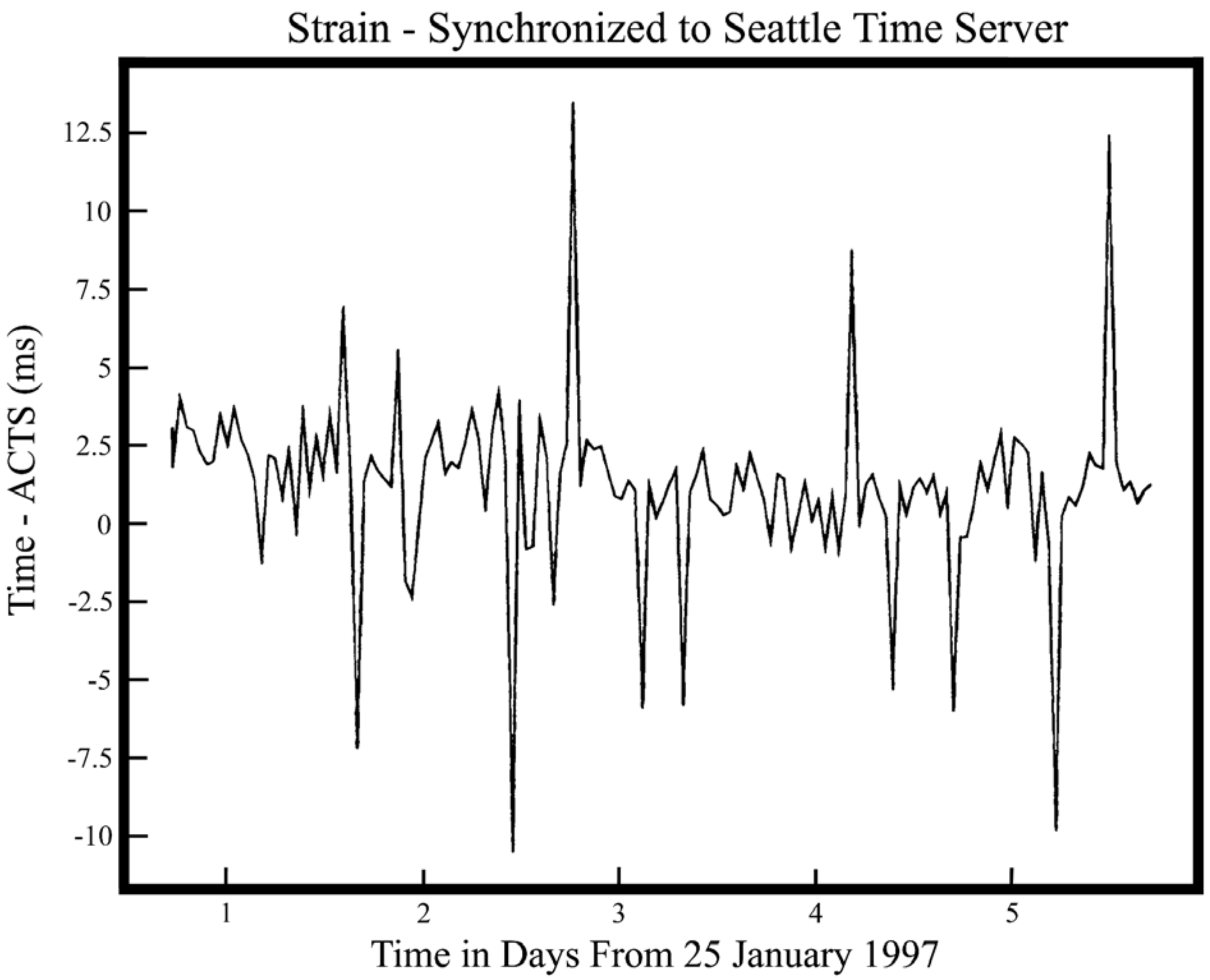


Figure 5. The time offset of the *strain* clock with respect to UTC as measured from the ACTS time signals. The ACTS comparisons are characterized by white phase noise of about 1 ms, so that fluctuations of that order or smaller are not significant.

If more uncertainty is acceptable, *interlock* works fine with long calibration intervals. A 3.5 d calibration interval, for example, results in a timing uncertainty of about 40 ms (1σ). The maximum synchronization error would almost always be less than 120 ms (3σ), which is probably adequate for most applications. One downside of these infrequent calibrations, however, is that problems will not be quickly detected. Using a 3.5 d calibration interval, for example, means that a glitch in the local clock will remain undetected (on average) for 1.75 d. Even so, it is clear that relaxing the uncertainty requirements can save bandwidth and reduce the number of public time servers that must be supported [7].

3. Autolock Algorithm

The *autolock* algorithm uses an adaptive FLL that configures itself to realize any specified performance level at minimum cost (measured in computer cycles or network bandwidth). The algorithm conserves network bandwidth and improves on previous algorithms by doing a better job of separating network noise from clock noise. In addition, *autolock* improves upon the *interlock* algorithm because it is self-configuring, and can adapt itself to fluctuations in the asymmetry of the network delay.

Autolock was developed in response to two observations made at NIST. The first observation was that ITS usage (section I.A.) has continued to grow at a rapid rate, and this growth has shown no signs of decreasing. The second observation is that most ITS users do not need millisecond level uncertainty. *Autolock* was designed to provide the best of both worlds. If the uncertainty requirements are reduced, it can reduce the bandwidth and processing time required to support existing clients, and therefore increase the capacity and reduce the cost of the ITS. At the same time, the underlying capability of the original *lockclock* algorithm is not compromised, so clients who do need millisecond level uncertainty can continue to receive it.

Details of the method - The *autolock* algorithm design is based on the principle of separation of variance, or the idea that it is possible to separate the clock noise from the measurement noise using statistical techniques. A second implicit assumption is that the variances of both the measurement noise and clock noise can be modeled with stochastic parameters. These models turn out to be good estimators of the performance of real hardware under typical conditions, although the second assumption tends to break down at averaging times approaching one day (see below). In order to achieve this separation, the program evaluates the following two statistics after each calibration has been completed:

- S-1. The standard deviation of a group of closely spaced time difference measurements that are made fast enough so that the time offset of the server clock has not changed significantly while they are being made. This requirement is easily satisfied if the measurements are completed within a few seconds.

- S-2. The error in predicting the currently observed average time difference from the previously measured value and the estimated frequency offset. This prediction is done using a simple linear relationship:

$$\hat{x}_i = x_{i-1} + \hat{y}_{i-1}\tau, \qquad \textbf{(16)}$$

where x_i is the time offset measured at time t_i, y_i is the frequency offset between the local clock and the server clock estimated at the same time, and τ is the time interval from t_{i-1} to t_i. We use equation (16) to predict the time difference; the error in the prediction on this cycle is the difference between the predicted and measured values:

$$\varepsilon_i = \left|x_i - \hat{x}_i\right|. \qquad \textbf{(17)}$$

Both statistics are maintained in two versions: the value determined in the current calibration cycle, and a sliding average over the most recent 12 hours or 3 calibration cycles, whichever is longer.

The first statistic (S-1) is sensitive to phase noise in the measurement process. The software first compares the average value of S-1 with the desired performance specified by the operator when the software was installed. The number of measurements in each group is adjusted once per day until the two are roughly equal; the size of the group is increased if the uncertainty is larger than requested, and decreased if the uncertainty is already better than requested. Assuming that the measurement noise is white phase noise, the uncertainty of a set of measurements varies as the square root of the size of the set. The specified performance level therefore has a quadratic effect on the cost of the algorithm. It might not be possible to reach the desired uncertainty with a reasonably sized data set (less than 25 or 50 samples). If an unrealistic number of samples are required, the software defaults to a specified maximum sample size. Likewise, the software never decreases the number of samples below a defined minimum size (usually three) no matter how small the uncertainty becomes.

The clock synchronization accuracy is limited by the uncertainty of the group of time offset measurements. No matter how stable the server clock frequency is, it cannot be synchronized more accurately than this measurement uncertainty. This uncertainty value may be as small as 75 to 100 μs when the server and the client are on the same network, but is more typically on the order of milliseconds for a continental length path. This limit is independent of the details of the synchronization procedure.

If the standard deviation of the current set of measurements is much larger than the average, the most likely cause is jitter in the symmetry of the network delay. Although the measurements are made fast enough so that the stability of the client and server clocks have not changed, the same cannot be said of the network. The magnitude of the network delay is not a problem, since it is measured on each cycle; only its asymmetry causes trouble. The algorithm assumes that asymmetries in the network delay are transient effects. Small asymmetries are removed by averaging. Large asymmetries are short lived and detected as outliers.

Under normal operating conditions, the second statistic (S-2) is sensitive primarily to the stability of the server clock, and its average value is proportional to the value of the Allan deviation for an averaging time equal to the calibration interval. The magnitude of S-2 is affected by two parameters: the calibration interval and the time constant of the frequency update loop [equation (3)].

The time constant of the frequency update loop represents the optimum averaging time for the frequency estimates, assuming that the process can be analyzed statistically. Averaging for less time does not optimally reduce the white frequency noise of the oscillator. Averaging for a longer time does not improve the result, since the remaining noise types are nonstationary (flicker and random walk). This time constant used depends upon the oscillator used, and needs to be determined only when the algorithm is started for the first time. It is measured using the Allan deviation when the local clock is free running. Typical time constants range from 10 000 s to 20 000 s, so the initial evaluation process usually takes less than 1 day.

Once the algorithm is running, the average value of S-2 and the desired uncertainty are used to adjust the calibration interval. If the average prediction error is much larger than the measurement noise (i.e., S-2 >> S-1), then the prediction error is due to stochastic frequency fluctuations in the local oscillator. If the prediction error is larger than the desired uncertainty, then the calibration interval is too long, which means that the oscillator is not stable enough to free run for that interval and needs to be calibrated more often. If the prediction error is smaller than the desired uncertainty, the calibration interval is acceptable.

If the prediction error is much smaller than the measurement noise, then the calibration interval is too small. In this case, the oscillator is stable enough to achieve the same uncertainty with fewer calibrations. Therefore, the calibration interval can be lengthened.

Measurement Results - The *autolock* algorithm was tested with two synchronization experiments. The first experiment was designed to synchronize the local clock as accurately as possible and to find out what uncertainty could be achieved and how expensive it would be to realize it. The second experiment was designed to relax the uncertainty specification to 1 s while using the same servers and network path as the first experiment. The goal was to see how much this relaxation of the uncertainty requirements would save in operating costs.

Both experiments synchronized a computer clock located at NIST in Boulder, Colorado, to a server on the West Coast in Redmond, Washington, and a server on the East Coast in Gaithersburg, Maryland. The time of the local server's clock was also measured using other systems in Boulder, but these measurements were for diagnostic purposes only and were not used to discipline the clock. The server in Redmond was the primary server; the one in Gaithersburg was used only for error evaluation.

The details of the paths between our client in Boulder and the two servers were not under our control. Although the path to Gaithersburg, Maryland had a nearly symmetrical delay, the path from Boulder to Redmond was often 42 ms longer than the return path (because the outbound path was routed through Atlanta, Georgia). When this asymmetry existed, the two servers seemed to have a time offset of 21 ms, an offset larger than the time dispersion due to all other causes for averaging times up to several hours. Static asymmetries of this type sometimes cannot be removed by any statistical procedure. Our solution was to make Gaithersburg the primary server when the delay from Redmond became asymmetric, and to use a third server (located in Reston, Virginia), if both Redmond and Gaithersburg had asymmetric delays.

Figure 6 shows the measured time difference between the computer clock in Boulder and UTC(NIST), and Figure 7 shows the calibration intervals for the same period. Note that the calibration interval is gradually increased during the first week of the experiment, without significantly changing the uncertainty of the computer clock. This situation changes abruptly when the calibration interval reaches 32 000 s, an interval comparable to the length of the working day, because both the local temperature and the network delay have deterministic fluctuations near this period. One of these fluctuations eventually exceeded the threshold, and the algorithm was forced to reduce the calibration interval to maintain the desired uncertainty. The same story is repeated on a smaller scale several more times until about day 28, a day that had both a large change in temperature and a failure of a network element when the same kind of problem was repeated. The standard deviation of the data set is about 30 ms with the two large phase shifts included, but only 10 ms with the phase shifts removed.

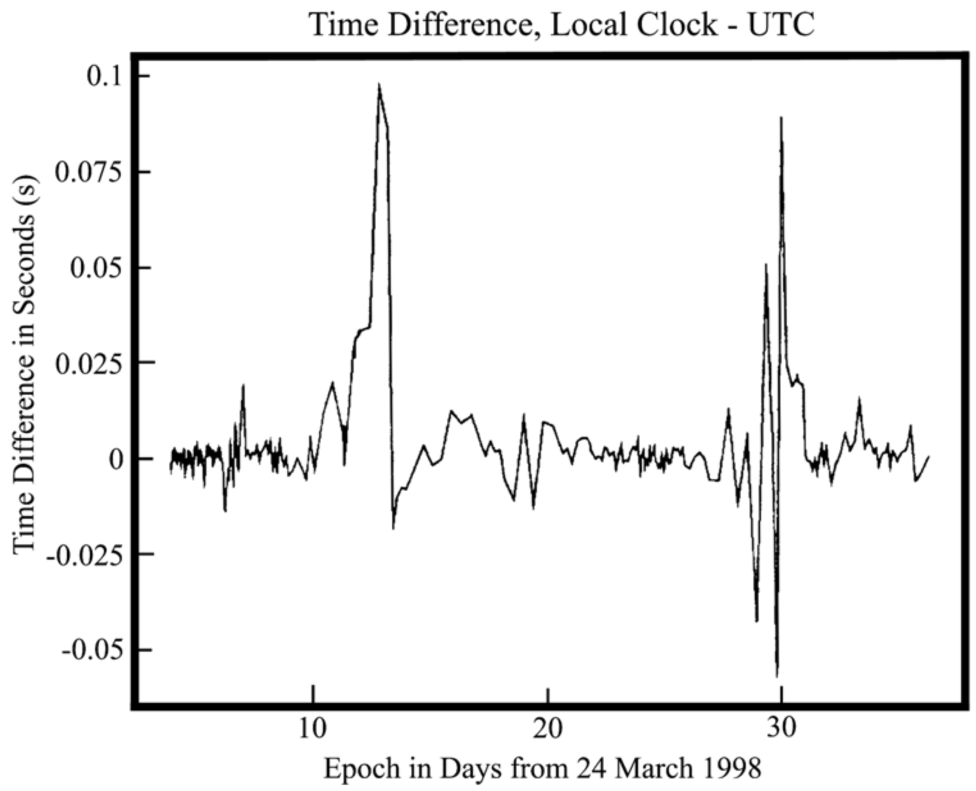


Figure 6. The time difference in seconds between the local clock and UTC(NIST).

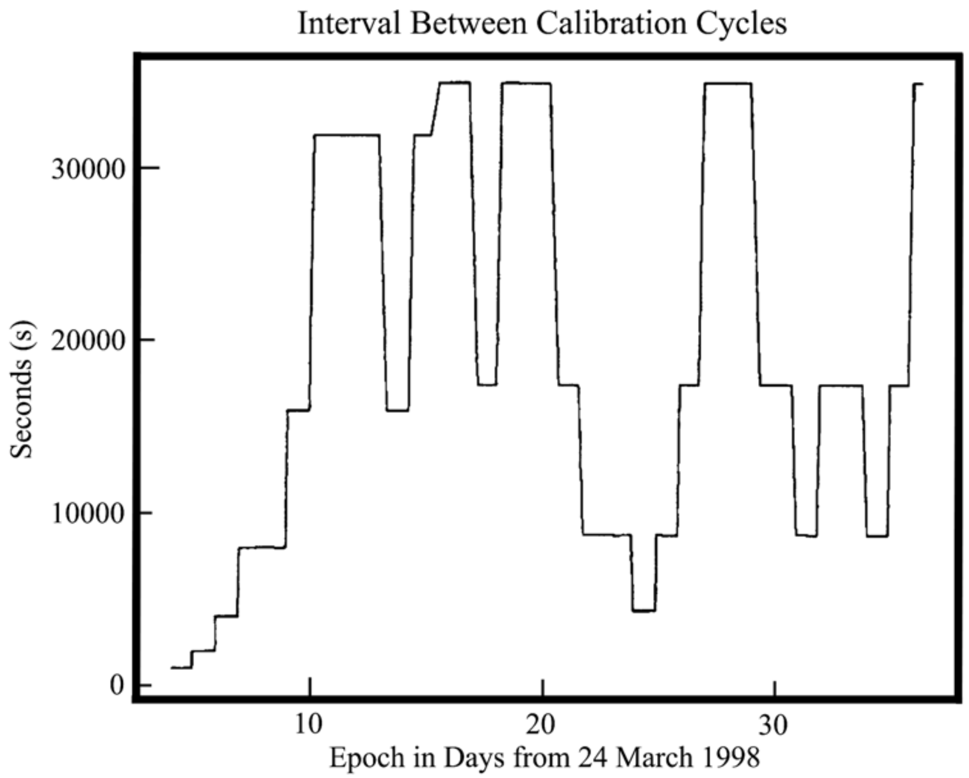


Figure 7. The calibration interval in seconds as a function of epoch for the first experiment.

