# Internet Time Synchronization: the Network Time Protocol[1, 2]

David L. Mills
Electrical Engineering Department
University of Delaware

**Abstract**

This paper describes the Network Time Protocol (NTP), which is designed to distribute time information in a large, diverse internet system operating at speeds from mundane to lightwave. It uses a symmetric architecture in which a distributed subnet of time servers operating in a self-organizing, hierarchical configuration synchronizes local clocks within the subnet and to national time standards via wire, radio or calibrated atomic clock. The servers can also redistribute time information within a network via local routing algorithms and time daemons.

This paper also discusses the architecture, protocol and algorithms, which were developed over several years of implementation refinement and resulted in the designation of NTP as an Internet Standard protocol. The NTP synchronization system, which has been in regular operation in the Internet for the last several years, is described along with performance data which shows that timekeeping accuracy throughout most portions of the Internet can be ordinarily maintained to within a few milliseconds, even in cases of failure or disruption of clocks, time servers or networks.

Keywords: network clock synchronization, standard time distribution, fault-tolerant architecture, maximum-likelihood principles, disciplined oscillator, internet protocol.

## 1. Introduction

Accurate, reliable time is necessary for financial and legal transactions, transportation and distribution systems and many other applications involving widely distributed resources. How do hosts in a large, dispersed networking community know what time it is? How accurate are their clocks? In a recent survey involving 94,260 hosts of the Internet system, 20,758 provided local time using three time-transfer protocols [24]. About half of the replies had errors greater than two minutes, while ten percent had errors greater than four hours. A few had errors over two weeks. Most local clocks are set by eyeball-and-wristwatch to within a minute or two and rarely checked after that. Many of these are maintained by some sort of battery-backed clock-calendar device using a room-temperature quartz oscillator that may drift as much as a second per day and can go for weeks between manual corrections. For many applications, especially distributed internet applications, much greater accuracy and reliability is required.

This paper presents an overview of the architecture, protocol and algorithms of the Network Time Protocol (NTP) used in the Internet system to synchronize clocks and coordinate time distribution. The Internet consists of over 100,000 hosts on over 1500 packet-switching networks interconnected by a similar number of gateways. In this paper the capitalized *Internet* refers to this particular system, while the uncapitalized *internet* refers to any generic system of multiple networks interconnected by gateways. While the Internet backbone networks and gateways are carefully engineered for good service, operating speeds and service reliability vary considerably throughout the system. This places severe demands on NTP, which must deliver accurate and reliable time in spite of component failures, service disruptions and possibly mis-engineered implementations.

In the remainder of this introductory Section 1, issues in the requirements, approaches and comparisons with previous work are discussed. The architecture of the NTP synchronization system, including the primary reference sources and distribution mechanisms, is described in Section 2. An overview of the NTP protocol and modes of operation is given in Section 3. Section 4 describes the algorithms used to improve the accuracy of measurements made over Internet paths and to select the best from among a set of available clocks for synchronization. Section 5 describes a local-clock design based on a type of phase-lock loop and capable of long-term accuracies to the order of a few milliseconds. The international NTP

1. Sponsored by: Defense Advanced Research Projects Agency contract number N00140-87-C-8901 and by National Science Foundation grant number NCR-89-13623.

2. To appear in: Mills, D.L. Internet time synchronization: the Network Time Protocol. *IEEE Transactions on Communications*. Distribution is restricted pending publication.

synchronization system of time servers now operating in the Internet is described and its performance assessed in Section 6. Section 7 discusses further development and issues for future research. This paper itself is an updated and condensed version of [23].

### 1.1. Definitions

In this paper the *stability* of a clock is how well it can maintain a constant frequency, the *accuracy* is how well its time compares with national standards and the *precision* is how precisely time can be resolved in a particular timekeeping system. The *offset* of two clocks is the time difference between them, while the *skew* is the frequency difference between them. The *reliability* of a timekeeping system is the fraction of the time it can be kept operating and connected in the network (without respect to stability and accuracy). Local clocks are maintained at designated *time servers*, which are timekeeping systems belonging to a *synchronization subnet*, in which each server measures the offsets between its local clock and the clocks of its neighbor servers or *peers* in the subnet. In this paper to *synchronize frequency* means to adjust the clocks in the subnet to run at the same frequency, to *synchronize time* means to set them to agree at a particular epoch with respect to Coordinated Universal Time (UTC), as provided by national standards, and to *synchronize clocks* means to synchronize them in both frequency and time.

### 1.2. Performance Requirements

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. In NTP address filtering is used for access control, while encrypted checksums are used for authentication. Network management presently uses a proprietary protocol with provisions to migrate to standard protocols where available.

### 1.3. Discussion of Approaches

There are many ways that time servers distributed throughout a large geographic area can synchronize clocks to UTC. In North America the U.S. and Canada operate broadcast radio services with a UTC timecode modulation which can be decoded by suitable receivers. One approach to time synchronization is to provide timecode receivers at every site where required. However, these receivers are specialized, moderately expensive and subject to occasional gross errors due to propagation and equipment failures. A comprehensive summary of radio synchronization techniques can be found in [4].

The U.S. National Institute of Standards and Technology (NIST) (formerly National Bureau of Standards), recently announced a computer time service available to the general public by means of a standard telephone modem [26]. The service is intended for use by personal workstations to set clock-calendars, for example, but

would not be suitable for a large population of clients calling on a frequent, regular basis without further redistribution.

In principle, it is possible to use special network facilities designed for time synchronization, such as timecode rebroadcasts on a dedicated FM or TV subcarrier or cable system. For many years AT&T has synchronized digital switching equipment to the Basic Synchronization Reference Frequency (BSRF), which consists of a master oscillator synchronized to UTC and a network of dedicated 2048-kHz links embedded in the transmission plant. AT&T and other carriers are planning to use the Global Positioning System and the LORAN-C radionavigation system to synchronize switches in various areas of the country. However, neither of these methods would be economically viable for widespread deployment in a large, diverse internet system.

Current network clock synchronization techniques have evolved from both linear systems and Byzantine agreement methodologies. Linear methods for digital telephone network synchronization are summarized in [16], while Byzantine methods for clock synchronization are summarized in [15]. While reliable clock synchronization has been studied using agreement algorithms [15], [33], in practice it is not possible to distinguish the *truechimer* clocks, which maintain timekeeping accuracy to a previously published (and trusted) standard, from the *falseticker* clocks, which do not, on other than a statistical basis. In addition, the algorithms discussed in the literature do not necessarily produce the most accurate and precise time on a statistical basis and may produce unacceptable network overheads and instabilities in a large, diverse internet system.

