EFFICIENT TIME TRANSFER USING THE INTERNET

Judah Levine
Time and Frequency Division and JILA
National Institute of Standards and Technology
Boulder, Colorado 80305
USA

Abstract - **The National Institute of Standards and Technology operates an ensemble of Internet time servers. The ensemble of servers receives about 450 million time-stamp requests per day (as of May, 2002). This demand is increasing at a compound rate of almost 9% per month, so that improving the efficiency of the time synchronization process is very important. In addition, these requests are not distributed uniformly among the servers, and the busiest servers are nearly saturated during peak periods. Therefore, the capacity of the system as a whole could be increased if the load were distributed more evenly, and we have investigated methods for achieving this active load balancing. We have also developed a number of algorithms based on the Network Time Protocol that try to make the best use of the time stamps and the available network bandwidth. In particular, the algorithms can be configured to trade off accuracy for cost (defined as the network bandwidth and computer cycles needed to realize a specific performance level), so that the many users who do not need the full accuracy of the system can receive satisfactory service at much lower cost.**

Keywords: **Internet time, Network Time Protocol, NTP**

INTRODUCTION

The National Institute of Standards and Technology (NIST) currently operates a network of 14 Internet time servers in the continental United States. In addition to these servers, NIST has collaborated with groups in Singapore and in Japan to install 7 additional identical time servers in these two countries. These servers respond to requests for time in a number of different formats. Most of the requests (currently about 90% of the requests to the US systems) are in the Network Time Protocol (NTP) format. The NTP format is based on the User Datagram Protocol (UDP) and therefore does not involve a large amount of network overhead. In addition, the NTP format and signaling protocol include explicit algorithms for estimating the network delay between the server and the client. Although we continue to support other time formats (such as the "daytime" and "time" protocols), we are strongly encouraging the use of this protocol for these reasons.

The servers in the continental US currently (as of May, 2002) receive about 450 million requests per day for time, and that number is increasing at a compounded rate of about 9% per month. In addition, these requests are not distributed uniformly among the servers, and the busiest servers are nearly saturated at times. In this paper we describe two methods for dealing with this problem. The first involves techniques for actively balancing the load among the servers. This would increase the effective overall capacity of the system by making better use of the hardware we already have installed. The second involves a study of possible changes to the NTP algorithm that might be useful in reducing the number of requests that were needed for a client system to realize a given synchronization accuracy. In addition, the accuracy that is provided by the current implementation of NTP is significantly better than many users require. Therefore, a significant reduction in the number of requests could be realized if the NTP algorithm had a parameter that supported an explicit tradeoff between the accuracy of the clock on the client system and the cost that was needed to realize that accuracy, where cost is measured in terms of network bandwidth and computer cycles.

THE CURRENT DISTRIBUTION OF REQUESTS TO THE NIST SERVERS

The current (as of May, 2002) distribution of the requests to the NIST servers is shown in the following figure. The servers in the Mountain Time zone receive about 45% of all requests, and one of the servers in that zone receives almost half of these, which is 21% of the requests to the entire network. The average load on this one server is about 1100 requests per second, and this part of the load is growing faster than many of the others. The peak load on this server is about a factor of 2.5× larger than the average load, and this server is nearly saturated by these peaks.

TECHNIQUES FOR LOAD BALANCING

In the current configuration, each server has a unique name and ip address. A crude method of balancing the load would be to have several servers share the same name but have different ip addresses. Each time a client requested the ip address that corresponded to this generic name, the name server for the mountain time zone would return a different ip address (either in random or in round-robin order). The main problem with this method is that most implementations of NTP are configured either to query a server directly by its ip address or to request the ip address once and then use it directly on subsequent requests. Thus this method tends to send most of the load to the oldest servers. A second problem is that this method is not dynamic and cannot adjust to a failure of any of the servers in its zone by shifting the load to a nearby system.