Figure 8 shows the average frequency offset of the clock as estimated by the algorithm, measured in parts in 10^6. Diurnal effects are visible on occasion, but they are smaller than the random walk frequency noise, which dominates the noise spectrum at this period. These frequency fluctuations have an amplitude of about 3×10^{-6} peak-to-peak with an irregular period of about 1 week. This data can be used to configure a synchronization loop designed to keep the clock on the machine accurate only to the nearest second.

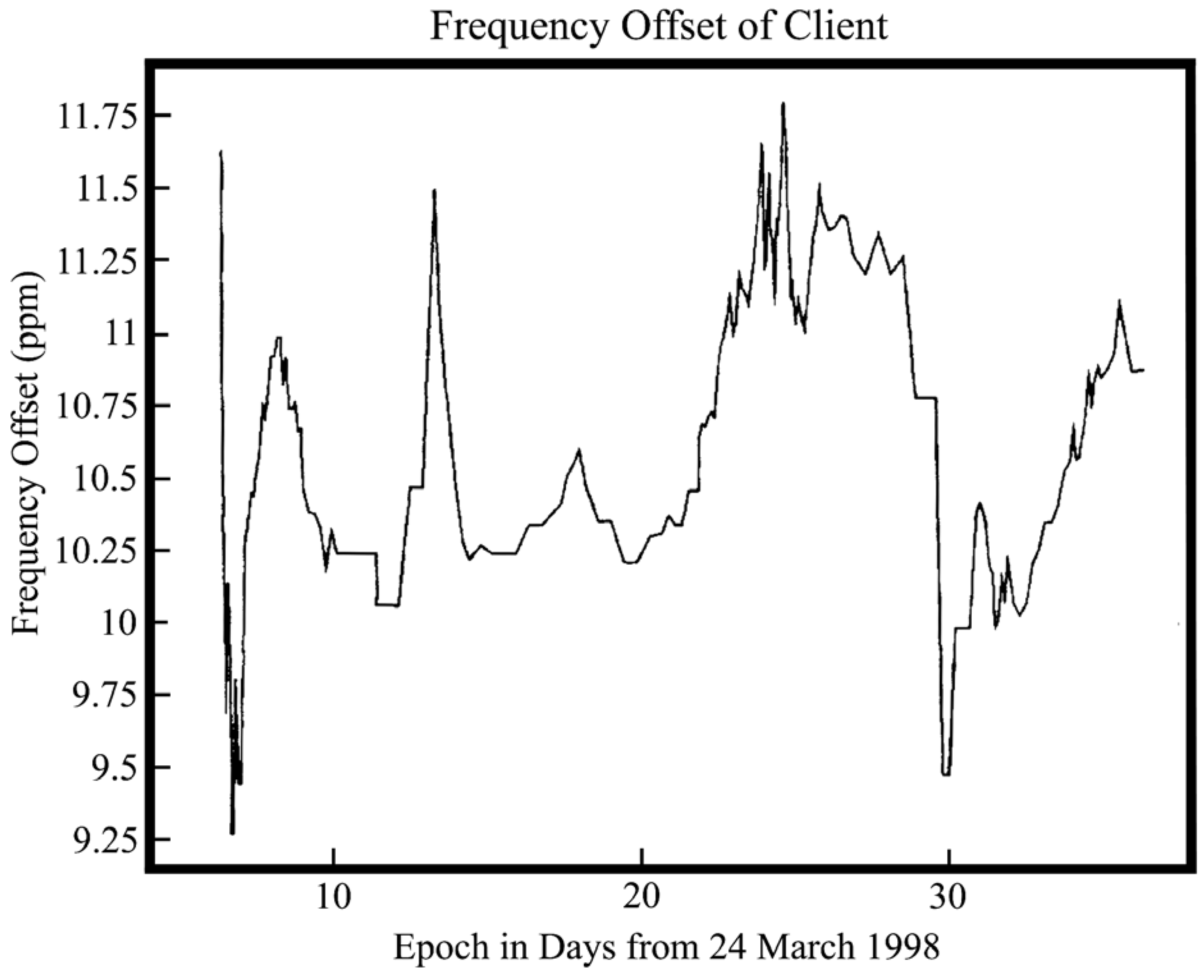


Figure 8. The frequency offset of the local clock as estimated by the autolock algorithm.

If a static frequency offset of about 10.5×10^{-6} was applied to the clock oscillator of this computer, and if the clock was allowed to free run with no other updates, then the time uncertainty over the one-month period shown in Figure 8 would still be < 1 s (1σ). Obviously, it would not be wise to allow the clock to free run for this long, since there is no way to detect a glitch in its operation. However, the result emphasizes that keeping a clock within 1 s of UTC(NIST) requires very little external information. Although this particular clock would gain almost 1 s/day if it were run with no correction at all, simple applying a static frequency offset allows us to predict its performance as far as one month into the future.

The second experiment configured the *autolock* algorithm for an uncertainty of 1 s, using the same servers as in the previous experiment. The calibration interval was set to 32 000 s, and specified a maximum calibration interval of 200 000 s. The time constant for the frequency update loop was left at 12 000 s; since the minimum interval was larger than this value, the effective time constant was three sample times, as discussed above.

It took just over 10 days for the calibration interval to reach the maximum specified value (200 000 s). The time differences during this period are shown in Figure 9. Note that the uncertainty increases with longer calibration intervals, but is still well below 1 s even when the calibration interval is 200 000 s. Since neither random walk noise nor temperature noise are modeled by the algorithm, the long-term performance of the clock is difficult to predict, but 1 s was easily held for a period of more than 2 weeks.

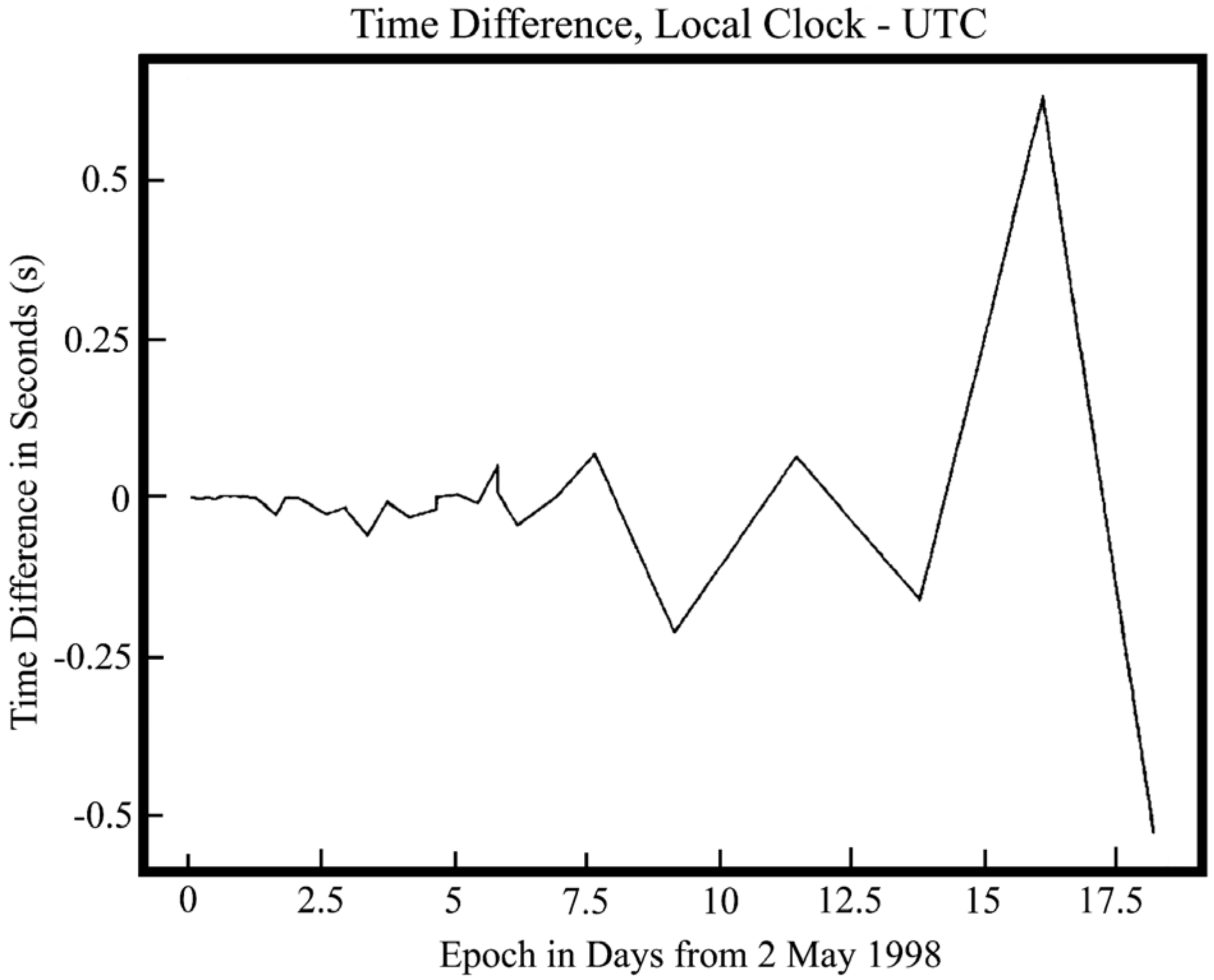


Figure 9. The time difference in seconds between the computer clock and UTC(NIST) when the algorithm is configured to realize an uncertainty of 1 s.

3. Technical Description of File Transfer and Diagnostic Software

All ITS time servers support standard anonymous file transfer protocol (FTP) read only access. This means that every time server is also an anonymous FTP server that distributes copies of the client software and documentation. All other network-based access is completely blocked or is enabled and governed by the security policy described in Section III.F.

The servers also include reporting and diagnostic software that monitor the status of the servers and send reports and alarms to NIST personnel when necessary. This software is described in the following section.

a) Reporting and diagnostic software

The reporting and diagnostic software was developed at NIST and uses both network and telephone interfaces to report status information and failures. The alarm system uses the telephone interface to call a numeric pager carried by NIST personnel; the network interface provides the more detailed weekly reports and related information.

The software monitors and records the server usage, and analyzes the performance of the synchronization daemons. It contains no special capabilities or time critical design features. Each time server is monitored both internally and externally as follows:

- *Internal Monitoring* - Each server monitors its own performance using a series of internal checks performed by the various daemons and by distinct watch dog processes that are activated periodically. These checks include tests of the network interface, the telephone line, and the hardware. A failure of these checks results in a pager alarm using the telephone system and a notification via email.

- *External Monitoring* - All time servers are monitored by an external process that runs on the NIST network in Boulder. This external process checks the time of each server, evaluates a snapshot of its performance received via the finger daemon, and so on. If a server fails any test, an alarm is sent to a radio pager.

Weekly reports that provide a snapshot of the performance of each server are sent to Boulder on Saturday morning. These reports include all activity for the previous seven days. In addition, each server maintains a more detailed copy of its internal log files. Two classes of log files are used: those maintained by the standard *syslog* utility of the kernel, and those unique to the daemon processes, whose log files are maintained by the daemons themselves. All files are write-locked against modification by non-daemon processes during system operation.

4. Technical Description of Client Software

NIST distributes sample client software for the ITS as freeware. This software can be downloaded from any ITS time server using the FTP protocol and an anonymous log-in. The software is available for several computer platforms, and in most cases, the source code and documentation are also available. For current downloading information, please see the ITS web page at:

http://tf.nist.gov/service/its.htm

There are many commercially available ITS time clients. Although NIST is not allowed to recommend, endorse, or review commercial products, it does maintain a list of all known software publishers. New publishers are added upon request. This list can be found at:

http://tf.nist.gov/general/softwarelist.htm

All client software supports one or more of the three time code formats supported by the ITS servers (Daytime, Time, and NTP). A client designed to support the NIST daytime format will work only with a NIST server. NTP and time protocol clients will work with any public server that supports those formats. If you have a generic NTP client, for example, you can use it to access a NIST server simply by typing in one of the IP addresses listed in Table 1.

NTP clients are divided into two types. A "true" NTP client runs continuously as a background task and periodically gets updates from one or more servers. It ignores responses from servers that appear to be sending the wrong time, and averages the results from those that appear to be correct. While doing so, it continuously steers the clock of the local machine to keep it on time. The NTP clock model is similar in some respects to the *interlock* and *autolock* algorithms described in Section II.B.2, but uses a phase locked loop (PLL) as opposed to the frequency locked loop (FLL) employed by the NIST algorithms. Many available NTP clients do no averaging or clock steering. These clients make a single timing request to a single server (just like a Daytime or Time client) and use the resulting time code to synchronize their clock. They are properly named SNTP (Simple Network Time Protocol) clients.

All ITS client software assumes it is connected to the Internet, either directly to a local-area network or indirectly via a dial-up connection. If you use a dial-up connection, you may have to complete this connection before using the client software, although some network interfaces will automatically start a dial-in sequence when a network request is posted. Many Digital Subscriber Line (DSL) and cable modems are configured to act as local area networks and are effectively connected to the Internet at all times.

C. Time Code Formats

The time code formats supported by the NIST Internet Time Service are specified by standard Internet timing protocols, as described below.

1. Daytime Protocol (RFC-867)

The RFC document does not specify an exact format for the Daytime time code, but does require that the time code is sent using standard ASCII characters. This means that the code will appear as standard text on a computer screen. The NIST version of the Daytime protocol is very similar to the time code transmitted by ACTS (Chapter 2), and is listed in Table 3. The server listens for daytime requests on port 13, and responds in either TCP/IP or User Datagram Protocol/Internet Protocol (UDP/IP) format.

Table 3. The NIST Daytime Time Code Format.

JJJJJ YR-MO-DA HH:MM:SS TT L H msADV UTC(NIST) OTM
where:
JJJJJ is the Modified Julian Date (MJD). The MJD is the last five digits of the Julian Date, which is simply a count of the number of days since January 1, 4713 B.C. To get the Julian Date, add 2.4 million to the MJD.
YR-MO-DA is the date. It shows the last two digits of the year, the month, and the current day of month.
HH:MM:SS is the time in hours, minutes, and seconds. The time is always sent as Coordinated Universal Time (UTC). An offset needs to be applied to UTC to obtain local time. For example, Mountain Time in the U.S. is 7 hours behind UTC during Standard Time, and 6 hours behind UTC during Daylight Saving Time.
TT is a two-digit code (00 to 99) that indicates whether the United States is on Standard Time (ST) or Daylight Saving Time (DST). It also indicates when ST or DST is approaching. This code is set to 00 when ST is in effect, or to 50 when DST is in effect. During the month in which the time change actually occurs, this number will decrement every day until the change occurs. For example, during the month of October, the U.S. changes from DST to ST. On October 1, the number will change from 50 to the actual number of days until the time change. It will decrement by 1 every day until the change occurs at 2 a.m. local time when the value is 1. Likewise, the spring change is at 2 a.m. local time when the value reaches 51.
L is a one-digit code that indicates whether a leap second will be added or subtracted at midnight on the last day of the current month. If the code is 0, no leap second will occur this month. If the code is 1, a positive leap second will be added at the end of the month. This means that the last minute of the month will contain 61 seconds instead of 60. If the code is 2, a second will be deleted on the last day of the month. Leap seconds occur at a rate of about one per year. They are used to correct for irregularity in the Earth's rotation. The correction is made just before midnight UTC (not local time).
H is a health digit that indicates the health of the server. If H=0, the server is healthly. If H=1, then the server is operating properly but its time may be in error by up to 5 s. This state should change to fully healthy within 10 minutes. If H=2, then the server is operating properly but its time is known to be wrong by more than 5 s. If H=3, then a hardware or software failure has occurred and the amount of the time error is unknown.
MsADV displays the number of milliseconds that NIST advances the time code to partially compensate for network delays.
The label UTC(NIST) is contained in every time code. It indicates that you are receiving Coordinated Universal Time (UTC) from the National Institute of Standards and Technology (NIST).
OTM (on-time marker) is an asterisk (*). The time values sent by the time code refer to the arrival time of the OTM. In other words, if the time code says it is 12:45:45, this means it is 12:45:45 when the OTM arrives.