In an internet system involving many networks and gateways a useful approach is to equip a few strategically located hosts or gateways with timecode receivers or calibrated atomic clocks and coordinate time distribution using a suitable protocol. Various Internet protocols have been used to record and transmit the time at which an event takes place, including the Daytime protocol [28], Time protocol [29], ICMP Timestamp message [7] and IP Timestamp option [34]. The Unix 4.3bsd time daemon *timed* uses an elected master host to measure offsets of a number of slave hosts and send periodic corrections to them [11]. While addressing no particular protocol architecture, the schemes proposed in [6] have features in common with NTP, including master-slave synchronization with provisions for failures and changing system load. However, none of these protocols includes engineered algorithms to compensate for the effects of statistical delay variations encountered in wide-area networks and are unsuitable for precision time distribution throughout the Internet.

It became evident, as the algorithms used in NTP evolved over a nine-year period of experiment and stepwise refinement, that accurate and reliable internet time synchronization can be achieved only through an integrated approach to system design, including the primary reference sources, time servers, synchronization subnets, protocols and synchronization mechanisms which are at the heart of this paper. From an analytical point of view the distributed system of NTP time servers operates as a hierarchically organized subnet of loosely coupled time servers which exchange periodic update messages containing precision timestamps to adjust local oscillator phase and frequency. The principal features of this design, described in more detail later in this paper, can be summarized as follows:

1. The synchronization subnet consists of a self-organizing, hierarchical network of time servers configured on the basis of estimated accuracy, precision and reliability.

2. The synchronization protocol operates in connectionless mode in order to minimize latencies, simplify implementations and provide ubiquitous interworking.

3. The synchronization mechanism uses a symmetric design which tolerates packet loss, duplication and mis-ordering, together with filtering, selection and combining algorithms based on maximum-likelihood principles.

4. The local-clock design is based on a type II, adaptive-parameter, phase-lock loop with corrections computed using timestamps exchanged along the arcs of the synchronization subnet.

5. Multiply redundant time servers and multiply diverse transmission paths are used in the synchronization subnet, as well as engineered algorithms which select the most reliable synchronization source(s) and path(s) using weighted-voting procedures.

6. System overhead is reduced through the use of dynamic control of phase-lock loop bandwidths, poll intervals and association management

## 2. Time Standards and Distribution

Since 1972 the time and frequency standards of the world have been based on International Atomic Time (TAI), which is currently maintained using multiple cesium-beam clocks to an accuracy of a few parts in $10^{12}$ [1]. The International Bureau of Weights and Measures uses astronomical observations provided by the U.S. Naval Observatory and other observatories to determine corrections for small changes in the mean solar rotation period of the Earth, which results in Coordinated Univer-

sal Time (UTC). UTC is presently slow relative to TAI by a fraction of a second per year, so corrections in the form of leap seconds must be inserted in TAI from time to time in order to maintain agreement. The U.S. and many other countries operate standard time and frequency broadcast stations covering most areas of the world, although only a few utilize a timecode modulation suitable for computer use.

The NTP system consists of a network of primary and secondary time servers, clients and interconnecting transmission paths. A primary time server is directly synchronized to a primary reference source, usually a timecode receiver or calibrated atomic clock. A secondary time server derives synchronization, possibly via other secondary servers, from a primary server over network paths possibly shared with other services. Under normal circumstances clock synchronization is determined using only the most accurate and reliable servers and transmission paths, so that the actual synchronization paths usually assumes a hierarchical configuration with the primary reference sources at the root and servers of decreasing accuracy at increasing levels toward the leaves.

Following conventions established by the telephone industry, the accuracy of each time server is defined by a number called the *stratum*, with the reference level (primary servers) assigned as one and each succeeding level towards the leaves (secondary servers) assigned as one greater than the preceding level [2]. Using existing stations and available timecode receivers with computed propagation-delay corrections, accuracies in the order of a millisecond can be achieved at the network interface of a primary server. As the stratum increases from one, the accuracies achievable will degrade depending on the network paths and local-clock stabilities.

The synchronization subnet is organized using a variant of the Bellman-Ford distributed routing algorithm to compute the minimum-weight spanning trees rooted at the primary reference sources [3]. The distance metric is determined first by stratum, then by total roundtrip path delay to the root, called the *synchronization distance*. The timekeeping quality at a particular peer is determined by a sum of weighted offset differences, called the *dispersion*. The total dispersion to the root due to all causes is called the *synchronization dispersion*.

## 3. Network Time Protocol

The Network Time Protocol (NTP), now established as an Internet Standard protocol [22], is used to organize and maintain a set of time servers and transmission paths as a synchronization subnet. NTP is built on the Internet Protocol (IP) [8] and User Datagram Protocol (UDP) [27], which provide a connectionless transport mechanism; however, it is readily adaptable to other protocol

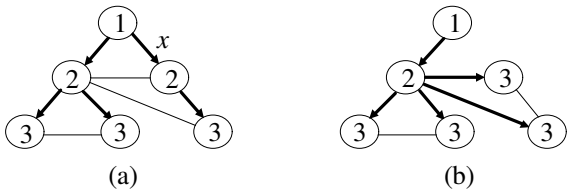

Figure 1. Subnet Synchronization

suites. It is evolved from the Time Protocol [29] and the ICMP Timestamp Message [7], but is specifically designed to maintain accuracy and reliability, even when used over typical Internet paths involving multiple gateways and unreliable networks.

There are no provisions for peer discovery, configuration or acquisition in NTP itself, although some implementations include these features. Data integrity are provided by the IP and UDP checksums. No circuit-management, duplicate-detection or retransmission facilities are provided or necessary. The protocol can operate in several modes appropriate to different scenarios involving private workstations, public servers and various network configurations. A lightweight association-management capability, including dynamic reachability and variable poll-interval mechanisms, is used to manage state information and reduce resource requirements. Optional features include message authentication based on crypto-checksums and provisions for remote control and monitoring. Since only a single NTP message format is used, the protocol is easily implemented and can be used in a variety of operating-system and networking environments.

In NTP one or more primary servers synchronize directly to external reference sources such as timecode receivers. Secondary time servers synchronize to the primary servers and others in the synchronization subnet. A typical subnet is shown in Figure 1a, in which the nodes represent subnet servers, with normal stratum numbers determined by the hop count to the root, and the heavy lines the active synchronization paths and direction of timing information flow. The light lines represent backup synchronization paths where timing 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 stratum 2 to stratum 3.