A more sophisticated algorithm would use a combination of hardware and software to redirect requests

Distribution of Requests by Location

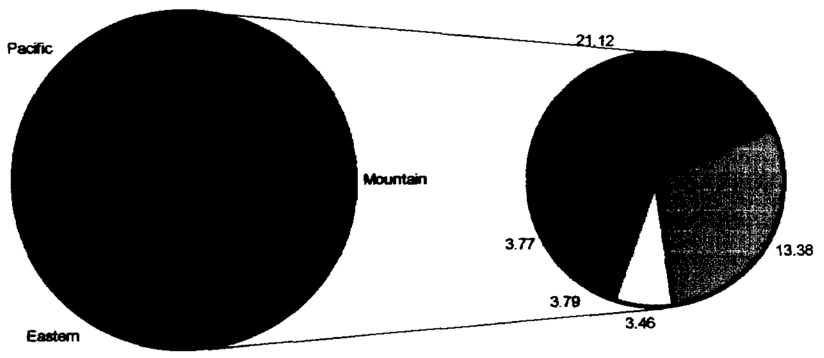

Fig. 1. The distribution of the time requests as a function of the geographical location of the servers. The left pie chart shows the distribution of the load for all of the servers operated by NIST in the US, and the right panel shows the distribution among the servers in the Mountain Time Zone. The percentages in the right chart are the fraction of the total number of requests handled by each server.

even if the client used the actual ip address of the server. In addition, this type of load balancing could automatically and transparently compensate for the failure of any one server by directing the request to another server in the same region. From the point of view of a user, all of the servers in the region would share a common ip address, and a user would not know (or need to know) which physical server actually responded.

The realization of this idea is complicated by the fact that the servers in each of the time zones shown in fig. 1 are not located in one place within that zone. This has the advantage that there is no single point of failure in any region, but it complicates the load-balancing algorithm since the different servers are on different local networks that are independently managed. A conceptual solution to this problem is shown in the following figure. All of the servers share a common name, which might be "mountain-time.nist.gov" and a single ip address that points to one of the hubs. The hubs exchange status and load information in real time, and redirect a request to one of the physical servers, which might be at one of the other sites. In addition to balancing the load dynamically, the system improves the overall reliability of the service by redirecting requests away from a component that has failed. An important aspect of the design is that the number of physical-server computer cycles needed to support the overhead of the load-balancing process must always be less than what would be required for the server to simply reply to the user directly. This requirement is realized by implementing the load balancing algorithm in separate hardware.

This system can be extended in a natural way to include servers in other time zones. In each case, the load balancing would be done in two steps – a system at each site to balance the load among the physical servers there and a zone system to balance the load among the sites in the zone. The global balance would be achieved with a meta-system that exchanged status and load information with the zone controllers. This global load balancing requires additional overhead, and would be justified only if the imbalance among the time zones becomes larger than is currently the case.

We have designed and tested this idea on a small test network in Boulder, and we are awaiting the delivery of additional hardware to complete a full-scale test using all of the servers in the Mountain Time Zone.

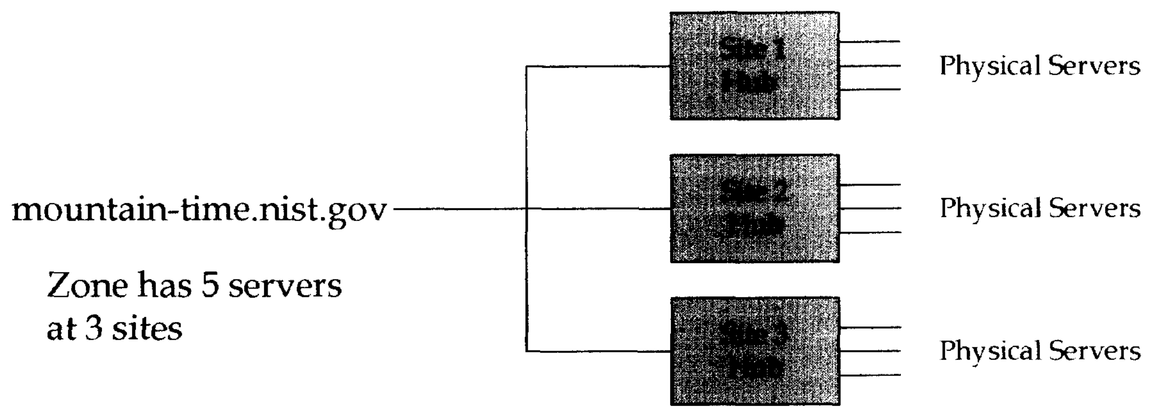

Fig. 2. The design for balancing the load on the time servers in the Mountain time zone. The servers are distributed among three sites, each one of which has a system to balance the load among the physical servers at that site. A global load balancer, implemented in parallel in all of the hubs, is used to balance the load among the sites. Both the hub at each site and the algorithm used to distribute requests among the sites will automatically cope with a failure of an entire site or of a physical server at a site by redirecting the requests to another system.