2. Time Protocol (RFC-866, RFC-868)

This simple protocol returns a 32-bit unformatted binary number that represents the time in UTC seconds since January 1, 1900. The server listens for Time Protocol requests on port 37, and responds in either TCP/IP or UDP/IP formats. Conversion to local time (if necessary) is the responsibility of the client software. The 32-bit binary format can represent times over a span of about 136 years with a resolution of 1 s. There is no provision for increasing the resolution or increasing the range of years.

The strength of the Time protocol time code is its simplicity. Since many computers keep time internally as the number of seconds since January 1, 1970 (or another date), converting the received time to the necessary format is often a simple matter of binary arithmetic. However, the format does not allow any additional information to be transmitted, such as advance notification of leap seconds or daylight savings time, or information about the health of the server.

3. Network Time Protocol (RFC-1305)

The Network Time Protocol (NTP) is the most complex and sophisticated of the time protocols, and the one that provides the best performance. NTP client software is typically used by large computers and workstations, and is often included with the operating system. The client software runs continuously as a background task that periodically gets updates from the server. The client software can be configured to query several servers, and to average the results. By querying multiple servers, it can ignore responses from servers that appear to be sending the wrong time.

The NIST servers listen for a NTP request on port 123, and respond by sending a UDP/IP data packet in the NTP format. The data packet includes a 64-bit time code containing the time in UTC seconds since January 1, 1900, with a resolution of 200 picoseconds.

D. Authentication of Time Signals

The time signals are passively authenticated based on the IP address used to connect to an Internet time server. A number of commercial services provide (or are planning to provide) additional authentication to the messages transmitted from the NIST servers based on encryption and digital signatures; these authenticated services will be overlays to the basic messages provided by NIST.

III. Operational Procedures of Internet Time Service

This section discusses the hardware and software maintenance required by ITS. It also discusses the ITS failure modes and security issues.

A. Hardware Maintenance

The servers are maintained by a maintenance contract with the manufacturer that provides for on-site service and includes all parts and labor. The cost of the maintenance contract is added to the purchase price. The terms and conditions of the maintenance agreement call for a four-hour response time. These warranty agreements typically expire after three years. When the three-year period ends, the service contract is either renewed or a replacement server with a new three-year maintenance agreement is installed.

B. Software Maintenance

The software maintenance is simplified by the fact that the time servers are essentially identical (except for local configuration parameters, which are specified in a number of small files that are unique to each system). A backup copy of the disk image of each system is saved on tape. This tape is recorded just before the system is shipped and after its final local configuration parameters have been specified. These tapes are largely redundant, of course, since most of the information they contain is the same on all servers. The tapes are stored in a fireproof safe, together with the log notebooks, which specify the detailed hardware configuration of each system, its software revision level, and so on. The location and combination of this safe are known by at least three senior members of the Time and Frequency Division.

Software upgrades are usually handled over the network. The revised software is first tested on private servers that are not part of the operational network. The private servers can be configured to look like operational systems, so there is a high degree of confidence that the software upgrades will work without problems when they become part of the network.

Notifications for upcoming leap seconds have to be manually entered. A single entry propagates this information to all ITS servers. This is the only required time code entry, since the operating system was modified to automatically compute the transition dates from ST to DST and from DST to ST. This computation is made using the current United States rules, which call for the ST to DST transition to occur on the first Sunday in April, and for the DST to ST transition to occur on the last Sunday in October.

C. Failure Modes

ITS failures can be divided into two groups: client side failures and server side failures, as described below. The service was designed to have no single point of failure.

1. Client Side Failures

Although server failures are rare, network failures are much more common. All of the client software developed at NIST (and much of the client software developed by others) will automatically query another time server if the first server cannot be reached. Since all of the

NIST time servers support the same services, the client software simply has to try another IP address after a query fails. The operational details vary from one client program to another; the NTP daemon, for example, will do this automatically and without special notice to the user. The NIST AUTOLOCK daemon switches to the alternate server automatically, but “remembers” the primary server and switches back to it when it becomes available. Other programs inform the user of the problem and give a menu of suggested alternatives in order of priority. The bottom line is that the failure of a single server, or even of multiple servers at a single site, is not a cause for much concern. In fact, network outages that have nothing to do with the ITS cause these sorts of problems much more often than we would like.

2. Server Side Failures

Server side failures can be divided into three groups: total hardware failures, partial hardware failures, and failures due to software or external signals.

a) Total hardware failure

The server simply disappears from the network in this case, and the disappearance is reported by the external monitor (II.B.3.a). However, a disappearing server is ambiguous, because it can also be caused by transient network problems. Isolating such problems can be quite difficult, and often requires consultation with the on-site staff at the remote location. When a total hardware failure does occur, it is usually caused by the power supply or display adapter. Both of these are covered by the service agreement and are usually replaced by Compaq the same day.

b) Partial hardware failure

If the hardware partially fails, the server is still visible on the network, but is reporting a bad status problem either through the finger daemon, through the NTP status bits, or by other equivalent status parameters. These situations are rather rare. The two problems that we have seen have been telephone related. The first type was caused by a telephone line failure, and the second was due to a problem with the ACTS servers in Boulder.

Telephone company problems have usually been solved within 24 hours. The server is up and running during this period, but its time is not correct, since it cannot communicate with the ACTS system.

There have been only two failures that were caused by ACTS: one during the first few hours of the Y2K rollover (January 2000), and the second in August 1999, when the ACTS system was overloaded by a configuration problem in certain third-party client programs.

Generally speaking, the server continues to respond to requests for time during a partial hardware failure with a message that includes a “bad status” bit. Most (but not all) client programs check this status bit and switch to another server when it is set. This check is not possible for the Time Protocol, since that format provides no status information.

We have experimented with a number of different solutions to the proper response to a “time” format request that is received by a server that is not healthy. Although the initial thought was simply to not respond at all, many client programs treat “no response” as a time in the year 2037. This produced about as many angry letters as transmitting the wrong time. The proper solution for a user is to use either the daytime or NTP formats, but the designers of some proprietary network operating systems have been unwilling to do that. The same question about how to respond when the server is not healthy also exists for the more robust formats, but the choice there is clearer. For various technical reasons, not responding in this case and depending on the resulting time-out does not always result in the client doing the "right thing," and sending a "server unhealthy" flag results in more robust and predictable behavior.

c) Other failure modes

We have never had a failure that was due to a general software problem, although one site did have a problem because its external reference oscillator failed and the failure was not diagnosed properly by the software. This problem has been addressed, but there are probably more things like this just waiting to happen. However, we have been at this for many years and they have not happened yet.

D. Record Keeping

The automated diagnostic and monitoring software described earlier produces files that are maintained and stored at NIST for record keeping purposes. All other information pertaining to the service, including documentation of software changes, system maintenance, outages, and so forth, is archived at NIST for a period of at least several years.

E. Security Issues

This section discusses the security of the network time servers and their vulnerability to misuse or attack.

1. Physical Security

The time servers are located in rooms that have keypad-activated locks so that entry to these locations can be monitored and controlled. In addition, the server software has been modified to protect the servers against both premeditated attacks and mistakes in the configuration of client systems.

2. Security Against External Command Entry or Log-ins

The ITS servers do not accept external commands that can modify the state of the server. The time daemons neither accept nor parse any input messages, beyond the requirements specified in RFC-1305 for time requests in NTP server/client mode. Although the NTP daemon itself supports external management and control, all of these functions are disabled in its configuration file. The other time daemons neither expect nor accept external input and simply ignore anything that arrives.

The server software is designed so that it is not necessary to log-in to the server under normal circumstances. Furthermore, there are no general-purpose user accounts on these systems. Therefore, logins are relatively rare events; that simplifies the security considerations.

3. Limitations on FTP Access

The time servers block inbound FTP access to all user names except for the username "anonymous," which provides access only to the publicly available software. This block is independent of the username/password system; user-level FTP access is blocked even to a legitimate user who knows the correct password. All publicly available directories are marked read-only.

4. Limitations on All Other Access (including telnet, rlogin, and rsh)

Access through *telnet* and other types of log-ins is limited to a particular set of IP addresses specified at NIST. Security is implemented as follows:

a) Log-In attempts from non-registered IP addresses

TCP wrappers are used to limit access to a particular set of IP addresses. Users who connect from a non-registered IP address receive a dummy login prompt that looks authentic, but that simply records whatever is entered by the user and exits when a carriage return is typed. This process will *never* result in a valid login (even if a legitimate username and password are entered), and is used to provide some information on the details of the attacker by capturing whatever is typed. No matter what the user types, an innocuous response (username/password not recognized) is sent as the reply. A response to the "login prompt" that is not null and that is deemed at least "semi-serious" by the control software is archived in the system log and is treated as a significant event by the report daemons.

b) Log-In attempts from registered IP addresses

Users who connect from an approved IP address receive a standard login prompt but are required to enter a special one-time password in addition to the normal password for that user name. Failure to enter the correct one-time password (on the *first* attempt) triggers an intrusion alarm (which sends a message to the numeric radio pager) and disables the terminal. Although the password is sent in clear text in this mode, it is a one-time password, and capturing it using a network monitor or "sniffer" is not worth much.

c) Secure shell connections

The servers support secure shell connections in which the entire session is encrypted using the IDEA algorithm (or something equivalent) with a unique session key. This method is the one that is usually used to connect to the servers to perform maintenance or a software upgrade.

In all modes, system-level users (such as usernames "root" or "service") cannot log-in over the network even with the correct password. Attempts to do so are trapped and logged by the server. System-level users must login as a normal user and then use the equivalent of "sudo" to perform maintenance. Since maintenance is normally performed through a secure shell connection, the system passwords are sent over the network as encrypted messages.

5. Alarm System

The reporting system described earlier (II.B.3.a) reports the status of the various security systems. The alarm system is implemented as a distinct process so that the reporting daemons can be modified without requiring changes to the lower-level processes that handle the details of the notification process.

6. Attacks and Responses

We have never had a successful attack against the wrappers and one-time password systems described above (II.G.4). There was one successful attack about five years ago against the server at Microsoft. This attack used a network "sniffer" to capture a user password, and our response was to install the one-time password system that we currently have in place. Our logging system has recorded lots of attempts, some of which are more humorous than dangerous (someone trying to login as root with passwords such as "its-me" or "pretty-please" or the often-tried usernames "tick" and "tock"). As discussed above, such attacks are blocked by the TCP wrappers, which also store the text in the system log files. They would fail in any case, since network access by root is blocked even with the correct password.

We have had a number of attacks that might be attempts at denial-of-service but might also be more innocent configuration errors in some versions of the NTP client software. These attacks consisted of sending thousands of time requests in a short period of time to a single server. Although it is not friendly or statistically useful, it is possible to configure a client program to do this without malicious intent. The security administrator at NIST reported a few attacks, and other attacks were reported by the time servers.

Although attacks that attempt to saturate a server by sending it thousands of requests for time are something of a nuisance, it is hard to cause real trouble using them. The reason is that a time packet is so small (typically 64 bytes or less) and the servers are so fast, that a typical network gateway cannot send requests fast enough to cause a problem for the server. Compare this to the time.gov web sites (Chapter 3), where a *single* time request for service requires a response consisting of many tens of thousands of bytes. In practice, denial-of-service attacks have tended to cause the gateway system to fail well before the time server is saturated.

Nevertheless, denial-of-service attacks represent a serious potential problem. The fact that it is the gateway that fails first rather than the server is not of much comfort to the user, who still cannot get access to the server. Having our servers distributed at various sites is helpful, since an attack against a single site has no effect on the other sites.

IV. Customers

This section describes the number of ITS users, the service capacity, and the ways that NIST interacts with and supports the users.

A. Estimated Number of Users

As illustrated in Figure 1, the ITS was receiving well over 300 million timing requests per day as of November 2001. There is no feasible way of identifying how many of these requests originate from distinct users. The time servers simply count the number of users; they do not store the IP address of each client, because doing so would result in a large administrative burden of log file management, and would reduce both the speed and capacity of the servers. It is clear that the number of timing requests far exceeds the number of unique users, since nearly all of the recent growth is accountable to increased usage of the NTP protocol, and NTP clients usually make multiple timing requests in the same day, sometimes three per hour. Even so, it is safe to say that the number of unique users now numbers in the millions, and probably in the tens of millions.

B. Capacity of Service

Each Internet time server can handle about 1,000 requests per second, so that the capacity of the ensemble of 14 servers is about 1.2×10^9 requests per day. The current load (November 2001) is about 3×10^8 requests per day, so that overall system has significant capacity to support future growth in demand. However, the usage is not uniformly distributed among the servers, and some are nearing saturation. Simply adding new servers does not always help, since much of the existing client software will not be aware of the new server's presence. We are currently planning to install load balancers at some locations. This will make it possible to use multiple servers on the same IP address.

C. Customer Interaction

We receive an average of about 6 to 10 comments per day from users (about 2000 to 2500 per year). Almost all of these are received by email either to jlevine@boulder.nist.gov or time@time.nist.gov (which both resolve to the same end-point location). A few email requests are sent to other offices at NIST and are forwarded to the addresses above, and a much smaller number (a few per month) are received by other means (mostly telephone calls). The vast majority of inquiries (about 85 %) are requests for more information about the services, requests for general time and frequency information (for example, how leap seconds are defined) and similar matters. A smaller number (perhaps 10 %) are reports of problems or difficulties in using the services. The most commonly reported problem with the ITS is caused by a firewall at the user's location that blocks the communication between the client program and the servers. Most remaining inquiries come from inexperienced users who are unfamiliar with their computers or operating systems, and who ask basic computer questions. About one request per month is a suggestion for an enhancement to the client programs or is a report of a possible problem with the system itself.

Nearly all requests are answered within 24 hours and the question or problem is generally resolved immediately. In some cases, when the initial message is ambiguous or incomplete, an email conversation is necessary. Most issues are resolved in two or three email messages.

Reports of serious problems are handled immediately, but these are rare. Suggestions for new features or modifications are considered and are implemented if possible. The client software distributed by NIST is generally upgraded about two or three times per year in response to these suggestions.

Most upgrades to the client software are in response to changes in the operating system or the operating environment of the client. For example, recent upgrades have added support for Windows 2000 and Me clients operating behind a proxy server, for firewalls that block TCP/IP port 13 (the standard Daytime port), and similar matters.

One recent problem fix concerned dealing with a problem in the daylight saving time tables in the Windows system for locations in the Southern Hemisphere (mostly users in New Zealand). Some recent enhancements include better error checking for errors or inconsistencies in the local time-zone configuration and the capability to run the client program in the background, where it does not require any space on the screen display or the Windows taskbar. These changes were implemented following customer requests.

D. Other Forms of Customer Support

NIST utilizes both its web site and FTP sites to help support the ITS, as described in this section.

1. Web Page

The ITS page is the most visited page on the NIST Time and Frequency Division web site. This page is used to distribute software, a current list of time servers, and to alert customers about any changes or outages to the service. It also provides instructions for using the service, and links to publishers of third-party software. The page can be found here:

http://tf.nist.gov/service/its.htm

2. FTP Files

All of the time servers support anonymous FTP access to documentation, client software source code and executable files, and similar information. Users can retrieve these files by connecting directly to any of the time servers or through a link on the ITS web page described in the previous section. The web page link points to a particular time server, but all time servers have identical copies of the material.

V. Measurement Uncertainties

This section discusses the measurement uncertainties of the signals transmitted by the ITS servers, and of the signals received by ITS clients. It also discusses establishing traceability to UTC(NIST).

A. Uncertainties of Transmitted Signals

The Internet time servers located at the NIST laboratories in Boulder, Colorado directly connect to the national time and frequency standard, UTC(NIST). The resolution of the receiving hardware is 1 μs and the measurement is characterized as white phase noise with a magnitude of 10 μs for a single measurement.

The time servers located at other sites are synchronized with periodic calls to the ACTS system. The stability of the ACTS system is about 500 μs (1σ), although individual telephone connections may be noisier than this value on occasion. The modems that are used for these calls are individually calibrated, so that the uncertainty of the time transmitted by the servers is approximately 1 ms (1σ) with respect to UTC(NIST).

