# A Comparison of Certain Timekeeping Systems and the Network Time Protocol[1]

David L. Mills
Electrical Engineering Department
University of Delaware
12 May 1991

**Abstract**

This report describes and compares three timekeeping systems in objectives, architecture and design. These systems are: Digital Time Synchronization Service (DTSS), Probabilistic Clock Synchronization (PCS) and Network Time Protocol (NTP), which have been submitted to the standards bodies as the basis of an international standard. Each of the three can be used to synchronize local clocks in a computer network with distributed and diversified services.

Author's note: This document is a preliminary draft and is subject to change before publication in final form. It is intended for informational purposes and used only in connection with professional activities. Please do not cite this document in any publication and do not redistribute it without this notice.

## 1. Introduction

This report discusses the scope, application and function of three network time-synchronization architectures and protocols with respect to possible standards-making activities. These include the Distributed Time Synchronization Service (DTSS) described in [DEC89], the Probabilistic Clock Synchronization (PCS) described in [CRI89b] and the Network Time Protocol (NTP) described in [MIL90b]. A document [ISO90] has been submitted to the ISO Working Group JTC1/SC21/WG7 proposing DTSS for consideration. Another document [CCI90] has been submitted to the CCITT Study Group VII proposing NTP for consideration. In addition, it is expected that PCS will also become a candidate for consideration. It may be that one of these three architectures will eventually be selected for ISO/CCITT standardization or some other architecture synthesized from one or more of them.

In most developed nations time is considered a metricated service, disseminated by national means, coordinated by international agreement and intended for ubiquitous civil access. Today, the timekeeing systems of most countries are based on Coordinated Universal Time (UTC), which is maintained by cooperative agreement using astronomical observations. Users of computer network applications expect local clocks to be synchronous with UTC with acceptable accuracy, stability and reliability.

As will be evident from later discussion, there are wide variations in the perceived requirements and expectations of the three timekeeping systems considered in this report; however, each of these systems shares the common goal of providing accurate, stable and reliable local clocks. In the following the *accuracy* of a clock is how well its time compares with a designated reference clock or national standard, the *stability* is how well it can maintain a constant frequency and the *precision* is to what degree time can be resolved in a particular timekeeping system. The *time offset* of two clocks is the time difference between them, while the *frequency offset* is the frequency difference between them. The *reliability* of a timekeeping system is the fraction of the time it can be kept operating and providing correct time with respect to stated accuracy and stability tolerances.

## 2. Time Synchronization Systems

An important feature of a computer network providing distributed services is the capability to synchronize the local clocks of the various processors in the network. This capability can be provided by a *timekeeping system* consisting of *time servers* (called *masters* in PCS) in the form of dedicated or shared processors which exchange timing messages between themselves and provide timing information to the *clients* (called *slaves* in PCS) throughout the network population. Many applications needing time synchronization also need to coordinate local time with standard time as disseminated from national standards laboratories via radio, telephone or satellite. This can be done by connecting some servers to a designated source of standard time, called a *primary clock* (*time provider* in DTSS). The servers connected to primary clocks are designated *primary servers*; others that may depend on them are designated *secondary servers*. The

1 Sponsored by: Defense Advanced Research Projects Agency under NASA Ames Research Center contract number NAG 2-638 and National Science Foundation grant number NCR-89-13623.

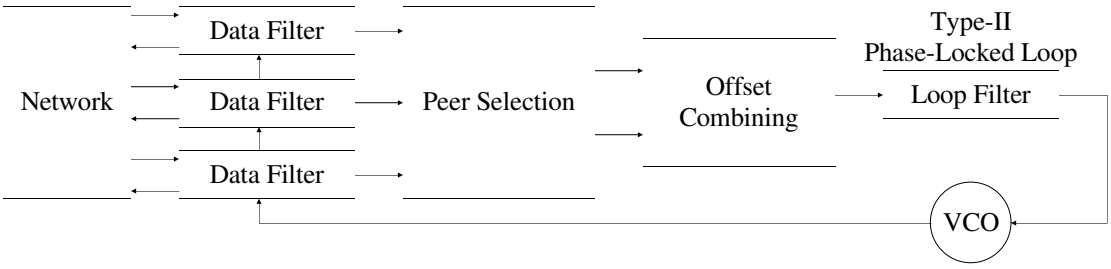


Figure 2. NTP System Model

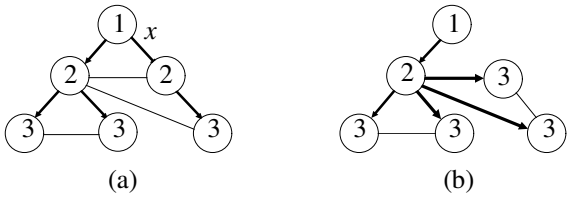


Figure 1. Subnet Synchronization

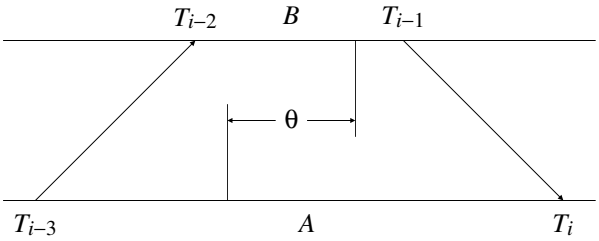


Figure 3. Delay and Offset Computation

secondary servers are clients of the primary servers and may themselves serve a client population including other secondary servers.

It is generally considered impractical to equip every processor with a primary clock such as a radio timecode receiver or telephone modem; therefore, a timekeeping system will usually employ one or more primary clocks and provide synchronization using a *synchronization protocol* such as DTSS, PCS or NTP. Using the protocol, servers and clients exchange timing messages at intervals depending on the accuracy required. Information in these messages can also be used to determine reachability between the servers and to organize the set of servers and clients as a hierarchical *synchronization subnet*.

In all three timekeeping systems one or more primary servers synchronize directly to national standards using methods such as described in [MIL91b]. Each secondary server or client selects a set of servers or *peers* from within the subnet population on the basis of accuracy, stability and reliability. Secondary time servers (called *couriers* in DTSS) synchronize to the primary servers and possibly others in the synchronization subnet. Clients (called *clerks* in DTSS) synchronize to possibly several servers located on the same or different LANs. In order to assure reliability, at least three peers operating over disjoint network paths are required. In PCS and NTP the selection of peers is engineered prior to operation; while, in DTSS the selection can be semi-automated.

A typical synchronization subnet is shown in Figure 1a, in which the nodes represent subnet servers, with normal level numbers determined by the hop count from the root (stratum 1), and the heavy lines the active synchronization paths and direction of timing information flow. The light lines represent backup synchronization paths where timing and reachability information is exchanged, but not necessarily used to synchronize the local clocks. Figure 1b shows the same subnet, but with the line marked *x* out of service. The subnet has re-configured itself automatically to use backup paths, with the result that one of the servers has dropped from level 2 to level 3. In the present designs the number of levels is limited to three for DTSS, two for PCS and fifteen for NTP.

