Network Working Group
Request for Comments: 4330
Obsoletes: 2030, 1769
Category: Informational

D. Mills
University of Delaware

January 2006

Simple Network Time Protocol (SNTP) Version 4 for IPv4, IPv6 and OSI

Status of This Memo

This memo provides information for the Internet community. It does not specify an Internet standard of any kind. Distribution of this memo is unlimited.

Copyright Notice

Abstract

This memorandum describes the Simple Network Time Protocol Version 4 (SNTPv4), which is a subset of the Network Time Protocol (NTP) used to synchronize computer clocks in the Internet. SNTPv4 can be used when the ultimate performance of a full NTP implementation based on RFC 1305 is neither needed nor justified. When operating with current and previous NTP and SNTP versions, SNTPv4 requires no changes to the specifications or known implementations, but rather clarifies certain design features that allow operation in a simple, stateless remote-procedure call (RPC) mode with accuracy and reliability expectations similar to the UDP/TIME protocol described in RFC 868.

This memorandum obsoletes RFC 1769, which describes SNTP Version 3 (SNTPv3), and RFC 2030, which describes SNTPv4. Its purpose is to correct certain inconsistencies in the previous documents and to clarify header formats and protocol operations for NTPv3 (IPv4) and SNTPv4 (IPv4, IPv6, and OSI), which are also used for SNTP. A further purpose is to provide guidance for home and business client implementations for routers and other consumer devices to protect the server population from abuse. A working knowledge of the NTPv3 specification, RFC 1305, is not required for an implementation of SNTP.

Table of Contents

1. Introduction

The Network Time Protocol Version 3 (NTPv3), specified in RFC 1305 [MIL92], is widely used to synchronize computer clocks in the global Internet. It provides comprehensive mechanisms to access national time and frequency dissemination services, organize the NTP subnet of servers and clients, and adjust the system clock in each participant. In most places of the Internet of today, NTP provides accuracies of 1-50 ms, depending on the characteristics of the synchronization source and network paths.

RFC 1305 specifies the NTP protocol machine in terms of events, states, transition functions and actions, and engineered algorithms to improve the timekeeping quality and to mitigate several synchronization sources, some of which may be faulty. To achieve accuracies in the low milliseconds over paths spanning major portions of the Internet, these intricate algorithms, or their functional equivalents, are necessary. In many applications, accuracies on the order of significant fractions of a second are acceptable. In simple home router applications, accuracies of up to a minute may suffice. In such cases, simpler protocols, such as the Time Protocol specified in RFC 868 [POS83], have been used for this purpose. These protocols involve an RPC exchange where the client requests the time of day and the server returns it in seconds past a known reference epoch.

NTP is designed for use by clients and servers with a wide range of capabilities and over a wide range of network jitter and clock frequency wander characteristics. Many users of NTP in the Internet of today use a software distribution available from www.ntp.org. The distribution, which includes the full suite of NTP options,

mitigation algorithms, and security schemes, is a relatively complex, real-time application. Although the software has been ported to a wide variety of hardware platforms ranging from personal computers to supercomputers, its sheer size and complexity is not appropriate for many applications. Accordingly, it is useful to explore alternative strategies using simpler software appropriate for less stringent accuracy expectations.

This memo describes the Simple Network Time Protocol Version 4 (SNTPv4), which is a simplified access paradigm for servers and clients using current and previous versions of NTP and SNTP. The access paradigm is identical to the UDP/TIME Protocol, and, in fact, it should be easy to adapt a UDP/TIME client implementation, say for a personal computer, to operate using SNTP. Moreover, SNTP is also designed to operate in a dedicated server configuration including an integrated radio clock. With careful design and control of the various latencies in the system, which is practical in a dedicated design, it is possible to deliver time accurate on the order of microseconds.

The only significant protocol change in SNTPv4 from previous SNTP versions is a modified header interpretation to accommodate Internet Protocol Version 6 (IPv6) (RFC 2460) and OSI (RFC 1629) addressing. However, SNTPv4 includes certain optional extensions to the basic NTP Version 3 (NTPv3) model, including a manycast mode and a public-key-based authentication scheme designed specifically for broadcast and manycast applications. Although the manycast mode is described in this memo, the authentication scheme is described in another RFC to be submitted later. Until such time that a definitive NTPv4 specification is published, the manycast and authentication features should be considered provisional. In addition, this memo introduces the kiss-o'-death message, which can be used by servers to suppress client requests as circumstances require.

When operating with current and previous versions of NTP and SNTP, SNTPv4 requires no changes to the protocol or implementations now running or likely to be implemented specifically for future NTP or SNTP versions. The NTP and SNTP packet formats are the same, and the arithmetic operations to calculate the client time, clock offset, and roundtrip delay are the same. To an NTP or SNTP server, NTP and SNTP clients are indistinguishable; to an NTP or SNTP client, NTP and SNTP servers are indistinguishable. Like NTP servers operating in non-symmetric modes, SNTP servers are stateless and can support large numbers of clients; however, unlike most NTP clients, SNTP clients normally operate with only a single server at a time.

The full degree of reliability ordinarily expected of NTP servers is possible only using redundant sources, diverse paths, and the crafted

algorithms of a full NTP implementation. It is strongly recommended that SNTP clients be used only at the extremities of the synchronization subnet. SNTP clients should operate only at the leaves (highest stratum) of the subnet and in configurations where no NTP or SNTP client is dependent on another SNTP client for synchronization. SNTP servers should operate only at the root (stratum 1) of the subnet, and then only in configurations where no other source of synchronization other than a reliable radio clock or telephone modem is available.