MODIFYING THE NTP ALGORITHM

The NTP standard defines the format of the messages that are exchanged between a client and a server, and all clients and servers must conform to these specifications if the protocol is to be generally useful. However, other aspects of the client side of the protocol, such as how to choose a server, how to set the interval between queries, and how to use the messages to discipline the local clock are not fundamental to the definition of the protocol, and can be modified without affecting the overall operation of the system.

If we assume that the clock on the NTP server is synchronized to a national time reference such as UTC(NIST) with a negligible time offset, then the accuracy of the overall synchronization process will be limited by three effects: (1) errors in the measurement of the transmission delay between the client and the server, (2) frequency instability of the clock oscillator in the client system and (3) jitter in the measurement of the time difference between the client clock and the server due to interrupt latency, processing delays, and similar effects. It is not too difficult to realize a system in which the standard deviation of the measurement process due to all of these effects is on the order of 10-20 ms, and it is quite common to find systems where the uncertainty is significantly less than these values. This performance is substantially better than many users need, and it is therefore useful to design algorithms that can explicitly trade-off time accuracy (defined as the RMS time difference between the client and the server) and the cost of realizing it (defined in terms of network bandwidth and computer cycles at both ends).

STABILITY OF COMPUTER TIME MEASUREMENTS

Computer clocks generally use quartz-crystal oscillators as a frequency reference. The oscillator generates periodic hardware interrupts, and the computer responds to these interrupts by incrementing a register or memory location. The value added on each interrupt is usually the nominal period of the clock oscillator in microseconds, so that the time register contains the time of day as the number of microseconds since some epoch.

Since the speed of the processor clock is much faster than the frequency of the clock interrupts, the computer typically will perform thousands of operations between each clock interrupt, and there is usually little or no correlation between the micro-state of the system when an interrupt occurs and the micro-state at the time of the previous interrupt. The latency in processing any interrupt is therefore essentially independent of the latency in processing the previous one, so that the time jitter associated with the processing of the interrupts can be modeled as white phase noise. The root-mean-square amplitude is usually on the order of ten microseconds or less.

Using the same sort of reasoning, the process of comparing the time of the clock with the time received over the network from a server can also be characterized as white phase noise with a comparable RMS amplitude. (Even a "slow" Ethernet cable can transmit about 60,000 packets per second, so that the latency in processing a network message cannot be greater than about 15 μs if the system is to work at all.)

The optimum strategy for dealing with white phase noise is to make repetitive measurements and average the results, because the underlying mean time difference is well-defined in this situation. However, this strategy is often not of much use in practical applications. Most applications that use the time of the computer clock associate its value with the occurrence of some external event, such as the reading of a voltage, the pressing of a keyboard character, the clicking of a mouse button or the arrival of an message from some external device. This external event happens once, and averaging the clock data is obviously not possible in these situations. The jitter in these time tags is therefore going to be determined by the typical system latency for a single measurement without averaging. The conclusion is that there is usually not much point in synchronizing a computer clock with an uncertainty of less than a few tens of microseconds, since, even if it could be done, most applications could not make use of that increased accuracy anyway. (Reliable microsecond-level accuracy generally requires special-purpose hardware, which generally includes an explicit connection between the hardware that generates the event and the clock that is going to provide the time-tag that will be associated with it.)