A timekeeping system maintaines synchronization by the exchange of timestamps and related information. The three systems DTSS, PCS and NTP differ somewhat in how they interact between peers and how the offset and related information are determined. Following is a summary of the operations performed by each of the three systems.

### 2.1. NTP Synchronization Model

Figure 2 shows the overall organization of the NTP synchronization model. Timestamps and related data are exchanged between a peer and possibly several other subnet peers to determine individual clock offsets and roundtrip delays. The periods between exchanges are determined as a function of required accuracy and error bounds calculated from the data. A set of state variables, including the most recently determined offset and delay, is maintained separately for each peer and updated as each message from that peer is received.

Figure 3 shows how NTP timestamps are numbered and exchanged between peers *A* and *B*. Let $T_i$, $T_{i-1}$, $T_{i-2}$, $T_{i-3}$ be the values of the four most recent timestamps as shown. Note that $T_i$ and $T_{i-3}$ are determined from the *A* local clock, while $T_{i-1}$ and $T_{i-2}$ are determined from the

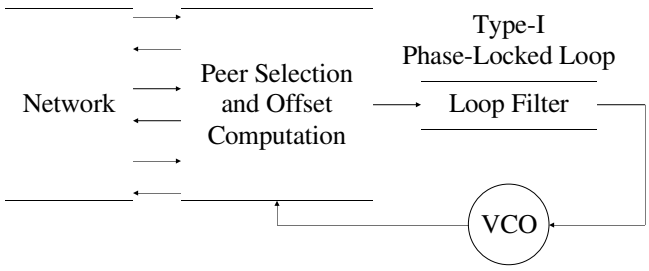

Figure 4. DTSS and PCS System Model

*B* local clock. For convenience, let $a = T_{i-2} - T_{i-3}$ and $b = T_{i-1} - T_i$.

If the effects of clock reading errors, frequency offsets and non-reciprocal delays between *A* and *B* are neglected, the clock offset θ and roundtrip delay δ of *B* relative to *A* at the end of the exchange are

$$\theta = \frac{a+b}{2} \quad \text{and} \quad \delta = a - b.$$

In addition to θ and δ the peer also calculates the *dispersion* ε from the known local-clock reading error and frequency tolerance. The dispersion is used later to determine the range over which θ can be assumed valid.

Each NTP message includes three timestamps $T_{i-1}$, $T_{i-2}$ and $T_{i-3}$, while the fourth timestamp $T_i$ is determined upon arrival of the message. Thus, both peers can independently calculate θ and δ using a single bidirectional message stream and cross-check each other, as well as quickly reconfigure should other peers or network paths fail. Since the timestamps received by one peer are included in the next message sent to the other, the scheme provides inherent protection against message loss or duplication. Among its advantages are that the transmission time and received message order are unimportant and that reliable delivery is not required.

The computed offsets for each peer are processed by the data-filter algorithm shown in Figure 2 to reduce incidental timing noise. As described in [MIL90b], this algorithm selects from among the last several samples the one with minimum δ and presents the associated θ as the output. The peer-selection algorithm determines from among all peers a suitable subset of peers capable of providing the most accurate and trustworthy time. In NTP this is done using a cascade of two subalgorithms, one an intersection algorithm modified from [MAR85] to detect and discard faulty clocks and the other a clustering algorithm [MIL91a] to improve expected accuracy.

The θ, δ and ε values for the peers selected by the subalgorithms are combined using a weighted-average algorithm [JON83]. The results are processed to produce the *local-clock offset* and a *confidence interval* over which this offset can be considered a valid estimator of the time at the primary server(s). The local-clock offset is then used to adjust the phase and frequency of the local clock as described later, while the confidence interval is provided to the user interface.

### 2.2. DTSS Model

Figure 4 shows the overall organization of the DTSS synchronization model. In periodic rounds a client selects at random a small set of servers from among a directory list and solicits the time and related data from each. The periods between rounds are determined as a function of required accuracy and the error bounds calculated from the data. In DTSS state is not preserved between rounds.

Referring to Figure 3, server *B* returns the $T_{i-2}$ timestamp and the quantity $v = T_{i-1} - T_{i-2}$, representing the holding time at the server. Upon receipt of the reply, client *A* calculates the quantity $u = T_i - T_{i-3}$. In addition, the server provides a quantity called the *inaccuracy interval* $I_{i-2}$, which represents the worst-case error implicit in the $T_{i-2}$ timestamp.

If the effects of clock reading errors, frequency offsets and non-reciprocal delays between *A* and *B* are neglected, the time $T_{i-3}(B)$ and inaccuracy interval $I_{i-3}(B)$ of *B* relative to *A* at the beginning of the exchange are

$$T_{i-3}(B) = T_{i-2} + \frac{v-u}{2} \quad \text{and} \quad I_{i-3}(B) = I_{i-2} + \frac{u-v}{2}.$$

Subtracting $T_{i-3}$ from the first equation and rearranging terms yields

$$\frac{T_{i-2} - T_{i-3}}{2} + \frac{T_{i-1} - T_i}{2} = \frac{a+b}{2} = \theta,$$

while substitution in the second equation yields

$$I_{i-3} = I_{i-2} + \frac{T_{i-2} - T_{i-3}}{2} - \frac{T_{i-1} - T_i}{2}$$
$$= I_{i-2} + \frac{a-b}{2} = I_{i-2} + \frac{\delta}{2}.$$

It is apparent that, while expressed differently, both DTSS and NTP perform similar calculations; although, as will be seen, their treatment of the error budget is

somewhat different. However, the DTSS scheme is not symmetric and does not utilize the data-filter algorithm, clustering algorithm or combining algorithm found in NTP.

In DTSS the peer-selection algorithm shown in Figure 4 determines from the set of servers surveyed the local-clock offset and confidence interval using the intersection algorithm described in [MAR85]. These values are then used to adjust the phase of the local clock as described later, while the confidence interval is provided to the user interface.

### 2.3. PCS Model

PCS operates in a manner similar to DTSS (Figure 4), in that state is not preserved between rounds and the data-filter, clustering and combining algorithms found in NTP are not used. In each round a client sends a message including $T_{i-3}$ to a single server, which returns the message with $T_{i-2}$ appended. By assumption, the message is returned immediately, so that $T_{i-2} = T_{i-1}$. The client can then calculate the clock offset and roundtrip delay in the same way as NTP. As in NTP and DTSS the periods between rounds are determined as a function of required accuracy and the error bounds calculated from the data.

