Internet Engineering Task Force
Internet-Draft
Intended status: Standards Track
Expires: 3 January 2026

M. Lichvar
Red Hat
T. Mizrahi
Huawei

2 July 2025

Network Time Protocol Version 5
draft-ietf-ntp-ntpv5-05

Abstract

The Network Time Protocol (NTP) is a widely deployed protocol that allows hosts to obtain the current time of day from time servers. This document specifies version 5 of the protocol (NTPv5), which adopts a client-server model as its sole mode of operation. Legacy operational modes supported in earlier versions have been removed to improve protocol robustness and clarity. While this specification defines the protocol used for time distribution, it does not define the algorithms or heuristics employed by clients to determine or adjust their local time.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 3 January 2026.

Copyright Notice

Table of Contents

1. Introduction

The Network Time Protocol (NTP) is a widely used protocol that enables hosts to synchronize their clocks over packet-switched, variable-latency data networks. Time is distributed hierarchically, beginning with primary time servers, often synchronized to Coordinated Universal Time (UTC) via external sources such as GNSS, and propagating through a network of clients. These clients may themselves act as servers for downstream systems, forming a time distribution tree. To enhance reliability, stability, and accuracy, clients can query multiple servers concurrently and apply local algorithms to select and combine time sources.

This document specifies version 5 of the protocol (NTPv5), which defines only the on-the-wire protocol used for time exchange. It does not cover client-side algorithms such as source selection, filtering of time measurements, or clock discipline. These aspects are considered out of scope.

NTPv5 adopts a simplified client-server model as its sole operational mode. For security and robustness, legacy modes from previous versions, such as symmetric active, symmetric passive, broadcast, control, and private modes, have been removed. Only client and server modes are retained in this version.

To support optional features and future extensibility, NTPv5 makes use of extension fields. This document defines several such fields, which may be included in protocol messages to convey additional information beyond the core header.

NTPv5 supports secure operation through two possible mechanisms originally defined for NTPv4. The first is the NTP Message Authentication Code (MAC) [RFC8573] mechanism, which provides basic message integrity. The second is Network Time Security (NTS) [RFC8915], which offers stronger security guarantees, including server authentication, replay protection, and secure key establishment.

This specification also introduces a feature aimed at improving synchronization accuracy. The Correction Extension Field allows clients and servers to communicate variable delays introduced by intermediate network devices such as switches and routers.

Backward compatibility is supported through version negotiation. A server that implements multiple protocol versions responds using the same version as the client’s request, provided that version is supported.

2. Conventions

2.1. Terminology

Abbreviations used in this document:

MAC	Message Authentication Code
NTP	Network Time Protocol
NTS	Network Time Security
NTS-KE	Network Time Security Key Establishment
ppm	parts per million
PTP	Precision Time Protocol
SI	International System of Units
TAI	International Atomic Time
TC	Transparent Clock
UT1	Universal Time 1
UTC	Coordinated Universal Time

2.2. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

3. Main Differences between NTPv5 and NTPv4

NTPv5 is very similar to NTPv4 [RFC5905]. The main differences are:

1. The protocol specification (this document) describes only the on-wire protocol. Filtering of measurements, security mechanisms, source selection, clock control, and other algorithms, are out of scope.

2. For security reasons, NTPv5 drops support for the symmetric active, symmetric passive, broadcast, control, and private modes. The symmetric and broadcast modes are vulnerable to

replay attacks. The control and private modes can be exploited for denial-of-service traffic amplification attacks. Only the client and server modes remain in NTPv5.

3. The NTPv5 message format differs from that of NTPv4. However, the Version Number field remains at the same offset in both formats, enabling protocol implementations to distinguish between the two versions.

4. Timestamps are clearly separated from values used as cookies.

5. Synchronized server clock is indicated in a separate flag instead of the leap indicator.

6. NTPv5 messages can be extended only with extension fields. The MAC field is wrapped in an extension field to avoid an ambiguity in parsing of the NTP message, which was later addressed for NTPv4 in RFC 7822 [RFC7822].

7. Extension fields can be of any length, even indivisible by 4, but are padded to a multiple of 4 octets. Extension fields specified for NTPv4 can be included in NTPv5 messages as specified for NTPv4.

8. NTPv5 adds support for other timescales than UTC.

9. The NTP era number is exchanged in the protocol, which extends the unambiguous time interval from 136 years to about 35000 years.

10. NTPv5 adds a flag to clearly identify the use of interleaved mode instead of comparing timestamps or cookies. The server’s receive timestamps do not need to be unique in order to support the interleaved mode.

11. NTPv5 works with sets of reference IDs to prevent synchronization loops over multiple hosts, even if they form over other NTP sources than the system peer. Reference IDs have 120 bits instead of 32 bits to minimize the rate of false positives.

12. Resolution of the root delay and root dispersion fields is improved from about 15 microseconds to about 4 nanoseconds.

13. Clients do not leak information about their clock (e.g. timestamps, estimated accuracy).

4. Basic Concepts

4.1. The NTP Message Exchange

An NTP client exchanges messages with one or more NTP servers; the client sends a request and the server sends a response. Both the client and server measure the time of transmission and reception of every message they send and receive. The servers provide their timestamps to the client.

The NTP message exchange is illustrated in Figure 1, depicting four timestamp values:

1. T1 - client's transmit timestamp of the request

2. T2 - server's receive timestamp of the request

3. T3 - server's transmit timestamp of the response

4. T4 - client's receive timestamp of the response

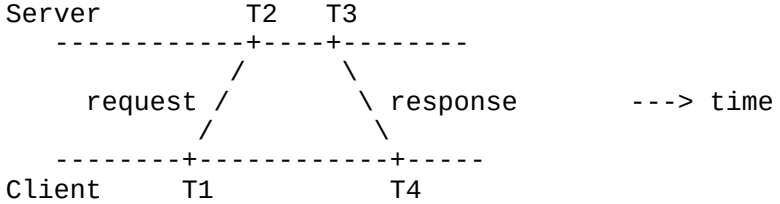

Figure 1: NTP message exchange

The timestamps T1 and T4 are recorded locally by the client and are not transmitted over the network. In contrast, the server includes timestamps T2 and T3 in its response to the client. As a result, by the end of the exchange, the client has access to all four timestamps, commonly referred to as a sample.