The following subsections contain an overview of the data formats, entities, state variables and procedures used in NTP. Further details are contained in the formal specification [22]. The specification is based on the implementation model illustrated below, but it is not intended that this model be the only one upon which a specification can be based. In particular, the specification is

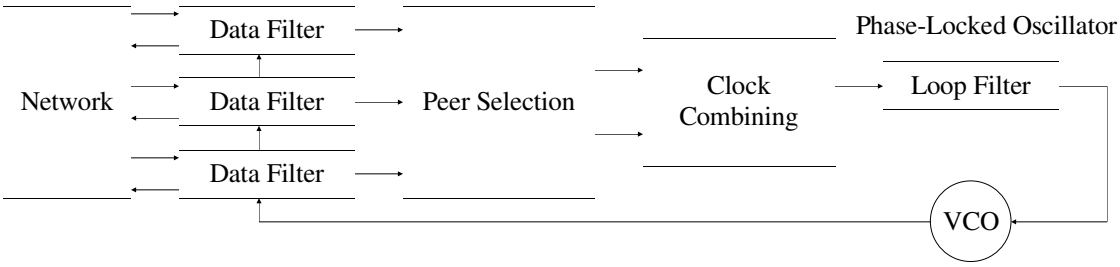


Figure 2. Network Time Protocol

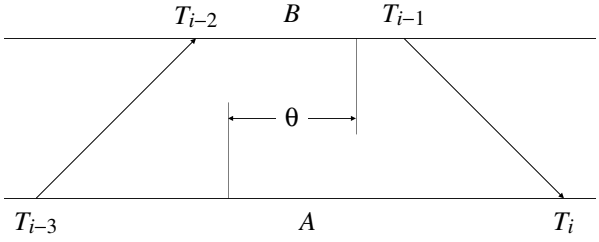


Figure 3. Measuring Delay and Offset

intended to illustrate and clarify the intrinsic operations of NTP and serve as a foundation for a more rigorous, comprehensive and verifiable specification.

### 3.1. Determining Time and Frequency

Figure 2 shows the overall organization of the NTP time-server model. Timestamps exchanged between the server and possibly many other subnet peers are used to determine individual roundtrip delays and clock offsets, as well as provide reliable error estimates. Figure 3 shows how the timestamps are numbered and exchanged between server $A$ and peer $B$. Let $T_i$, $T_{i-1}$, $T_{i-2}$, $T_{i-3}$ be the values of the four most recent timestamps as shown and let $a = T_{i-2} - T_{i-3}$ and $b = T_{i-1} - T_i$. Then, the roundtrip delay $\delta_i$ and clock offset $\theta_i$ of $B$ relative to $A$ at time $T_i$ are:

$$\delta_i = a - b \quad \text{and} \quad \theta_i = \frac{a+b}{2}.$$

In the present NTP version (2) errors due to local-clock resolution and skew are minimized by the control-feedback design shown in Figure 2. In practice, errors due to stochastic network delays dominate; however, it is not usually possible to characterize network delays as a stationary random process, since network queues can grow and shrink in chaotic fashion and arriving customer traffic is frequently bursty.

Nevertheless, it is a simple exercise to calculate bounds on network 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

$$\theta_i - \frac{\delta_i}{2} \leq \theta \leq \theta_i + \frac{\delta_i}{2}.$$

In other words, the true clock offset must lie in the interval of size equal to the measured delay and centered about the measured offset.

Each NTP message includes the latest 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 the server and the peer can independently calculate delay and offset using a single message stream. This can be described as a symmetric, continuously sampled, time-transfer method similar to those used in some digital telephone networks [25]. Among its advantages are that the transmission times and received message orders are unimportant and that reliable delivery is not required. Obviously, the accuracies achievable depend upon the statistical properties of the outbound and inbound data paths. Further analysis and experimental results bearing on this issue can be found below and in [5], [19] and [20].

As shown in Figure 2, the computed delays and offsets are processed in the data filters to reduce incidental timing noise and the most accurate and reliable subset determined by the peer-selection algorithm. The resulting offsets of this subset are first combined on a weighted-average basis and then processed by a phase-lock loop (PLL). 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 the above calculations.

### 3.2. Modes of Operation

NTP time servers can operate in one of three service classes: multicast, procedure-call and symmetric. These classes are distinguished by the number of peers involved, whether synchronization is to be given or received and whether state information is retained. The multicast class is intended for use on high speed LANs with numerous workstations and where the highest accuracies are not required. In the typical scenario one or more time servers operating in multicast mode send periodic NTP broadcasts. The workstation peers operating in client mode then determine the time on the basis of an assumed delay in the order of a few milliseconds. By operating in multicast mode the server announces its willingness to provide synchronization to many other peers, but to accept NTP messages from none of them.

The procedure-call class is intended for operation with file servers and workstations requiring the highest accuracies or where multicast mode is unavailable or inappropriate. In the typical scenario a time server operating in client mode sends an NTP message to a peer operating in server mode, which then interchanges the addresses, inserts the requested timestamps, recalculates the checksum and optional authenticator and returns the message immediately. By operating in client mode a server announces its willingness to be synchronized by, but not provide synchronization to a peer. By operating in server mode a server announces its willingness to provide synchronization to, but not be synchronized by a peer.

While the multicast and procedure-call classes may suffice on LANs involving public time servers and perhaps many private workstation clients, the full generality of NTP requires distributed participation of a number of time servers arranged in a dynamically reconfigurable, hierarchically distributed configuration. This is the motivation for the symmetric modes (active and passive). By operating in these modes a server announces its willingness to synchronize to or be synchronized by a peer, depending on the peer-selection algorithm. Symmetric active mode is designed for use by servers operating near the leaves (high stratum levels) of the synchronization subnet and with pre-configured peer addresses. Symmetric passive mode is designed for use by servers operating near the root (low stratum levels) and with a relatively large number of peers on an possibly intermittent basis.

