Network Working Group
Request for Comments: 958

D.L. Mills
M/A-COM Linkabit

September 1985

Network Time Protocol (NTP)

Status of this Memo

This RFC suggests a proposed protocol for the ARPA-Internet community, and requests discussion and suggestions for improvements. Distribution of this memo is unlimited.

Table of Contents

1. Introduction

This document describes the Network Time Protocol (NTP), a protocol for synchronizing a set of network clocks using a set of distributed clients and servers. NTP is built on the User Datagram Protocol (UDP) [13], which provides a connectionless transport mechanism. It is evolved from the Time Protocol [7] and the ICMP Timestamp message [6] and is a suitable replacement for both.

NTP provides the protocol mechanisms to synchronize time in principle to precisions in the order of nanoseconds while preserving a non-ambiguous date, at least for this century. The protocol includes provisions to specify the precision and estimated error of the local clock and the characteristics of the reference clock to which it may be synchronized. However, the protocol itself specifies only the data representation and message formats and does not specify the synchronizing algorithms or filtering mechanisms.

Other mechanisms have been specified in the Internet protocol suite to record and transmit the time at which an event takes place, including the Daytime protocol [8] and IP Timestamp option [9]. The NTP is not meant to displace either of these mechanisms. Additional information on network time synchronization can be found in the

References at the end of this document. An earlier synchronization protocol is discussed in [3] and synchronization algorithms in [2], [5], [10] and [12]. Experimental results on measured roundtrip delays and clock offsets in the Internet are discussed in [4] and [11]. A comprehensive mathematical treatment of clock synchronization can be found in [1].

2. Service Model

The intent of the service for which this protocol is designed is to connect a few primary reference clocks, synchronized by wire or radio to national standards, to centrally accessable resources such as gateways. These gateways would use NTP between them to cross-check the primary clocks and mitigate errors due to equipment or propagation failures. Some number of local-net hosts, serving as secondary reference clocks, would run NTP with one or more of these gateways. In order to reduce the protocol overhead, these hosts would redistribute time to the remaining local-net hosts. In the interest of reliability selected hosts might be equipped with less accurate but less expensive radio clocks and used for backup in case of failure of the primary and/or secondary clocks or communication paths between them.

In the normal configuration a subnetwork of primary and secondary clocks will assume a hierarchical organization with the more accurate clocks near the top and the less accurate below. NTP provides information that can be used to organize this hierarchy on the basis of precision or estimated error and even to serve as a rudimentary routing algorithm to organize the subnetwork itself. However, the NTP protocol does not include a specification of the algorithms for doing this, which is left as a topic for further study.

3. Protocol Overview

There is no provision for peer discovery, acquisition, or authentication in NTP. Data integrity is provided by the IP and UDP checksums. No reachability, circuit-management, duplicate-detection or retransmission facilities are provided or necessary. The service can operate in a symmetric mode, in which servers and clients are indistinguishable yet maintain a small amount of state information, or in an unsymmetric mode in which servers need maintain no client state other than that contained in the client request. Moreover, only a single NTP message format is necessary, which simplifies implementation and can be used in a variety of solicited or unsolicited polling mechanisms.

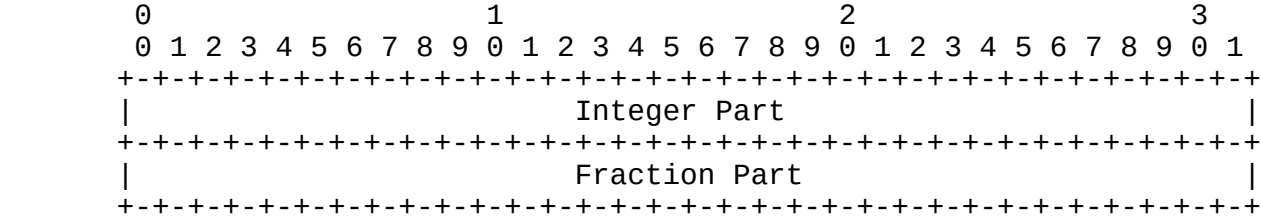
In what may be the most common (unsymmetric) mode a client sends an