An important provision in this memo is the interpretation of certain NTP header fields that provide for IPv6 [DEE98] and OSI [COL94] addressing. The only significant difference between the NTP and SNTPv4 header formats is the four-octet Reference Identifier field, which is used primarily to detect and avoid synchronization loops. In all NTP and SNTP versions providing IPv4 addressing, primary servers use a four-character ASCII reference clock identifier in this field, whereas secondary servers use the 32-bit IPv4 address of the synchronization source. In SNTPv4 providing IPv6 and OSI addressing, primary servers use the same clock identifier, but secondary servers use the first 32 bits of the MD5 hash of the IPv6 or NSAP address of the synchronization source. A further use of this field is when the server sends a kiss-o'-death message, documented later in this memo.

> NTP Version 4 (NTPv4), now in deployment, but not yet the subject of a standards document, uses the same Reference Identifier field as SNTPv4.

In the case of OSI, the Connectionless Transport Service (CLTS) is used as in [ISO86]. Each SNTP packet is transmitted as the TS-Userdata parameter of a T-UNITDATA Request primitive. Alternately, the header can be encapsulated in a Transport Protocol Data Unit (TPDU), which itself is transported using UDP, as described in RFC 1240 [DOB91]. It is not advised that NTP be operated at the upper layers of the OSI stack, such as might be inferred from RFC 1698 [FUR94], as this could seriously degrade accuracy. With the header formats defined in this memo, it is in principle possible to interwork between servers and clients of one protocol family and another, although the practical difficulties may make this inadvisable.

> In the following, indented paragraphs such as this one contain information not required by the formal protocol specification, but considered good practice in protocol implementations.

This memo is organized as follows. Section 2 describes how the protocol works, the various modes, and how IP addresses and UDP ports are used. Section 3 describes the NTP timestamp format, and Section

4 the NTP message format. Section 5 summarizes SNTP client operations, and Section 6 summarizes SNTP server operations. Section 7 summarizes operation and management issues. Section 8 describes the kiss-o'-death message, newly minted with functions similar to the ICMP Source Quench and ICMP Destination Unreachable messages. Section 9 summarizes design issues important for good network citizenry and presents an example algorithm designed to give good reliability while minimizing network and server resource demands.

1.1. Specification of Requirements

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [BRA97].

2. Operating Modes and Addressing

Unless excepted in context, a reference to broadcast address means IPv4 broadcast address, IPv4 multicast group address, or IPv6 address of appropriate scope. Further information on the broadcast/multicast model is in RFC 1112 [DEE89]. Details of address format, scoping rules, etc., are beyond the scope of this memo. SNTPv4 can operate with either unicast (point to point), broadcast (point to multipoint), or manycast (multipoint to point) addressing modes. A unicast client sends a request to a designated server at its unicast address and expects a reply from which it can determine the time and, optionally, the roundtrip delay and clock offset relative to the server. A broadcast server periodically sends an unsolicited message to a designated broadcast address. A broadcast client listens on this address and ordinarily sends no requests.

Manycast is an extension of the anycast paradigm described in RFC 1546 [PAR93]. It is designed for use with a set of cooperating servers whose addresses are not known beforehand. The manycast client sends an ordinary NTP client request to a designated broadcast address. One or more manycast servers listen on that address. Upon receiving a request, a manycast server sends an ordinary NTP server reply to the client. The client then mobilizes an association for each server found and continues operation with all of them. Subsequently, the NTP mitigation algorithms operate to cast out all except the best three.

> Broadcast servers should respond to client unicast requests, as well as send unsolicited broadcast messages. Broadcast clients may send unicast requests in order to measure the network propagation delay between the server and client and then continue operation in listen-only mode. However, broadcast servers may

choose not to respond to unicast requests, so unicast clients should be prepared to abandon the measurement and assume a default value for the delay.

The client and server addresses are assigned following the usual IPv4, IPv6 or OSI conventions. For NTP multicast, the IANA has reserved the IPv4 group address 224.0.1.1 and the IPv6 address ending :101 with appropriate scope. The NTP broadcast address for OSI has yet to be determined. Notwithstanding the IANA reserved addresses, other multicast addresses can be used that do not conflict with others assigned in scope. The scoping, routing, and group membership procedures are determined by considerations beyond the scope of this memo.

It is important to adjust the time-to-live (TTL) field in the IP header of multicast messages to a reasonable value in order to limit the network resources used by this (and any other) multicast service. Only multicast clients in scope will receive multicast server messages. Only cooperating manycast servers in scope will reply to a client request. The engineering principles that determine the proper values to be used are beyond the scope of this memo.

In the case of SNTP as specified herein, there is a very real vulnerability that SNTP broadcast clients can be disrupted by misbehaving or hostile SNTP or NTP broadcast servers elsewhere in the Internet. It is strongly recommended that access controls and/or cryptographic authentication means be provided for additional security in such cases.

It is intended that IP broadcast addresses will be used primarily in IP subnets and LAN segments including a fully functional NTP server with a number of dependent SNTP broadcast clients on the same subnet, and that IP multicast group addresses will be used only in cases where the TTL is engineered specifically for each service domain. However, these uses are not integral to the SNTP specification.

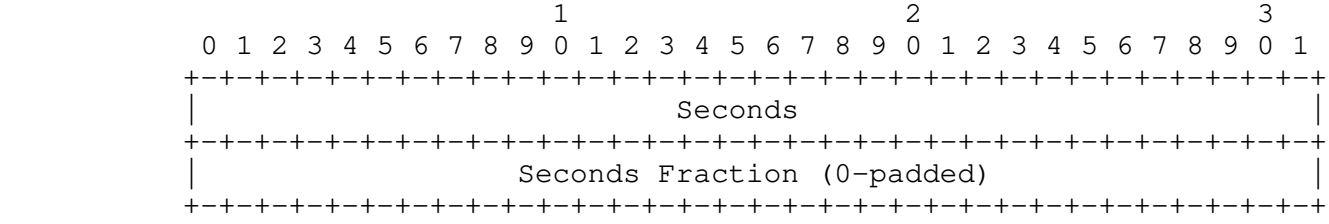
3. NTP Timestamp Format

SNTP uses the standard NTP timestamp format described in RFC 1305 and previous versions of that document. In conformance with standard Internet practice, NTP data are specified as integer or fixed-point quantities, with bits numbered in big-endian fashion from 0 starting at the left or most significant end. Unless specified otherwise, all quantities are unsigned and may occupy the full field width with an implied 0 preceding bit 0.