Keep in mind that the uncertainty of the transmitted signal is essentially meaningless to the user. The uncertainty of the received ITS signal (next section) is usually much larger since it is dominated by the variation in the network delay between the user and the server.

B. Uncertainties of Received Signals

The uncertainty of the received signals is limited by the jitter in the transmission delay between the server and the client. The uncertainty of the time received at a client site depends on the quality of the network connection between the client and the server.

In all time transfer systems, the systematic uncertainty introduced by path delay, in this case the time required for the signal to travel from the server to the client, must be accounted for. Since some client software can measure and account for path delay, the uncertainty is limited not by the path delay itself, but by its asymmetry. It is assumed that the one-way path delay equals one-half of the measured round-trip path delay. Any deviation of the delay from this symmetric situation introduces an offset into the reception time of the time signals. This offset is bounded by ±0.5 times the measured round-trip delay, so that minimizing the round-trip delay (by locating servers at many different sites, for example), helps reduce the uncertainty.

The time offset between the servers is continuously monitored and recorded by a special processor in Boulder. The deviation between the time difference of this special server and the time received from any of the operational servers is the sum of the network asymmetry and the time offset of the remote server. This total time offset varies with network traffic and other factors, but is typically about 5 ms (1 σ), with a spectrum that is approximated by white frequency noise (NOT white phase noise). Since the monitoring processor connects to the servers exactly as any client would, the 5 ms value is a good estimate of the uncertainty that a typical client would realize in using the service. However, this uncertainty might be significantly better or worse for clients whose network connections are particularly good or poor, respectively. For example, a client on the same local network as a server can come close to preserving the timing uncertainty of the servers themselves. On the other hand, a client in Japan contacting a server on the East Coast of the United States will receive messages with an uncertainty of about 25 ms (1σ). The actual worst case can be much, much worse, with uncertainties of more than 1 s possible if the network connection is extremely poor.

C. Establishing Traceability to UTC(NIST)

Users can establish traceability to UTC(NIST) through the ITS for time-of-day or time interval measurements. The ITS should not be used for the measurement of frequency, since the large amount of phase noise introduced by variations in network path delay make it nearly impossible to measure frequency precisely, even if very long averaging times are used. A sample traceability chain for time-of-day information is listed in Table 4. This example lists typical uncertainties. Keep in mind that while the uncertainty of link B is smaller for a time server directly connected to UTC(NIST) than for a server connected to ACTS, the uncertainty of time received by the client will still be the same. Also, as previously noted, the uncertainty of link D could be much larger than listed in the table (approaching 1 s in extreme cases) for users with a noisy network connection or a long network path.

Table 4. The ITS Traceability Chain.

Link	Reference	Compared To:	Uncertainty (2σ)
A	SI units	UTC(NIST)	<100 ns
B	UTC(NIST)	NIST Automated Computer Time Service (ACTS)	<200 μs
C	ACTS	NIST Internet Time Service (ITS)	2 ms
D	ITS	NTP Client Daytime or Time Client	10 ms <100 ms

Chapter 2. Automated Computer Time Service (ACTS)

The Automated Computer Time Service (ACTS) was the first computer time service offered by NIST, making its debut on March 9, 1988. Since that date, ACTS has distributed Coordinated Universal Time (UTC) kept at NIST, called UTC(NIST), to thousands of computers daily, using ordinary telephone lines and conventional analog modems. While the majority of users have migrated to the Internet Time Service (ITS), ACTS is still widely used to synchronize clocks in computers and other types of electronic equipment that are not connected to the Internet, but that do have access to a telephone line. In addition, ACTS fulfills the extremely important role of providing the time reference for the ITS. This chapter describes ACTS in detail. It provides a physical and technical description of the service, and explains how NIST provides and maintains ACTS for the American public.

I. Physical Description of ACTS

ACTS is based on standard rack-mounted servers running the DOS operating system. Each server supports from two to six modems, and is directly connected to the UTC(NIST) time scale via a coaxial cable. Additional servers continuously monitor the ACTS time code and send alarms to NIST personnel whenever an error is found. This section provides a physical description of the service.

A. Introduction and History

The original version of ACTS went on-line on March 9, 1988. The initial system was designed using custom computers built at NIST using the Z80 microprocessor. One computer was required for each phone line, and the initial system had 12 phone lines. Each line could handle a connection at either 300 or 1200 bits/s, which were the most commonly used speeds at that time. A shortened time code was sent to clients who connected at 300 bits/s, because the full time code could not be transmitted within 1 s at that speed.

The initial servers were complemented with a set of client programs that were designed to work on as many different computer configurations as possible. Most of the initial clients were MS-DOS computers, but UNIX workstations and the VAX/VMS operating system were supported with very similar client software. The client programs were distributed both as source code and as compiled and ready-to-run versions for a number of common configurations. [10,11]

The initial hardware was replaced in 1994 with a design based on standard PCs, and various features have since been added to both the servers and client software. The hardware improvements have included support for multi-speed modems. The software improvements have added new features and provided support for additional types of client systems. The current service supports 26 phone lines in Boulder, Colorado (24 lines on a public rotary and 2 on a private rotary) and 4 phone lines at NIST radio station WWVH in Hawaii (the Hawaii service originated in December 1997).

The basic service has not changed since its inception, and every change was designed to preserve compatibility with the previous versions. Although few PCs still run MS-DOS, the original client program is still distributed and supported for use on that platform and on most UNIX workstations.

The service receives about 10,000 telephone calls per day. Many of these requests come early in the morning, especially just before the start of the working day on the East coast, and all lines are busy for a few minutes each day.

B. How ACTS is Used by its Customers

As with the ITS, the identity of ACTS customers is revealed to NIST only when they contact us requesting information or assistance. Thus our description of the uses of ACTS is very likely incomplete. Nevertheless, we can identify some general classes of users:

- E-commerce and e-business users who synchronize the clocks of systems used to time stamp commercial and financial transactions. This class of users also includes commercial applications similar to the function of a Notary Public or to the Registered Mail and Return Receipt Requested services provided by the United States Postal Service. There are several commercially available clocks used by brokerage firms that were specifically designed to call ACTS.

- Special-purpose time stamp hardware, such as employee time clocks, parking ticket systems, and so on. Many mechanical clocks used to time stamp documents use either ITS or ACTS system to establish traceability to NIST.

- Computers not connected to the Internet. Older PCs that are not connected to the Internet but have an analog modem and access to an ordinary telephone still depend upon ACTS for time synchronization. We believe that some computer users with Internet access continue to use ACTS instead of ITS since they are long-time users of the ACTS service and are reluctant to change, or for some reason have not heard of ITS. However, as the availability of high-speed Internet access continues to spread, it is likely that most computers will no longer contain an analog (dial-up) modem, and will not be able to access ACTS. The digital modems used for high-speed Internet access (including digital subscriber line (DSL), cable, and wireless modems) are incompatible with ACTS.

- Computers behind an Internet firewall. Some ACTS users are located in organizations where they do have Internet access, but where a firewall prevents them from accessing the ITS time servers. The availability of ACTS allows these users to continue to synchronize their computer clocks.

C. ACTS Facilities

The ACTS servers are located in two separate rooms in the NIST Boulder laboratories. The rooms are located on different floors, but both rooms are equipped with direct signals from the UTC(NIST) time scale. The incoming telephone lines are divided roughly equally between the two rooms. Two rotaries, a public rotary containing 24 telephone lines, and a 2-line rotary reserved for calls made by ITS time servers, are located in Boulder. A single ACTS server residing on a 4-line rotary is located at NIST Radio Station WWVH in Kauai, Hawaii. Table 5 provides information about each of the three rotaries.

Table 5. The ACTS Telephone Rotaries.

Location of Rotary	Access	Phone Lines	Phone Number	Connection Speed	Capacity (phone calls per day)
NIST, Boulder	Public	24	(303) 494-4774	300 to 19,200 bits/s	60,000
NIST, Boulder	Private	2	Not Published	300 to 1200 bits/s	NA
NIST, Hawaii	Public	4	(808) 335-4721	300 to 19,200 bits/s	10,000

The rotaries all "hunt" for the first available line. A busy signal is received only if all lines are busy. Once a caller drops off of any line, that line immediately becomes available for the next caller. The connection speed is really irrelevant, since the time codes are sent only once per second and even the slowest modems can receive all of the transmitted information.

However, supporting speeds up to 19,200 bits/s was done to make it easier for the fastest modems (56K) to negotiate a connection.

The private rotary used to synchronize the ITS time servers is equipped with 1200 bit/s modems. These modems have more stable delays than higher speed modems since they do not rely on any error correction or data compression schemes. [10] Therefore, they are much better suited for time transfer. This allows the ITS time servers to synchronize to UTC(NIST) with an uncertainty of 1 ms (1σ) as discussed in Chapter 1.

One or more time servers is connected to each rotary. All the time servers are capable of servicing at least four phone lines, and some servers can service six phone lines. The time servers are located in environmentally controlled rooms or laboratories. At the NIST Boulder laboratories, backup power from uninterruptible power supplies (UPS) can run the hardware for about 20 minutes. The UPS devices are connected to a generator that starts automatically when the power fails. The generator is normally running within 60 seconds after a power failure, and can support the entire load indefinitely. The time server located in Hawaii is also connected to a local UPS and a backup generator.

1. Configuration of ACTS Time Servers

Each time server contains from four to six serial ports (RS-232) that control external modems configured to accept incoming calls. The modems are configured as inbound only devices; they do not make outgoing calls. The serial interface of the modem connected to line 1 on each server is monitored by an external computer that continously verifies the time code accuracy.

Each time server also has a direct connection to a 1 Hz signal from the NIST primary time and frequency standard. These 1 Hz pulses are used to phase lock the internal system clock to UTC(NIST). Each server has a keyboard, and can respond to commands issued by NIST personnel. The servers do not respond to commands issued remotely by modem, other than to identify themselves and provide the time code, or to provide a brief help message.

The ACTS servers do not write to disk, so the required disk capacity is just a few megabytes. The servers use solid state disks for reliability purposes.

D. Organizational Control of ACTS

ACTS is managed, controlled, and maintained by the Services Group of the NIST Time and Frequency Division. The Group Leader of the Services Group reports to the Division Chief of the Time and Frequency Division, who in turn reports to the Director of the NIST Physics Laboratory. The Physics Laboratory is one of the seven operational units of NIST, an agency of the United States Department of Commerce.

II. Technical Description of ACTS

This section summarizes the technical operation of ACTS, including a discussion of the hardware used, and the client and server side software. For a much more detailed discussion, please see the *Technical Reference Manual for the NIST Automated Computer Time Service*. [12]

A. Technical Description of Hardware

ACTS is built entirely from commercially available hardware that has not been modified by NIST. This section describes the specifications of the hardware.

1. Computer Systems

Most 486 or Pentium computers (33 MHz or faster) running the DOS operating system are capable of serving as ACTS servers. Both Microsoft MS-DOS 6.22 and Datalight ROMDOS 6.22 have been successfully tested. The computer requires any video card capable of displaying text, 4 MB of RAM, a 3.5 inch floppy drive, and a hard disk or electronic flash disk. Depending on the configuration, the computer might need as many as five ISA (Industry Standard Architecture) bus slots. At this writing (November 2001) it is usually necessary to use a passive backplane design with a CPU card when building new ACTS servers, since motherboards with more than one ISA bus slot are now difficult to find.

Each ACTS time server is periodically checked by another computer called the ACTS Monitor (ACTSM). One ACTSM can check up to four time servers. The ACTSM hardware consists of a PC (with either a 486 or Pentium processor), an ACTSM ISA bus card made at NIST, the monitor eavesdrop box, a 1200 bit/s (or faster) modem, and an optional Ethernet card for connection to the Internet. The ACTSM board contains the REF_CLK, the I/O port interface circuit and the circuit required to synchronize the REF_CLK to the external reference 1 pps. The REF_CLK requires a stable 5 MHz or 10 MHz reference signal from the UTC(NIST) time scale. The external reference 1 pps is required only while setting the REF_CLK.

2. Modems

ACTS uses standard analog external modems. Each modem is connected to its own serial port, and each serial port has a unique address and Interrupt Request Line (IRQ). A number of different modems ranging in speed from 1200 bits/s to 33,600 bits/s have been successfully tested and used with the ACTS servers. The server provides full support for call supervision, including detection of the RING, CARRIER DETECT, and DATA SET READY signals.

3. Internal Clock Board

Each server contains an internal clock board that it uses to connect to the UTC(NIST) time scale. The current version uses the GT401 Event Timing and Controller Board manufactured by Guide Technology, but equivalent hardware from other manufacturers could also be used. The Guide Technology board contains a free-running crystal oscillator with a typical, uncalibrated frequency offset of $\pm 5 \times 10^{-6}$ per day (0.5 s per day) and a stability of 2×10^{-7} per month. The clock board is used to perform several functions:

- This oscillator is used to measure the round-trip path delay during an ACTS phone call with a resolution of 0.1 ms.

- It keeps the server's operating system clock (DOS clock) fast with respect to UTC by about 0.77 s (14 ±1 DOS-clock ticks). The DOS clock is steered by ±1 tick as necessary to maintain this relationship.

- It determines the transmission time for the On-Time Marker (OTM). The software uses a fixed advance of 45 ms if the user does not echo the OTM character, and will adjust the advance using the actual measured delay if the user does echo this character. The advance is driven from the millisecond value of the clock board in both cases. As described below, the time code transmitted to a user is computed from the internal DOS clock and is therefore transmitted to the user starting about 0.8 s early. The OTM is transmitted when the clock board is set to 955 ms in fixed advance mode. The variable advance value is determined dynamically for each connection.

4. External Reference Oscillator

All ACTS servers are directly connected to the UTC(NIST) time scale via a 1 Hz signal. This connection is made using standard coaxial cable and a BNC connector with 50 Ω termination.

B. Technical Description of Server Software

The server software consists of the operating system, and the software developed at NIST that handles timing requests.

1. Operating System

The ACTS server software was written for the DOS operating system. Both Microsoft MS-DOS 6.22 and Datalight ROMDOS 6.22 have been successfully tested and used with the ACTS servers.

2. Time Server Software

The time server software was written in the C programming language without using any of the extended features of C++. The software is built in the small memory model, and therefore runs on a standard DOS machine with 640 kilobytes of memory. The text segment of the software is almost 64 kilobytes long, which is the limit for the small memory model.

The software consists of a main program named ACTS.C and a number of utility subroutines. Input, output, and control of the serial ports are managed by a library of subroutines that are built into the software during the linking process. These routines provide support for interrupt-driven I/O. The software also controls the DTR line using direct input and output commands to the appropriate control registers of each of the serial lines. The clock board has its own driver (supplied by the manufacturer), which is called by the time server software. This driver enables the time of the clock board to be locked to the integer second using an external 1 pps pulse from UTC(NIST).

The software is not designed to operate in a multi-tasking environment and may not work properly in the "DOS Window" of another operating system that does not allow software routines to directly access hardware. In particular, the software must be able to write directly to the VGA screen memory and to control the lines of the serial ports.

a) Software operating principles

The time server software controls up to six dial-in modems through standard serial ports. The software provides full modem control and line supervision and operates at any speed supported by the modems, including 300 bits/s.

The start-up code configures the serial ports and the modems. When the modems have been tested and the initialization of the other devices is complete, the software enters the steady-state operating mode. The steady-state operation of the server is driven by two parameters: the system clock and a state vector giving the status of each of the serial ports as an integer.

1. The System Clock

The clock oscillator inside an ACTS time server runs at a frequency of 1.19318 MHz. Dividing this clock frequency by 65 536 produces a new frequency used to generate an interrupt on every cycle. The interrupt frequency is thus 18.206 Hz (1 193 180 / 65 536). The processor increments a 32-bit register on each interrupt and resets the register to 0 at 00:00:00 every day. The register therefore keeps time in units of the tick period, which is 1/18.206 s or about 55 ms. Converting the internal time in ticks to hours, minutes, and seconds is the responsibility of the operating system. This conversion is complicated by the fact that there is not an integral number of ticks per second, so that most exact second times do not exist. The time 00:00:01, for example must be found by interpolation between the 18th tick after midnight, which is time 00:00:00.98, and the 19th tick after midnight, which is time 00:00:01.04. This is a serious issue for the ACTS system since it transmits the time in the usual way with an exact integer value for the seconds. If the system clock were used for the time reference without some form of correction or interpolation, the time at which a message was transmitted would vary by up to 1 tick (55 ms). As an additional complication, the DOS clock frequency is not accurate or stable, and the time of a typical PC may drift by several seconds per day or more, which is on the order of milliseconds per minute. This frequency aging has a deterministic component driven by the temperature, but there are also other factors that are not constant or predictable. Thus, although it would be simple to use the PC clock as the time reference, it is simply not up to the task.