NTP message to one or more servers and processes the replies as received. The server interchanges addresses and ports, fills in or overwrites certain fields in the message, recalculates the checksum and returns it immediately. Information included in the NTP message allows each client/server peer to determine the timekeeping characteristics of its other peers, including the expected accuracies of their clocks. Using this information each peer is able to select the best time from possibly several other clocks, update the local clock and estimate its accuracy.

It should be recognized that clock synchronization requires by its nature long periods and multiple comparisons in order to maintain accurate timekeeping. While only a few comparisons are usually adequate to maintain local time to within a second, primarily to protect against broken hardware or synchronization failure, periods of hours or days and tens or hundreds of comparisons are required to maintain local time to within a few tens of milliseconds. Fortunately, the frequency of comparisons can be quite small and almost always non-intrusive to normal network operations.

4. State Variables and Formats

NTP timestamps are represented as a 64-bit fixed-point number, in seconds relative to 0000 UT on 1 January 1900. The integer part is in the first 32 bits and the fraction part in the last 32 bits, as shown in the following diagram.



This format allows convenient multiple-precision arithmetic and conversion to Time Protocol representation (seconds), but does complicate the conversion to ICMP Timestamp message representation (milliseconds). The low-order fraction bit increments at about 0.2-nanosecond intervals, so a free-running one-millisecond clock will be in error only a small fraction of one part per million, or less than a second per year.

In some cases a particular timestamp may not be available, such as when the protocol first starts up. In these cases the 64-bit field is set to zero, indicating the value is invalid or undefined.

Following is a description of the various data items used in the protocol. Details of packet formats are presented in the Appendices.

Leap Indicator

This is a two-bit code warning of an impending leap-second to be inserted in the internationally coordinated Standard Time broadcasts. A leap-second is occasionally added or subtracted from Standard Time, which is based on atomic clocks, to maintain agreement with Earth rotation. When necessary, the corrections are notified in advance and executed at the end of the last day of the month in which notified, usually June or December. When a correction is executed the first minute of the following day will have either 59 or 61 seconds.

Status

This is a six-bit code indicating the status of the local clock. Values are assigned to indicate whether it is operating correctly or in one of several error states.

Reference Clock Type

This is an eight-bit code identifying the type of reference clock used to set the local clock. Values are assigned for primary clocks (locally synchronized to Standard Time), secondary clocks (remotely synchronized via various network protocols) and even eyeball-and-wristwatch.

Precision

This is a 16-bit signed integer indicating the precision of the local clock, in seconds to the nearest power of two. For instance, a 60-Hz line-frequency clock would be assigned the value -6, while a 1000-Hz crystal clock would be assigned the value -10.

Estimated Error

This is a 32-bit fixed-point number indicating the estimated error of the local clock at the time last set. The value is in seconds, with fraction point between bits 15 and 16, and is computed by the sender based on the reported error of the reference clock, the precision and drift rate of the local clock and the time the local clock was last set. For statistical purposes this quantity can be assumed equal to the estimated or computed standard deviation, as described in [12].

Estimated Drift Rate

This is a 32-bit signed fixed-point number indicating the estimated drift rate of the local clock. The value is dimensionless, with fraction point to the left of the high-order bit. While for most purposes this value can be estimated based on the hardware characteristics, it is possible to compute it quite accurately, as described in [12].

Reference Clock Identifier

This is a 32-bit code identifying the particular reference clock. The interpretation of its value depends on value of Reference Clock Type. In the case of a primary clock locally synchronized to Standard Time (type 1), the value is an ASCII string identifying the clock. In the case of a secondary clock remotely synchronized to an Internet host via NTP (type 2), the value is the 32-bit Internet address of that host. In other cases the value is undefined.