The client can use the set of four timestamps to estimate both the clock offset relative to the server and the network delay. The offset of the server clock relative to the client clock can be calculated using Eq. (1), and the two-way delay between the client and server can be estimated using Eq. (2).

(1) offset = ((T2 + T3) - (T4 + T1)) / 2
(2) delay = |(T4 - T1) - (T3 - T2)|

The offset and delay estimates given in the two equations above are based on a single timestamp sample. In practice, a client typically maintains ongoing estimates of offset and delay by aggregating multiple samples collected over several message exchanges.

4.2. Hierarchical Time Distribution

NTP supports a hierarchical time distribution scheme. This hierarchy is defined by stratum levels, which indicate the distance from the reference time source. Stratum 1 servers are directly synchronized with an external reference clock. Servers that synchronize with stratum 1 servers are classified as stratum 2, and the hierarchy continues in this manner.

An NTP response message contains two fields, root delay and root dispersion, which together provide an estimate of the server's time error accumulated along the synchronization path.

Root delay represents the cumulative delay along the path to the reference time source used by the primary time server. Each server in the synchronization chain adds its own delay estimate based on the server it deems most accurate. This estimate includes both the measured two-way delay towards the server and any processing delays between the timestamping and actual sending or receiving of NTP messages. To estimate the potential error caused by asymmetric delays, half of the root delay is typically used as a bound on the clock's maximum error.

Root dispersion provides an estimate, in time units, of the maximum potential error in the clock due to both the instability of clocks along the synchronization path and the variability in NTP measurements. Each server in the chain contributes its own dispersion value to the cumulative root dispersion. The method for estimating dispersion can vary between implementations and often depends on the underlying clock model. In a simple model, dispersion may be calculated using a constant Dispersion Rate (DR), as shown in Equation (3). The constant DR represents a relative frequency error, typically within the range (0, 1). For instance, a DR value of 0.000015 (15 ppm) is suggested in [RFC5905].

(3) dispersion = |T4 - T1| * DR

The root distance is defined as the sum of the root dispersion and half of the root delay. It represents an estimate of the maximum possible time error of the clock, accounting for the assumed clock stability and the worst-case scenario of asymmetric network delays. Although root distance is not transmitted in NTP messages, it can be computed and used by the client to assess the accuracy of time sources.

A synchronization loop can occur when clients attempt to synchronize with each other--either directly or indirectly through other servers. Such loops are undesirable in an NTP network, as they create positive feedback cycles that can degrade synchronization stability. To help detect and prevent these loops, servers use randomly generated reference IDs, which are exchanged in NTP messages as described in Section 7.4.

4.3. Leap Seconds

An NTPv5 server can distribute time using one of four timescales: UTC, TAI, UT1, or leap-smeared UTC. The specific timescale in use is indicated by the Timescale field in the NTP message. Of these options, UTC and leap-smeared UTC are subject to leap second adjustments.

A leap second is a one-second adjustment occasionally applied to UTC to maintain alignment with solar time. This adjustment can be either positive, by inserting an extra second, or negative, by removing one. To date, only positive leap seconds have been introduced. Therefore, the following discussion focuses on the more common case of a positive leap second, though the behavior of a negative leap second can be inferred.

When time is distributed using the UTC timescale, a system clock may handle a positive leap second in one of two ways: it may either roll back by one second at the end of the leap second, or it may pause, holding the same time, for the duration of the leap second, as outlined in [RFC8877]. As a result, any NTP message exchange that occurs outside of a leap second will yield timestamps that reflect server’s clock according to the actual UTC time.

When using the leap-smeared timescale, the system clock is gradually slowed down during the leap second and surrounding time intervals, allowing the time to smoothly adjust until it aligns with the correct value. This adjustment period, known as a leap smear, can span from a few seconds to several hours before, during, and/or after the scheduled leap second.

The Leap Indicator (LI) field in an NTP message signals that a leap second is scheduled to occur within the next 14 days.

5. Data Types

NTPv5 uses few different data types. They are all in the network order. Beside signed and unsigned integers, it has also the following fixed-point types:

time32
: A 32-bit unsigned fixed-point type containing values in seconds. It has 4 bits describing the unsigned integral part and 28 bits describing the fractional part. The maximum value is 16 seconds and the resolution is about 3.7 nanoseconds. Note that this is different from the 32-bit time format in NTPv4.

timestamp64
: A 64-bit unsigned fixed-point type containing a timestamp specified in seconds. It has 32 signed integer bits and 32 fractional bits. It spans an interval of about 136 years and has a resolution of about 0.23 nanoseconds. It can be used in different timescales. In the UTC timescale it is the number of SI seconds since 1 Jan 1972 plus 2272060800 (number of seconds since 1 Jan 1900 assuming 86400-second days), excluding leap second adjustments. Timestamps in the TAI timescale are the same except they include leap seconds and extra 10 seconds for the original difference between TAI and UTC in 1972, when leap seconds were introduced. A value of 0 indicates an unknown or invalid timestamp. One interval covered by the type is called an NTP era. The era starting at the epoch is era number 0, the following era is number 1, and so on.

Some fields use a logarithmic scale, where an 8-bit signed integer represents the rounded log2 value of seconds. For example, a log2 value of 4 is 2^4 (2 to the power of 4, 16) seconds, or a log2 value of -2 is 2^-2 (0.25 seconds).

6. Message Format

NTPv5 servers and clients exchange messages as UDP datagrams. Clients send requests to servers and servers send them back responses. The server’s UDP port in NTP messages is 123, as assigned by IANA. The client’s UDP port can be any number consistent with the local policy. The format of the UDP payload is shown in Figure 2.