When a pair of servers operating in symmetric modes first exchange messages, a loosely coupled connection or *association* is created. Each server creates an instantiation of the NTP protocol machine with persistent state variables; however, the main purpose of the protocol machine is not to assure delivery but to preserve timestamps and related information. In symmetric modes the servers refresh reachability status as each message is received and dissolve the association and recover state storage if this status has not been refreshed for a considerable time.

| LI | VN | Mode | Stratum | Poll | Precision |
|---|---|---|---|---|---|
| Synchronizing Distance | | | | | |
| Synchronizing Dispersion | | | | | |
| Reference Timestamp (64 bits) | | | | | |
| Originate Timestamp (64 bits) | | | | | |
| Receive Timestamp (64 bits) | | | | | |
| Transmit Timestamp (64 bits) | | | | | |
| Authenticator (optional) (96 bits) | | | | | |

Figure 4. NTP Packet Header

### 3.3. Data Formats

All mathematical operations assumed in the protocol are two's-complement arithmetic with integer or fixed-point operands. Since NTP timestamps are cherished data and, in fact, represent the main product of the protocol, a special format has been established. An NTP timestamp is a 64-bit unsigned fixed-point number, with the integer part in the first 32 bits and the fraction part in the last 32 bits and interpreted in standard seconds relative to UTC. When UTC began at $0^h$ on 1 January 1972 the NTP clock was set to 2,272,060,800.0, representing the number of standard seconds since this time at $0^h$ on 1 January 1900 (assuming no prior leap seconds).

This format allows convenient multiple-precision arithmetic and conversion to other formats used by various protocols of the Internet suite. The precision of this representation is about 232 picoseconds, which should be adequate for even the most exotic requirements. Note that since some time in 1968 the most significant bit of the 64-bit field has been set and that the field will overflow some time in 2036. Should NTP be in use in 2036, some external means will be necessary to qualify time relative to 1900 and subsequent 136-year cycles. Historic timestamped data of such precision and requiring such qualification will be so precious that appropriate means should be readily conceived.

Timestamps are determined by copying the current value of the local clock to a timestamp variable when some

significant event occurs, such as the arrival of a message. In some cases a particular variable may not be available, such as when the server is rebooted or the protocol is restarted. In these cases the 64-bit field is set to zero, indicating an invalid or undefined value. There exists a 232-picosecond interval, henceforth ignored, every 136 years when the 64-bit field will naturally become zero and thus be considered invalid.

### 3.4. State Variables

Following is a summary description of the important variables and parameters used by the protocol. In the symmetric modes a set of state variables is maintained for each association. In other modes these variables have a fleeting persistence lasting only until the reply message has been formulated and sent. Further discussion on some of these variables is given later in this paper. A complete description is given in [22].

Figure 4 shows the NTP packet-header format, which follows the IP and UDP headers. Following is a short description of the various fields.

Leap Indicator (LI). Warns of an impending leap second to be inserted or deleted in the UTC timescale at the end of the current day.

Version Number (VN). Identifies the present NTP version (2).

Mode, Stratum, Precision. Indicate the current operating mode, stratum and local-clock precision.

Poll Interval (Poll). The current desired interval between NTP messages sent. Each server uses the minimum of its own poll interval and that of the peer.

Synchronization Distance, Synchronization Dispersion. Indicates the total roundtrip delay and total dispersion, respectively, to the primary reference source.

Reference Clock Identifier, Reference Timestamp. Identifies the type of reference clock and the time of its last update; intended primarily for management functions.

Originate Timestamp. The peer time when the last received NTP message was originated, copied from its transmit timestamp field upon arrival ($T_{i-3}$ above).

Receive Timestamp. The local time when the latest NTP message was received ($T_{i-2}$ above).

Transmit Timestamp. The local time when this NTP message was transmitted ($T_{i-1}$ above).

Authenticator (optional). The key identifier and encrypted checksum of the message contents.

The NTP protocol machine maintains state variables for each of the above quantities and, in addition, other state variables, including the following:

Addresses and Ports. The 32-bit Internet addresses and 16-bit port numbers of the server and peer, which serve to identify the association.

Peer Timer. A counter used to control the intervals between transmitted NTP messages.

Reachability Register. A shift register used to determine the reachability status of a peer.

Filter Register. A shift register used by the data-filtering algorithm, where each stage stores a tuple consisting of the measured delay and offset associated with a single delay/offset sample.

Delay, Offset, Dispersion. Indicate the current roundtrip delay, clock offset and filter dispersion produced by the data-filtering algorithm.

Synchronization Source. Identifies the peer currently used to synchronize the local clock, as determined by the peer-selection algorithm.

Local Clock. The current local time as derived from the local clock.

### 3.5. Procedures

The significant events of interest in NTP occur upon expiration of a peer timer, one of which is dedicated to each association, and upon arrival of an NTP message. An event can also occur as the result of an operator command or detected system fault, such as a primary reference source failure. This subsection briefly summarizes the procedures invoked when these events occur.

The transmit procedure is called when a peer timer decrements to zero. When this occurs the peer timer is reset and an NTP message is sent including the addresses, variables and timestamps described above. The value used to reset the timer is called the poll interval and is adjusted dynamically to reflect dispersive delays and reachability failures.

The receive procedure is called upon arrival of an NTP message, which is then matched with the association indicated by its addresses and ports. This results in the creation of a persistent association for a symmetric mode or a transient one for the other modes. Following a set of sanity checks the raw roundtrip delay and raw clock offset sample are calculated as described previously. A weighted voting procedure described in Section 4 determines the best in a sequence of raw samples and also an error estimator called the *filter dispersion*. The final values of roundtrip delay, clock offset and filter disper-

sion are determined using the minimum-filter algorithm described in Section 4.

The update procedure is called when a new set of estimates becomes available. A weighted voting procedure described in Section 4 determines the best peer, which may result in a new synchronization source, and also an error estimator called the *select dispersion*. If the synchronization source is the peer for which the estimates have just been produced, the estimated offset is used to adjust the local clock as described in Section 5. If due to a significant discrepancy the local clock is reset, rather than gradually slewed to its final value, the procedure expunges all timing information, resets the poll intervals and re-selects the synchronization source, if necessary. A new synchronization source will be determined when the data filters fill up again and the dispersions settle down.

### 3.6. Robustness Issues

It has been the experience of the Internet community that something somewhere in the system is broken at any given time. This caveat applies to timecode receivers, time servers and synchronization paths, any of which can misbehave to produce a bogus timestamp popularly known as a *timewarp*. The very nature of time synchronization requires that it be extremely robust against timewarps and capable of operation even when major breakdowns or attempted break-ins occur. This subsection describes some of the measures taken to deal with these problems, including reachability, authentication and poll control.

As shown previously, reliable time synchronization does not require reliable message delivery; however, in order to minimize resource requirements, resist using very old data and manage the memory resources required, a simple reachability protocol is used in which a peer is considered unreachable if no messages are received during eight consecutive poll intervals. In the active modes the peer is marked unreachable, but polls continue; while, in the passive modes the association is dissolved and its resources reclaimed for subsequent use.