These problems were addressed using a two-tiered system. The first tier uses the PC clock to define the integer second only. It is set roughly 0.77 s (14 ± 1 ticks) fast, and this relationship is maintained using an external reference as described below. The PC time is split into two quantities: the date and time to the integer second, and the fraction of a second expressed as a number of ticks ranging from 0 to 18 (or 19). The time to the integer second is used to construct the time code, and the tick value is used to drive the state of the system as described in more detail below. The second tier uses an external hardware pulse to specify the actual arrival of the second. This pulse is used to transmit the OTM and to keep the PC clock synchronized as described below.

2. The Steady-State Cycle

This cycle is implemented in the program *ACTS.C*. It is driven by the tick value described above. This is an integer giving the time in ticks since the last integer second and ranges from 0 to 18 or 19.

The steady state cycle starts as soon as possible after the 0th tick. The software calls a subroutine that generates the time message using the current system time. This subroutine in turn calls other subroutines to calculate the Modified Julian Day (MJD) number, the daylight saving time flag, and the information included in the time code. The time code is stored in a character array and displayed on screen.

The next portion of the code loops through all of the serial ports and responds to requests for time codes by users connected to the ports. This portion begins when the time code is ready and the housekeeping is done. The action depends on the state value for each port. The state values are integers and are maintained independently for each port. The action in the loop starts with the previous value of the state for that port and continues until a branch indicating "end of action" is reached (all state values start at 0). The operation continues with the next port, and when all ports have been tested the next phase of the loop begins. This loop over all serial ports is not executed again until the next second.

3. Verify Clock Phase

This portion of the software checks the local clock against other time sources. The DOS clock time is compared to the internal clock board, which is in turn synchronized to UTC(NIST) using an external 1 pps signal. This comparison has a dynamic range of 24 hours and a resolution of 1 ms. The purpose of this comparison is to keep the DOS clock fast by 14 ± 1 ticks. Since both the DOS clock and the internal clock board can be read with millisecond resolution, the difference can be computed at any point in the cycle, and the comparison is made at this point for convenience. If the difference between the DOS clock and the internal clock board does not equal 14 ± 1 ticks, the DOS clock is steered by ±1 tick in the appropriate direction. If the difference is equal to the expected value, then no action is taken. Note that both clocks can be read with negligible delay so that the time comparison (and any necessary adjustment) can be completed immediately.

The next step is to compare the internal clock board to the external 1 pps signal from the UTC(NIST) time scale. The 1 pps that synchronizes the Guide board clock is also connected to the "time-tag" input on the board. This input stores the internal board time whenever a pulse is received. Since these pulses are used to synchronize the clock, the time-tags should be exact integer seconds. Thus, the existence of the time-tags means that the 1 pps signal is present and the fact that all time tags are exact integral seconds means that the clock is synchronized to the external 1 pps. A message is displayed on the server screen if either tests fails. The time tags also contain more significant digits, including the minute, hour, and so on, but these are not used in this process; only the existence of the time tags and the fact that they are exact integral seconds is important.

4. Sending On-Time Markers (OTM) to Each Active Port

The next phase of the loop is sending the time codes and OTMs to each active port (a port with a caller currently connected). This is done using the internal clock board. The software initially advances the OTM by 45 ms, so the OTMS are transmitted when the millisecond field of the time read from the internal clock board is 955 ms. The software enters a tight loop waiting for this event starting at the 10th tick of the DOS clock. Since the DOS clock is nominally 14 ticks fast, the time on the internal clock board is actually about 750 ms in the previous second, so that this loop is executed for about 250 ms, which sets the maximum value for the OTM advance. The character used for an OTM with the standard 45 ms advance is an asterisk (*).

After an OTM is sent, the next step is to check whether the client software returns (or "echos") the OTM back to the ACTS server. If the OTM is returned, the server can measure the round-trip path delay between the server and the client, and adjust the OTM advance so that it equals the one-way path delay. There are five or more ticks of the DOS clock during which the software looks for echoed OTMs. This represents a time interval of about 275 ms, which is nearly the same duration as the maximum advance described earlier.

The routine that measures the round-trip path consists of a nested set of tight loops. The outermost loop begins after all the OTMs have been sent and runs for about 275 ms. If an OTM is received during this interval, the transmission delay is measured and the advance value is modified. To illustrate how this works, let T be the millisecond portion of the clock board time when the OTM was sent, and let R be the corresponding time at which it was received. The initial value of T is 955 ms, as discussed above. If $R > T$, then the previous advance was too large since the millisecond value has not yet wrapped around to the next second. The one-way transit time is $(R - T) / 2$, and the next transmission time should be set to be $1000 - (R - T) / 2$, so that the next OTM arrives just as the clock board reaches the even second.

If $R < T$, then the millisecond value of the clock board has wrapped around to the next second. The round-trip delay is $(1000 - T) + R$, so the one-way delay is $500 + (R - T) / 2$. The transmission time should be 1000 minus this value or $500 + (R - T) / 2$, so that the next OTM will arrive just as the clock board reaches the even second as above.

Once the transmitted OTM is actually advanced by the amount of the one-way path delay (and not by the standard advance of 45 ms), the OTM character is changed to a pound sign (#). However, there are some cases where the path delay measurements are rejected and the OTM advance remains at 45 ms (and the OTM character remains an "*") even if the client software is returning the OTM. For example, OTM advances of less than 30 ms and more than 300 ms are known to be in error and are discarded. Also, each OTM advance is compared to the two previous OTM advances. The current value is used only if it agrees with both previous measurements to within ±12 ms. Thus, the software will not change the OTM advance until the third OTM, and then only if a consistent delay is received. If any of these tests fails, the OTM is reset to "*" and the advance is reset to 45 ms.

5. Dynamic Range of the Path Delay Measurements

The dynamic range of the path delay measurements is limited by the fact that the DOS clock is running fast by 13 to 15 ticks. The time code is generated at the start of the DOS clock second. Under worst-case conditions (the DOS clock is only 13 ticks fast), the time code construction begins when the actual time is no later than 286 ms into the previous second. The actual time is more likely to be about 231 ms in the previous second on the average. The transmission of the time code begins immediately. The time code consists of 51 characters and can take no longer than 425 ms to transmit at 1200 bits per second so that the transmission will be finished by 711 ms (425 + 286) of the previous second at the latest. Again assuming the worst case (the DOS clock is 15 ticks fast), the time code for the next second will be constructed when the actual time is 176 ms into the second. Thus, an interval of 465 ms (1176 − 711) exists between the latest time at which the transmission of one time code can finish and the earliest time at which the construction of the next one must begin. This period of 465 ms can be used to measure delays up to 232 ms without using interrupts or interfering with other portions of the code. This delay is long enough for almost all paths, although it might not be long enough for a call connected through a geostationary satellite.

C. Technical Description of Monitoring Software

The monitoring software is written in BASIC for the MS-DOS operating system. It might not work properly using Microsoft Windows or another multi-tasking environment. The ACTS Monitor (ACTSM) software performs the following functions:

- Compares the ACTS time codes to the time codes generated by its own independent clock (REF_CLK)
- Inspects the availability of access to the ACTS servers
- Collects the statistics (number of calls) from the ACTS servers

Unusual incidents are reported in log files. If ACTSM detects a date/time error (the time difference between ACTS and ACTSM exceeds the specified limit), an alarm signal (open circuit) will be sent to the ACTS alarm system. If no ACTS server can be reached during a scheduled inspection, an alarm message is displayed on the ACTSM screen. The ACTSM can monitor up to four ACTS servers. The ACTSM will handle leap years automatically. Leap seconds will be handled according to the leap second flag status in the ACTS time codes.

1. Comparing ACTS time codes to REF_CLK

The REF_CLK is an accurate, real time clock that serves as a reference time source to the ACTSM. The REF_CLK is a 12.288 MHz voltage controlled crystal oscillator (VCXO) that is locked to an external 5 MHz or 10 MHz reference frequency. The 12.288 MHz signal is then divided by 375 to generate the 32.768 kHz time base for driving the real time clock chip (HM58167 or HM68167). The clock chip outputs the month, day of month, day of week (not used), hours, minutes, seconds, tenths and hundredths of seconds, and milliseconds. The year and MJD are managed by the monitor software. The 5 MHz or 10 MHz external reference frequency can be selected by setting a jumper on the board.

The monitoring software obtains time codes from the ACTS servers and compares them to REF_CLK. The time codes are obtained in two different ways, by eavesdropping on the server's serial lines, or by calling the server through a modem and telephone line (in the same fashion as any ACTS customer). By periodically dialing the telephone line, the monitor is able to insure that the modem and server are working properly and are available for normal access.

During normal operation, each ACTS time server is checked every 10 minutes, and 10 time codes are collected during each check. An error is reported if | REF_CLK – ACTS | > 100 ms. Keep in mind that a date error is clearly more than 100 ms. When errors occur, the ACTS alarm system is activated and NIST personnel are called. The REF_CLK card also contains a relay that is used to activate the alarm system.

2. Collecting Statistics (Number of Calls) from the ACTS Servers

The monitoring systems at NIST in Boulder download statistics from the ACTS servers at the beginning of each UTC day. This feature is not implemented in Hawaii. This information is then uploaded to the Internet (via FTP) where it can be viewed remotely using a standard web browser. The Boulder monitors are connected to the Internet using a DOS compatible Ethernet card, and a DOS TCP/IP stack and FTP program are installed on each system.

C. Technical Description of Client Software

ACTS client software uses an analog modem to dial the telephone, connect to an ACTS server, receive one or more time codes from the server, and then use the information contained in the time code to synchronize a local clock. Each ACTS client is actually a simple terminal program that can establish a connection with a remote computer. However, unlike a terminal program (which could be used only to view ACTS time codes and not to set a clock), an ACTS client can also perform the synchronization.

When the client software first connects to ACTS, it receives a time code whose last character is an asterisk (*). The asterisk is called the On-Time Marker (OTM). The time values sent by the time code refer to the arrival time of the OTM. In other words, if the time code says it is 12:45:45, this means it is 12:45:45 when the OTM arrives. Of course, there is some delay between the time the OTM leaves the ACTS server and the time it arrives at your computer. Some of this delay is the actual data transmission time. However, most of the delay occurs when the modem processes the incoming data and sends it to the computer.

Since there is some path delay as the OTM travels from ACTS to your computer, the OTM is sent out 45 ms early. The 45 ms advance was chosen based on experiments conducted at NIST in 1989 using 1200 baud modems. [11] This 45 ms includes the 8 ms that it takes to send the OTM at 1200 baud, 7 ms transmission time to allow for travel from NIST to an average user in the United States, and 30 ms to allow for the modem's differential delay. As modem technology advanced and the speeds became faster, these average delays have less meaning. For example at 9600 baud, OTM transmission time drops to 1 ms, but modem delays become less stable due to data compression and could be much larger than 30 ms.

Advancing the OTM by 45 ms always removes some of the path delay. However, to reduce the timing uncertainty as much as possible, the ACTS server needs to advance the OTM by the amount of the actual path delay. ACTS can do this by using a *loop-back* technique to calibrate the path. The loop-back technique works if the client software returns the OTM to NIST after it is received. Each time the OTM is returned, the server (as described earlier) measures the round trip path delay; or the amount of time it took for the OTM to go from the server to the client and back to the server. The round-trip path delay is divided by 2 to obtain the one-way path delay. After three consecutive measurements have been made, the server advances the time by the amount of the one-way path delay. For example, if the one-way path delay is 50.4 ms, the server sends the OTM out 50.4 ms (instead of 45 ms) early. At this point, the path delay is calibrated, and OTM changes from an asterisk to a pound sign (#). To get the least amount of uncertainty, a time code with a pound sign character as its OTM is then used to synchronize the local clock.

D. Time Code Format

The ACTS time code is described in Table 6. Note that the Daytime Time Code (Table 3) used by the Internet Time Service was based on the ACTS time code, and is very similar.

Table 6. The ACTS Time Code.

JJJJJ YR-MO-DA HH:MM:SS TT L UT1 msADV UTC(NIST) <OTM>
JJJJJ is the Modified Julian Date (MJD). The MJD is the last five digits of the Julian Date, which is simply a count of the number of days since January 1, 4713 B.C. To get the Julian Date, add 2.4 million to the MJD.
YR-MO-DA is the date. It shows the last two digits of the year, the month, and the current day of month.
HH:MM:SS is the time in hours, minutes, and seconds. The time is always sent as Coordinated Universal Time (UTC). An offset needs to be applied to UTC to obtain local time. For example, Mountain Time in the U. S. is 7 hours behind UTC during Standard Time, and 6 hours behind UTC during Daylight Saving Time.
TT is a two-digit code (00 to 99) that indicates whether the United States is on Standard Time (ST) or Daylight Saving Time (DST). It also indicates when ST or DST is approaching. This code is set to 00 when ST is in effect, or to 50 when DST is in effect. On the day of the transition from DST to ST, the code is set to 01. On the day of the transition from ST to DST, the code is set to 51. The client software is responsible for implementing the change at 2 a.m. on the day of the transition. During the month of the transition, the code is decremented every day until the change occurs. For example, October is the month of the transition (in the United States) from DST to ST. On October 1, the number changes from 50 to the actual number of days until the time change. It will decrement by 1 every day, and reach 01 on the day of the transition. It will be set to 00 the day after the transition, and will remain there until the following April.
L is a one-digit code that indicates whether a leap second will be added or subtracted at midnight on the last day of the current month. If the code is 0, no leap second will occur this month. If the code is 1, a positive leap second will be added at the end of the month. This means that the last minute of the month will contain 61 s instead of 60. If the code is 2, a second will be deleted on the last day of the month. Leap seconds occur at a rate of about one per year. They are used to correct for irregularity in the earth's rotation.
UT1 is a correction factor for converting UTC to an older form of universal time. It always ranges from -0.9 to $+0.9$ s. This number is added to UTC to obtain UT1.
msADV is a five-digit code that displays the number of milliseconds that NIST advances the time code. It is originally set to 45.0 ms. If you return the on-time marker (OTM) three consecutive times, it will change to reflect the actual one-way line delay.
The label UTC(NIST) is contained in every time code. It indicates that you are receiving Coordinated Universal Time (UTC) from the National Institute of Standards and Technology (NIST).
The on-time marker (OTM) is a single character sent at the end of each time code. The OTM is originally an asterisk (*) and changes to a pound sign (#) if ACTS has successfully calibrated the path delay.

III. Operational Procedures of ACTS

This section discusses the hardware and software maintenance required by ACTS. It also discusses the ACTS failure modes and security issues.

A. Hardware Maintenance

The ACTS servers are maintained by NIST Time and Frequency Division personnel, who have been trained in the procedure of repairing and/or replacing a server. At least one operational spare server is kept in the same room as the on-line servers, and a small inventory of spare parts is stored in the same room. The most likely cause of a failure is a faulty modem or power supply. While a server is being repaired or replaced, the phone lines connected to that server are "busied out" so that those lines are skipped by the telephone rotary. This temporarily reduces the service capacity, but does not normally cause a problem.

B. Software Maintenance

Since ACTS is a mature system (on-line since 1989), the server software is seldom changed. Since the software is small, new files are copied to the servers when necessary from a floppy disk. A single floppy installation disk can restore an entire ACTS system, and copies of these installation disks are stored in the Time and Frequency Division.

Notifications for upcoming leap seconds and the current UT1 correction have to be manually entered. These entries are made from the keyboard of each server. These are the only required time code entries, since the servers automatically compute the transition dates from ST to DST and from DST to ST. This computation is made using the current United States rules, which call for the ST to DST transition to occur on the first Sunday in April, and for the DST to ST transition to occur on the last Sunday in October.

C. Failure Modes

ACTS failures can be divided into two groups: client side failures, and server side failures, as described below. The service was designed to have no single point of failure.

1. Client Side Failures

Client side failures are usually caused by a problem with the client's software or modem, a busy signal, or a telephone line failure. All failure modes are fairly rare, and NIST personnel will assist the client with technical support. If the client software redials when receiving a busy signal, it should get through to the server within a few minutes at most, since the capacity of ACTS far exceeds the current client load.