The distinguishing feature of PCS is that the user of the time service is expected to provide a fixed error tolerance, with the expectation that the service will either return a correct time within this tolerance or will abandon the effort and report failure. A round begins when a client sends a request to a server. For each reply the client calculates $\theta$, $\delta$ and $\varepsilon$ in a manner similar to NTP, and thus the local-clock offset and confidence interval. If the confidence interval exceeds the error tolerance, the offset value is used to adjust the local clock and the client reports success. If not, or in case of timeout, the client sends the request again. If no valid reply arrives before a specified number of retransmissions, the client reports failure.

### 2.4. Local Clock Models

The result of each of the three timekeeping protocols is to produce a local-clock offset correction and confidence interval over which this offset can be considered valid. In DTSS and PCS (without the self-adjusting feature) this correction is used to adjust the phase of the local clock so that, following an amortization interval, the time of the local clock will equal, within a computed error bound, the time of the selected server(s). This is done using a feature of the operating system which allows a small (signed) adjustment (called *tickadj* in some Unix systems) to be added to the local clock at each tick for an interval calculated from the value of the correction. This amounts to the activation of a system-specific frequency offset for the period of amortization.

In NTP the local clock is modelled as a component of a control-feedback system called a phase-lock loop (PLL), as shown in Figure 2. In the PLL the combined effects of the filtering, selection and combining operations are to produce a phase-correction term, which is processed by the loop filter to control the local clock, which functions as a voltage-controlled oscillator (VCO). The VCO furnishes the timing (phase) reference to produce the timestamps used in all timing calculations. The NTP local-clock model has been implemented on Unix systems using the same operating-system feature described above, but with the addition of a facility which periodically adds to the local clock a (signed) correction to account for the calculated frequency error.

While it might appear that the DTSS/PCS and NTP local-clock models are quite different, they are in fact both based on well known control-theoretic principles and both are represented by variations of phase-locked loops. The various systems incorporate different PLL models based on assumptions of the accuracy and stability required. A type-I PLL, as used in DTSS and PCS, can correct for time offset, but not frequency offset; therefore, it requires periodic corrections depending on the intrinsic frequency offset and accuracy required. Such PLLs are unconditionally stable for any choice of loop parameters, although they may display a considerable degree of phase noise or timing jitter. This might be of some concern in subnets including multiple levels of hierarchy.

A type-II PLL, as used in NTP, can correct for both time and frequency offsets, but requires an initial interval or *training* in order to stabilize the frequency-offset estimate. While not identified explicitly as such, the PCS model with the self-adjusting feature is also a type-II PLL. A type-II PLL measures the intrinsic frequency offset of each local clock individually and introduces automatic corrections as required. However, type-II PLLs may become unstable for some choices of loop parameters, so require engineering to specified convergence criteria. In practice, the allowable parameter tolerances are quite generous and the criteria reliably predicted [MIL90b]. A type-II PLL can significantly reduce the message rate and increase the accuracy during possibly lengthy periods when contact with all servers is lost. However, real clocks exhibit some degree of instability as the result of aging, environmental changes, etc., so the improvement in performance using a type-II PLL is limited. These issues are discussed in further detail in a later section of this document.

### 2.5. Error Analysis

Real systems are subject to stochastic errors due to local clock resolution and stability. If these errors can be bounded by design and manufacture to specific tolerances, then it is possible to calculate their affect on the confidence interval. All three timekeeping systems include provisions to calculate these bounds as a byproduct of normal synchronization activities and include their contribution in the resulting confidence intervals provided to the user interface.

In all three timekeeping systems the error budget is calculated from increments of the form

$$\varepsilon = \rho + \varphi\tau,$$

where $\rho$ is the error in reading the clock, $\varphi$ the frequency tolerance of the clock and $\tau$ the interval since the clock was last read. Since the error budget always increases with time, it is possible to use it to determine when the next time-request message should be sent, which is in fact done in DTSS and PCS. Since DTSS and NTP are based on a hierarchical subnet topology, provisions are necessary to include in the error budget the effects of all increments accumulated on the synchronization paths to the primary clock, including the effect of all intervening servers, local clocks and the primary clock itself. This portion of the error budget is called the dispersion in NTP.

In addition to the error increments described above, it is necessary to include in the error budget contributions that can result from non-reciprocal delays on the network paths between peers. It is a simple exercise to calculate bounds on these errors as a function of measured delay. The true offset of *B* relative to *A* is called $\theta$ in Figure 3. Let *x* denote the actual delay between the departure of a message from *A* and its arrival at *B*. Therefore, $x + \theta = T_{i-2} - T_{i-3} \equiv a$. Since *x* must be positive in our universe, $x = a - \theta \geq 0$, which requires $\theta \leq a$. A similar argument requires that $b \leq \theta$, so surely $b \leq \theta \leq a$. This inequality can also be expressed

$$b = \frac{a+b}{2} - \frac{a-b}{2} \leq \theta \leq \frac{a+b}{2} + \frac{a-b}{2} = a\,,$$

which is equivalent to

$$\theta - \frac{\delta}{2} \leq \hat{\theta} \leq \theta + \frac{\delta}{2}\,.$$

In other words, the true clock offset $\hat{\theta}$ must lie in an interval of size equal to the measured delay and centered about the measured offset. All three timekeeping protocols carry out this calculation and include this interval in the error budget. The total of all contributions to the error budget is interpreted as the confidence interval.

One of the requirements placed on NTP is the ability to calculate not only the clock offset, but the roundtrip delay and dispersion to each peer. This requirement arises both from the perceived user need to control the departure of a message to arrive at a peer at a designated time, which is necessary for multi-media conferencing and other real-time, distributed, synchronized applications, as well as the perceived user need for an estimate of expected accuracy, as well as guaranteed accuracy. For these reasons the delay contributions and the dispersion contributions to the error budget are kept separate and accumulate separately along the synchronization path to the primary clock.

## 3. Issues and Discussion

The preceding overview should provide some insight on the architectures, protocols and algorithms adopted by the DTSS, PCS and NTP timekeeping systems. However, there are important issues which characterize the design approach in the three systems which need to be addressed in further detail. The following sections address some of the most important of these.

### 3.1. Frequency Compensation

The NTP local-clock model includes the capability to estimate the intrinsic frequency of the local clock and compensate for its error with respect to standard frequency, as determined from the synchronization subnet. This section contains an overview of the issues and rationale for the inclusion of this feature. It should be pointed out that nothing in the NTP local-clock model precludes its adaptation to other timekeeping systems.

NTP uses a type-II PLL designed to stabilize time and frequency offset and automatically adjust the message rate and error bounds based on observed timing noise and clock stability. The various design parameters were determined using a mathematical model and verified both by simulation and measurement in several implementations. While not identified explicitly as such, the PCS self-adjusting, logical-clock feature is in fact a type-II PLL. The DTSS design uses a type-I PLL stating as rationale (private communication) its increased complexity and vulnerability to mis-implementation.