Because NTP timestamps are cherished data and, in fact, represent the main product of the protocol, a special timestamp format has been established. NTP timestamps are represented as a 64-bit unsigned fixed-point number, in seconds relative to 0h on 1 January 1900. The integer part is in the first 32 bits, and the fraction part in the last 32 bits. In the fraction part, the non-significant low-order bits are not specified and are ordinarily set to 0.

It is advisable to fill the non-significant low-order bits of the timestamp with a random, unbiased bitstring, both to avoid systematic roundoff errors and to provide loop detection and replay detection (see below). It is important that the bitstring be unpredictable by an intruder. One way of doing this is to generate a random 128-bit bitstring at startup. After that, each time the system clock is read, the string consisting of the timestamp and bitstring is hashed with the MD5 algorithm, then the non-significant bits of the timestamp are copied from the result.

The NTP format allows convenient multiple-precision arithmetic and conversion to UDP/TIME message (seconds), but does complicate the conversion to ICMP Timestamp message (milliseconds) and Unix time values (seconds and microseconds or seconds and nanoseconds). The maximum number that can be represented is 4,294,967,295 seconds with a precision of about 232 picoseconds, which should be adequate for even the most exotic requirements.



Note that since some time in 1968 (second 2,147,483,648), the most significant bit (bit 0 of the integer part) has been set and that the 64-bit field will overflow some time in 2036 (second 4,294,967,296). There will exist a 232-picosecond interval, henceforth ignored, every 136 years when the 64-bit field will be 0, which by convention is interpreted as an invalid or unavailable timestamp.

As the NTP timestamp format has been in use for over 20 years, it is possible that it will be in use 32 years from now, when the seconds field overflows. As it is probably inappropriate to archive NTP timestamps before bit 0 was set in 1968, a convenient way to extend the useful life of NTP timestamps is the following convention: If bit 0 is set, the UTC time is in the range 1968-2036, and UTC time is reckoned from 0h 0m 0s UTC on 1 January

1900. If bit 0 is not set, the time is in the range 2036-2104 and UTC time is reckoned from 6h 28m 16s UTC on 7 February 2036. Note that when calculating the correspondence, 2000 is a leap year, and leap seconds are not included in the reckoning.

The arithmetic calculations used by NTP to determine the clock offset and roundtrip delay require the client time to be within 34 years of the server time before the client is launched. As the time since the Unix base 1970 is now more than 34 years, means must be available to initialize the clock at a date closer to the present, either with a time-of-year (TOY) chip or from firmware.

4. Message Format

Both NTP and SNTP are clients of the User Datagram Protocol (UDP) specified in RFC 768 [POS80]. The structures of the IP and UDP headers are described in the cited specification documents and will not be detailed further here. The UDP port number assigned by the IANA to NTP is 123. The SNTP client should use this value in the UDP Destination Port field for client request messages. The Source Port field of these messages can be any nonzero value chosen for identification or multiplexing purposes. The server interchanges these fields for the corresponding reply messages.

This differs from the RFC 2030 specifications, which required both the source and destination ports to be 123. The intent of this change is to allow the identification of particular client implementations (which are now allowed to use unreserved port numbers, including ones of their choosing) and to attain compatibility with Network Address Port Translation (NAPT) described in RFC 2663 [SRI99] and RFC 3022 [SRI01].

Figure 1 is a description of the NTP and SNTP message format, which follows the IP and UDP headers in the message. This format is identical to the NTP message format described in RFC 1305, with the exception of the Reference Identifier field described below. For SNTP client messages, most of these fields are zero or initialized with pre-specified data. For completeness, the function of each field is briefly summarized below.