Special sanity checks are provided to avoid disruptions due to system reboot, protocol restart or malfunction. For instance, if the transmit timestamp of a message is identical to one previously received, the message is a duplicate or replay and may contain bogus data. Since precision timestamps are difficult to spoof, the originate timestamp makes a fairly effective one-time pad. If a message contains an originate timestamp that does not match the transmit timestamp of the last message transmitted, the message is either out of order, from a previous association or bogus. Additional checks protect against using very old time information and from associations not completely synchronized.

Where security considerations require the highest level of protection against message modification, replay and other overt attacks, the NTP specification includes optional cryptographic authentication procedures. The procedures are used to insure an unbroken chain of authenticated associations within the synchronization subnet to the primary servers. An authenticator, consisting of a key identifier and encrypted checksum, is computed using the DES encryption algorithm [9] and DES cipher block-chaining method [10]. Some implementations incorporate special provisions to compensate for the delays inherent in the encryption computations.

Careful consideration was given during design to factors affecting network overheads. Some of the present Internet time servers operate with over 100 peers and a few operate with many more than that. Therefore, it is important that the longest poll intervals consistent with the required accuracy and stability be used. When reachability is first confirmed and when dispersions are high it is necessary to use a relatively wide PLL bandwidth, which requires a poll interval no greater than about a minute. When the association has stabilized and dispersions are low, the PLL bandwidth can be reduced to improve stability, which allows the poll interval to be increased substantially. In the present design the poll interval is increased gradually from about one minute to about 17 minutes as long as the filter dispersion is below an experimentally determined threshold; otherwise, it is decreased gradually to its original value.

## 4. Filtering, Selection and Combining Operations

At the very heart of the NTP design are the algorithms used to improve the accuracy of the estimated delays and offsets between the various servers, as well as those used to select a particular peer for synchronization. The complexity of these algorithms depends on the statistical properties of the transmission path, as well as the accuracies and precisions required. Since Internet paths often involve multiple hops over networks of widely varying characteristics, it is not possible to design one set of algorithms optimized for a particular path. Another factor considered is to avoid the use of multiply/divide operations in favor of simple shifts in order to facilitate implementation on dedicated microprocessors.

A good deal of research has gone into mechanisms to synchronize clocks in a community where some clocks cannot be trusted. Determining whether a particular clock can be trusted is an interesting abstract problem which can be attacked using methods such as described in [14], [15], [18] and [31]. A number of algorithms for filtering, smoothing and classifying timekeeping data have been described in the literature [1], [6], [12], [13], [19], including convergence algorithms, which attempt

to reduce errors by repeatedly casting out statistical outlyers, and consistency algorithms, which attempt to classify subsets of clocks as trusted or not by comparing statistics such as mean and variance. The NTP data-filtering algorithm, which attempts to improve the offset estimate for a single clock, given a series of observations, belongs to the former class. The NTP peer-selection algorithm, which attempts to find the best (i.e., the most reliable) clocks from a population, belongs to the latter class.

### 4.1. Data-Filtering Algorithm

Interactive convergence algorithms use statistical clustering techniques such as the fault-tolerant average (FAT) algorithm of [12], the CNV algorithm of [17], the majority-subset algorithm of [19], the non-Byzantine algorithm of [30] and the egocentric algorithm of [31]. A variation on the FAT algorithm suggested in a recent paper [6] attempts to bound the offset errors when reading a remote clock by casting out readings where the measured roundtrip delay is above a specified value. This algorithm has features in common with the NTP data-filtering algorithm, but does not take advantage of the improved accuracy possible using a statistical analysis such as described in this section.

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 roundtrip delay $\delta$ and clock offset $\theta$ are computed from the four most recent timestamps. Without making any assumptions about the delay distributions due to packet queueing in either direction along the path, but assuming the skew between the server and peer clocks is 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 approach used in the design of the data-filtering algorithm was suggested by the observation that packet-switching networks are most often operated well below the knee of the throughput-delay curve, which means that packet queues are mostly small with relatively infrequent surges. In addition, the routing algorithm most often operates to minimize the number of packet-switch hops and thus the number of queues. Thus, not only is the probability that an NTP packet finds a busy queue in one direction relatively low, but the probability of packets from a single exchange finding busy queues in both directions is even lower. Therefore, the best offset samples should occur at the lowest delays,

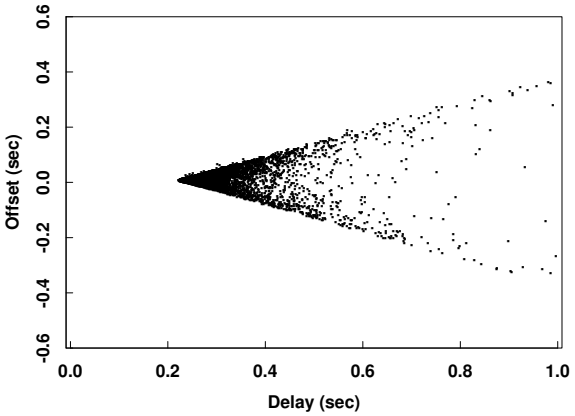


Figure 5. Offset vs Delay

This observation suggests the design of 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 $(\hat{\delta}, \hat{\theta})$. Several experiments were made to evaluate this design using measurements between NTP primary servers, so that delays and offsets could be determined independently of the measurement procedure itself [24]. The experiments were performed over several paths involving ARPANET, NSFNET and various LANs and using minimum filters and various other algorithms based on median and trimmed-mean statistics. The results show consistently lower errors for the minimum filter when compared the other algorithms. Perhaps the most dramatic result with the minimum filter is the greatly reduced maximum error under conditions of high levels of network traffic.

The delay/offset characteristics of a typical Internet path are illustrated in Figure 5, 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. This particular path involves seven networks and twelve gateways and is among the most complex in the NTP synchronization subnet. 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.

In the reference implementation, samples $(\delta_i, \theta_i)$ are shifted into an eight-stage shift register from one end, causing old samples to shift off the other. Then, all eight samples are placed on a temporary list and sorted in order

of increasing $\delta$. The first sample on the list $(\delta_0, \theta_0)$ represents the estimators $(\hat{\delta}, \hat{\theta})$, which are recorded for each peer separately for later processing by the selection and combining algorithms.