An oscillator is characterized by its frequency tolerance and stability. A tolerance specification is usually in terms of a maximum frequency deviation with respect to a calibrated source. Typical values range from $10^{-4}$ for an uncompensated quartz-crystal oscillator to $10^{-12}$ for a cesium-beam oscillator. A stability specification is usually in terms of a maximum frequency change over a specified interval. Typical values range from $10^{-6}$ per day for an quartz oscillator under room-temperature conditions to $10^{-12}$ per year for a cesium oscillator. However, quartz oscillators also exhibit an aging effect which

varies from unit to unit and can result in a long term gradual frequency change as much as $10^{-5}$ per year. In addition, quartz oscillators without temperature compensation or control exhibit frequency variations dependent on ambient temperature of about $1^{0-6}$ per degree Celsius.

The main reason to be concerned about tolerance and stability is the message rate necessary to keep a local clock within a specified accuracy. For instance, if a local clock is to be synchronized to within a millisecond and has an oscillator with a frequency offset of $10^{-4}$, it must be updated at least once every ten seconds. If this oscillator is allowed to run for a day without outside correction, it will be in error by almost ten seconds. On the other hand, if the intrinsic frequency of each oscillator can be estimated and corrected by some means, the intervals between corrections can be considerably increased. This feature has considerable practical benefits in cases where servers dispense time to several hundred clients, as is now the case with many Internet NTP primary servers.

While there are considerable advantages in using frequency estimates, there are limits imposed by the intrinsic stability of the local-clock oscillator, which is usually not temperature compensated. Measurements made with workstations and mainframe computers located in typical office environments show some oscillators with intrinsic frequency errors as high as $10^{-4}$, but with stability after frequency compensation as low as $10^{-7}$ per day; however, these data vary somewhat between various equipments and environments. With accurate frequency compensation and stabilities in this order, message rates can in principle be reduced to about one in about three hours to maintain millisecond accuracy.

Considerable attention was paid in the NTP design to the issue of how to optimize the performance in the face of widely varying tolerance and stability specifications without requiring specific design configuration for each individual oscillator. The particular design adopted, called an adaptive-parameter, type-II, phase-lock loop has the capability to automatically sense the in-operation stability regime of the local oscillator and then adjust the PLL characteristics (bandwidth) and message intervals accordingly. The design has been tested using many types of equipment using compensated and uncompensated quartz oscillators, as well as precision laboratory standards with stabilities ranging to $10^{-12}$ per day.

### 3.2. Reliability Expectations

Of major concern in the dissemination of timekeeping information is the overwhelming importance of reliability. Many papers and articles have been written on this issue with profound theoretical and practical implications. Issues in hardware, software and hybrid hardware/software methods for synchronizing clocks in a distributed timekeeping system where some clocks may

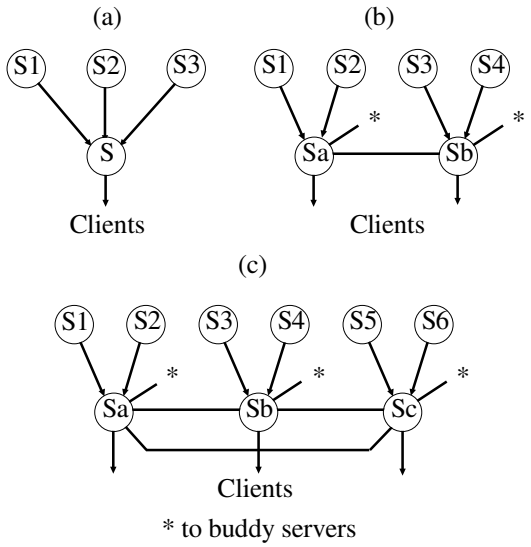


Figure 5. Subnet Topologies

exhibit faults of one kind or another, including Byzantine faults, are discussed in [RAM90] and references cited there. Typical methods are based on a voting procedure involving a number of peers (at least three), together with an multi-round algorithm that seeks a majority of them according to some criterion.

For the purposes of this discussion, the reliability of a timekeeping system refers to its ability to sustain peer connectivity and correct synchronization in the face of misbehaving servers or outages on the links between them. Both DTSS, PCS and NTP explicitly address these issues using the principles of redundancy and diversity. A highly redundant system employs multiple servers at each level of the hierarchy, while a highly diverse system employs multiple disjoint peer paths with few common points of failure. Experience in the Internet shows that these features are necessary and vital in order to provide accurate and trusted time.

However, along with the issues of redundancy and diversity go the issues of how to make efficient use of the multiple assets required and here the proposed systems differ. In the DTSS model clients on a LAN or extended LAN send periodic requests to a number of local time servers randomly selected among those registered in a well-known directory service. Local time servers periodically exchange timing information with each other and, optionally, with global time servers presumably in an extended WAN. If all local servers are lost, a client can directly poll a member of a configured global set.

In NTP peers of the synchronization subnet exchange messages over paths which are independently configured. This establishes the topology of the subnet over which messages are sent continuously, although usually

at relatively long intervals. Using a distance measure based on maximum-likelihood estimates of roundtrip delay and total error accumulated at each level (stratum) from the primary server, a subset of presumed accurate and reliable peers is selected and their readings combined as a weighted average. While the subnet topology must be engineered on the basis of anticipated physical interconnectivity, the actual topology formed by the system results in the lowest distance and thus lowest expected error at each level.

While the configuration engineering required in NTP can be burdensome, it is vital to the robustness of the service, especially in the unmanaged Internet where time servers are frequently operated on a volunteer basis. Figure 5 shows some of the configurations frequently employed. Figure 4a shows a configuration used by a client or a local server with few clients. The local server S peers with three remote servers S1-S3 (primary or secondary) at the next lower stratum, any one of which can fault without effecting the time delivered by S.

Figure 4b shows a configuration used by a campus with a sizable population of dependent servers and clients. Each of two servers Sa and Sb peers with two out of four distinct servers S1-S4 at the next lower stratum and, in addition, each other and one "buddy" server at the same stratum and located in a different administrative domain. This configuration can tolerate multiple faults of S1-S6, but requires Sa and Sb to remain faithful. The buddy server provides a third source of "outside" time to help insulate the campus time from local server software maintenance faults.

For the most reliable service in large campus complexes configurations such as shown in Figure 4c are required. This consists of at least three local servers Sa-Sc, each of which peer with two out of six remote (usually primary) servers S1-S6, plus each of the other two local servers plus a buddy server. Dependent servers and clients peer with all three local servers. This configuration can survive multiple faults of the remote and buddy servers, as well as a fault of one of the local servers.

The configuration adopted for the Internet primary (stratum-1) servers is completely connected. Each of over a dozen primary servers peers with all of the others. If the primary clock at a particular server fails in a detectable way, that server continues operation at stratum-2 synchronized with its neighbors. If any combination of three or fewer clocks or servers displays out of tolerance or turns Byzantine, the remainder will vote them faulty and disregard their indications.