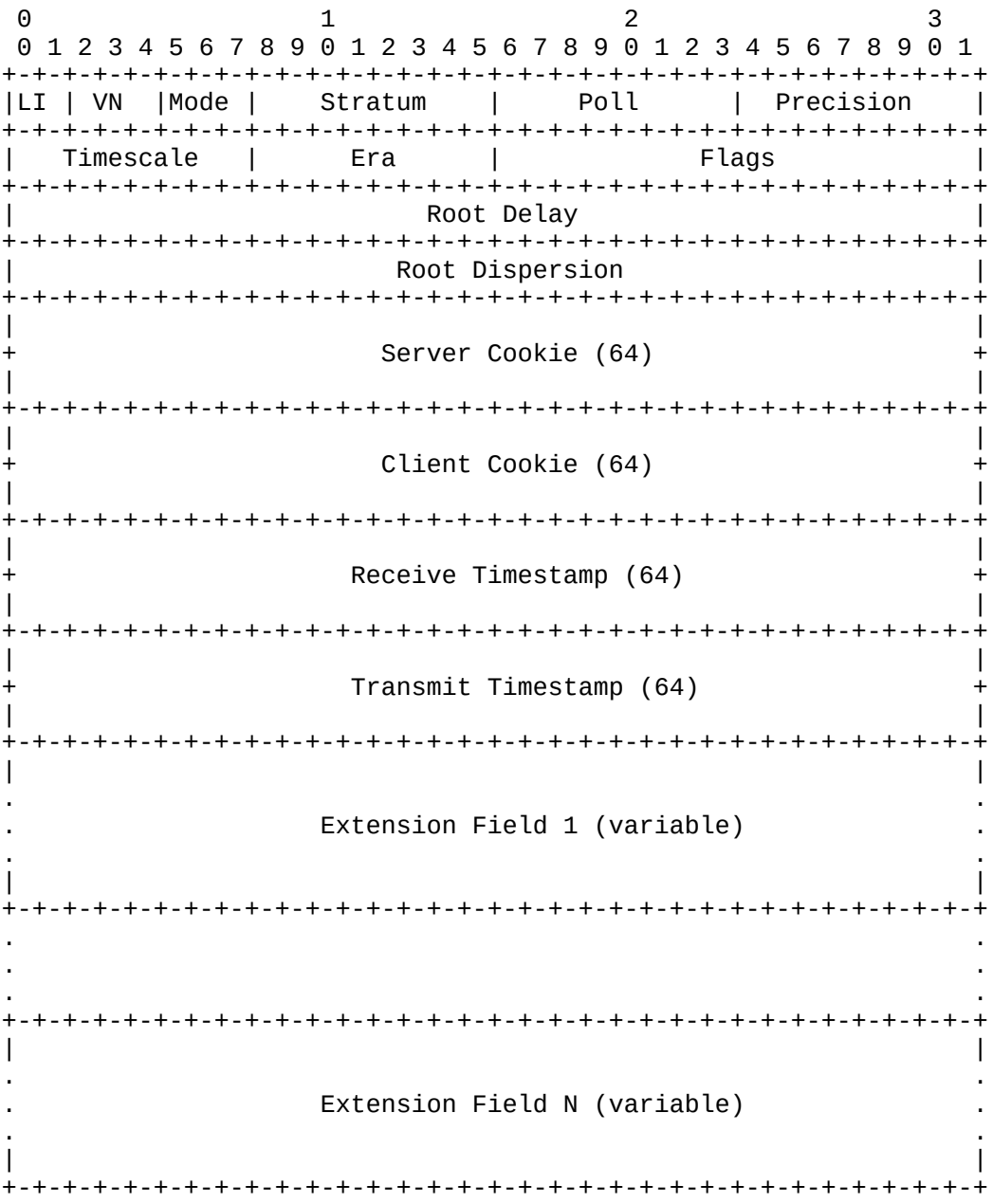

Figure 2: Format of NTPv5 messages

Each NTPv5 message has a header containing the following fields:

Leap indicator (LI)
A 2-bit field indicating upcoming leap seconds. In requests it is always 0. In responses it has one of the following values:

0: Normal
No leap second will be inserted or deleted in the next 14 days, or the server is responding in a leap-smeared timescale (i.e. the client is not expected to be handling leap seconds).

1: Insert leap second
A leap second will be inserted in the next 14 days at the end of the current month.

2: Delete leap second
A leap second will be deleted in the next 14 days at the end of the current month.

3: Unknown
The server does not have a time source or other source providing information about leap seconds.

Version Number (VN)
A 3-bit field containing the value 5.

Mode
A 3-bit field containing the value 3 (request) or 4 (response).

Stratum
An 8-bit field containing the stratum of the server. Primary time servers have a stratum of 1, their clients have a stratum of 2, and so on. The value of 0 indicates an unknown or infinite stratum. In requests it is always 0. When 0 in a response, it indicates the server was unable or unwilling to provide valid time information. Servers advertising a stratum above 16 should not be synchronized to, except when the client is explicitly configured to do so by the end-user.

Poll
An 8-bit signed integer containing the minimum allowed polling interval as a log2 value in seconds. In requests it is always 0. In responses it is the server’s value that the client is expected to follow. This field is valid even when stratum is 0. A value of 127 indicates the server does not want to hear from the client again. Note that the poll value has a different meaning than in NTPv4. It supersedes the NTPv4 Kiss-o’-Death (KoD) RATE and DENY codes.

Precision
: An 8-bit signed integer containing the precision of the timestamps included in the message as a rounded log2 value in seconds. In requests, which do not contain any timestamps, it is always 0.

Timescale
: An 8-bit identifier of the timescale. In requests it is the requested timescale. In responses it is the timescale of the receive and transmit timestamps. Defined values are:

  0: UTC

  1: TAI

  2: UT1

  3: Leap-smeared UTC

Era
: An 8-bit unsigned NTP era number corresponding to the receive timestamp. In requests it is always 0.

Flags
: A 16-bit integer that can contain the following flags:

  0x1: Synchronized
  : In requests it is 0. In responses a value of 1 indicates the server's clock is synchronized and the provided timestamps can be used by the client for its own synchronization.

  0x2: Interleaved mode
  : In requests a value of 1 is a request for a response in the interleaved mode. In responses a value of 1 indicates the response is in the interleaved mode.

  0x4: Authentication NAK
  : In requests it is 0. In responses a value of 1 indicates that the server failed to authenticate the request. A server setting this bit SHOULD also set the stratum of the message to 0.

Root Delay
: A field using the time32 type. In responses it is the server's root delay. In requests it is always 0.

Root Dispersion
: A field using the time32 type. In responses it is the server's root dispersion. In requests it is always 0.

Server Cookie
: A 64-bit field containing a number generated by the server which enables the interleaved mode. In requests it is 0, or a copy of the server cookie from the last response.

Client Cookie
: A 64-bit field containing a random number generated by the client. Responses contain a copy of the field from the corresponding request, which allows the client to verify that the responses are related to the requests.

Receive Timestamp
: A field using the timestamp64 type. In requests it is always 0. In responses it is the time when the request was received by the server. The timestamp corresponds to the end of the reception.