Since hardware jitter in the client is usually not the problem, the real limits to synchronizing computer clocks are usually the frequency stability of the local clock oscillator (which sets the maximum interval between calibration queries) and the uncertainties in measuring the network delay between the client that is to be synchronized and the server that provides the information to do this (which sets the accuracy that can be realized with any single query). The design of the optimum algorithm would balance these two problems – the data received from the remote servers over the network would be used to adjust the local clock if and only if its estimated jitter was less than the free running stability of the local clock oscillator itself. For most wide-area network (including the Internet), the stability of the remote server itself doesn't matter, since the client can see it only through the noisier network.

STABILITY OF COMPUTER CLOCK OSCILLATORS

Figure 3 shows the free-running stability of a typical computer clock oscillator. (The general shape of the curve is typical of most systems. The actual values shown here are from one of the systems that is used as a NIST time server and are better than average.) To perform this analysis, the time of the clock was compared to UTC(NIST) using two different methods. The first method, whose Allan deviation is identified in the figure using "*" characters, was performed using a special-purpose device on the system bus, which received 1 pps signals on a coaxial cable from the NIST clock ensemble.

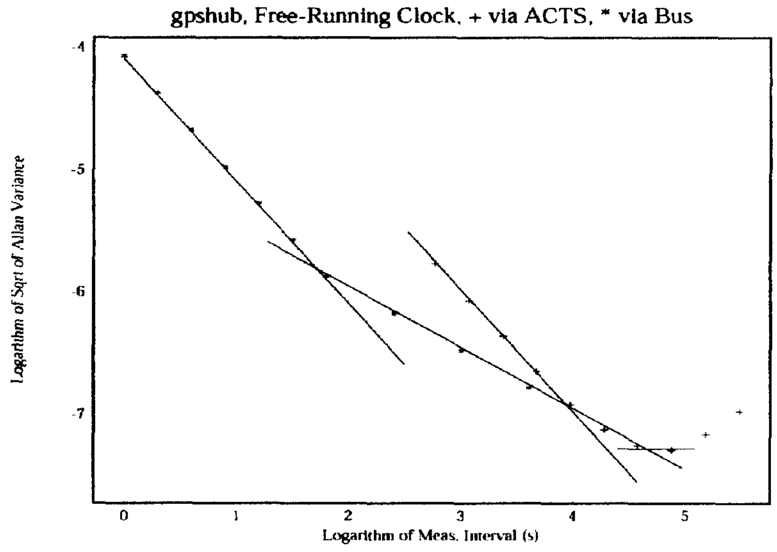

Fig. 3. Allan deviation of clock in computer gpshub measured using two methods as described in the text.

The cable delay was about 0.5 μs and its variation was negligible on the scale of interest here. The second method, whose Allan deviation is identified in the figure using "+" characters, used the NIST ACTS dial-up telephone service to perform the time comparison. The ACTS software on the client system calls the server at NIST using a standard modem connected to the system using one of the serial ports. The delay through the telephone connection is measured by the ACTS server at NIST in real time, and the server adjusts the advance of each message so that it will arrive on-time at the client system. The adjustment algorithm depends on the assumption that the inbound and outbound delays are equal, and this assumption is usually quite accurate for the dial-up telephone system. Even when the inbound and outbound delays through the modems are not quite equal, the asymmetry for a given modem is usually very stable in time, and the reproducibility of the ACTS system itself (using the same hardware from one connection to the next) is usually of order 0.3 ms. (The asymmetry in the inbound and outbound delays may vary by several milliseconds from one brand of modem to another.)

At short times, the noise in either method is well characterized as white phase noise, with the only difference between the two being the level: the bus device has a measurement noise of about 30 μs, while the noise in the ACTS system (including the jitter in processing the characters from the serial port) is about 0.5 ms. Since the overhead in using the ACTS system is dominated by the work needed to establish the telephone connection, it is relatively inexpensive to improve the performance of the ACTS link by averaging several consecutive measurements. However, it is clearly not practical to reduce the noise below about 0.1 ms for a single telephone connection, since the maximum connection time supported by the ACTS system is about 30 s, and the jitter in the mean time difference only improves as the square root of the number of measurements used to compute it.

The shape of the Allan deviation shown in fig. 3 is consistent with what we would expect from a quartz crystal oscillator. At short times, the deviation is characterized by a slope of –1 due to the white phase noise of each measurement method that is used to evaluate the clock. The accuracy is limited by the capabilities of the measurement process, and varying the interval between measurements clearly has no effect on the time accuracy of the synchronization procedure.