The design of the voting procedure is obviously a critical issue affecting the success of the reliability strategy. The approach followed in DTSS utilizes an algorithm due to Marzullo and Owicki [MAR85], in which an interval called the inaccuracy interval is associated with the apparent clock value for each peer. The algorithm computes the intersection of these intervals in order to find the smallest interval containing the apparently correct time. In earlier versions of NTP a probabilistic approach was taken based on maximum-liklihood methodology familiar in communications systems design. In simulation studies with the Marzullo algorithm and actual offset data collected over particularly troublesome Internet paths, the algorithm proved to be effective, although accuracy suffered considerably. In NTP Version 3 the Marzullo algorithm is used in tandem with the original maximum-likelihood clustering algorithm, with result that the original accuracy is preserved.

Of concern to the early users of NTP (Kerberos, part of Project Athena at MIT) was the vulnerability of the timekeeping system to hostile attack, since incorrect time could result in denial of service (premature key expiration, for example). An extensive security analysis by the Privacy and Security Research Group under the auspices of the Internet Activities Board concluded that it was not possible to secure NTP against protocol attack other than through use of cryptographic authentication [BIS90]. This feature was subsequently introduced in NTP and is now in regular operation for critical and potentially high-risk servers.

In principle, cryptographic authentication could be introduced in other timekeeping systems as well, including DTSS and PCS, either as a component of the protocol or, preferably as a component that can be used by any service on request. It would not seem possible to safely deploy a ubiquitous, multinational, distributed timekeeping system or set of interoperable timekeeping systems without this feature. It should be noted that cryptographic techniques are computationally intensive, so that special care has to be taken to preserve the accuracy of the synchronization service.

### 3.3. Accuracy Expectations

In many, perhaps most, distributed applications requiring synchronized local clocks, accuracies in the order of seconds are acceptable. Applications where inconsistent state in the network can be tolerated for such periods include distributed archiving and electronic mail. However, there are many others where accuracies in the order of milliseconds are required, such as transaction journaling, distributed software and hardware maintenance, real-time conferencing and long-baseline scientific experiments. There are a few others where accuracies in the order of nanoseconds are required, but these applications typically rely on special-purpose time-transfer networks, usually via satellite.

Probably few would dispute the ranking of various applications in the order of increasing accuracy require-

ments, but there may be some disagreement on where to draw a reasonable line between those that would be considered feasible using a shared, global packet-switched medium with stochastic delays and those that require a dedicated medium with predictable delays. With today's technology using 10-Mbps LANs interconnected by 1.5-Mbps WANs, the line might be drawn in the low milliseconds [MIL90a]; however, considering the recent High Performance Computer and Communication Initiative, which could lead to widespread deployment of 1-Gbps links, the line might be drawn in the microseconds. It would seem, then, that any timekeeping system intended for wide deployment should contain provisions to accommodate accuracies of this order.

For the highest accuracy it is usually desirable to implant the synchronization protocol at the lower protocol layers of the reference model. While the particular layer is not stated in PCS, both DTSS and NTP are implemented at the transport layer and operated in connectionless mode. Therefore, the protocol itself must provide protection from lost or duplicate messages and determine whether its peers are reachable for the purposes of synchronization. A lightweight association-management capability, including various features for directory cacheing, dynamic reachability, peer selection and variable message-interval mechanisms is used by both DTSS, PCS and NTP to manage state information and reduce resource requirements. Optional features may include message authentication based on crypto-checksums, as in NTP, and provisions for remote management, as in DTSS and NTP.

In general, it is not possible, absent a detailed engineering analysis of each particular scenario, to predict the expected accuracy of a timekeeping system. In principle, both DTSS, PSC and NTP can maintain an accuracy regime consistent with the underlying resource provisioning. While no specific examples can be cited for DTSS, experiments with both PCS and NTP suggest accuracies to a millisecond can be expected for all three protocols when operated over a high speed LAN. However, while NTP is specifically engineered for minimum error operated over an extended WAN such as the Internet, there is some concern about the attainable accuracy of DTSS and the robustness of PCS when operated in such environments.

Most users of a distributed timekeeping system are most concerned about the maximum error that can be displayed, rather than the expected error, which is usually much smaller. Both DTSS, PCS and NTP include algorithms designed to avoid malfunctioning servers and provide to the user confidence intervals which bracket the source of correct time. DTSS does this by collecting samples from at least three servers, computing the intersection of their confidence intervals and providing this

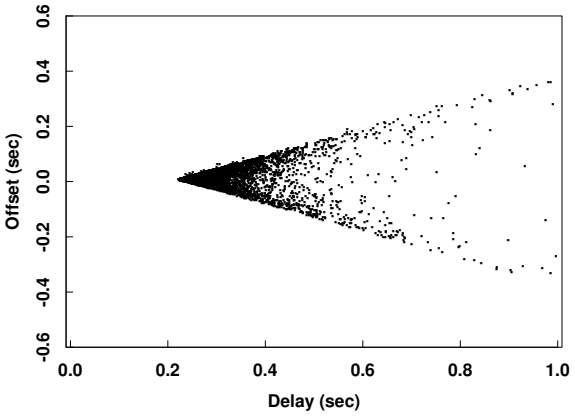


Figure 6. Offset vs Delay

and its midpoint to the user. PCS takes a different approach where the user provides an error tolerance and the protocol discards all samples with confidence intervals larger than the error tolerance.

While NTP includes an algorithm similar to DTSS and with the same claimed robustness, it also includes a considerable burden of procedures designed to provide a best-effort accuracy, even in the face of severe timing noise that normally occurs as the result of stochastic network delays and congestion. The reason for this complexity is to serve both communities - those expecting reliable error bounds, but can tolerate diminished expected accuracy, and those expecting the highest accuracy attainable under current environmental conditions.

There are two features of NTP which contribute to high accuracy expectations. The first is the data-filter algorithm which attempts to find the best sample from among a series of clock-offset samples from each peer. The NTP data-filtering algorithm, which has been evolved over several years of experimentation and experience with Internet paths, is designed specifically to provide high accuracy together with low computational burden.

Recall that the clock offset $\theta$ and roundtrip delay $\delta$ are computed from the four most recent timestamps. Without making any assumptions about the error distributions due to non-reciprocal queueing delays in either direction along the path, but assuming the frequency offsets are relatively small, let $(\delta, \theta)$ represent the delay and offset when the path is otherwise idle and thus the true values. The problem is to produce an accurate estimator $(\hat{\delta}, \hat{\theta})$ from a sample population $(\delta_i, \theta_i)$ collected for the path over an appropriate interval under normal traffic conditions.

The NTP design is based on a *minimum filter*, which selects from the n most recent samples $(\delta_i, \theta_i)$, $(\delta_{i-1}, \theta_{i-1})$, ..., $(\delta_{i-n+1}, \theta_{i-n+1})$ the sample with lowest delay $\delta_j$ and produces $(\delta_j, \theta_j)$ as the estimator $(\hat{\delta}, \hat{\theta})$.