Transmit Timestamp
: A field using the timestamp64 type. In requests it is always 0. In responses it is the server's time denoting the beginning of the transmission of a response to the client. Which response it refers to depends on the selected mode (basic or interleaved). See Measurement Modes (Section 8) for detail.

The header has 48 octets, which is the minimum length of a valid NTPv5 message. A message can contain optional extension fields (zero or more). The maximum length is not specified, but the length MUST be divisible by 4 octets.

7. Extension Fields

The format of NTPv5 extension fields is shown in Figure 3.

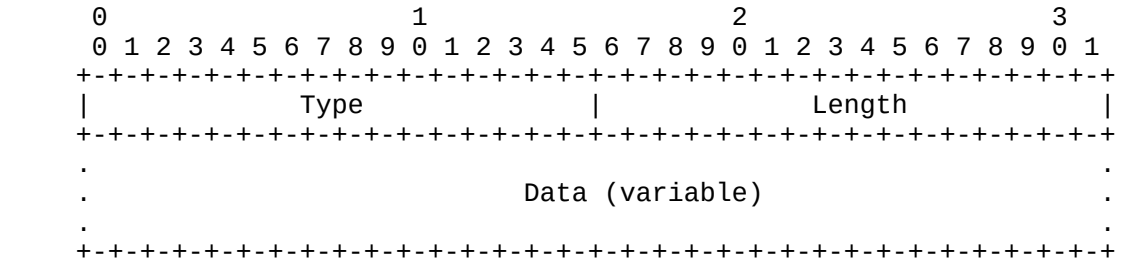

Figure 3: Format of NTPv5 extension fields

Each extension field has a header which contains a 16-bit type and 16-bit length. The length is in octets and it includes the header. The minimum length is 4, i.e. an extension field does not have to contain any data. If the length is not divisible by 4, the extension field is padded with zeros to the smallest multiple of 4 octets.

If a request contains an extension field, the server MUST include this extension field in the response unless the specification of the extension field states otherwise, or the server does not support the extension field. If the server excludes one or more extension fields in the response, it MUST include a Padding extension field that compensates for the excluded fields, making the request and response symmetric in length. A client can interpret the absence of an expected extension field in a response as an indication that the server does not support the extension field.

Extension fields specified for NTPv4 can be included in NTPv5 messages as specified for NTPv4.

The rest of this section describes extension fields specified for NTPv5. Clients are not required to use or support any of these extension fields, but servers are required to support the following extension fields: Padding, Server Information, Reference IDs Request, Reference IDs Response.

7.1. Draft Identification Extension Field

Note to the editors: this section must be removed before final publication.

This field, with type 0xF5FF, is used to indicate which draft of the specification an implementation is based upon. It MUST be included in NTPv5 requests produced by an implementation based on a draft of this specification, and MUST NOT be included in NTPv5 requests produced by an implementation based on the final version of this specification. Server MUST use this field if and only if responding to a request containing this field and the server is a draft implementation.

The contents of this field MUST be the full name, including version number, of the draft upon which the implementation is based, encoded as an ASCII string. If the server string is longer than the client string, the server MUST NOT respond in that version to prevent traffic amplification.

A server MUST NOT respond to requests with a draft identification it does not recognize. If it responds, it SHOULD respond according to the same draft specification as given by the client.

Note: the content of this field MUST NOT be null-terminated (the extension field in the NTP message may need to be padded with up to 3 octets). When comparing the strings be compared in their full length, i.e. a longer string containing a shorter string is not sufficient.

7.2. Padding Extension Field

This field, with type [[TBD]] (draft: 0xF501), can have any length. The data field within this extension field SHOULD contain zeros and it MUST be ignored by the receiver.

Servers can use this extension field to pad the response to match the length of the request if the response does not contain all requested extension fields, or some have a variable length. If the request message includes a padding extension, the server can increase its length if necessary. If the request message does not include a padding extension, the server can add it to the response. Clients can include the padding extension in the request message, allowing the extension to be used for internal purposes. For instance, an NTP client receiving a response message can use this extension field to transfer the reception time from a hardware module to a software module.

This field MUST be supported on servers.

7.3. Message Authentication Code Extension Field

This field, with type [[TBD]] (draft: 0xF502), authenticates the NTPv5 message with a symmetric key. Implementations SHOULD use the Message Authentication Code (MAC) specified in RFC8573 [RFC8573]. The MAC MUST be computed over the NTP header and all the extension fields, if present, located prior to the Message Authentication Extension Field. The extension field MUST be the last extension field in the message unless an extension field is specifically allowed to be placed after a MAC or another authenticator field, such as the Correction Extension Field (Section 7.6).

7.4. Reference IDs Request and Response Extension Fields

Each NTPv5 server has a randomly generated 120-bit reference ID (it will be split into 10 12-bit values). The extension fields described in this section are used to exchange sets of reference IDs in order to detect synchronization loops, i.e. when a client is synchronizing (directly or indirectly) to one of its own clients, or more generally detect presence of a specific time source in the synchronization chain.

As each client can be synchronized to an unlimited number of servers (and there can be up to 15 strata of servers), the reference IDs are exchanged as a Bloom filter [Bloom] instead of a list to limit the amount of data that needs to be exchanged.

The Bloom filter is an array of 4096 bits. When empty, all bits are zero. To add a reference ID to the filter, the 120-bit value of the reference ID is split into 10 12-bit values and the bits of the array at the 10 positions given by the 12-bit values are set to one.

A server maintains a copy of the filter for each server it acquires time from. The filter provided by the server to clients is the union of the filters (using the bitwise OR operation) of the server's sources selected for synchronization and the server's own reference ID.

If the server uses a previous version of NTP for some of its sources, the reference IDs added to the filter are generated from their IP addresses as the first 120 bits of the MD5 [RFC1321] sum of the address in network order. If the server uses a reference clock, the reference ID is the first 120 bits of the MD5 sum of the 4-octet zero-padded ASCII string from the NTP Reference Identifier Codes registry maintained by IANA, or a string beginning with the uppercase letter X, which are reserved for private and experimental use.

A client checking whether the server's set of reference IDs contains the client's own reference ID checks whether the bits at the 10 positions corresponding to the 12-bit values from the reference ID are all set to one.