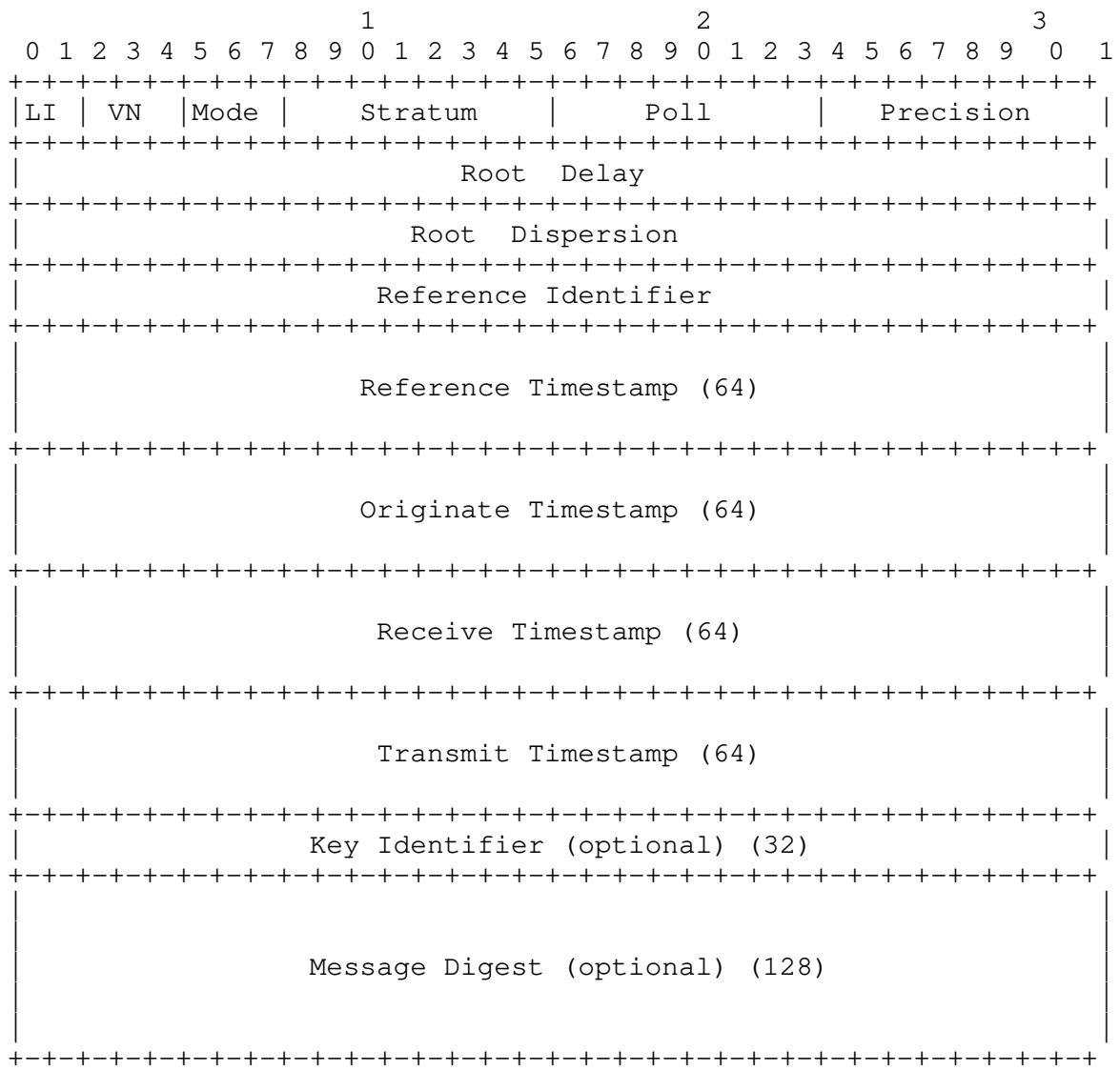


Figure 1. NTP Packet Header

Leap Indicator (LI): This is a two-bit code warning of an impending leap second to be inserted/deleted in the last minute of the current day. This field is significant only in server messages, where the values are defined as follows:

LI	Meaning
0	no warning
1	last minute has 61 seconds
2	last minute has 59 seconds
3	alarm condition (clock not synchronized)

On startup, servers set this field to 3 (clock not synchronized), and set this field to some other value when synchronized to the primary reference clock. Once set to a value other than 3, the field is never set to that value again, even if all synchronization sources become unreachable or defective.

Version Number (VN): This is a three-bit integer indicating the NTP/SNTP version number, currently 4. If necessary to distinguish between IPv4, IPv6, and OSI, the encapsulating context must be inspected.

Mode: This is a three-bit number indicating the protocol mode. The values are defined as follows:

Mode	Meaning
0	reserved
1	symmetric active
2	symmetric passive
3	client
4	server
5	broadcast
6	reserved for NTP control message
7	reserved for private use

In unicast and manycast modes, the client sets this field to 3 (client) in the request, and the server sets it to 4 (server) in the reply. In broadcast mode, the server sets this field to 5 (broadcast). The other modes are not used by SNTP servers and clients.

Stratum: This is an eight-bit unsigned integer indicating the stratum. This field is significant only in SNTP server messages, where the values are defined as follows:

Stratum	Meaning
0	kiss-o'-death message (see below)
1	primary reference (e.g., synchronized by radio clock)
2-15	secondary reference (synchronized by NTP or SNTP)
16-255	reserved

Poll Interval: This is an eight-bit unsigned integer used as an exponent of two, where the resulting value is the maximum interval between successive messages in seconds. This field is significant only in SNTP server messages, where the values range from 4 (16 s) to 17 (131,072 s -- about 36 h).

Precision: This is an eight-bit signed integer used as an exponent of two, where the resulting value is the precision of the system clock in seconds. This field is significant only in server messages, where the values range from -6 for mains-frequency clocks to -20 for microsecond clocks found in some workstations.

Root Delay: This is a 32-bit signed fixed-point number indicating the total roundtrip delay to the primary reference source, in seconds with the fraction point between bits 15 and 16. Note that this variable can take on both positive and negative values, depending on the relative time and frequency offsets. This field is significant only in server messages, where the values range from negative values of a few milliseconds to positive values of several hundred milliseconds.

Code	External Reference Source
LOCL	uncalibrated local clock
CESM	calibrated Cesium clock
RBDM	calibrated Rubidium clock
PPS	calibrated quartz clock or other pulse-per-second source
IRIG	Inter-Range Instrumentation Group
ACTS	NIST telephone modem service
USNO	USNO telephone modem service
PTB	PTB (Germany) telephone modem service
TDF	Allouis (France) Radio 164 kHz
DCF	Mainflingen (Germany) Radio 77.5 kHz
MSF	Rugby (UK) Radio 60 kHz
WWV	Ft. Collins (US) Radio 2.5, 5, 10, 15, 20 MHz
WWVB	Boulder (US) Radio 60 kHz
WWVH	Kauai Hawaii (US) Radio 2.5, 5, 10, 15 MHz
CHU	Ottawa (Canada) Radio 3330, 7335, 14670 kHz
LORC	LORAN-C radionavigation system
OMEG	OMEGA radionavigation system
GPS	Global Positioning Service

Figure 2. Reference Identifier Codes

Root Dispersion: This is a 32-bit unsigned fixed-point number indicating the maximum error due to the clock frequency tolerance, in seconds with the fraction point between bits 15 and 16. This field is significant only in server messages, where the values range from zero to several hundred milliseconds.