2. Server Side Failures

Server side failures can be divided into two groups: hardware failures and time code failures.

a) Hardware failures

Hardware failures are usually reported by the alarm system and repaired immediately by NIST personnel. The most common failure mode is a defective modem that will not accept calls (removing one line from the ACTS rotary), or a server power supply failure (removing from four to six lines from the ACTS rotary). Since ACTS has a lot of excess capacity based on the current system load, hardware failures are usually transparent to clients.

b) Time code failures

The ACTS monitor described in the Section II.C of this chapter is able to find and report time code failures within 10 minutes of their occurrence. The ACTS alarm system then notifies the security desk at the NIST Boulder laboratories. Personnel at the security desk then alert Time and Frequency Division employees either at work or at home, so that they can report to the ACTS room and fix the problem. A list of multiple employees is on file at the security office in case an individual employee cannot be contacted. Time code failures have been very rare, and occur at a rate of much less than one per year.

D. Record keeping

The automated diagnostic and monitoring software described earlier produces files that are maintained and stored at NIST for record keeping purposes. All other information pertaining to the service, including documentation of software changes, system maintenance, outages, and so forth, are archived at NIST for an indefinite period of at least several years.

E. Security issues

This section discusses the security of the ACTS time servers and their vulnerability to misuse or attack.

1. Physical Security

The ACTS servers are located in rooms that have keypad-activated locks so that entry to these locations can be monitored and controlled. Only trained members of the Time and Frequency Division are permitted to enter these rooms.

2. Security Against External Command Entry or Log-ins

The ACTS time servers do not accept external commands that can modify the state of the server or change the time code that is broadcast to users. Since the servers run a single-tasking operating system, it is not possible to log-on to an ACTS server remotely.

IV. Customers

This section describes the number of ACTS users, the service capacity, and the ways that NIST interacts with and supports users.

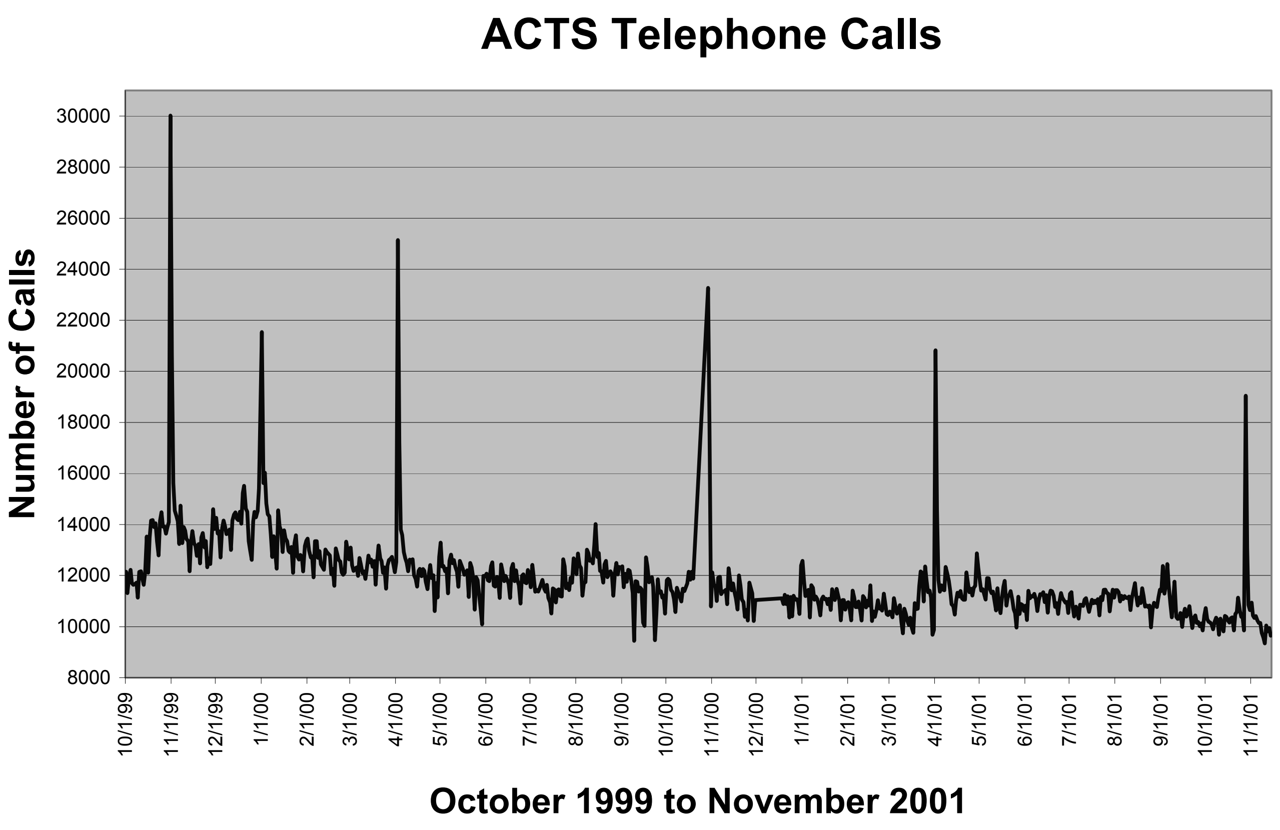


Figure 10. Daily time requests received by ACTS.

A. Estimated number of users

ACTS currently receives about 10,000 calls per day in Boulder, Colorado and about 400 calls per day in Hawaii. Figure 10 graphs the usage of the 24-line Boulder public rotary from 10/01/1999 to 11/14/2001 (25 months). The number of phone calls received by ACTS during the most recent UTC day can be viewed here:

http://gpsmonitor.timefreq.bldrdoc.gov/actscalls.htm

Figure 10 shows that ACTS usage is gradually declining over the 25 month period. We believe that much of this decline has to do with ACTS customers switching over to the Internet Time Service, and also with ACTS customers switching to digital modems, which makes further access to the analog time servers impossible. The large spikes shown on the graph represent the transition days from standard time to daylight saving time (the first Sunday in April) and from daylight saving time to standard time (the last Sunday in October). There was also a large spike on the year 2000 (Y2K) rollover on January 1, 2000. It is interesting to note that the number of calls during these peak calling periods have declined at a linear rate. The transition day to standard time netted over 30,000 phone calls in 1999, but fewer than 20,000 calls in 2001.

Two other events shown on the Figure 10 graph are of interest. The unexpected increase in calls that took place in March 2001 is believed to be due to an article about the NIST primary frequency standard that appeared in *Discover* magazine. This article mentioned ACTS, but not the Internet Time Service. This proves that there are still a substantial number of people who are unaware of NIST computer time services and who can be reached through the popular press. The second event relates to the sharp decrease in ACTS calls that occurred immediately after the terrorist attack on the United States on September 11, 2001. It is believed that hundreds of computer systems in the New York City financial district function as ACTS clients, and that many of these systems were destroyed in the attack.

B. Capacity of Service

The public ACTS rotary in Boulder, Colorado, has a capacity of about 60,000 calls per day; the Hawaii rotary has a capacity of about 10,000, as listed in Table 5. Based on current call levels (November 2001), the Boulder system is operating at less than 20 % of capacity, and the Hawaii system is operating at less than 5 % of capacity.

C. Customer Interaction

The huge increase in usage of the Internet Time Service (Chapter 1) has greatly reduced the number of new ACTS customers, and therefore most of the customer interaction. However, NIST personnel promptly respond to telephone calls and emails regarding ACTS and are usually able to quickly solve the customer's problem.

D. Other Forms of Customer Support

NIST utilizes both its web site and FTP sites to help support ACTS, as described in this section.

1. Web Page

The ACTS page is used to distribute software, and to alert customers about any changes or outages to the service. It also provides instructions for using the service, and links to publishers of third-party software. The page can be found here:

http://tf.nist.gov/service/acts.htm

2. FTP Files

All of the ITS time servers (Table 1) support anonymous FTP access to documentation, client software source code and executable files, and similar information regarding ACTS. Users can retrieve these files by connecting directly to any of the time servers. A current list of time servers can be obtained here:

http://tf.nist.gov/service/its.htm

V. Measurement Uncertainties

This section discusses the measurement uncertainties of the signals transmitted by the ACTS servers, and of the signals received by ACTS clients. It also discusses establishing traceability to UTC(NIST).

A. Uncertainties of Transmitted Signals

The ACTS time servers located at the NIST laboratories in Boulder, Colorado, directly connect to the national time and frequency standard, UTC(NIST). The Hawaii ACTS system is referenced to the WWVH station clock, which is typically within 100 ns of UTC(NIST). The resolution of the receiving hardware on the ACTS servers is 1 μs, and the stability of the server time is <100 μs (1σ).

Keep in mind that the uncertainty of the transmitted signal is essentially meaningless to the user. The uncertainty of the received ACTS signal (next section) is usually much larger, since it is dominated by the variation in modem, telephone line, and operating system delays.

B. Uncertainties of Received Signals

The best-case uncertainty of ACTS was demonstrated with an experiment [11] that used 300 bit/s modems on both the server and client computers, and a path that was entirely contained in a local telephone network. This experiment produced a timing uncertainty of about 90 μs (1σ) using the individual time codes with no averaging. The same experiment was repeated using 1200 bit/s modems (with larger internal delay variations than the 300 bit/s modems) and the uncertainties were 2 to 3 times larger. The 300 bit/s measurement was then repeated between an ACTS time server located in Boulder, Colorado, and an ACT time client located at NIST radio station WWVH in Hawaii. The results were similar, suggesting that the variations in the modem delays (and not in the telephone path delays) were the dominant sources of uncertainty.

Obtaining uncertainies nearly as small as those cited above might still be possible, but only if a single-tasking or real-time operating system is used by the client, and if a dedicated 300 or 1200 bits/s modem is used by both the server and the client. This is the type of connection the Internet Time Service (Chapter 1) uses to synchronize to the private ACTS rotary. However, in practice, the uncertainty of ACTS is much larger for most users. This is due to several factors. The first factor is the widespread use of higher speed modems (such as 28.8 and 56 kilobit/s devices) that use data compression and error correction schemes to deliver more data using the same amount of bandwidth. These modems have much larger internal delays than the 300 and 1200 bit/s modems that contained no data compression or error correction software. The second factor is the widespread use of multitasking operating systems that can introduce latencies of several milliseconds or more when servicing the modem. A third, but less important factor, is that the PC-based ACTS servers introduced in 1994 are less stable than the original, custom-built ACTS servers. As a result of these factors, more recent measurements using higher speed modems have shown uncertainties of approximately 5 ms (1σ) using the single tasking MS-DOS operating system, and ranging from 15 to 20 ms (1σ) using the multitasking Windows operating system [13]. These uncertainties are still smaller than the clock resolution (typically 55 ms) of most personal computers.

C. Establishing Traceability to UTC(NIST)

ACTS is normally used to establish traceability to UTC(NIST) for time-of-day measurements only, although time interval measurements are sometimes made using ACTS as a reference, and even frequency measurements have been made in very rare instances.

The time interval measurements made using ACTS are generally used to calibrate stop watches and timers [14]. Since the ACTS time code is character-based, it can be read from the screen by a human operator, and used to start and stop a stopwatch. Typically, one call is made to ACTS using a terminal program or specially designed client program to start the timer, and another call is made (usually several hours later) to stop the timer.

The original ACTS documentation proposed two schemes for measuring frequency using ACTS. Both involved circuits [10, 15] that would enable the client computer to generate a 1 pps pulse during an ACTS connection. One circuit required special client software and generated a pulse at the arrival time of the OTM using the computer's parallel port. The other circuit was connected directly to the modem or modem cable and also generated a pulse at the arrival time of the OTM. The output pulse could then be connected to a time interval counter and compared to an oscillator whose output frequency was divided to 1 pps. Using this method, it was shown [12] that frequency could be measured with an uncertainty of 1×10^{-8} or less using this scheme over an interval of one or more days.

Even if 1×10^{-8} meets the requirements of a calibration customer, it is not recommended that ACTS be used for frequency traceability, unless no other alternative exists. There are several obvious reasons for this recommendation. One reason is that the length of an ACTS connection is limited to 48 s, so there is no way to continuously measure frequency. Instead, repeated phone calls to ACTS are required at periodic intervals. A second reason is that much better references exist for frequency, including the Global Positioning System (GPS) satellite broadcasts, and the 60 kHz broadcast from NIST radio station WWVB. Even the high frequency radio broadcast from NIST radio stations WWV and WWVH have less variation in their path delay than ACTS, and are better suited for low level frequency measurements. For these reasons, ACTS should normally be used to establish traceability only for time-of-day or time interval measurements.

A sample ACTS traceability chain with typical uncertainties is listed in Table 7.

Table 7. The ACTS Traceability Chain.

Link	Reference	Compared To:	Uncertainty (2σ)
A	SI units	UTC(NIST)	<100 ns
B	UTC(NIST)	NIST Automated Computer Time Service (ACTS)	<200 μs
C	ACTS	ACTS Client	<40 ms

Chapter 3 – time.gov Web Sites

NIST first provided a time-of-day clock on the Time and Frequency Division web site in 1997. The popularity of the clock prompted NIST to create two separate web sites dedicated to presenting the current time-of-day using the time.gov domain name. The original site, time.gov went on-line in 1999. In 2001, the nist.time.gov web site made its debut. These sites allow any user with Internet access and a web browser to view the current time in any United States time zone. This chapter describes the time.gov web sites in detail. It provides a physical and technical description of these services, and explains how NIST provides and maintains the time.gov web sites for the American public.

I. Physical Description of time.gov Web Sites

The time.gov and nist.time.gov web sites are hosted on web servers maintained in Boulder, Colorado by the Information Technology Laboratory (ITL) of NIST. The web sites provide a time-of-day clock display (usually of the user's local time) referenced to UTC(NIST) via the Internet Time Service (Chapter 1).

A. Introduction and History

The original web clock produced by the NIST Time and Frequency Division was part of the Division's web site located at **http://tf.nist.gov**. This clock was added to the web site in 1997, and was easily the site's most popular feature. Due to its popularity, the decision was made to create a new web site, independent of the Division's site, whose sole purpose would be to display the correct time. As a result, the time.gov domain name was registered on June 23, 1999, and the time.gov web site went on-line immediately afterwards.

Although the time.gov web sites were designed and are maintained by NIST, the decision to launch the original site was made jointly with the United States Naval Observatory (USNO), which, like NIST, is an official source of time in the United States. The information displayed on the time.gov pages pertains to both NIST and the USNO as part of a mutual recognition agreement between the agencies. The nist.time.gov site was launched on January 5, 2001. Although its inner workings are essentially the same as time.gov, it displays some information that is specific to NIST. The time.gov web sites are already the most visited web sites maintained by NIST, and their usage is expected to increase at a rapid rate during the coming years.

The sites are simple to use. Visitors to either **http://time.gov** or **http://nist.time.gov** are presented with a map of the United States (Figure 11). Time zones are shown in alternating colors and outlines of the states are visible. As the cursor passes over different time zones, a message is displayed listing the time zone in which the cursor lies. After the user clicks on one of the time zones, a time-of-day clock with the correct time in the selected time zone is displayed. Also, a gray line map appears, showing the parts of the Earth that are currently in light or darkness.

B. How the time.gov Web Sites are Used by Their Customers

Since the time.gov web sites only display the time, and are not capable of automatically synchronizing the computer's clock, we assume that most customers use the sites simply to see what time it is, or to manually synchronize other clocks (such as their computer clock, wall clock, or wrist watch). The sites serve as a convenient time reference that costs nothing to use if you already have Internet access. We have heard from at least one customer who uses the site as the primary time reference in their facility, and continuously displays the clock on a large display in the entryway of their building, using a dedicated Internet connection. While this works fine, this is obviously not recommended. Radio-controlled clocks (such as those that receive NIST radio station WWVB) are available that cost much less than a dedicated computer and Internet connection, provide better accuracy, and preserve bandwidth, since the same time signal can be simultaneously reach millions of receivers.