When a client that also serves time to other clients as an NTPv5 server detects its reference ID in a server's set of reference IDs, it SHOULD assume it is in a synchronization loop, and stop using that server for synchronization. When the client's reference ID is no longer detected in the server's filter, it SHOULD wait for a random number of polling intervals (e.g. between 0 and 4) before selecting the server again. The random delay helps with stabilization of the selection in longer loops. If a recurring loop is detected, it is recommended to increase the random delay in order to avoid livelock scenarios.

False positives are possible. The probability of a collision grows with the number of reference IDs in the filter. With 26 reference IDs it is about 1e-12. With 118 IDs it is about 1e-6. The client MAY avoid selecting a server which has too many bits set in the filter (e.g. more than half) to reduce the probability of the collision for its own clients. A client which detected a synchronization loop MAY change its own reference ID to limit the duration of the potential collision.

Bloom filters are free from false negatives. However, there is a transient period between the addition of a reference ID to a server's Bloom filter and the propagation of this information through a chain of clients. During this period, the new reference ID may not be detected, potentially causing a temporary synchronization loop. For instance, if two servers inquire each other about the list of reference IDs roughly at the same time, neither will detect its reference ID in the other's Bloom filter, resulting in a temporary loop.

Generally, when a server updates the Bloom filter it distributes to clients, temporary synchronization loops might occur before converging to an acyclic distribution tree.

The filter can be exchanged as a single 512-octet array, or it can be exchanged in smaller chunks over multiple NTP messages, making them shorter, but delaying the detection of the synchronization loop.

The request extension field specifies the offset of the requested chunk in the filter as a number of octets. The requested length of the chunk is given by the length of data in the request extension field. The response extension field MUST have the same length as the request extension field. If the request contains an invalid offset for the length, i.e. it is larger than 512 minus length of data in the extension field, the extension field MUST be ignored.

When a filter is sent to a client in multiple chunks, the server might update the Bloom filter to a new value after some chunks have been sent. This can cause the subsequent chunks to be inconsistent with the previously sent ones. The partially updated Bloom filter might cause false negative or false positive detections. This transient issue is resolved once the server completes sending the updated Bloom filter.

The client SHOULD use requests of a constant length for the association to avoid adding a variation to the measured NTP delay.

The format of the Reference IDs Request is shown in Figure 4.

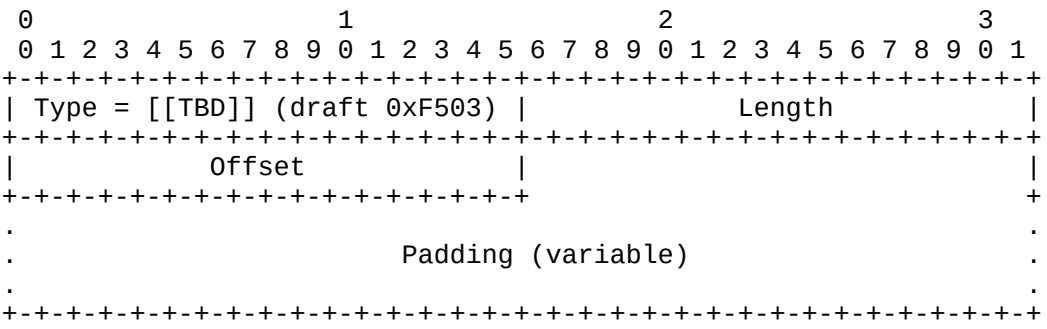

Figure 4: Format of Reference IDs Request Extension Field

The format of the Reference IDs Response is shown in Figure 5.

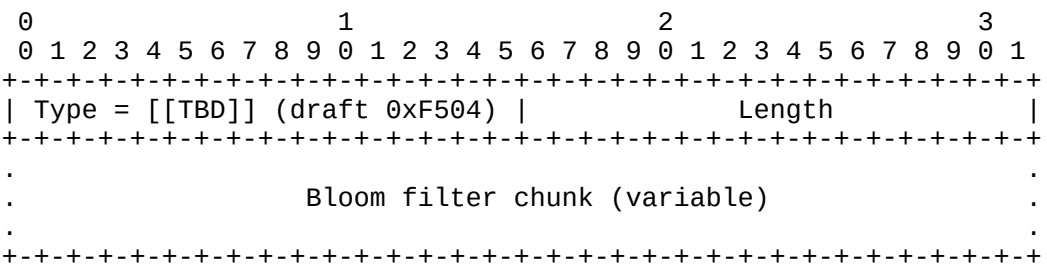

Figure 5: Format of Reference IDs Response Extension Field

These fields MUST be supported on servers which can be synchronized to other NTP servers (i.e. they can be in a synchronization loop).

7.5. Server Information Extension Field

This field provides clients with information about which NTP versions are supported by the server, i.e. whether it can respond to requests conforming to the specific version. It contains a 16-bit field with flags indicating support for NTP versions in the range of 1 to 16, where the least significant bit corresponds to the version 1. The extension field has a fixed length of 8 octets. In requests, all data fields of the extension are 0.

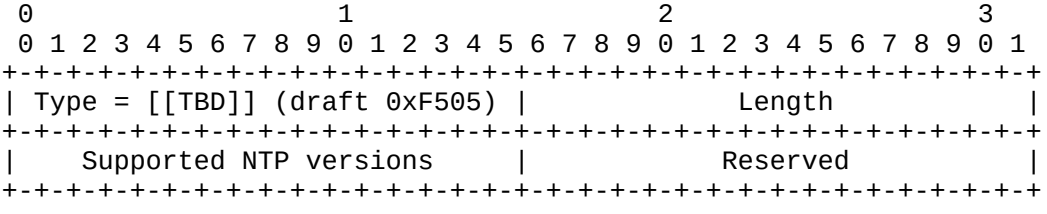

Figure 6: Format of Server Information Extension Field

This field MUST be supported on servers.

7.6. Correction Extension Field

Processing and queueing delays in network switches and routers may be a significant source of jitter and asymmetry in network delay, which has a negative impact on accuracy and stability of clocks synchronized by NTP. A solution to this problem is defined in the Precision Time Protocol (PTP) [IEEE1588], which is a different protocol for synchronization of clocks in networks. In PTP a special type of switch or router, called a Transparent Clock (TC), updates a correction field in PTP messages to account for the time messages spend in the TC. This is accomplished by timestamping the message at the ingress and egress ports, taking the difference to determine time in the TC and adding this to the Delay Correction. Clients can account for the accumulated Delay Correction to determine a more accurate clock offset and network delay.