Reference Identifier: This is a 32-bit bitstring identifying the particular reference source. This field is significant only in server messages, where for stratum 0 (kiss-o'-death message) and 1 (primary server), the value is a four-character ASCII string, left justified and zero padded to 32 bits. For IPv4 secondary servers, the value is the 32-bit IPv4 address of the synchronization source. For IPv6 and OSI secondary servers, the value is the first 32 bits of the MD5 hash of the IPv6 or NSAP address of the synchronization source.

Primary (stratum 1) servers set this field to a code identifying the external reference source according to Figure 2. If the external reference is one of those listed, the associated code should be used. Codes for sources not listed can be contrived, as appropriate.

> In previous NTP and SNTP secondary servers and clients, this field was often used to walk-back the synchronization subnet to the root (primary server) for management purposes. In SNTPv4 with IPv6 or OSI, this feature is not available, because the addresses are longer than 32 bits, and only a hash is available. However, a walk-back can be accomplished using the NTP control message and the reference identifier field described in RFC 1305.

Reference Timestamp: This field is the time the system clock was last set or corrected, in 64-bit timestamp format.

Originate Timestamp: This is the time at which the request departed the client for the server, in 64-bit timestamp format.

Receive Timestamp: This is the time at which the request arrived at the server or the reply arrived at the client, in 64-bit timestamp format.

Transmit Timestamp: This is the time at which the request departed the client or the reply departed the server, in 64-bit timestamp format.

Authenticator (optional): When the NTP authentication scheme is implemented, the Key Identifier and Message Digest fields contain the message authentication code (MAC) information defined in Appendix C of RFC 1305.

5. SNTP Client Operations

An SNTP client can operate in unicast, broadcast, or manycast modes. In unicast mode, the client sends a request (NTP mode 3) to a designated unicast server and expects a reply (NTP mode 4) from that server. In broadcast client mode, it sends no request and waits for a broadcast (NTP mode 5) from one or more broadcast servers. In manycast mode, the client sends a request (NTP mode 3) to a designated broadcast address and expects a reply (NTP mode 4) from one or more manycast servers. The client uses the first reply received to establish the particular server for subsequent unicast operations. Later replies from this server (duplicates) or any other server are ignored. Other than the selection of address in the request, the operations of manycast and unicast clients are identical.

> Client requests are normally sent at intervals depending on the frequency tolerance of the client clock and the required accuracy. However, under no conditions should requests be sent at less than one minute intervals. Further discussion on this point is in Section 9.

A unicast or manycast client initializes the NTP message header, sends the request to the server, and strips the time of day from the Transmit Timestamp field of the reply. For this purpose, all the NTP header fields shown above are set to 0, except the Mode, VN, and optional Transmit Timestamp fields.

NTP and SNTP clients set the mode field to 3 (client) for unicast and manycast requests. They set the VN field to any version number that is supported by the server, selected by configuration or discovery, and that can interoperate with all previous version NTP and SNTP servers. Servers reply with the same version as the request, so the VN field of the request also specifies the VN field of the reply. A prudent SNTP client can specify the earliest acceptable version on the expectation that any server of that or a later version will respond. NTP Version 3 (RFC 1305) and Version 2 (RFC 1119) servers accept all previous versions, including Version 1 (RFC 1059). Note that Version 0 (RFC 959) is no longer supported by current and future NTP and SNTP servers.

Although setting the Transmit Timestamp field in the request to the time of day according to the client clock in NTP timestamp format is not necessary in a conforming client implementation, it is highly recommended in unicast and manycast modes. This allows a simple calculation to determine the propagation delay between the server and client and to align the system clock generally within a few tens of milliseconds relative to the server. In addition, this provides a

simple method for verifying that the server reply is in fact a legitimate response to the specific client request and thereby for avoiding replays. In broadcast mode, the client has no information to calculate the propagation delay or to determine the validity of the server, unless one of the NTP authentication schemes is used.

To calculate the roundtrip delay d and system clock offset t relative to the server, the client sets the Transmit Timestamp field in the request to the time of day according to the client clock in NTP timestamp format. For this purpose, the clock need not be synchronized. The server copies this field to the Originate Timestamp in the reply and sets the Receive Timestamp and Transmit Timestamp fields to the time of day according to the server clock in NTP timestamp format.

When the server reply is received, the client determines a Destination Timestamp variable as the time of arrival according to its clock in NTP timestamp format. The following table summarizes the four timestamps.

Timestamp Name	ID	When Generated
Originate Timestamp	T1	time request sent by client
Receive Timestamp	T2	time request received by server
Transmit Timestamp	T3	time reply sent by server
Destination Timestamp	T4	time reply received by client

The roundtrip delay d and system clock offset t are defined as:

$$d = (T4 - T1) - (T3 - T2) \qquad t = ((T2 - T1) + (T3 - T4)) / 2.$$

Note that in general both delay and offset are signed quantities and can be less than zero; however, a delay less than zero is possible only in symmetric modes, which SNTP clients are forbidden to use. The following table summarizes the required SNTP client operations in unicast, manycast, and broadcast modes. The recommended error checks are shown in the Reply and Broadcast columns in the table. The message should be considered valid only if all the fields shown contain values in the respective ranges. Whether to believe the message if one or more of the fields marked "ignore" contain invalid values is at the discretion of the implementation.

Field Name	Unicast/Manycast Request	Unicast/Manycast Reply	Broadcast
LI	0	0-3	0-3
VN	1-4	copied from request	1-4
Mode	3	4	5
Stratum	0	0-15	0-15
Poll	0	ignore	ignore
Precision	0	ignore	ignore
Root Delay	0	ignore	ignore
Root Dispersion	0	ignore	ignore
Reference Identifier	0	ignore	ignore
Reference Timestamp	0	ignore	ignore
Originate Timestamp	0	(see text)	ignore
Receive Timestamp	0	(see text)	ignore
Transmit Timestamp	(see text)	nonzero	nonzero
Authenticator	optional	optional	optional

Although not required in a conforming SNTP client implementation, it is wise to consider a suite of sanity checks designed to avoid various kinds of abuse that might happen as the result of server implementation errors or malicious attack. Following is a list of suggested checks.