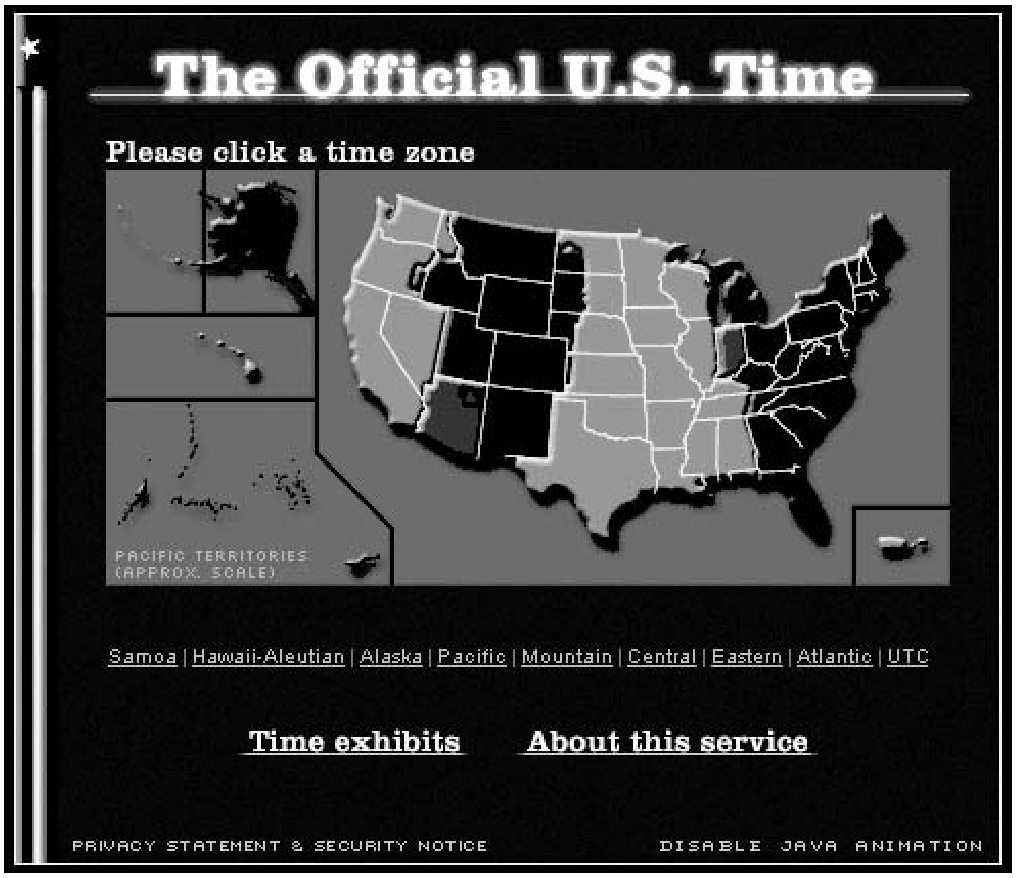

The Official U.S. Time
Please click a time zone
PACIFIC TERRITORIES (APPROX. SCALE)
Samoa | Hawaii-Aleutian | Alaska | Pacific | Mountain | Central | Eastern | Atlantic | UTC
Time exhibits
About this service
PRIVACY STATEMENT & SECURITY NOTICE
DISABLE JAVA ANIMATION

Figure 11. The time zone map displayed by the time.gov web site.

C. time.gov Web Site Facilities

The time.gov web sites are hosted on web servers maintained in environmentally controlled computer rooms in the Information Technology Laboratory (ITL) of NIST. The servers are connected to local UPS units that can provide backup power for about 30 minutes. A natural gas backup power generator provides power indefinitely if the electricity to the site is lost.

D. Organizational Control of time.gov Web Sites

The content and timekeeping functions of the time.gov web sites are managed, controlled, and maintained by the Services Group of the NIST Time and Frequency Division. The Group Leader of the Services Group reports to the Division Chief of the Time and Frequency Division, who in turn reports to the Director of the NIST Physics Laboratory. The Physics Laboratory is one of the seven operational units of NIST, and NIST is an agency of the United States Department of Commerce.

The web servers that host the time.gov web sites are managed, controlled, and maintained by the Information Technology Laboratory (ITL) of NIST. The Time and Frequency staff works closely with the ITL staff to keep the sites running smoothly.

II. Technical Description of time.gov Web Sites

This section summarizes the technical operation of the time.gov web sites, including a discussion of the hardware used and the server and client software.

A. Technical description of hardware

The time.gov web sites run on commercially available web servers as described below. Since the software is highly portable, it should be possible to rehost the sites to another platform if necessary.

1. Web Server

The Java clock runs on *Bldweb*, a web server located at NIST's Boulder laboratories. The server is IBM RS6000 with a 350 MHz clock speed, four 64-bit processors, and four gigabytes of memory. It uses an HA70 two-node cluster for failure protection. This is referred to as a highly available system because two identical nodes run simultaneously, and if one fails, the other takes over within several minutes.

2. Available Bandwidth

The web server shares a pipeline out of Boulder, Colorado, with other government agencies located near the NIST Boulder laboratories. The pipeline consists of multiple T1 lines (1.544 megabits per second). Bandwidth is allocated on a first-come, first-served basis and the maximum available bandwidth is at least 10 megabits per second.

B. Technical description of server software

The server software consists of the operating system, and the software developed at NIST that displays the time-of-day. A standard web browser is the only client software required to use the service.

1. Operating System and Web Server Software

The web server runs IBM AIX 4.3.3 (IBM's version of UNIX) and Sun/Netscape's Iplanet 4.1 web server software.

2. Time Server Software

The sites were built using a combination of HyperText Markup Language (HTML), Perl, Java and Javascript code. When a customer first connects to the site, a time zone map (Figure 11) is displayed. The time zone map is a bitmapped image, displayed by an HTML page. Each time zone on the map is a link. When a user clicks a link, it sends a string of parameters to a Common Gateway Interface (CGI) script written in Perl (*timezone.cgi*). Four parameters are sent: the name of the selected time zone (Pacific, Mountain, etc.), a character indicating whether the time zone observes Daylight Saving Time (DST), the time offset in hours for the selected time zone from UTC, and whether the site should display a running clock or a static clock that provides a "snapshot" of the time. A call to the CGI script might look like this:

http://www.time.gov/timezone.cgi?Eastern/d/-5/java

The parameters indicate that the Eastern time zone was selected. The "d" signifies that the Eastern time zone does observe Daylight Saving Time. Hawaii, most of Arizona and parts of

Indiana are areas that do not observe DST. The "−5" indicates that there is a 5 hour time offset between Eastern standard time and UTC. The HTML code attempts to determine whether the client's web browser supports Java. If it does, the word "java" is included with the parameters and the server will display a running clock to the client. If it does not, the word "java" is omitted and the server will display a static "snapshot" of the time to the client. Likewise, if the user clicks on the link below the time zone map entitled "DISABLE JAVA ANIMATION", the word "java" is also omitted from the string.

After it is launched, *timezone.cgi* first checks the current time and date by getting a time string from the server. It calculates whether it is currently Daylight Saving Time or Standard Time using the current DST rules for the United States. DST is in effect from 2:00 a.m. on the first Sunday in April until 2:00 a.m. on the last Sunday on October. In April, clocks are moved one hour ahead. In October, clocks are moved one hour back. Because the DST rollover happens at 2:00 a.m. locally, the script has to determine whether it is currently before or after 2:00 a.m. locally for the specific time zone chosen.

The script then checks the parameter string to see if this is a Java or non-Java request. Both types of requests are described below.

a) Java time requests

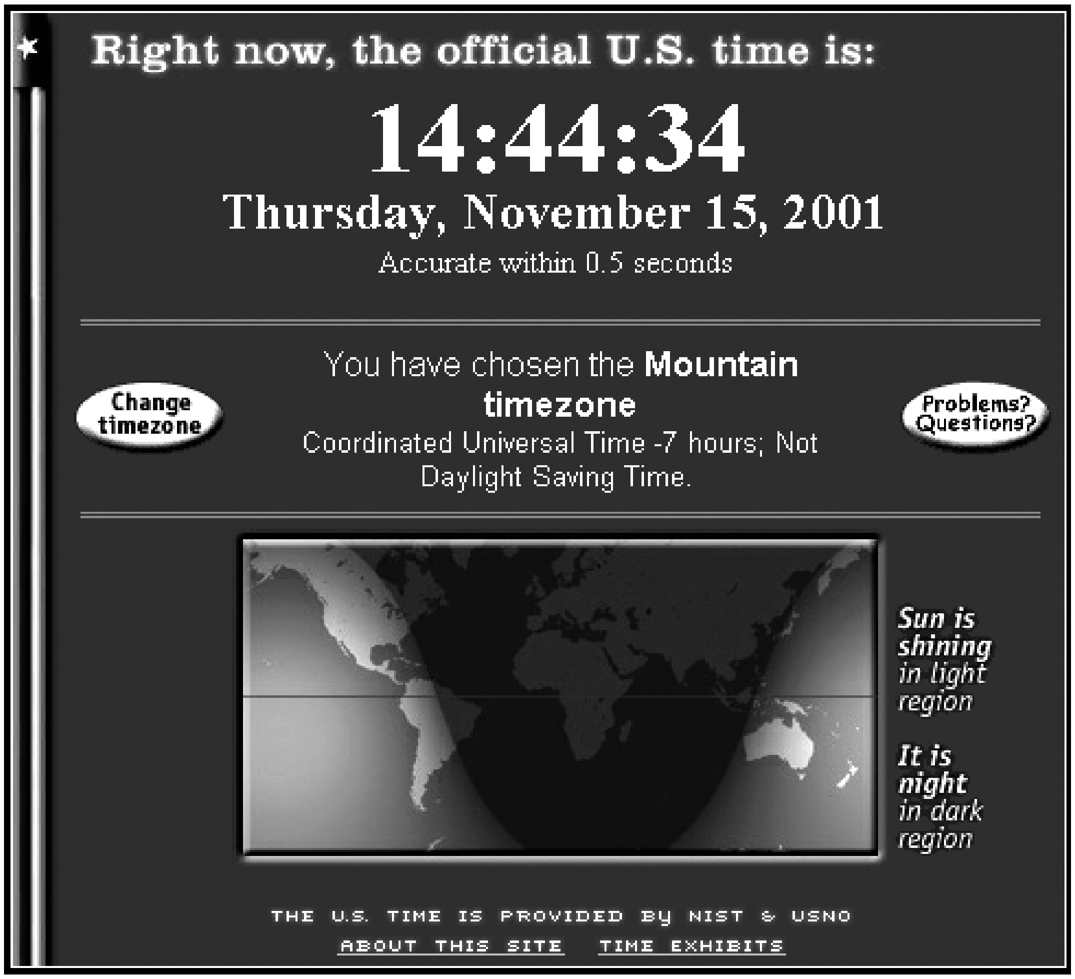
If a Java time request is made, the script prepares to display an animated clock and a gray line map on the client's web browser. The HTML file *clock_top.html* is sent to the browser, and variables for a Javascript program are initialized. The Javascript program is used to display the correct gray line map for the current solar position. An additional HTML file, *clock_mid.html*, is then sent to the browser to continue the formatting of the page.

The *timezone.cgi* script then sends the Java applet to the client's web browser. The code contained in the file *utcnist4.class* first verifies that the applet is being sent from a NIST domain, so that non-NIST servers cannot retransmit the applet and display it on other web sites. The applet also calls methods contained in the *correctnist4.class* file.

First, the method *measure* is called. It assigns a variable to the client's current clock time, and then calls the method *GetDeltaT*, which in turn calls *GetServerDateString* to get UTC(NIST) from the server clock. *GetServerDateString* opens a connection back to the web server using port 8013 (an arbitrary port assignment chosen by NIST) to rendezvous with the *utcgate.class* code running as a daemon. The server clock time is passed to the applet through the TCP connection, and the connection is closed. At this point, the *GetDeltaT* has the time of both the server and client clocks, expressed with a resolution of 1 ms. Both time strings are then converted to the number of elapsed milliseconds since January 1, 1970, a time epoch used by the UNIX operating system. The two values are subtracted, and the result is the time offset between the server and client clocks. This value is stored as the variable *deltaT*. The user's computer time and date do not need to be close to the correct values for this to work correctly. For example, if the client clock were behind by one year, then *deltaT* would exceed 3×10^{10} ms. This value is returned to the original applet method "measure" and stored as the variable *c.value*. The time displayed in the client's web browser is the client clock time corrected by *c.value*.

The time uncertainty of the display is roughly equal to the amount of time required to compare the server and client clocks. This includes the time it takes for the applet to call the time server, query the user's computer clock, convert the strings to time since the epoch, calculate the difference, and send the variable back. The goal is to complete this comparison in less than 1 s, and not display the time at all if the comparison takes more than 3 s. Up to three attempts are made to make a valid comparison. If the time offset is obtained in <1 s (which usually happens on the first try), the comparison is accepted. If it takes more than 2 s to make the comparison, the process times out. If the comparison takes more than 1 s but less than 2 s, the delay value is stored. If two delay values >1 s have been obtained, then the shorter delay is used. If no delay values of <2 s are obtained after three tries, the system stops trying to display the time, and instead displays the message "Net Congestion".

Figure 12 shows the result of a successful Java time request. Note that the text below the time reads "Accurate within 0.5 seconds". This message serves as a coarse, "worst case" estimate of the time uncertainty. The actual uncertainty of the displayed time should always be smaller.



Right now, the official U.S. time is:

14:44:34

Thursday, November 15, 2001

Accurate within 0.5 seconds

You have chosen the **Mountain timezone**

Coordinated Universal Time -7 hours; Not Daylight Saving Time.

Sun is shining in light region

It is night in dark region

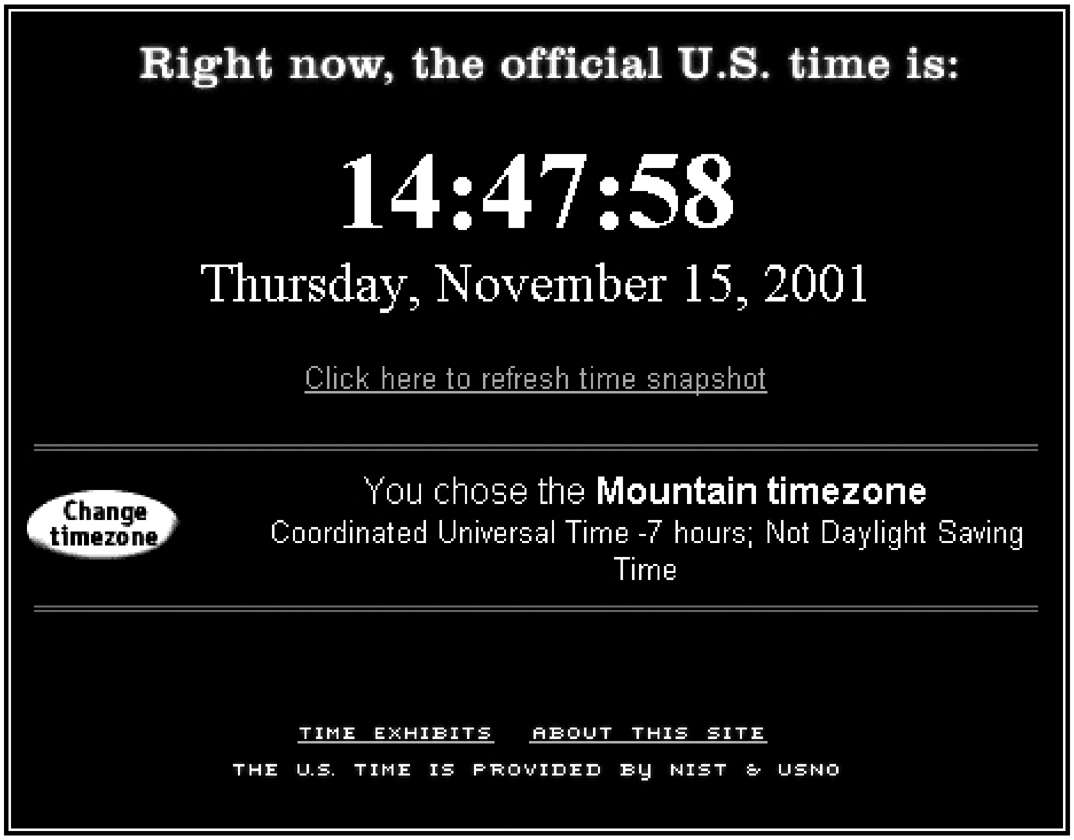
THE U.S. TIME IS PROVIDED BY NIST & USNO

Figure 12. The animated web clock displayed in a client's web browser.