The NTPv5 Delay Correction has the same format as the PTP correctionField to make it easier for manufacturers of switches and routers to implement NTP corrections. The format of the Correction Extension Field is shown in Figure 7.

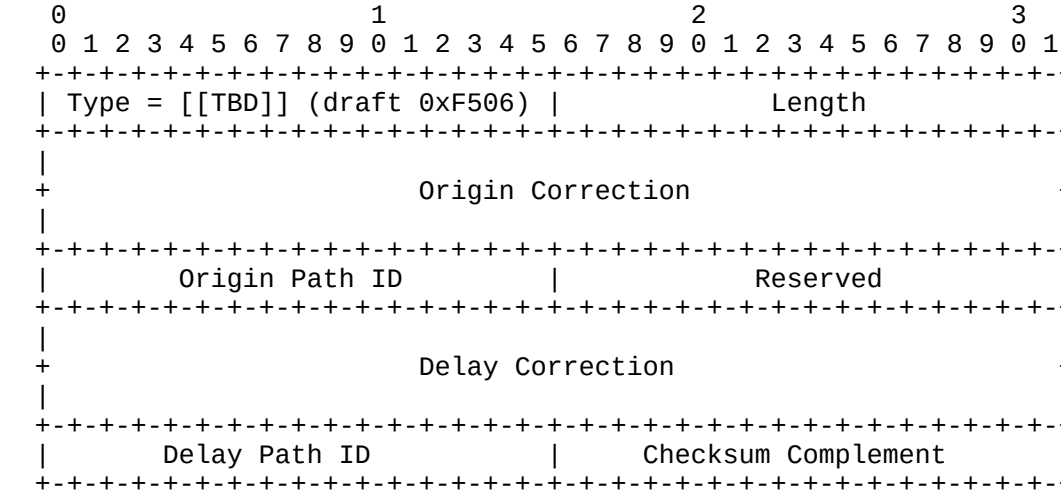

Figure 7: Format of Correction Extension Field

Field Type
: The type which identifies the Correction extension field (value TBD).

Length
The length of the extension field, which is 28 octets.

Origin Correction
A field which contains a copy of the accumulated delay correction from the request packet in the NTP exchange.

Origin Path ID
A field which contains a copy of the final path ID from the request packet in the NTP exchange.

Reserved
16 bit reserved for future specification by the IETF. Transmit with all zeros.

Delay Correction
A signed fixed-point number of nanoseconds with 48 integer bits and 16 binary fractional bits, which represents the current correction of the network delay that has accumulated for this packet on the path from the source to the destination. A value of one in all bits, except the most significant, of the field indicates that the correction is too big to be represented. The format of this field is identical to the PTP correctionField.

Path ID
A 16-bit identification number of the path where the delay correction was updated.

Checksum Complement
A field which can be modified in order to keep the UDP checksum of the packet valid. This allows the UDP checksum to be transmitted before the Correction Field is received and modified. The same field is described in RFC 7821 [RFC7821].

An NTP client with enabled support for network corrections SHOULD add the Correction Extension Field to its requests, with all fields of the extension field set to zero. It MUST be the last extension field in the NTP message.

A network device forwarding packets (e.g. a switch or router) with enabled support for NTP corrections MUST modify only packets which meet all of the following requirements:

1. It is an IPv4 or IPv6 packet

2. The IP protocol number is 17 (UDP)

3. The UDP source port or destination port is 123 (NTP)

4. The NTP version is 5

5. The NTP message contains the Correction Extension Field

The network device SHOULD add the difference between the beginning of the NTP message retransmission and the end of the message reception to the Delay Correction value in the Correction Extension Field. Note that this time difference might be negative, e.g. in a cut-through switch. If the packet is transmitted at the same speed as it was received and the length of the packet does not change (e.g. due to adding or removing a VLAN tag), the beginning and end of the interval may correspond to any point of the reception and transmission as long as it is consistent for all forwarded packets of the same length. If the transmission speed or length of the packet is different, the beginning and end of the interval SHOULD correspond to the end of the reception and beginning of the transmission respectively.

The receive timestamp of the ingress port and transmit timestamp of the egress port MUST be from the same clock, or two clocks that are synchronized to each other. The clocks do not need to be synchronized to an external reference if their frequency is accurate enough for the accuracy of measured NTP delay required by the application. The correction field is updated before or during the transmission of the message. It is a one-step end-to-end transparent clock in the PTP terminology.

The network device SHOULD have a randomly generated 16-bit ID number assigned to each of its ports. When it modifies the Correction Extension Field, it SHOULD update the Path ID field of the extension field by adding to it the values of the incoming and outgoing port ID. The Path ID values can be used by clients to determine if the NTP request and response have likely traversed the same network path.

If the network device modified any fields of the Correction Extension Field, it MUST also update the Checksum Complement field in order to keep the existing UDP checksum valid, or update the UDP checksum in the UDP header itself. The network device MUST NOT modify any other data in the UDP payload.

If an NTP server supports the Correction Extension Field and receives a request which contains this extension field, it SHOULD include the extension field in the response. If it is included, it MUST be the last extension field in the message. It MUST copy the Delay Correction and Path ID from the request to the Origin Correction and Origin Path ID fields in the response respectively. It SHOULD set the Delay Correction and Path ID fields of the response to zero.

The Correction Extension Field is required to be the last extension field in the message to provide an indicator at a fixed offset from the ending of the message reception (which might simplify hardware implementation of the correction) and to avoid being covered by the Message Authentication Code Extension Field, or other authenticator extension fields (e.g. Network Time Security). It is the exception in the requirement specified for the Message Authentication Code Extension Field. If the corrections were covered by the authenticator fields, the network devices would need to have access to the keys and have a significant additional complexity in order to update the authenticator fields when they modify the Correction Extension Field.

As the Correction Extension Field is not protected, NTP clients MUST validate the corrections before their application as specified in Measurement Modes (Section 8). Clients MUST ignore an unexpected Correction Extension Field in the response, i.e. if it was not included in the request.

7.7. Reference Timestamp Extension Field

This field contains the time of the last update of the clock. It has a fixed length of 12 octets. In requests, that timestamp is always 0.

(Is this really needed? It was mostly unused in NTPv4.)