The slope of the Allan deviation changes to –0.5 at longer measurement intervals. The break point in the slope occurs when the Allan deviation due to the white phase noise of the measurement process becomes equal to the white frequency noise of the clock oscillator itself. At still longer averaging times (on the order of 1 day) the flicker of frequency floor is reached. The minimum value of the Allan deviation is about 5×10^{-8} at an averaging time of about 1 day.

The stability of this clock is considerably better than the clocks found in cheap PCs and many larger work stations, and this particular hardware is used in the NIST time servers for this reason. Although the general shape of the function and the break-points in its slope are essentially the same for most other systems, the magnitude of the Allan deviation can vary by as much as an order of magnitude between systems from different manufacturers. However, based on our experience with a rather small sample, there is not much variation from system to system of the same type from the same supplier.

STABILITY OF THE NETWORK DELAY

The other contribution to the uncertainty budget for the synchronization process is the jitter in the network delay. In order to estimate this, we used three systems that were independently synchronized to UTC(NIST). The test system was located in our laboratory in Boulder, Colorado, and we measured the delays between this system and two other systems located in Seattle, Washington and Gaithersburg, Maryland. Time packets traveling from Boulder to either of these systems pass through a number of routers and gateways along the way, and the overall path delay is usually dominated by the delays in these network elements.

Fig. 4 below shows the results of these measurements for three different configurations. The data near the bottom of the figure (shown with "+" characters) shows the Allan deviation in a "loopback" configuration. The requests from the client system are returned to the client in this configuration, so that the clock is effectively synchronized to itself. These data estimate the fluctuations in the processing delays within the client system itself. (The data are insensitive to the average value of this processing delay, which is automatically removed in all cases by the measurement algorithm.) As can be seen from the figure, these data are well characterized as white phase noise at all measurement intervals. The amplitude is about 5 μs, which is about 2% of the total processing delay, which was about 240 μs. This level of white phase noise can be realized between two separate systems, but only if the client and server are on a short, dedicated network segment with no other traffic. (Although both the processing delay and its jitter can be reduced to some extent by using faster hardware, much of the recent increase in the hardware speed of general-purpose computers has been balanced by an increase in the complexity of the operating system software that controls them, especially when the operating system is based on a graphical user interface.)

The upper plots show the Allan deviation of the measurements between a client and a server on the same local network (shown by "X") and between a client and a server that is about 1200 km away (shown by "*"). We have superimposed a line with a slope of –0.5 on the upper two plots. This line is copied from fig. 3, and is the estimate of the statistical performance of the clock oscillator in the client system that we deduced from the first set of experiments.

The goal of the synchronization procedure is to improve the stability and accuracy of the local clock. Therefore, adjustments of its time or frequency should be limited to those measurement intervals where the calibration data add information, that is, where the noise of the received data is less than that of the free-running local oscillator. Although the detailed results of fig. 4 are a function of the particular systems and network that we used, the general characteristics are essentially the same for all systems. In particular, at short intervals, the time dispersion due to the frequency noise of the local clock oscillator is almost always less than the noise of the time server as seen through the noisy channel, so that some sort of averaging of the calibration messages is usually required to realize an optimum design.

Using the results shown in the figure, if we plan to synchronize the system using a time server that is connected to the same local area network, then the variance of the calibration signals received over the network will fall below the variance due to the frequency noise of the clock for averaging times longer than about 150 s. This is the minimum measurement interval that we should use in this situation. If we use a shorter interval then the noise in the network delay will degrade the inherent stability of the local clock oscillator.