While not discussed in detail here, the algorithm also calculates the dispersion for each sample as the quantity

$$\varepsilon = \rho + \varphi\tau,$$

where the variables have the same interpretation as described previously.

The delay/offset characteristics of a typical Internet path are illustrated in Figure 6, which is a scatter diagram plotting $\theta$ versus $\delta$ points for a path between primary servers on the east and west coasts over an interval of about a week. Under low-traffic conditions the points are concentrated about the apex of the wedge and begin to extend rightward along the extrema lines as the network traffic increases. As the traffic continues to increase, the points begin to fill in the wedge as it expands even further rightward. This behavior is characteristic of typical Internet paths involving ARPANET, NSFNET and regional networks. From these data it is obvious that good estimators $(\hat{\delta}, \hat{\theta})$ are points near the apex, which is exactly what the minimum filter is designed to produce. The minimum filter greatly reduces the expected offset error, often by factors of 100 or more.

The second feature contributing to high accuracy expectations is the combining algorithm, which combines the offset estimates of those peers that pass the intersection algorithm. Recall that most NTP subnet configurations involve the use of several servers, at least three and sometimes many more. Assuming these servers have been declared non-faulty by the intersection algorithm, there remains the opportunity to make use of all the timing information collected, not just the single server selected by the selection algorithm. This is done by a clustering algorithm based on maximum-likelihood principles. In successive rounds the algorithm discards offset extrema, leaving a set of peers depending on the predicted error and reading error of the local clock. The combining algorithm then computes an average weighted by the predicted error. This reduces the expected offset error, usually by a factors of ten or more.

### 3.4. Scaling and the Need for Hierarchy

Since time is ubiquitous, it may develop that most or all computers in a network or internet will be members of the synchronization subnet, so there is some question as to the ability of the synchronization protocol to scale up in the number of servers and clients in the subnet. There are of course natural boundaries imposed by administrative, legal and political constraints and these impose natural boundaries on topology and reachability. However, there are other boundaries imposed by technical reasons, such as the quality and utilization of transmission links, the speed of the processors and so forth. Nevertheless, DTSS and NTP are specifically intended for use in networks including many thousands or even millions of synchronized processors.

Normally, primary clocks cannot be made available for all clients of a timekeeping system. Even if there were, some means would have to be provided to compare their indications, since in practice they are not completely trustworthy. Therefore, there will be a set of servers, at least one server for every primary clock, and each may serve multiple clients or other servers. If the number of clients or servers supported by a particular server exceeds its capacity or the capacity of its connected network, it may be necessary to create a multi-level, tree-structured, hierarchical system, with primary servers represented by the root(s) of the tree and clients represented by its leaves.

While the available PCS documentation does not address hierarchy issues, DTSS and NTP are both hierarchical systems. In its present form DTSS is limited to three levels of hierarchy: one corresponding to the global-server set, another to the local-server set and the third to the clerk set. Each timekeeping entity is designated as either a server or a clerk, with synchronization always flowing from server to clerk.

NTP was developed in the context of the Internet and is blessed or cursed by its characteristics. Extrapolating from a recent survey in Noway, where some eight percent of about 5,000 hosts responded to NTP messages, and a recent estimate of well over 100,000 hosts in the aggregate system, there are probably over 8,000 NTP-speaking hosts on the Internet. This does not count a sizeable number of NTP-capable hosts that do not participate continuously, but choose to read a server clock vicariously every few hours using designer protocols. The present Internet with almost 3,000 networks is hierarchically organized in levels from the NSFNET backbone, through about sixteen regional consortiums to hundreds of autonomously administered domains.

Therefore, NTP provides multiple levels of hierarchy, with each level identified by the stratum number. The stratum number and subnet topology are automatically determined as a function of timekeeping quality, with synchronization always flowing from the root to the leaves. The present NTP synchronization subnet routinely operates with five or more levels of hierarchy. However, it may happen that the subnet is reconfigured as the result of a broken or deteriorated primary clock or server, so that the stratum number direction of synchronization flow may at times be reversed between some servers.

### 3.5. Configuration Strategies

Experience with NTP has proven that configuring a timekeeping system with a constantly changing topol-

ogy, multiple levels of hierarchy and thousands of servers and clients can be a daunting task. Carried to extreme, this could mean that every computer in the Internet must be made aware of at least three servers with which to synchronize. This issue is not addressed in the available PCS or NTP documentation other than to relegate it to the management functions.

In practice, NTP configuration relies on directory services (Domain Name System) augmented by informal databases. Server and client configuration consists of manually selecting a number of likely upstream servers, selecting a operating mode and building a configuration file. In most cases the upstream servers do not need to know about their downstream clients and the configuration files on a particular LAN are usually identical. In order to handle cases when a server is down, clients normally include more than three servers in this file and the protocol selects the best ones from among the set.

DTSS also relies on directory services; however, it includes an elaborate server-discovery scheme based on multicasting and periodically "enumerating the globals;" that is, querying the directory services to extract a list of available servers. The assumption is that DTSS requires no manual configuration at all, other than to set certain architectural constants which presumably are invariant over a management domain.

While not detracting from the DTSS scheme as a valuable management tool, it is not clear whether the particular dynamic enumeration scheme used by DTSS would be feasible in an environment such as the Internet with an estimated over 8,000 servers on almost 3,000 networks operating in hundreds of administrative domains. Presumably the directory information would have to be stratified in hierarchical levels or the information cached at strategic places. There is also an argument, as there also is in the case of cryptographic authentication, that these kinds of services should be management-based and available for use beyond timekeeping services. These are issues appropriate for further study.

### 3.6. Synchronization Strategies

There are considerable differences between DTSS, PCS and NTP on the strategy for discovering servers and using them. As mentioned previously, DTSS discovers servers with the aid of multicasting and directory services and an architected discovery protocol, while NTP relies on directory services and handcrafted configuration tables. The available PCS documentation does not discuss how to do this, but presumes some means is readily available. The principle difference between the timekeeping systems is the scheme used to select among the discovered servers and the strategy of their use.

In DTSS a set of local servers is cached by the clerk. At each round separated by intervals determined by the required accuracy and current confidence interval the client selects one of them at random and attempts to read its clock, which results in a new sample including time offset and confidence interval. The clerk stops at the first response and makes a new selection if a maximum number of attempts is reached. Operation continues in this way until a minimum number of servers have responded. If insufficient local servers have been found, the process continues with a set of global servers. The resulting sample set is then processed to obtain the final clock offset and confidence interval. Note that only one reading from each server is obtained at each round; however, in the case of a time provider, multiple samples may be accumulated in order to assess the health of the device.