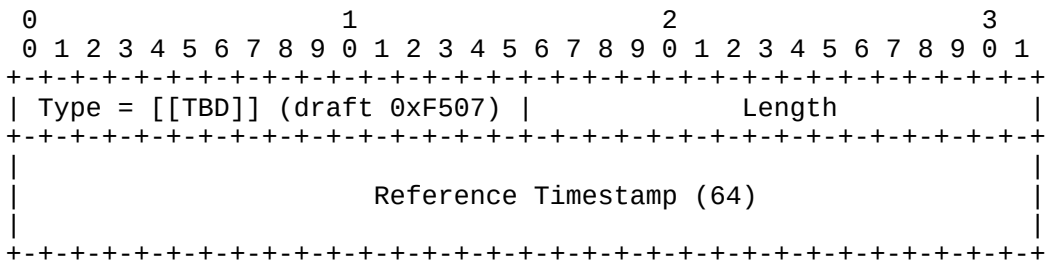

Figure 8: Format of Reference Timestamp Extension Field

7.8. Monotonic Receive Timestamp Extension Field

When a clock is synchronized to a time source, there is a compromise between time (phase) accuracy and frequency accuracy, because the frequency of the clock has to be adjusted to correct time errors that accumulate due to the frequency error (e.g. caused by changes in the temperature of the crystal). Faster corrections of time can minimize the time error, but increase the frequency error, which transfers to clients using that clock as a time source and increases their

frequency and time errors. This issue can be avoided by transferring time and frequency separately using different clocks.

The Monotonic Receive Timestamp Extension Field contains an extra receive timestamp with a 32-bit epoch ID captured by a clock which does not have corrected phase and can better transfer frequency than the clock which captures the receive and transmit timestamps in the header. The extension field has a constant length of 16 octets. In requests, the counter and timestamp are always 0.

The epoch ID is a random number which is changed when frequency transfer needs to be restarted, e.g. due to a step of the clock.

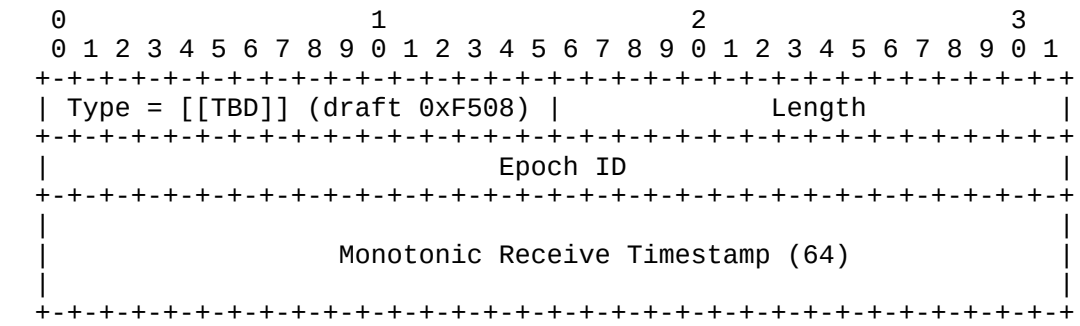

Figure 9: Format of Monotonic Receive Timestamp Extension Field

The client can determine the frequency-transfer offset from the time-transfer offset and difference between the two receive timestamps in the response. It can use the frequency-transfer offset to better control the frequency of its clock, avoiding the frequency error in the server’s time-transfer clock.

7.9. Secondary Receive Timestamp Extension Field

This extension field provides an additional receive timestamp of the client request in a selected timescale. It enables the client to get the same receive timestamp in different timescales in order to calculate the current offset between the timescales.

In requests, the Timescale field selects the requested timescale. The other data fields in the extension field MUST be set to 0.

The Timescale, Era, and Secondary Receive Timestamp fields in a response have the same meaning as the Timescale, Era, and Receive Timestamp fields in the header respectively.

If the server does not support the requested timescale, it MUST ignore the extension field in the request. If the server supports the timescale, but does not have a reliable timestamp (e.g. due to being close to a leap second), it SHOULD set the timestamp field to 0.

The server MAY provide in this extension field timestamps in timescales which it does not provide in the header, e.g. it can provide UTC in addition to leap-smeared UTC to enable its clients to measure the current smearing offset.

A request MAY contain multiple instances of this extension field, but each timescale MUST be requested at most once, not counting the timescale in the header. The server SHOULD include in its response timestamps in all requested timescales it supports.

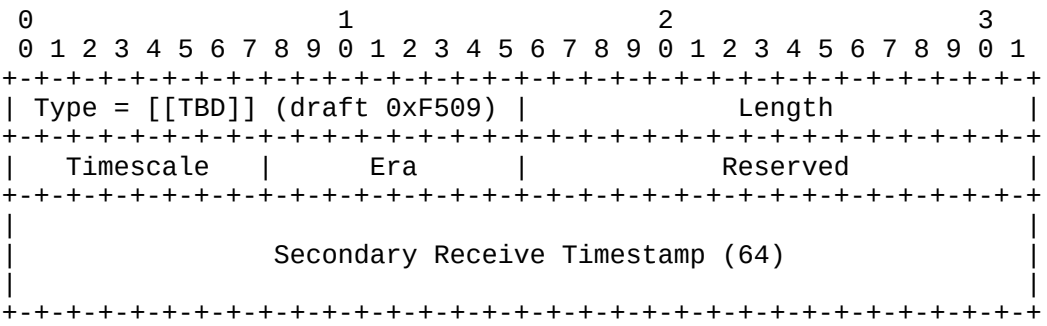

Figure 10: Format of Secondary Receive Timestamp Extension Field

8. Measurement Modes

At the end of an NTP message exchange, the client has access to four timestamps, which it can use to estimate the clock offset and the network delay, as described in Section 4.1.

If the Correction Extension Field is used and the corrections are known for both the request and response, a corrected offset and delay is calculated:

* offset_c = offset + (Cd - Co) / 2

* delay_c = delay - (Cd + Co - Drx - Dtx) * (1 - FC)|

where

* Co is the Origin Correction from the response to the request corresponding to timestamps T1 and T2

* Cd is the Delay Correction from the response corresponding to timestamps T3 and T4

* FC is the maximum expected frequency error of devices providing the delay corrections (e.g. 100 ppm)

* Drx is the time it took to receive the frame containing the response corresponding to T3 and T4

* Dtx is the time it took to transmit the frame containing the request corresponding to T1 and T2. If unknown, it SHOULD be set to Drx.