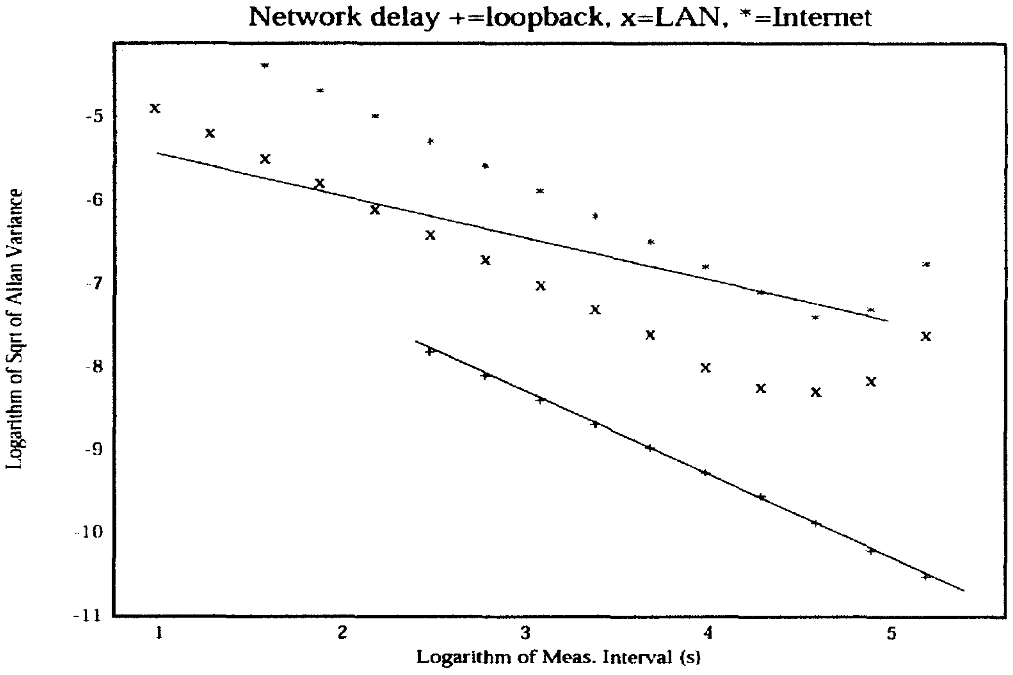

Fig. 4. The Allan deviation of the network delay measured in three different configurations. The deviation shown using (+) shows a "loopback" measurement, the plot using (X) shows measurements on a local network, and the plot using (*) shows measurements to a remote system on the Internet.

If, on the other hand, we plan to use the remote server to synchronize the clock of the client system, then it is important to note that the larger variance in the data from the remote server imply that the free-running stability of the local clock is better than the stability of the server seen through the network out to an averaging time of about 5.3 hours, and that using the shorter averaging time found in the previous paragraph degrades the stability of the local clock by introducing network noise. This conclusion may be counter-intuitive; when the data are noisy there is a temptation to shorten the interval between measurements and increase the number of them, whereas this analysis shows that exactly the opposite should be done because the local clock has a smaller time dispersion than the remote server seen through the network. The best thing that we could do at short intervals is to leave the clock alone. (This understanding comes from the fact that we have independently evaluated the free-running stability of the oscillator in the client computer, so that we can distinguish between the contribution to the measured time dispersion due to its frequency noise and the component due to the jitter in the network delay. This ability to separate the variance is an important prerequisite for implementing the techniques in this paper.)

This optimum interval of several hours between calibration cycles is much longer than the longest interval that is used by standard NTP clients, because the NTP client is generally implemented using a phase-lock loop rather than the frequency-based algorithm that is implied by these analyses. As we can see from this analysis, the phase lock loop is unlikely to be optimum in this situation, because of the large variance in the network noise at short periods. Furthermore, many implementations of the NTP algorithm do not make explicit use of the stability of the local clock oscillator. (There is often no simple way for an average user to remedy this deficiency, since evaluating the stability of the local clock oscillator requires a connection to a server using a channel whose delay fluctuations are small compared to the time jitter due to the frequency dispersion of the clock under test.)

Even though the network is quite noisy, the performance of this loop using an interval of 5.3 hours between measurements would be of order $\sqrt{2}\sigma_y(19000)19000 \cong 0.002$ s, and this level of performance would be realized with only about 4 or 5 measurements per day. This is a significant improvement in performance over what can be realized with standard NTP, and this performance is realized with far fewer measurements that are required by the standard NTP software. The main reason for this improvement is the use of a frequency loop and the recognition that the free-running stability of the local clock oscillator at short averaging times is much better than the stability of the remote server as seen through the network.