If the client's web browser is left open while connected to the site, the time-of-day will be continuously displayed. The Java applet makes a new time request every 10 minutes, and resynchronizes the time-of-day display. The process is also started over if the client's computer clock is adjusted, or the browser window is resized. The displayed gray line map is chosen from thousands of files that show where the sun is shining on Earth, based on the time of day and the time of year. These calculations are performed by the Javascript program, and displayed in an HTML file named *clock-bot.html*.

b) Non-Java Time Requests

If a non-JAVA time request is made, the Java applet is not used. Instead, the HTML file *clock-nojava.html* is sent to the client's web browser using Port 80, the standard port for a HyperText Transfer Protocol (HTTP) request. The file contains a time code synchronized to UTC(NIST) and adjusted for the selected time zone. This static time code is displayed in the client's browser, and the client must refresh the page to get a new "snapshot" of the time. Figure 13 shows the results of a successful non-Java time request.


Right now, the official U.S. time is:

14:47:58

Thursday, November 15, 2001

Click here to refresh time snapshot

You chose the **Mountain timezone**
Coordinated Universal Time -7 hours; Not Daylight Saving Time

Figure 13. A static "snapshot" of the time displayed in a client's web browser.

III. Operational Procedures of time.gov Web Sites

This section discusses the hardware and software maintenance required by the time.gov web sites. It also discusses the failure modes and security issues.

A. Hardware Maintenance

The web server and operating system software are covered by a maintenance agreement with IBM that provides 24-hour, on-site service. This maintenance agreement is paid for by ITL and is renewed every year.

B. Software Maintenance

Time and Frequency Division personnel maintain the Perl, Java, Javascript, HTML, and graphics files used by the time.gov web sites, and are responsible for their proper operation. As with most web sites, these files are modified regularly, and the look, feel, and performance of the sites will surely evolve during the coming years. A test platform on NIST's internal web site is used to test new features and software changes. ITL personnel are requested to assist with the software development and configuration of the sites when necessary.

No maintenance is required on the time code, since it is obtained from the Internet Time Service, and all time code corrections have already been made by that service.

C. Failure Modes

Failures can be divided into two groups: client side failures, and server side failures, as described below. The service was designed to have no single point of failure.

1. Client Side Failures

Client side failures are usually caused by a problem with the client's web browser or local firewall. If the client's web browser is not Java-enabled, for example, an older browser or a browser with Java support turned off, a Java time request will fail. If the client resides behind a firewall that has port 8013 blocked, a Java time request will also fail. Non-Java time requests are seldom blocked by a firewall, since they use port 80, the standard port for an HTTP request. This type of request is used to access all web sites.

2. Server Side Failures

Server side failures can be divided into two groups: hardware failures and time code failures.

a) Hardware failures

Hardware failures are usually detected immediately by ITL, and IBM is called in if necessary to perform on-site maintenance. Total failures are rare since the web server has two identical nodes that run simultaneously, and if one fails, the other quickly takes over.

b) Time code failures

Since the time.gov web sites obtain their time codes from the ITS, the ITS monitoring system is used to detect time code failures (Chapter 1, II.B.3.a). Although it has not happened in the past, we are prepared to take the web sites off-line if the wrong time is being transmitted.

D. Record Keeping

Information pertaining to the service, including documentation of software changes, system maintenance, outages, and so forth, is archived at NIST for at least several years.

E. Security Issues

This section discusses the security of the time.gov web servers and their vulnerability to misuse or attack.

1. Physical Security

The time.gov web servers are located in rooms that have keypad-activated locks so that entry to these locations can be monitored and controlled. Only trained members of ITL are permitted to enter these rooms. The content for the time.gov web sites is periodically copied from master copies stored on a host computer inside the NIST firewall. All of the data stored inside the firewall are automatically copied to backup media for redundancy.

2. Security Against External Command Entry or Log-ins

ITL employs a full-time security administrator and several other personnel trained in security issues at the NIST Boulder laboratories. These employees regularly test the web sites for security holes and install patches to the operating system if a server is found to be vulnerable. Security procedures appropriate for a large web site are routinely performed. If a web server is compromised, ITL works as quickly as possible to return it to its normal state and restore service.

IV. Customers

This section describes the number of time.gov users, the service capacity, and the ways that NIST interacts with and supports the users.

A. Estimated number of users

It is difficult to estimate the number of client computers that access the time.gov web sites. Unlike the ITS described in Chapter 1, the time.gov web sites store log files that record the Internet protocol (IP) address of every client that accesses the sites, and the number of unique IP addresses logged currently (November 2001) numbers in the hundreds of thousands per week. However, the number of unique IPs tends to overstate the number of actual clients, since most customers use Internet Service Providers (ISPs) that dynamically assign a new IP address each time they log-on to the Internet. This means that the same customer could access the time.gov web sites each day of the current month and be counted as a unique IP address every time. Conversely, some ISPs share IP addresses among multiple users and more than one unique user could be counted as a single unique IP address. Although an exact count of unique users is impossible, it is safe to say that the sites are accessed by tens of thousands (if not hundreds of thousands) of client computers every month, and that the number is rapidly growing.

It is possible to precisely count the number of time requests. All time requests (both Java and non-Java) access the *timezone.cgi* Perl script file, so the number of requests for this file equals the number of time requests. Figure 14 shows the number of time requests for both time.gov and nist.time.gov from the date when each site went on-line through October 2001. Note that the sites handled nearly 5 million combined time requests in October 2001. Keep in mind that the metric typically used to state the popularity of a web site is page views, or the number of pages displayed by a site. If page views were shown on the graph instead of time requests, the numbers would probably be higher by a factor of two or more, since most clients view at least two pages during each visit.

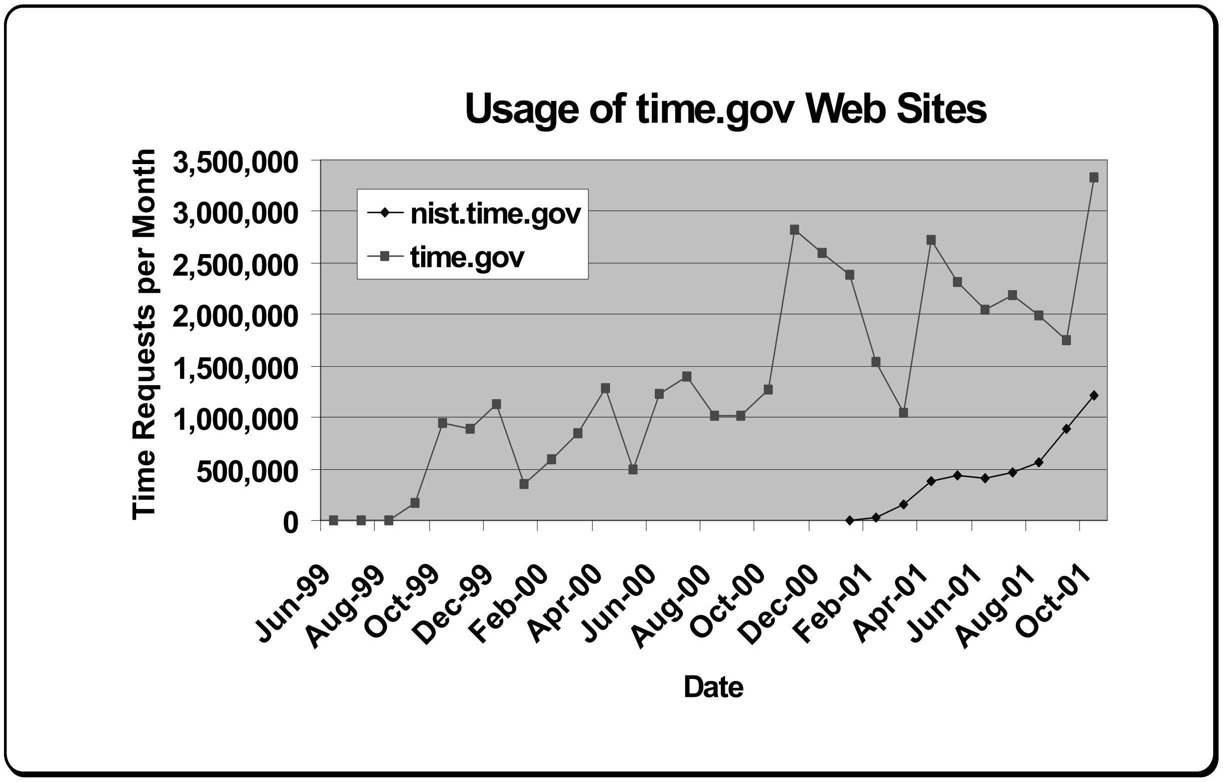


Figure 14. Time requests received by the time.gov web sites.

B. Capacity of Service

The actual capacity of the service is currently limited more by the amount of processing required by the software, than by the available bandwidth or capacity of the web server. On peak days (such as the transition day for daylight saving and standard time), the sites have been saturated for several hours or more, and in some cases, we have modified the sites to accept non-Java requests only to reduce the amount of processing required. Steps are being taken to optimize the software so it makes better use of the available bandwidth, and continues to handle the increased load expected in the coming years. If necessary, NIST is also prepared to increase the amount of available bandwidth and/or the capacity of the web server to continue to support the service.

C. Customer Interaction

Since an email link to NIST is found on the time.gov web sites, nearly all customer interaction is handled through email. The support email address is timeinfo@boulder.nist.gov. More than 50 emails are received each week. Although NIST cannot always guarantee a response, an attempt is made to answer each email, and most questions regarding the sites are easily answered.

V. Measurement Uncertainties

This section discusses the measurement uncertainties of the signals transmitted by the time.gov servers, and of the signals received by the time.gov clients. It also recommends that the time.gov signals should not be used to establish traceability to UTC(NIST).

A. Uncertainties of Transmitted Signals

The web server clock is synchronized to UTC(NIST) via the ITS (Chapter 1), and steered in between calibrations with the *autolock* algorithm described in Chapter 1 and reference [8]. The uncertainty of the server clock should never exceed 100 ms, even if it is unable to access the ITS for more than 1 day. Under normal conditions, the uncertainty relative to UTC(NIST) is a few milliseconds.

Keep in mind that the uncertainty of the transmitted signal is essentially meaningless to the user. The uncertainty of the time displayed in the clients web browser is usually much larger since it is dominated by processing and network delays.

B. Uncertainties of Received Signals

Our goal for the time.gov web sites is to reliably display the time within 1 s of UTC(NIST), and this goal is realized for most time requests. If a Java time request is made, the page displays the estimated error of the time display. This serves as a very coarse estimation of the uncertainty of the displayed time, with a resolution of 100 ms (0.1 s). However, the instabilities in this delay are much larger than the resolution. Due to processing delays introduced by both the server and client (the dominant source of uncertainty), and/or network delays introduced by a busy ISP, it is impossible to provide time within 1 s of UTC(NIST) to all users. In some cases the uncertainty might be as large as 3 s, so the measurement uncertainty for a typical user is very difficult to characterize.

C. Establishing Traceability to UTC(NIST)

Since the measurement uncertainties are so difficult to characterize, the time.gov web sites should be used as a time-of-day service only. They **should not be used** to establish traceability to UTC(NIST), even for measurements (such as stop watch or timer calibrations) with the most modest requirements. If traceability is required, we recommend using the services described in Chapters 1 and 2.

Cited References

[1] Postel, J. Daytime protocol. Network Working Group Report RFC-867. USC Information Sciences Institute; 1983 May.

[2] Postel, J. Time protocol. Network Working Group Report RFC-868. USC Information Sciences Institute; 1983 May.

[3] Mills, David L. Network Time Protocol (Version 3); Specification, Implementation and Analysis. DARPA Network Working Group Report RFC-1305, University of Delaware; 1992.

[4] Additional material on NTP is at the NTP web site: www.eecis.udel.edu/~ntp. The NIST time servers implement NTP as specified except that the authentication features based on encryption using either symmetric or asymmetric methods are not supported.

[5] Levine, Judah; Mills, David L. Using the Network Time Protocol (NTP) to Transmit International Atomic Time. Proc., 32nd Precise Time and Time Interval Planning and Applications Meeting (PTTI); 2000 November. 431-440.

[6] Levine, Judah. An Algorithm to Synchronize the time of a Computer to Universal Time. IEEE/ACM Trans. Networking (3): 42-50; 1995.

[7] Levine, Judah. Time Synchronization over the Internet. IEEE Trans. UFFC 45(2): 450-460; 1998.

[8] Levine, Judah. Time Synchronization over the Internet Using an Adaptive Frequency-Locked Loop. IEEE Trans. UFFC 46(4): 888-896; 1999.

[9] Sullivan, D.B.; Allan, D.W.; Howe, D. A.; Walls, F.L., ed. Characterization of Clocks and Oscillators. Natl. Inst. Stand. Technol. Tech. Note 1337; 1990 March. 352 p.

[10] Levine, J.; Weiss, M.; Davis, D.D.; Sullivan, D.B. The NIST Automated Computer Time Service. Natl. Inst. Stand. Technol. J. Res. 94: 311-321; 1989.

[11] Allan, D.W.; Davis, D.D.; Levine, J.; Weiss, M.A.; Hironka, N.; Okayama, D. New Inexpensive Frequency Calibration Service from NIST. Proc., Frequency Control Symp., Baltimore, MD. 1990 June. 107-116.

[12] Levine, Judah; Lombardi, Michael A.; Nelson, Lisa M.; Zhang, Victor S. Technical Reference Manual for NIST Automated Computer Time Service (ACTS). Natl. Inst. Stand. Technol. NISTIR 6611; 2001 July. 89 p.

[13] Shan, Y.; Chua, H. A.; Kyaw, A. M.; and Levine, J. Analysis of Delays in Transmitting Time Code Using an Automated Computer Time Distribution System. Proc., 31st Precise Time and Time Interval Planning and Applications Meeting (PTTI), Dana Point, CA. 1999 December. 323-328.

[14] Anderson, Ross J.; Harris, Georgia L. Specifications and Tolerances for Field Standard Stopwatches. Natl. Inst. Stand. Technol. 105-5; 1997 October. 10 p.

[15] Sullivan, D.B. ACTS sets computer time and much more. Broadcast Engr.; 112-116, 1989 November.

Other References (listed alphabetically by author)

Allan, D.W.; Davis, D.D.; Levine J.; Weiss, M.A.; Hironka, N.; Okayama, D. New Inexpensive Frequency Calibration Service from NIST. Proc., 44th Frequency Control Symp., Baltimore, MD. 1989 May. 107-116.

Levine, J. Authenticating Time and Frequency Signals. IEEE Trans. Ultrasonics, Ferroelectrics, Freq. Cont., Proc. SFC/EFTF, Besancon, France. 1999 April. 304-308.

Levine, J. Precision Synchronization of Computer Network Clocks, in Encyclopedia of Computer Science and Technology, Vol. 37. New York: Marcel Dekker, Inc.; 1997. 281-305.

Levine, J. Authentication, Time-Stamping and Digital Signatures. Proc., 27th Precise Time and Time Interval Planning and Applications Meeting (PTTI), San Diego, CA. 1995 November. 439-445.

Levine, J. The NIST Internet Time Services. Proc., 25th Precise Time and Time Interval Planning and Applications Meeting (PTTI), Marina Del Rey, CA. 1993 November. 505-511.

Levine, J.; Weiss, M.; Davis, D.D.; Allan, D.W.; Sullivan, D.B. The NIST Digital Time Service. Proc., 21st Precise Time and Time Interval Planning and Applications Meeting (PTTI), Redondo Beach, CA. 1989 November. 181-190.

Lombardi, M. A. Keeping Time on Your PC. Byte Magazine. 1993 October. 57-62.

Lombardi, M. A. Traceability in Time and Frequency Metrology. Cal Lab Int. J. Metrology 6: 33-40; 1999 September-October.

Mills, David L. Precision Synchronization of Computer Network Clocks. ACM Comp. Comm. Rev. 24(2): 16 p.; 1994 April.

Mills, David L. On the Chronometry and Metrology of Computer Network Timescales and their Application to the Network Time Protocol. ACM Comp. Comm. Rev. 21(5): 8-17; 1991 October.

Mills, D.L. Internet Time Synchronization: The Network Time Protocol. IEEE Trans. Comm. (10): 1482-1493; 1991 October.

Mills, D.L. Internet Timekeeping Around the Globe. Proc., 29th Precise Time and Time Interval Planning and Applications Meeting (PTTI), Long Beach, CA. 1997 December. 365-371.

Monington, K.E.; Levine J. Principles of Internet Time Synchronization. Proc., 1997 Int. Frequency Control Symp., Orlando, FL. 1997 May. 395-403.