Like DTSS, at each round separated by intervals determined by the required accuracy and error tolerance interval, the PCS client attempts, possibly more than once, to read the clock of a predetermined server. The attempts succeed if the confidence interval is less than the error tolerance and fails if a maximum number of attempts is reached. Extensions to the protocol provide for server failures in three ways. One called active master set is to send a multicast request to a number of redundant servers and save the first reply. Another called ranked master group requires each client to use a single default server unless directed otherwise by an unstated server-client protocol. The third called active master ring, in which multiple servers are arranged on a ring. At each round a client selects one of them at random. If attempts to synchronize fail, the client tries the next server on the ring and so on.

It is important to note that both DTSS and PCS "forget" all past history at each round. In effect, each round is statistically independent, with the only state memory the local clock and adjustment procedure. The only exception to this was noted with respect to the DTSS time-provider interface, where a history of samples is maintained for purposes of evaluating the health of the device. However, the NTP model specifically includes a limited amount of state history for the purposes of improving timing accuracy and error statistics.

One of the problems with systems that forget state at each round is that the clock offset and error statistics for each round can be quite different, depending on the particular statistics of the server and implied network paths selected at each round. A user of the service is then faced with the problem of how to interpret the differences and possibly to maintain a quality indicator based on memory of the reported confidence intervals themselves. In the case of PCS this variance can be reduced through judicious selection of the maximum error bound; however, this

may result in an unacceptable rate of outages and searches for redundant servers.

On the other hand, NTP includes specific provisions to remember state in the form of the data filters shown in Figure 2. It has been experimentally verified (see Figure 6) that major improvements in accuracy can be obtained using the minimum filter described previously. The state involved includes the last few measurement samples received from each peer, with special provision to avoid retention of very old samples.

### 3.7. Flexible Access

Although DTSS, PCS and NTP are designed to somewhat different models, they have a common goal of accurate, reliable service. In order to accomplish this goal, all three require exacting conformance to the specifications and possibly costly implementations. However, there may be cases where the cost to implement the full protocol is not justified with respect to the perceived requirements. In DTSS a line is drawn in that the protocol can operate in only one way and all clients must implement the full suite of (client) protocol mechanisms. This issue is not addressed in the available documentation on PCS; however, several alternatives for slave-server access procedures are suggested.

In NTP it is possible for a client or sever to select among several modes of operation, including multicasting and client-server (synchronization flows only from server to client), and peer-peer (synchronization information flows either way, depending on timekeeping quality). It should be pointed out that some of these modes, especially the multicasting mode, where servers simply broadcast the time at designated intervals, do not enjoy all the advantages of the fully implemented protocol; however, it cases involving simple workstations and personal computers, they seem justified.

### 3.8. Leap Seconds

A timekeeping system with profound reliability requirements and accuracy expectations of less than a second is always confronted with the issue of how to deal with leap seconds, which are introduced from time to time in the UTC timescale. There are many issues involved, some of which are addressed in [MIL91b], the issues come down to whether to allow the clocks of the timekeeping system to converge to a newly leaped timescale at their individual intrinsic rates, or to require that all clocks assume the new timescale at the instant of the leap. The former is the case with DTSS and, by impute, PCS. In DTSS the problem is solved simply by increasing the confidence interval at each server by one second just before the end of the UTC month.

The design approach taken in NTP was to require the accuracy expectation to be preserved always, including during, at and beyond the leap event. This has introduced a degree of complexity, since the protocol must provide for the advance distribution of leap-second warning, together with appropriate provisions in the local-clock algorithm. It is the expectation in the design that leap-second warnings are made available from the primary clocks as decreed by national standards bodies. While this expectation has been fulfilled in most time and frequency dissemination services in the U.S., it has not yet been fulfilled by all.

## 4. References and Bibliography


[ALL74] Allan, D.W., J.H. Shoaf and D. Halford. Statistics of time and frequency data analysis. In: Blair, B.E. (Ed.). *Time and Frequency Theory and Fundamentals*. National Bureau of Standards Monograph 140, U.S. Department of Commerce, 1974, 151-204.

[ALL89] Allan, D.W., M.A. Weiss and T.K. Peppler. In search of the best clock. *IEEE Trans. Instrumentation and Measurement 38, 2* (April 1989), 624-630.

[BAR87] Barnes, J.A., and S.R. Stein. Application of Kalman filters and ARIMA models to digital frequency and phase lock loops. *Proc. Nineteenth Annual Precise Time and Time Interval (PTTI) Applications and Planning Meeting*, (Redondo Beach, CA, December 1988), 311-323..

[BEL86] Bell Communications Research. Digital Synchronization Network Plan. Technical Advisory TA-NPL-000436, 1 November 1986.

[BIS90] Bishop, M. A security analysis of the NTP protocol (draft). Department of Mathematics and Computer Science Report, Dartmouth College, June 1990.

[CAL86] "Calendar." *The Encyclopaedia Britannica Macropaedia*, 15th ed., vol. 15, pp. 460-477. Encyclopaedia Britannica Co., New York, NY, 1986.

[CCI90] Time Synchronization Service. CCITT Study Group VII Temporary Document 433, International Telephone and Telegraph Consultative Committee, February 1990.

[CRI89a] Cristian, F. Probabilistic clock synchronization. IBM Almaden Research Center Report RJ 6432 (62550), March 1989.

[CRI89b] Cristian, F. A probabilistic approach to distributed clock synchronization. *Proc. Ninth IEEE International Conference on Distributed Computing Systems* (June 1989), 288-296.

[DER90] Dershowitz, N., and E.M. Reingold. Calendrical Calculations. *Software Practice and Experience 20, 9* (September 1990), 899-928.

[DOD81a] Defense Advanced Research Projects Agency. Internet Control Message Protocol. DARPA Network Working Group Report RFC-792, USC Information Sciences Institute, September 1981.

[DOD81b] Defense Advanced Research Projects Agency. Internet Protocol. DARPA Network Working Group Report RFC-791, USC Information Sciences Institute, September 1981.

[DEC89] Digital Time Service Functional Specification Version T.1.0.5. Digital Equipment Corporation, 1989.

[ISO90] Overview of a Distributed Time Synchronization Service (DTSS). ISO Document ISO/IEC JTC 1/SC21 N4503, International Standards Organization, 1990.

[JON83] Jones, R.H., and P.V. Tryon. Estimating time from atomic clocks. *J. Research of the National Bureau of Standards 88, 1* (January-February 1983), 17-24.

[JOR85] Jordan, E.C. (Ed). *Reference Data for Engineers, Seventh Edition.* H.W. Sams & Co., New York, 1985.

[KRI85] Krishna, C.M., K.G. Shin and R.W. Butler. Ensuring fault tolerance of phase-locked clocks. *IEEE Trans. Computers C-34, 8* (August 1985), 752-756.

[LIN80] Lindsay, W.C., and A.V. Kantak. Network synchronization of random signals. *IEEE Trans. Communications COM-28, 8* (August 1980), 1260-1266.

[MAR85] Marzullo, K., and S. Owicki. Maintaining the time in a distributed system. *ACM Operating Systems Review 19, 3* (July 1985), 44-54.