The filter dispersion is interpreted as a quality indicator, with increasing values associated with decreasing quality and weight in the selection and combining algorithms. A good estimator which counts samples near the apex of the wedge most heavily and is easily computable is the weighted differences of the $\theta_i$ in the sorted temporary list relative to $\theta_0$. Assume the list has $n > 1$ entries ($n = 8$ in this case) with $(\delta_j, \theta_j)$ ($j = 0, 1, ..., n - 1$) samples in order of increasing $\delta_i$. The filter dispersion $\varepsilon$ is defined

$$\varepsilon = \sum_{j=0}^{n-1} |\theta_j - \theta_0| \, v^j ,$$

where $v$ is an experimentally adjusted weight factor, $v = 0.5$ in the reference implementation. The filter dispersion is recorded for each peer separately for later processing by the selection and combining algorithms.

### 4.2. Peer-Selection and Combining Algorithms

The single most important contributing factor in maintaining high reliability with NTP is the peer-selection and combining algorithms. When new offset estimates are produced for a peer or are revised as the result of timeout, this mechanism is used to determine which peer should be selected as the synchronization source and how to adjust the local-clock, stratum and related variables.

Interactive consistency algorithms are designed to tolerate faulty clock processes which might indicate grossly inconsistent offsets in successive readings or to different readers. These algorithms use an agreement protocol involving successive rounds of readings, possibly relayed and possibly augmented by digital signatures. Examples include the fireworks algorithm of [12] and the optimum algorithm of [33]. However, these algorithms as described require an excessive burden of messages, especially when large numbers of clocks are involved, and require statistically awkward assumptions in order to certify correctness.

While drawing upon the technology of agreement algorithms, the NTP peer-selection algorithm is not strictly one of them, but an adaptation based on maximum-likelihood statistical principles and the pragmatic observation that the highest reliability is usually associated with the lowest stratum and synchronization dispersion, while the highest accuracy is usually associated with the lowest stratum and synchronization distance. A key design assumption is that truechimers are relatively numerous and represented by random variables narrowly distributed about UTC in the measurement space, while falsetickers are relatively rare and represented by random variables widely distributed throughout the measurement space.

The peer-selection algorithm begins by constructing a list of candidate peers sorted first by stratum and then by synchronization dispersion. To be included on the candidate list a peer must pass several sanity checks designed to detect blatant errors and defective implementations. If no peers pass the sanity checks, the existing synchronization source, if any, is cancelled and the local clock free-runs at its intrinsic frequency. The list is then pruned from the end to a predetermined maximum size and maximum stratum.

The next step is designed to detect falsetickers or other conditions which might result in gross errors. The candidate list is re-sorted in the order first by stratum and then by synchronization distance. Let $m > 0$ be the number of candidates remaining in the list and let $\theta_j$ be the offset of the $j$th candidate. For each $j$ ($0 \le j < m$) the select dispersion $\varepsilon_j$ relative to candidate $j$ is defined

$$\varepsilon_j = \sum_{k=0}^{m-1} |\theta_j - \theta_k| \, w^k ,$$

where $w$ is a factor experimentally adjusted for the desired characteristic (see below). Then discard the candidate with maximum $\varepsilon_j$ or, in case of ties the maximum $j$, and repeat the procedure. The procedure terminates when the maximum select dispersion over all candidates remaining on the list is less than the minimum filter dispersion of any candidate or until only a single candidate remains.

The above procedures are designed to favor those peers near the beginning of the candidate list, which are at the lowest stratum and lowest delay and presumably can provide the most accurate time. With proper selection of weight factor $w$, outlyers will be discarded from the end of the list, unless some other entry disagrees significantly with respect to the remaining entries, in which case that entry is discarded. For example, with $w = 0.75$ as used in the reference implementation, a single stratum-2 server at the end of the candidate list will swing the vote between two stratum-1 servers that disagree with each other. While these outcomes depend on judicious choice of $w$, the behavior of the algorithm is substantially the same for values of $w$ between 0.5 and 1.0.

The offsets of the peers remaining on the candidate list are statistically equivalent, so any of them can be chosen to adjust the local clock. Some implementations combine them using a weighted-average algorithm similar to that described in [1], in which the offsets of the peers remaining on the list are weighted by estimated error to produce a combined estimate. In these implementations the error

estimate is taken to be the reciprocal of synchronization dispersion.

The update procedure also sets the local stratum to one greater than the stratum of the selected peer. In addition, the server synchronization distance - the sum of the total roundtrip delays to the root of the synchronization subnet, as well as the server synchronization dispersion - the sum of the total dispersions to the root of the synchronization subnet, are calculated and recorded in a system state variable. All three of these quantities are included in the NTP message header.

## 5. Local-Clock Design

Precision timekeeping requires an exceptionally stable local oscillator reference in order to deliver accurate time when the synchronization path to a primary server has failed. Furthermore, the oscillator and control loop must maintain accurate time and stable frequency over wide variations in synchronization path delays. In the NTP local-clock model the fundamental system time reference, or logical clock, increments at some standard rate such as 1000 Hz and can be adjusted to precise time and frequency by means of periodic corrections determined by NTP, a timecode receiver or a calibrated atomic clock.

The model shown in Figure 6 can be described as a type-II, adaptive-parameter, phase-lock loop (PLL), which continuously corrects local oscillator phase and frequency variations relative to received updates. The difference between the peer time and server time $T_B - T_A$; that is, the offset $\theta$ shown in Figure 3, is processed by the phase detector PD to produce the output $V_d$. The filtering, selection and combining algorithms shown in Figure 2 operate as a variable delay network to produce the output $V_s$. The loop filter produces the output $V_c$, which is used to adjust the frequency of the voltage-controlled oscillator VCO in order to reduce the offset $\theta$.

Using familiar techniques of analysis [32], the (open-loop) transfer function of the PLL can be approximated as

$$F(s) = \frac{\omega_c^2}{s^2\tau^2}\left(1 + \frac{s\tau}{\omega_z}\right) e^{-sT},$$

where $\omega_c$ is the gain (crossover frequency), $\omega_z$ the corner frequency of the lead network (necessary for PLL stability), $T$ is the data-filter delay and $\tau$ is a parameter used for bandwidth control. Simulation of the entire PLL with the variables and constants specified in [22] results in the following characteristics: At the widest bandwidth (smallest $\tau$) and a 100-ms phase change the PLL reaches zero error in 39 minutes, overshoots 7 ms in 54 minutes and settles to less than 1 ms in about six hours.