1. When the IP source and destination addresses are available for the client request, they should match the interchanged addresses in the server reply.

2. When the UDP source and destination ports are available for the client request, they should match the interchanged ports in the server reply.

3. The Originate Timestamp in the server reply should match the Transmit Timestamp used in the client request.

4. The server reply should be discarded if any of the LI, Stratum, or Transmit Timestamp fields is 0 or the Mode field is not 4 (unicast) or 5 (broadcast).

5. A truly paranoid client can check that the Root Delay and Root Dispersion fields are each greater than or equal to 0 and less than infinity, where infinity is currently a cozy number like one second. This check avoids using a server whose synchronization source has expired for a very long time.

6. SNTP Server Operations

A SNTP server operating with either an NTP or SNTP client of the same or previous versions retains no persistent state. Because an SNTP server ordinarily does not implement the full suite of grooming and mitigation algorithms intended to support redundant servers and diverse network paths, it should be operated only in conjunction with a source of external synchronization, such as a reliable radio clock or telephone modem. In this case it operates as a primary (stratum 1) server.

A SNTP server can operate with any unicast, manycast, or broadcast address or any combination of these addresses. A unicast or manycast server receives a request (NTP mode 3), modifies certain fields in the NTP header, and sends a reply (NTP mode 4), possibly using the same message buffer as the request. A manycast server listens on the designated broadcast address, but uses its own unicast IP address in the source address field of the reply. Other than the selection of address in the reply, the operations of manycast and unicast servers are identical. Broadcast messages are normally sent at intervals from 64 s to 1024 s, depending on the expected frequency tolerance of the client clocks and the required accuracy.

Unicast and manycast servers copy the VN and Poll fields of the request intact to the reply and set the Stratum field to 1.

> Note that SNTP servers normally operate as primary (stratum 1) servers. Although operating at higher strata (up to 15) while synchronizing to an external source such as a GPS receiver is not forbidden, this is strongly discouraged.

If the Mode field of the request is 3 (client), the reply is set to 4 (server). If this field is set to 1 (symmetric active), the reply is set to 2 (symmetric passive). This allows clients configured in either client (NTP mode 3) or symmetric active (NTP mode 1) to interoperate successfully, even if configured in possibly suboptimal ways. For any other value in the Mode field, the request is

discarded. In broadcast (unsolicited) mode, the VN field is set to 4, the Mode field is set to 5 (broadcast), and the Poll field set to the nearest integer base-2 logarithm of the poll interval.

> Note that it is highly desirable that a broadcast server also supports unicast clients. This is so a potential broadcast client can calculate the propagation delay using a client/server exchange prior to switching to broadcast client (listen-only) mode. By design, a manycast server is also a unicast server. There does not seem to be a great advantage for a server to operate as both broadcast and manycast at the same time, although the protocol specification does not forbid it.

A broadcast or manycast server does not send packets if not synchronized to a correctly operating reference source. It may or may not respond to a client request if it is not synchronized, but the preferred option is to respond because this allows reachability to be determined regardless of synchronization state. If the server has never synchronized to a reference source, the LI field is set to 3 (unsynchronized). Once synchronized to a reference source, the LI field is set to one of the other three values and remains at the last value set even if the reference source becomes unreachable or turns faulty.

If the server is synchronized to a reference source, the Stratum field is set to 1, and the Reference Identifier field is set to the ASCII source identifier shown in Figure 2. If the server is not synchronized, the Stratum field is set to zero, and the Reference Identifier field is set to an ASCII error identifier described below.

The Precision field is set to reflect the maximum reading error of the system clock. For all practical cases it is computed as the negative base-2 logarithm of the number of significant bits to the right of the decimal point in the NTP timestamp format. The Root Delay and Root Dispersion fields are set to 0 for a primary server.

The timestamp fields in the server message are set as follows. If the server is unsynchronized or first coming up, all timestamp fields are set to zero, with one exception. If the message is a reply to a previously received client request, the Transmit Timestamp field of the request is copied unchanged to the Originate Timestamp field of the reply. It is important that this field be copied intact, as an NTP or SNTP client uses it to avoid bogus messages.

If the server is synchronized, the Reference Timestamp is set to the time the last update was received from the reference source. The Originate Timestamp field is set as in the unsynchronized case above. The Transmit Timestamp field is set to the time of day when the

message is sent. In broadcast messages the Receive Timestamp field is set to zero and copied from the Transmit Timestamp field in other messages. The following table summarizes these actions.

Field Name	Unicast/Manycast Request	Unicast/Manycast Reply	Broadcast
LI	ignore	as needed	as needed
VN	1-4	copied from request	4
Mode	3	4	5
Stratum	ignore	1	1
Poll	ignore	copied from request	log2 poll interval
Precision	ignore	-log2 server significant bits	-log2 server significant bits
Root Delay	ignore	0	0
Root Dispersion	ignore	0	0
Reference Identifier	ignore	source ident	source ident
Reference Timestamp	ignore	time of last source update	time of last source update
Originate Timestamp	ignore	copied from transmit timestamp	0
Receive Timestamp	ignore	time of day	0
Transmit Timestamp	(see text)	time of day	time of day
Authenticator	optional	optional	optional

There is some latitude on the part of most clients to forgive invalid timestamps, such as might occur when the server is first coming up or during periods when the reference source is inoperative. The most important indicator of an unhealthy server is the Stratum field, in

which a value of 0 indicates an unsynchronized condition. When this value is displayed, clients should discard the server message, regardless of the contents of other fields.