PROCESSING COST

In addition to providing an estimate of the uncertainty that will be realized using a given averaging interval, it is also possible to use these results to estimate the minimum averaging interval that would be necessary to realize a given level of performance. This is a useful feature, since many applications do not need millisecond-level accuracy, and it would be very useful to increase the interval between calibrations to the point where the resulting performance just satisfied the requirements. In order to evaluate this trade-off, we must examine how the synchronization accuracy and cost of the process vary with the time between calibration cycles between the client and the server.

Given the interval between calibration cycles, τ, the cost of a synchronization procedure is inversely proportional to this interval, assuming that each calibration cycle requires the same amount of network bandwidth and the same number of computer cycles in the client and the server. On the other hand, the synchronization uncertainty using a time interval between calibrations of τ will be proportional to $\tau \times \sigma_y(\tau)$. Therefore, whenever the slope of the Allan deviation as a function of averaging time is negative, increasing the interval between calibration cycles reduces the cost of the procedure faster than it degrades the uncertainty, so that this is a favorable choice as long as the resulting uncertainty meets the requirements of the application. Conversely, decreasing the interval between calibration cycles increases the cost faster than it improves the performance over most of the range of averaging times shown in the figure. Even when the slope of the Allan deviation as a function of averaging time increases to 0, the uncertainty increases only at the same rate as the cost decreases. Although this is not as favorable as the preceding case, the resulting uncertainty is often adequate anyway. Using the results of fig. 2, for example, a client could be synchronized to a server with an RMS uncertainty of only a few milliseconds using just one calibration per day. This level of performance would satisfy almost all of our current users, using only a tiny fraction of the number of calibrations that are currently used by most NTP clients.

A similar argument suggests that routinely contacting more than one server is not optimum. Using the most optimistic assumption that the difference between the two servers can be characterized as white phase noise, contacting two servers increases the cost of the process by a factor of 2 but decreases the uncertainty by only $\sqrt{2}$. Although contacting a second server may have some advantages in outlier detection, outliers are relatively rare events, and the second server should only be contacted when a preliminary analyses of the measurement from the first server suggests that the results are not consistent with the modeled stability of the local clock and the network.

Although the Allan deviation of the clock oscillator itself is based on its hardware design and is therefore a constant, the statistical performance of the network is a dynamic quantity that must be continuously re-evaluated. Our prototype software does this every day, and re-computes the appropriate measurement interval based on these revised estimates of the Allan deviation. The detailed design of this software is described in [1].

Since the jitter in measuring the time difference can be characterized approximately as white phase noise, it is also possible to improve the performance of the synchronization algorithm by replacing each time difference measurement with a group of many measurements made rapidly enough so that the clock and the network do not change. Even if the assumption that both the delay through the network and the parameters of the clock oscillator do not change during the group of measurements, the cost of doing this increases linearly with the number of measurements in the group, while the standard deviation of the mean improves only as the square root of this number (at best). Thus the cost/benefit ratio is always unfavorable, and a given incremental improvement becomes very expensive very quickly once the number of measurements in the group increases beyond 4 or 5. Nevertheless, for a fixed number of measurements, grouping them in this way is usually a better strategy than spreading them out uniformly in time, since the fluctuations in the frequency of the clock oscillator have little effect over a short time interval so that dispersion in a group is more likely to be characterized as white phase noise. The mean of the measured time differences is a robust estimate in this situation.

CONCLUSIONS

The load on the NIST Internet time servers is increasing by 9% per month, and we have investigated methods that can be used to address this increasing demand without a corresponding increase in the number of servers. We have described two methods that show promise for achieving this goal. The first would use active load balancing hardware to improve the reliability of the system and to equalize the number of requests processed by each server. The second would increase the effective capacity of the service by improvements to the NTP algorithm that would permit users to decrease the number of requests from any client system. These improvements should allow NIST to continue to satisfy the exponentially increasing demand for Internet time service without a significant increase in the operating costs for the foreseeable future.

References

[1] Judah Levine, "Time synchronization over the Internet using an adaptive frequency-locked loop," IEEE Trans. UFFC, Vol. 46, pp. 888-896, 1999.