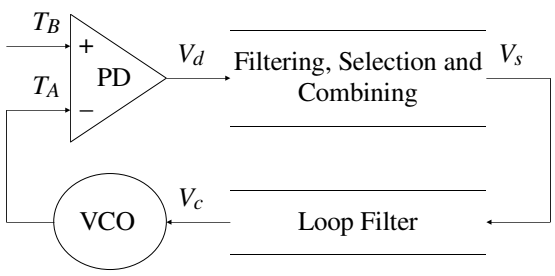


Figure 6. Phase-Lock Loop Model

Bandwidth control is necessary to match the PLL dynamics to varying levels of timing noise due to the intrinsic stability of the local oscillator and the prevailing delay variances in the network. On one hand, the PLL must track room-temperature quartz oscillators found in common computing equipment, where the frequency may be accurate to only .01 percent and may vary several parts-per-million (ppm) as the result of normal room-temperature variations. On the other hand, after the frequency errors have been tracked for several days, and assuming the local oscillator is appropriately compensated, the loop must maintain stabilities to the order of .01 ppm. The NTP PLL is designed to adapt automatically to these regimes by measuring the dispersions and adjusting $\tau$ over a five-octave range. Design details are discussed in [22] and performance assessed in [24].

## 6. NTP in the Internet System

The penetration of NTP in the Internet has steadily increased over the last few years. It is estimated that well over 2000 hosts presently synchronize local clocks to UTC using NTP and the Internet primary servers. In this section an overview of the various NTP implementations and subnet configurations is presented along with an evaluation of performance expected in routine operation.

The Fuzzball [21] is a software package consisting of a fast, compact operating system and an array of application programs for network protocol development, testing and evaluation. It usually runs on a DEC LSI-11 personal workstation, which functions as an experiment platform capable of millisecond timing accuracies and supports several types of timecode receivers and precision timebases. Since NTP and its forebears were developed and tested on the Fuzzball, the present implementation is the reference one for the NTP specification. An implementation of NTP for Unix systems was built by Michael Petry and Louis Mamakos at the University of Maryland. An implementation of NTP for Unix systems and for a dedicated Motorola 68000 microprocessor was built by Dennis Ferguson at the University of Toronto. Both Unix implementations adjust the local-clock phase and frequency using kernel primitives designed for this purpose and support various types of timecode receivers. Other

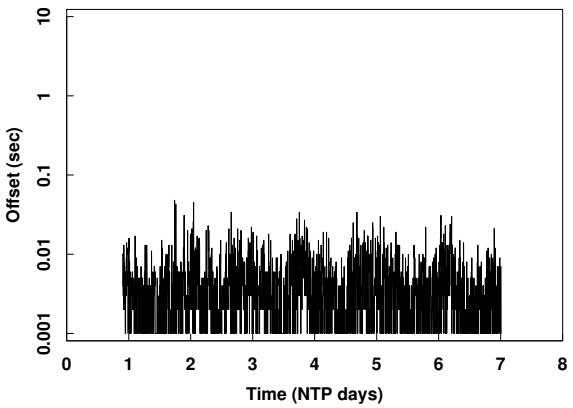


Figure 7. Filtered Offsets

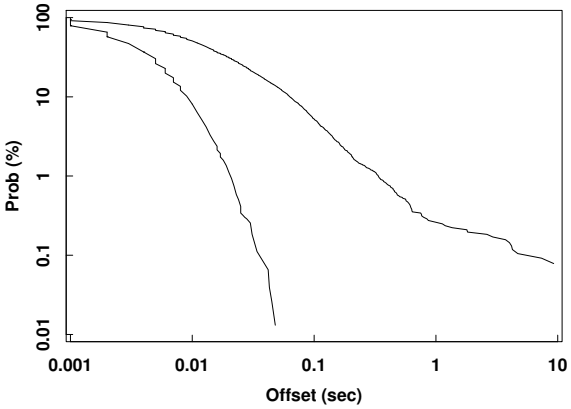


Figure 8. Error Distribution

implementations are in progress at Hewlett-Packard Laboratories, University College London and University of Delaware.

The NTP primary synchronization subnet now operating in the Internet consists of over two dozen Fuzzball and Unix primary time servers located in the U.S., Canada, the United Kingdom and Norway. All servers are synchronized to UTC via radio or satellite. Two servers use calibrated atomic clocks and two use LORAN-C timing receivers as precision timebases. Six servers are connected directly to national backbone networks, including NSFNET and ARPANET, and are intended for ubiquitous access, while the remainder are connected to regional networks and intended for regional and local access. All primary servers continuously exchange NTP messages with most of the other primary servers, which provides an exceptional level of redundancy and protection against failures. For instance, if a timecode receiver fails, a primary (stratum-1) server synchronizes via NTP to the nearest primary peer and continues operation as a secondary (stratum-2) server. If a primary server turns falseticker, discrepancies become apparent to its NTP peers, which then deselect the server as the result of the algorithms described previously.

The NTP secondary synchronization subnet presently includes an estimated total of over 2000 secondary time servers using some thousands of transmission paths on hundreds of networks. A secondary server operating at stratum $n > 1$ ordinarily operates with at least three peers, two at stratum $n - 1$ and one or more at stratum $n$. In the most robust configurations a set of servers agree to provide backup service for each other, so run NTP with some of the stratum-$(n - 1)$ servers and some of the other stratum-$n$ servers in the same set. In a typical example configuration used at the University of Illinois and the University of Delaware the institution operates three stratum-2 campus servers, each operating with two out of six different stratum-1 primary servers and with each other. The three campus servers in turn provide synchronization for several stratum-3 department servers, each operating with all three campus servers. Department servers, many of which also function as file servers, then deliver time to possibly hundreds of stratum-4 workstations in client/server or multicast modes.

As part of normal operations the Fuzzball time servers monitor delay and offset data from each of their peers. Periodically, these data are collected and analyzed to construct scatter diagrams, time-series diagrams and distribution functions. Scatter diagrams such as Figure 5 have proven exquisitely sensitive indicators of network performance and possible malfunctions. Time-series diagrams showing absolute offsets such as Figure 7, constructed from the same data as Figure 5, are useful for assessing algorithm performance and systematic errors. Distribution functions plotted on log-log axes such as Figure 8, also constructed from the same data, are useful in evaluating the performance of data-filtering algorithms. The figure shows the absolute raw offsets (upper curve) and filtered offsets (lower curve), from which it is apparent that the maximum error after the filter is less than about 30 ms for all but about one percent of the samples and less than about 50 ms for all samples. A companion paper [24] contains an extended discussion of performance issues and concludes that, using the adaptive-parameter PLL model described above together with the new combining algorithm, timing accuracies to a few milliseconds and frequency stabilities to a few milliseconds per day are regularly achieved.