7. Configuration and Management

Initial setup for SNTP servers and clients can be done using a web client, if available, or a serial port, if not. Some folks hoped that in-service management of NTP and SNTPv4 servers and clients could be performed using SNMP and a suitable MIB to be published, and this has happened in some commercial SNTP servers. But, the means that have been used in the last two decades and probably will be used in the next is the NTP control and monitoring protocol defined in RFC 1305. Ordinarily, SNTP servers and clients are expected to operate with little or no site-specific configuration, other than specifying the client IP address, subnet mask, and gateway.

Unicast clients must be provided with one or more designated server names or IP addresses. If more than one server is provided, one can be used for active operation and one of the others for backup should the active one fail or show an error condition. It is not normally useful to use more than one server at a time, as with millions of SNTP-enabled devices expected in the near future, such use would represent unnecessary drain on network and server resources.

Broadcast servers and manycast clients must be provided with the TTL and local broadcast or multicast group address. Unicast and manycast servers and broadcast clients may be configured with a list of address-mask pairs for access control, so that only those clients or servers known to be trusted will be accepted. Multicast servers and clients must implement the IGMP protocol and be provided with the local broadcast or multicast group address as well. The configuration data for cryptographic authentication is beyond the scope of this memo.

There are several scenarios that provide automatic server discovery and selection for SNTP clients with no pre-specified server configuration. For instance, a role server with CNAME such as pool.ntp.org returns a randomized list of volunteer secondary server addresses, and the client can select one or more as candidates. For an IP subnet or LAN segment including an NTP or SNTP server, SNTP clients can be configured as broadcast clients. The same approach can be used with multicast servers and clients. In both cases, provision of an access control list is a good way to ensure that only trusted sources can be used to set the system clock.

In another scenario suitable for an extended network with significant network propagation delays, clients can be configured for manycast addresses, both upon initial startup and after some period when the currently selected unicast source has not been heard. Following the defined protocol, the client binds to the server from which the first reply is received and continues operation in unicast mode.

8. The Kiss-o'-Death Packet

In the rambunctious Internet of today, it is imperative that some means be available to tell a client to stop making requests and to go somewhere else. A recent experience involved a large number of home/office routers all configured to use a particular university time server. Under some error conditions, a substantial fraction of these routers would send packets at intervals of one second. The resulting traffic spike was dramatic, and extreme measures were required to diagnose the problem and to bring it under control. The conclusion is that clients must respect the means available to targeted servers to stop them from sending packets.

According to the NTP specification RFC 1305, if the Stratum field in the NTP header is 1, indicating a primary server, the Reference Identifier field contains an ASCII string identifying the particular reference clock type. However, in RFC 1305 nothing is said about the Reference Identifier field if the Stratum field is 0, which is called out as "unspecified". However, if the Stratum field is 0, the Reference Identifier field can be used to convey messages useful for status reporting and access control. In NTPv4 and SNTPv4, packets of this kind are called Kiss-o'-Death (KoD) packets, and the ASCII messages they convey are called kiss codes. The KoD packets got their name because an early use was to tell clients to stop sending packets that violate server access controls.

In general, an SNTP client should stop sending to a particular server if that server returns a reply with a Stratum field of 0, regardless of kiss code, and an alternate server is available. If no alternate server is available, the client should retransmit using an exponential-backoff algorithm described in the next section.

The kiss codes can provide useful information for an intelligent client. These codes are encoded in four-character ASCII strings left justified and zero filled. The strings are designed for character displays and log files. Usually, only a few of these codes can occur with SNTP clients, including DENY, RSTR, and RATE. Others can occur more rarely, including INIT and STEP, when the server is in some special temporary condition. Figure 3 shows a list of the kiss codes currently defined. These are for informational purposes only; the list might be modified or extended in the future.

Code	Meaning
ACST	The association belongs to a anycast server
AUTH	Server authentication failed
AUTO	Autokey sequence failed
BCST	The association belongs to a broadcast server
CRYP	Cryptographic authentication or identification failed
DENY	Access denied by remote server
DROP	Lost peer in symmetric mode
RSTR	Access denied due to local policy
INIT	The association has not yet synchronized for the first time
MCST	The association belongs to a manycast server
NKEY	No key found. Either the key was never installed or is not trusted
RATE	Rate exceeded. The server has temporarily denied access because the client exceeded the rate threshold
RMOT	Somebody is tinkering with the association from a remote host running ntpdc. Not to worry unless some rascal has stolen your keys
STEP	A step change in system time has occurred, but the association has not yet resynchronized

Figure 3. Kiss Codes

9. On Being a Good Network Citizen

SNTP and its big brother NTP have been in explosive growth over the last few years, mirroring the growth of the Internet. Just about every Internet appliance has some kind of NTP support, including Windows XP, Cisco routers, embedded controllers, and software systems of all kinds. This is the first edition of the SNTP RFC where it has become necessary to lay down rules of engagement in the form of design criteria for SNTP client implementations. This is necessary to educate software developers regarding the proper use of Internet time server resources as the Internet expands and demands on time servers increase, and to prevent the recurrence of the sort of problem mentioned above.

10. Best Practices

NTP and SNTP clients can consume considerable network and server resources if they are not good network citizens. There are now consumer Internet commodity devices numbering in the millions that are potential customers of public and private NTP and SNTP servers. Recent experience strongly suggests that device designers pay particular attention to minimizing resource impacts, especially if large numbers of these devices are deployed. The most important

design consideration is the interval between client requests, called the poll interval. It is extremely important that the design use the maximum poll interval consistent with acceptable accuracy.

1. A client MUST NOT under any conditions use a poll interval less than 15 seconds.

2. A client SHOULD increase the poll interval using exponential backoff as performance permits and especially if the server does not respond within a reasonable time.

3. A client SHOULD use local servers whenever available to avoid unnecessary traffic on backbone networks.

4. A client MUST allow the operator to configure the primary and/or alternate server names or addresses in addition to or in place of a firmware default IP address.

5. If a firmware default server IP address is provided, it MUST be a server operated by the manufacturer or seller of the device or another server, but only with the operator's permission.