[MIL89] Mills, D.L. Network Time Protocol (Version 2) specification and implementation. DARPA Network Working Group Report RFC-1119, University of Delaware, September 1989.

[MIL90a] Mills, D.L. On the accuracy and stability of clocks synchronized by the Network Time Protocol in the Internet system. *ACM Computer Communication Review 20, 1* (January 1990), 65-75.

[MIL90b] Mills, D.L. Network Time Protocol (Version 3) specification, implementation and analysis. Electrical Engineering Department Report 90-6-1, University of Delaware, June 1990.

[MIL91a] Mills, D.L. Internet time synchronization: the Network Time Protocol. *IEEE Trans. Communications*, September 1991 (to appear).

[MIL91b] Mills, D.L. On the chronometry and metrology of computer network timescales and their application to the Network Time Protocol. *ACM Computer Communication Review*, (to appear).

[MOR83] Morley, S.G., G.W. Brainerd and R.J. Sharer. *The Ancient Maya*, 4th ed., pp. 598-600. Stanford University Press, Stanford, CA, 1983.

[MOY82] Moyer, G. The Gregorian Calendar. *Scientific American 246, 5* (May 1982), 144-152.

[NBS88] *Automated Computer Time Service (ACTS).* NBS Research Material 8101, U.S. Department of Commerce, 1988.

[PER78] Percival, D.B. The U.S. Naval Observatory clock time scales. *IEEE Trans. Instrumentation and Measurement 27, 4* (December 1978), 376-385.

[POS80] Postel, J. User Datagram Protocol. DARPA Network Working Group Report RFC-768, USC Information Sciences Institute, August 1980.

[POS83] Postel, J. Time protocol. DARPA Network Working Group Report RFC-868, USC Information Sciences Institute, May 1983.

[RAM90] Ramanathan, P., D.D. Kandlur and K.G. Shin. Hardware-assisted software clock synchronization for homogenous distributed systems. *IEEE Trans. Computers C-39, 4* (April 1990), 514-524.

[RAM90] Ramanathan, P., K.G. Shin and R.W. Butler. Fault-tolerant clock synchronization in distributed systems. *IEEE Computer 23, 10* (October 1990), 33-42.

[RAW87] Rawley, L.A., J.H. Taylor, M.M. Davis and D.W. Allan. Millisecond pulsar PSR 1937+21: a highly stable clock. *Science 238* (6 November 1987), 761-765.

[SCI91] ScienceScope. Sounding out the threat of global warning. Science 251 (8 February 1991), 615.

[SHI87] Shin, K.G., and P. Ramanathan. Clock synchronization of a large multiprocessor system in the presence of malicious faults. *IEEE Trans. Computers C-36, 1* (January 1987), 2-12.

[SHI88] Shin, K.G., and P. Ramanathan. Transmission delays in hardware clock synchronization. *IEEE Trans. Computers C-37, 11* (November 1988), 1465-1467.

[TIM86] "Time." *The Encyclopaedia Britannica Macropaedia*, 15th ed., vol. 28, pp. 652-664. Encyclopaedia Britannica Co., New York, NY, 1986.

[TRY83] Tryon, P.V., and R.H. Jones. Estimation of parameters in models for cesium beam atomic

clocks. *J. Research of the National Bureau of Standards 88, 1* (January-February 1983), 3-11.

[VAS88] Vasanthavada, N., and P.N. Marinos. Synchronization of fault-tolerant clocks in the presence of malicious failures. *IEEE Trans. Computers C-37, 4* (April 1988), 440-448.

[WEI89] Weiss, M.A., D.W. Allan and T.K. Peppler. A study of the NBS time scale algorithm. *IEEE Trans. Instrumentation and Measurement 38, 2* (April 1989), 631-635.

## 5. Appendix. Requirements Statements

The following sections contain requirements statements excerpted from the specification documents.

### 5.1. Distributed Time Synchronization Service (DTSS) [ISO90]

DTSS was designed to meet a number of significant technical goals to provide a firm underpinning for large, commercial networks. The design goals include the following:

1. Maximize the probability of a client obtaining the correct time.
2. Rely on specific measurement, rather than averages or experimentally determined parameters, to accommodate all network topologies with[out] operator intervention.
3. Use a client-server model to place complexity on the servers wherever possible. HEAD = 1.
4. Provide a simple and conventional view of time to consumers.
5. Associate a quality with every value of time. The quality can be expressed quantitatively as an inaccuracy measurement.
6. Be fault-tolerant; withstand the arbitrary failure of a small number of servers.
7. Scale from very small networks to networks of at least $10^5$ to $10^6$ real systems.
8. Be highly self-configuring to limit the amount of effort necessary to set up the service and keep it running.
9. Perform efficiently; do not use unreasonable amounts of resources.
10. Since time always advances, clocks too must always advance monotonically.
11. Allow totally decentralized management to avoid the dis-economies of scale in attempting to manage all the resources in a large computer network from one control point.

### 5.2. Network Time Protocol (NTP) [MIL90c]

Internet transmission paths can have wide variations in delay and reliability due to traffic load, route selection and facility outages. Stable frequency synchronization requires stable local-clock oscillators and multiple offset comparisons over relatively long periods of time, while reliable time synchronization requires carefully engineered selection algorithms and the use of redundant resources and diverse transmission paths. For instance, while only a few offset comparisons are usually adequate to determine local time in the Internet to within a few tens of milliseconds, dozens of measurements over some days are required to reliably stabilize frequency to a few milliseconds per day. Thus, the performance requirements of an internet-based time synchronization system are particularly demanding. A basic set of requirements must include the following:

1. The primary reference source(s) must be synchronized to national standards by wire, radio or calibrated atomic clock. The time servers must deliver continuous local time based on UTC, even when leap seconds are inserted in the UTC timescale.
2. The time servers must provide accurate and precise time, even with relatively large delay variations on the transmission paths. This requires careful design of the filtering and combining algorithms, as well as an extremely stable local-clock oscillator and synchronization mechanism.
3. The synchronization subnet must be reliable and survivable, even under unstable network conditions and where connectivity may be lost for periods up to days. This requires redundant time servers and diverse transmission paths, as well as a dynamically reconfigurable subnet architecture.
4. The synchronization protocol must operate continuously and provide update information at rates sufficient to compensate for the expected wander of the room-temperature quartz oscillators used in ordinary computer systems. It must operate efficiently with large numbers of time servers and clients in continuous-polled and procedure-call modes and in multicast and point-to-point configurations.
5. The system must operate in existing internets including a spectrum of machines ranging from personal workstations to supercomputers, but make minimal demands on the operating system and supporting services. Time-server software and especially client software must be easily installed and configured.

In addition to the above, and in common with other generic, promiscuously distributed services, the system must include protection against accidental or willful intrusion and provide a comprehensive interface for network management. [remaining text deleted]