Reference Timestamp

This is a 64-bit timestamp established by the server or client host as the timestamp (presumably obtained from a reference clock) most recently used to update the local clock. If the local clock has never been synchronized, the value is zero.

Originate Timestamp

This is a 64-bit timestamp established by the client host and specifying the local time at which the request departed for the service host. It will always have a nonzero value.

Receive Timestamp

This is a 64-bit timestamp established by the server host and specifying the local time at which the request arrived from the client host. If no request has ever arrived from the client the value is zero.

Transmit Timestamp

This is a 64-bit timestamp established by the server host and specifying the local time at which the reply departed for the client host. If no request has ever arrived from the client the value is zero.

5. Protocol Operation

The intent of this document is to specify a standard for data representation and message format which can be used for a variety of synchronizing algorithms and filtering mechanisms. Accordingly, the information in this section should be considered a guide, rather than a concise specification. Nevertheless, it is expected that a standard Internet distributed timekeeping protocol with concisely specified synchronizing and filtering algorithms can be evolved from the information in this section.

5.1. Protocol Modes

The distinction between client and server is significant only in the way they interact in the request/response interchange. The same NTP message format is used by each peer and contains the same data relative to the other peer. In the unsymmetric mode the client periodically sends an NTP message to the server, which then responds within some interval. Usually, the server simply interchanges addresses and ports, fills in the required information and sends the message right back. Servers operating in the unsymmetric mode then need retain no state information between client requests.

In the symmetric mode the client/server distinction disappears. Each peer maintains a table with as many entries as active peers, each entry including a code uniquely identifying the peer (e.g. Internet address), together with status information and a copy of the Originate Timestamp and Receive Timestamp values last received from that peer. The peer periodically sends an NTP message to each of these peers including the latest copy of these timestamps. The interval between sending NTP messages is managed solely by the sending peer and is unaffected by the arrival of NTP messages from other peers.

The mode assumed by a peer can be determined by inspection of the UDP Source Port and Destination Port fields (see Appendix A). If both of these fields contain the NTP service-port number 123, the peer is operating in symmetric mode. If they are different and the Destination Port field contains 123, this is a client request and the receiver is expected to reply in the manner described above. If they are different and the Source Port field contains 123, this is a server reply to a previously sent client request.

5.2. Message Processing

The significant events of interest in NTP occur usually near the times the NTP messages depart and arrive the client/server. In order to maintain the highest accuracy it is important that the timestamps associated with these events be computed as close as possible to the hardware or software driver associated with the communications link and, in particular, that departure timestamps be recomputed for each retransmission, if used at the link level.

An NTP message is constructed as follows (see Appendix B). The source peer constructs the UDP header and the LI, Status, Reference Clock Type and Precision fields in the NTP data portion. Next, it determines the current synchronizing source and constructs the Type and Reference Clock Identifier fields. From its timekeeping algorithm (see [12] for examples) it determines the Reference Timestamp, Estimated Error and Estimated Drift Rate fields. Then it copies into the Receive Timestamp and Transmit Timestamp fields the data saved from the latest message received from the destination peer and, finally, computes the Originate Timestamp field.

The destination peer calculates the roundtrip delay and clock offset relative to the source peer as follows. Let t1, t2 and t3 represent the contents of the Originate Timestamp, Receive Timestamp and Transmit Timestamp fields and t4 the local time the NTP message is received. Then the roundtrip delay d and clock offset c is:

$$d = (t4 - t1) - (t3 - t2) \quad \text{and} \quad c = (t2 - t1 + t3 - t4)/2 .$$

The implicit assumption in the above is that the one-way delay is statistically half the roundtrip delay and that the intrinsic drift rates of both the client and server clocks are small and close to the same value.