6. A client SHOULD use the Domain Name System (DNS) to resolve the server IP addresses, so the operator can do effective load balancing among a server clique and change IP address binding to canonical names.

7. A client SHOULD re-resolve the server IP address at periodic intervals, but not at intervals less than the time-to-live field in the DNS response.

8. A client SHOULD support the NTP access-refusal mechanism so that a server kiss-o'-death reply in response to a client request causes the client to cease sending requests to that server and to switch to an alternate, if available.

The following algorithm can be used as a pattern for specific implementations. It uses the following variables:

Timer: This is a counter that decrements at a fixed rate. When it reaches zero, a packet is sent, and the timer is initialized with the timeout for the next packet.

Maximum timeout: This is the maximum timeout determined from the given oscillator frequency tolerance and the required accuracy.

Server Name: This is the DNS name of the server. There may be more than one of them, to be selected by some algorithm not considered here.

Server IP Address: This is the IPv4, IPv6, or OSI address of the server.

If the firmware or documentation includes specific server names, the names should be those the manufacturer or seller operates as a customer convenience or those for which specific permission has been obtained from the operator. A DNS request for a generic server name, such as ntp.mytimeserver.com, should result in a random selection of server IP addresses available for that purpose. Each time a DNS request is received, a new randomized list is returned. The client ordinarily uses the first address on the list.

> When candidate SNTP or NTP servers are selected, it is imperative to respect the server operator's conditions of access. Lists of public servers and their conditions of access are available at www.ntp.org. A semi-automatic server discovery scheme using DNS is described at that site. Some ISPs operate public servers, although finding them via their help desks can be difficult.

A well-behaved client operates as follows (note that steps 2-4 constitute a synchronization loop):

1. Consider the specified frequency tolerance of the system clock oscillator. Define the required accuracy of the system clock, then calculate the maximum timeout. For instance, if the frequency tolerance is 200 parts per million (PPM) and the required accuracy is one minute, the maximum timeout is about 3.5 days. Use the longest maximum timeout possible given the system constraints to minimize time server aggregate load, but never make it less than 15 minutes.

2. When the client is first coming up or after reset, randomize the timeout from one to five minutes. This is to minimize shock when 3000 PCs are rebooted at the same time power is restored after a blackout. Assume at this time that the IP address is unknown and that the system clock is unsynchronized. Otherwise, use the timeout value as calculated in previous loop steps. Note that it may be necessary to refrain from implementing the aforementioned random delay for some classes of International Computer Security Association (ICSA) certification.

3. When the timer reaches zero, if the IP address is not known, send a DNS query packet; otherwise, send an NTP request packet to that address. If no reply packet has been heard since the last timeout, double the timeout, but do not make it greater than the maximum timeout. If primary and secondary time servers have been configured, alternate queries between the primary and secondary servers when no successful response has been received.

4. If a DNS reply packet is received, save the IP address and continue at step 2. If a KoD packet is received, remove that time server from the list, activate the secondary time server, and continue at step 2. If a received packet fails the sanity checks, drop that packet and also continue at step 2. If a valid NTP packet is received, update the system clock, set the timeout to the maximum, and continue at step 2.

11. Security Considerations

Without cryptographic authentication, SNTPv4 service is vulnerable to disruption by misbehaving or hostile SNTP or NTP broadcast servers elsewhere in the Internet. It is strongly recommended that access controls and/or cryptographic authentication means be provided for additional security. This document includes protocol provisions for adding such security mechanisms, but it does not define the mechanisms themselves. A separate document [MIL03] in preparation will define a cryptographic security mechanism for SNTP.

12. Acknowledgements

Jeff Learman was helpful in developing the OSI model for this protocol. Ajit Thyagarajan provided valuable suggestions and corrections.

13. Contributors

D. Plonka

J. Montgomery

14. Informative References

[BRA97] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[COL94] Colella, R., Callon, R., Gardner, E., and Y. Rekhter, "Guidelines for OSI NSAP Allocation in the Internet", RFC 1629, May 1994.

[DEE89] Deering, S., "Host extensions for IP multicasting", STD 5, RFC 1112, August 1989.

[DEE98] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, December 1998.

[DOB91] Shue, C., Haggerty, W., and K. Dobbins, "OSI connectionless transport services on top of UDP: Version 1", RFC 1240, June 1991.

[FUR94] Furniss, P., "Octet Sequences for Upper-Layer OSI to Support Basic Communications Applications", RFC 1698, October 1994.

[ISO86] International Standards 8602 - Information Processing Systems - OSI: Connectionless Transport Protocol Specification. International Standards Organization, December 1986.

[MIL92] Mills, D., "Network Time Protocol (Version 3) Specification, Implementation and Analysis", RFC 1305, March 1992.

[MIL03] Mills, D., "The Autokey Security Architecture, Protocol and Algorithms", http://eecis.udel.edu/~mills/database/reports/stime/stime.pdf, August 2003.

[PAR93] Partridge, C., Mendez, T., and W. Milliken, "Host Anycasting Service", RFC 1546, November 1993.

[POS80] Postel, J., "User Datagram Protocol", STD 6, RFC 768, August 1980.

[POS83] Postel, J. and K. Harrenstien, "Time Protocol", STD 26, RFC 868, May 1983.

[SRI99] Srisuresh, P. and M. Holdrege, "IP Network Address Translator (NAT) Terminology and Considerations", RFC 2663, August 1999.

[SRI01] Srisuresh, P. and K. Egevang, "Traditional IP Network Address Translator (Traditional NAT)", RFC 3022, January 2001.

Author's Address

David L. Mills
Electrical and Computer Engineering Department
University of Delaware
Newark, DE 19716

Phone: (302) 831-8247
EMail: mills@udel.edu

Full Copyright Statement

Intellectual Property

Acknowledgement

Funding for the RFC Editor function is provided by the IETF Administrative Support Activity (IASA).