## 7. Future Directions

The IRIG-H timecode format established in 1970 and used since then by NBS/NIST radio broadcast services does not include either year information or advance notice of leap-second insertion. Currently, this information must be provided at the primary servers by other means. It is reported (personal communication) that this information will soon be available in at least some radio

services. In fact, the recently introduced NIST telephone time service [26] already includes both the year and advance leap-second information.

The current mechanism of time delivery using dedicated radio systems and multifunction radionavigation and land-resources satellite systems requires relatively expensive timecode receivers subject to occasional disruption due to propagation path or radio failure. A plan once proposed by NIST using national television networks for time transfer has been generally thwarted by the growing use of buffered frame regeneration at the local stations. However, the growing penetration of cable television systems suggests a new opportunity for time distribution, such as providing incentives for cable operators to rebroadcast WWV, for example. An agenda should be pursued to promote the installation of NTP primary servers with Internet connectivity at various national standards laboratories. In fact, a pilot project is now in operation at the Norwegian Telecommunications Administration Research Establishment, in which Fuzzball primary NTP servers are synchronized directly to the Norwegian national standards.

As experience accumulates, improvements are being made continuously to the filtering and selection algorithms described in this paper. Recent improvements now being tested include engineered budgets for reading errors and skew-error accumulation, as well as an improved peer-selection algorithm based on the work of Marzullo and Owicki [18]. The goal is to provide reliable timing and timing-error information while preserving correctness, stability and accuracy. There may also be room for additional improvements in the offset-combination algorithm recently introduced, for example, as well as methods to compensate for asymmetric delays commonly found on Internet paths. Other improvements being considered include automatic subnet configuration and dynamic activation of peer associations when other peer associations become unreachable. These features are intended to reduce the network overhead when a large number of possible peers are available, but only a few are needed for reliable synchronization.

At present, NTP support is available only for Fuzzball and Unix systems. Support is needed for other systems, including mainframes and personal workstations of various manufacture. While NTP has been evolved within the Internet protocol suite, there is obvious application to the OSI protocol suite, in particular the protocols of the connectionless (CNLS) branch of that suite. Perhaps the most attractive methodology would be to integrate NTP functions directly into the ES-IS and IS-IS routing functions and network management systems.

## 8. Acknowledgments

Acknowledgments are due the referees for valuable suggestions on this paper. Thanks are due to the tribe of volunteer timetrekkers, too numerous to list here, who have assisted in the implementation and testing projects leading to the deployment of NTP. Thanks are also due the U.S. Coast Guard, who kindly provided a cesium clock and LORAN-C receiver on loan, as well as the U.S. Naval Observatory, who provided calibration assistance and much useful advice.

## 9. References

1. Allan, D.W., J.E. Gray and H.E. Machlan. The National Bureau of Standards atomic time scale: generation, stability, accuracy and accessibility. In: Blair, B.E. (Ed.). *Time and Frequency Theory and Fundamentals*. National Bureau of Standards Monograph 140, U.S. Department of Commerce, 1974, 205-231.

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

3. Bertsekas, D., and R. Gallager. *Data Networks*. Prentice-Hall, Englewood Cliffs, NJ, 1987.

4. Blair, B.E. Time and frequency dissemination: an overview of principles and techniques. In: Blair, B.E. (Ed.). *Time and Frequency Theory and Fundamentals*. National Bureau of Standards Monograph 140, U.S. Department of Commerce, 1974, 233-313.

5. Cole, R., and C. Foxcroft. An experiment in clock synchronisation. *The Computer Journal 31*, 6 (1988), 496-502.

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

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

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

9. *Data Encryption Standard*. Federal Information Processing Standards Publication 46. National Bureau of Standards, U.S. Department of Commerce, 1977.

10. *DES Modes of Operation*. Federal Information Processing Standards Publication 81. National Bu-

reau of Standards, U.S. Department of Commerce, December 1980.

11. Gusella, R., and S. Zatti. TEMPO - A network time controller for a distributed Berkeley UNIX system. *IEEE Distributed Processing Technical Committee Newsletter 6, NoSI-2* (June 1984), 7-15. Also in: *Proc. Summer 1984 USENIX* (Salt Lake City, June 1984).

12. Halpern, J.Y., B. Simons, R. Strong and D. Dolly. Fault-tolerant clock synchronization. *Proc. Third Annual ACM Sympos. on Principles of Distributed Computing* (August 1984), 89-102.

13. Kopetz, H., and W. Ochsenreiter. Clock synchronization in distributed real-time systems. *IEEE Trans. Computers C-36,* 8 (August 1987), 933-939.

14. Lamport, L., Time, clocks and the ordering of events in a distributed system. *Comm. ACM 21, 7* (July 1978), 558-565.

15. Lamport, L., and P.M. Melliar-Smith. Synchronizing clocks in the presence of faults. *JACM 32, 1* (January 1985), 52-78.

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

17. Lundelius, J., and N.A. Lynch. A new fault-tolerant algorithm for clock synchronization. *Proc. Third Annual ACM Sympos. on Principles of Distributed Computing* (August 1984), 75-88.

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

19. Mills, D.L. Algorithms for synchronizing network clocks. DARPA Network Working Group Report RFC-956, M/A-COM Linkabit, September 1985.

20. Mills, D.L. Experiments in network clock synchronization. DARPA Network Working Group Report RFC-957, M/A-COM Linkabit, September 1985.

21. Mills, D.L. The Fuzzball. *Proc. ACM SIGCOMM 88 Symposium* (Palo Alto, CA, August 1988), 115-122.

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

23. Mills, D.L. Internet time synchronization: the Network Time Protocol. DARPA Network Working Group Report RFC-1129, University of Delaware, October 1989.

24. 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.

25. Mitra, D. Network synchronization: analysis of a hybrid of master-slave and mutual synchronization. *IEEE Trans. Communications COM-28, 8* (August 1980), 1245-1259.

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

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

28. Postel, J. Daytime protocol. DARPA Network Working Group Report RFC-867, USC Information Sciences Institute, May 1983.

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

30. Rickert, N.W. Non Byzantine clock synchronization - a programming experiment. *ACM Operating Systems Review 22, 1* (January 1988), 73-78.

31. Schneider, F.B. A paradigm for reliable clock synchronization. Department of Computer Science Technical Report TR 86-735, Cornell University, February 1986.

32. Smith, J. *Modern Communications Circuits*. McGraw-Hill, New York, NY, 1986.

33. Srikanth, T.K., and S. Toueg. Optimal clock synchronization. *JACM 34, 3* (July 1987), 626-645.

34. Su, Z. A specification of the Internet protocol (IP) timestamp option. DARPA Network Working Group Report RFC-781. SRI International, May 1981.