5.3. Network Considerations

The client/server peers have an opportunity to learn a good deal about each other in the NTP message exchange. For instance, each can learn about the characteristics of the other clocks and select among them the most accurate to use as reference clock, compute the estimated error and drift rate and use this information to manage the dynamics of the subnetwork of clocks. An outline of a suggested mechanism is as follows:

Included in the table of timestamps for each peer are state

variables to indicate the precision, as well as the current estimated delay, offset, error and drift rate of its local clock. These variables are updated for each NTP message received from the peer, after which the estimated error is periodically recomputed on the basis of elapsed time and estimated drift rate.

Assuming symmetric mode, a polling interval is established for each peer, depending upon its normal synchronization source, precision and intrinsic accuracy, which might be determined in advance or even as the result of observation. The delay and clock-offset samples obtained can be filtered using maximum-likelihood techniques and algorithms described in [12].

From time to time a local-clock correction is computed from the offset data accumulated as above, perhaps using algorithms described in [10] and [12]. The correction causes the local clock to run slightly fast or slow to the corrected time or to jump instantaneously to the correct time, depending on the magnitude of the correction. See [5] and [11] for a discussion of local-clock implementation models and synchronizing algorithms. Note that the expectation here is that all network clocks are maintained by these algorithms, so that manual intervention is not normally required.

As a byproduct of the above operations an estimate of local-clock error and drift rate can be computed. Note that the magnitude of the error estimate must always be greater than that of the selected reference clock by at least the inherent precision of the local clock. It does not take a leap of imagination to see that the estimated error, delay or precision, or some combination of them, can be used as a metric for a simple min-hop-type routing algorithm to organize the subnetwork so as to provide the most accurate time to all peers and to provide automatic fallback to alternate sources in case of failures.

A variety of network configurations can be included in the above scenario. In the case of networks supporting a broadcast function, for example, NTP messages can be broadcast from one or more server hosts and picked up by client hosts sharing the same cable. Since typical networks of this type have a very low propagation delay, the roundtrip-delay calculation can be omitted and the clients need not broadcast in return. Thus, the requirement to save per-peer timestamps is removed, so that the Receive Timestamp and Transmit Timestamp fields can be set to zero and the local-clock offset becomes simply the difference between the Originate Timestamp and the local time upon arrival. In the case of long-delay satellite networks with broadcast capabilities,

an accurate measure of roundtrip delay is usually available from the channel-scheduling algorithm, so the per-peer timestamps again can be avoided.

5.4. Leap Seconds

A standard mechanism to effect leap-second correction is not a part of this specification. It is expected that the Leap Indicator bits would be set by hand in the primary reference clocks, then trickle down to all other clocks in the network, which would execute the correction at the specified time and reset the bits.

6. References

1. Lindsay, W.C., and A.V. Kantak. Network Synchronization of Random Signals. IEEE Trans. Comm. COM-28, 8 (August 1980), 1260-1266.

2. Mills, D.L. Time Synchronization in DCNET Hosts. DARPA Internet Project Report IEN-173, COMSAT Laboratories, February 1981.

3. Mills, D.L. DCNET Internet Clock Service. DARPA Network Working Group Report RFC-778, COMSAT Laboratories, April 1981.

4. Mills, D.L. Internet Delay Experiments. DARPA Network Working Group Report RFC-889, M/A-COM Linkabit, December 1983.

5. Mills, D.L. DCN Local-Network Protocols. DARPA Network Working Group Report RFC-891, M/A-COM Linkabit, December 1983.

6. Postel, J. Internet Control Message Protocol. DARPA Network Working Group Report RFC-792, USC Information Sciences Institute, September 1981.

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

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

9. Su, Z. A Specification of the Internet Protocol (IP) Timestamp Option. DARPA Network Working Group Report RFC-781. SRI International, May 1981.

10. Marzullo, K., and S. Owicki. Maintaining the Time in a Distributed System. ACM Operating Systems Review 19, 3 (July 1985), 44-54.

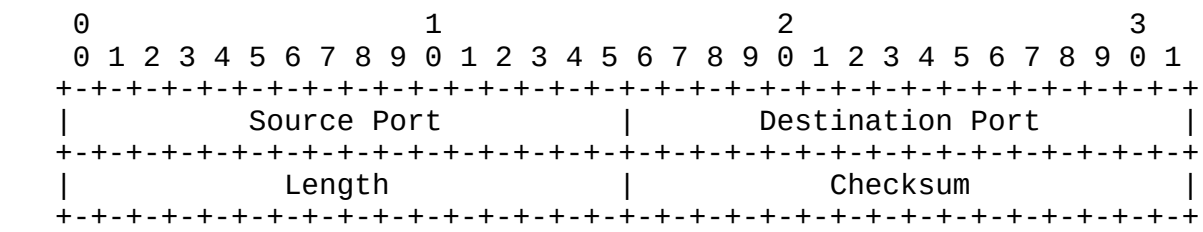
11. Mills, D.L. Experiments in Network Clock Synchronization. DARPA Network Working Group Report RFC-957, M/A-COM Linkabit, August 1985.

12. Mills, D.L. Algorithms for Synchronizing Network Clocks. DARPA Network Working Group Report RFC-956, M/A-COM Linkabit, September 1985.

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

Appendix A. UDP Header Format

An NTP packet consists of the UDP header followed by the NTP data portion. The format of the UDP header and the interpretation of its fields are described in [13] and are not part of the NTP specification. They are shown below for completeness.



Source Port

UDP source port number. In the case of unsymmetric mode and a client request this field is assigned by the client host, while for a server reply it is copied from the Destination Port field of the client request. In the case of symmetric mode, both the Source Port and Destination Port fields are assigned the NTP service-port number 123.

Destination Port

UDP destination port number. In the case of unsymmetric mode and a client request this field is assigned the NTP service-port number 123, while for a server reply it is copied form the Source Port field of the client request. In the case of symmetric mode, both the Source Port and Destination Port fields are assigned the NTP service-port number 123.

Length

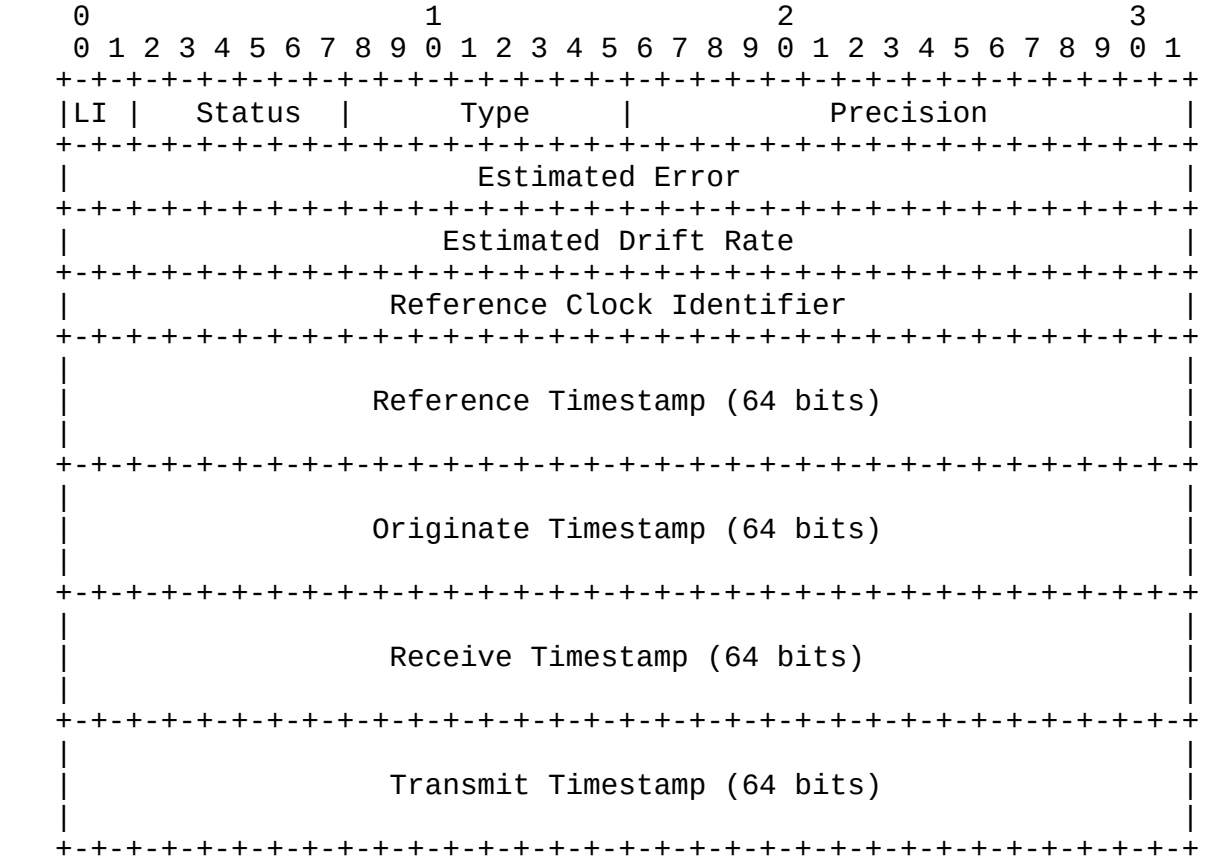
Length of the request or reply, including UDP header, in octets.

Checksum

Standard UDP checksum.

Appendix B. NTP Data Format

The format of the NTP data portion, which immediately follows the UDP header, is shown below along with a description of its fields.



Leap Indicator (LI)

Code warning of impending leap-second to be inserted at the end of the last day of the current month. Bits are coded as follows:

00	no warning
01	+1 second (following minute has 61 seconds)
10	-1 second (following minute has 59 seconds)
11	reserved for future use

Status

Code indicating status of local clock. Values are defined as follows:

0 clock operating correctly
1 carrier loss
2 synch loss
3 format error
4 interface (Type 1) or link (Type 2) failure
(additional codes reserved for future use)

Reference Clock Type
(Type)

Code identifying the type of reference clock. Values are defined as follows:

0 unspecified
1 primary reference (e.g. radio clock)
2 secondary reference using an Internet host via NTP
3 secondary reference using some other host or protocol
4 eyeball-and-wristwatch
(additional codes reserved for future use)

Precision

Signed integer in the range +32 to -32 indicating the precision of the local clock, in seconds to the nearest power of two.

Estimated Error

Fixed-point number indicating the estimated error of the local clock at the time last set, in seconds with fraction point between bits 15 and 16.

Estimated Drift Rate

Signed fixed-point number indicating the estimated drift rate of the local clock, in dimensionless units with fraction point to the left of the high-order bit.

Reference Clock
Identifier

Code identifying the particular reference clock. In the case of type 1 (primary reference), this is a left-justified, zero-filled ASCII string identifying the clock, for example:

WWVB WWVB radio clock (60 KHz)

GOES GOES satellite clock (468 HMz)
WWV WWV radio clock (2.5/5/10/15/20 MHz)
(and others as necessary)

In the case of type 2 (secondary reference) this is the 32-bit Internet address of the reference host. In other cases this field is reserved for future use and should be set to zero.

Reference Timestamp

Local time at which the local clock was last set or corrected.

Originate Timestamp

Local time at which the request departed the client host for the service host.

Receive Timestamp

Local time at which the request arrived at the service host.

Transmit Timestamp

Local time at which the reply departed the service host for the client host.