The corrected offset and delay MUST NOT be accepted if any of delay_c, Co and Cr is negative. The uncorrected delay MUST always be used for calculation of root delay.

The client can make measurements in the basic mode, or interleaved mode if supported on the server. In the basic mode, the transmit timestamp in the server response corresponds to the message which contains the timestamp itself. In the interleaved mode it corresponds to a previous response identified by the server cookie. The interleaved mode enables the server to provide the client with a more accurate transmit timestamp which is available only after the response was formed or sent.

An example of cookies and timestamps in an NTPv5 exchange using the basic mode is shown in Figure 11.

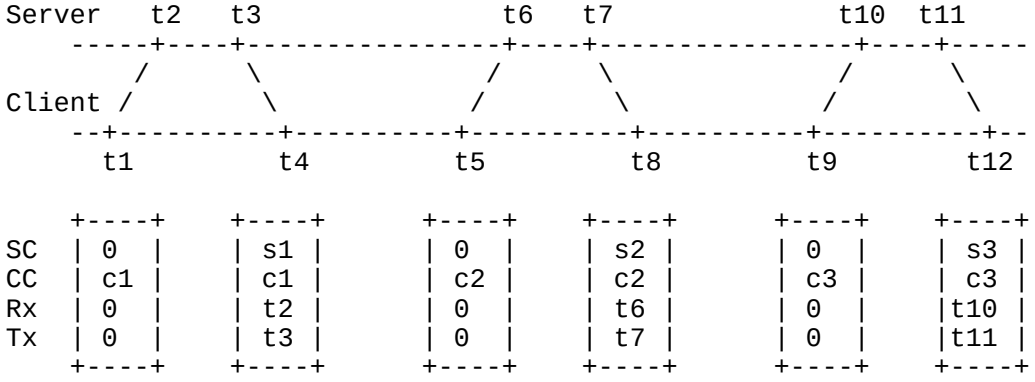

Figure 11: Cookies and timestamps in basic mode

From the three exchanges in this example, the client would use the the following sets of timestamps:

* (t1, t2, t3, t4)

* (t5, t6, t7, t8)

* (t9, t10, t11, t12)

The NTPv4 interleaved client-server mode is described in RFC 9769 [RFC9769]. The difference between NTPv5 and NTPv4 is that in NTPv5 the interleaved mode is enabled explicitly by a flag and the previous transmit timestamp on the server is identified by the server cookie instead of the receive timestamp, which avoids the requirements for receive timestamps to be unique and not equal to transmit timestamps. Therefore, the NTPv5 interleaved mode is easier to implement. A server implementation that should support both NTPv4 and NTPv5 does not need to process interleaved requests and save timestamps separately for the different NTP versions. It can reuse the NTPv4 support in NTPv5 by setting the server cookie to the (unique) receive timestamp.

An example of an NTPv5 exchange using the interleaved mode is shown in Figure 12. The messages in the basic and interleaved mode are indicated with B and I respectively. The timestamps t3′ and t11′ correspond to the same transmissions as t3 and t11, but they may be less accurate (e.g. due to being captured in software before the transmission). The first exchange is in the basic mode followed by a second exchange in the interleaved mode. For the third exchange, the client request is in the interleaved mode, but the server response is in the basic mode, because the server no longer had the timestamp t7 (e.g. it was dropped to save timestamps for other clients using the interleaved mode).

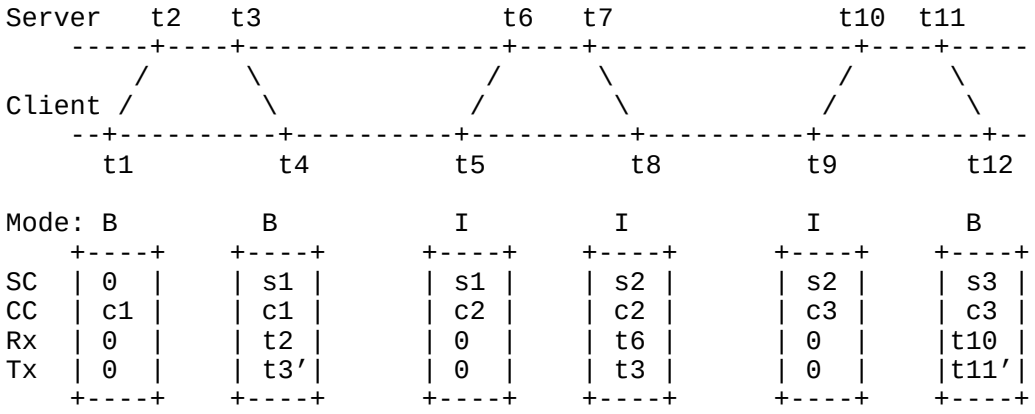

Figure 12: Cookies and timestamps in interleaved mode

From the three exchanges in this example, the client would use the following sets of timestamps:

* (t1, t2, t3’, t4)

* (t1, t2, t3, t4) or (t5, t6, t3, t4)

* (t9, t10, t11’, t12)

9. Client Operation

An NTPv5 client can use one or multiple servers. It has a separate association with each server. It makes periodic measurements of its offset and delay to the server. It can filter the measurements and compare measurements from different servers to select and combine the best servers for synchronization. It can adjust its clock in order to minimize its offset and keep the clock synchronized. These algorithms are not specified in this document. The client MAY use the NTPv4 [RFC5905] algorithms, or any more or less advanced algorithms optimized for the same or different conditions.

The polling interval can be adjusted for the network conditions and stability of the clock. When polling a public server on Internet, the client SHOULD use a polling interval of at least 2^6 (64) seconds, increasing in normal conditions up to at least 2^10 (1024) seconds to avoid excessive load on the server in case the implementation is used on a very large number of systems. On start, the client MAY send a burst of up to 8 requests using a poll interval of only 2^1 (2) seconds in order to make the initial correction of the clock sooner.

The polling interval SHOULD contain a small random part (e.g. up to 2% of the target interval in a uniform distribution) in order to spread the load on the server if a large number of identical clients is (re)started at the same time.

The client SHOULD follow the minimum allowed polling interval provided by the server in the poll field of the last valid response, but only up to a reasonable maximum value to limit the impact of misconfigurations, bugs, and potential denial-of-service attacks where the client would accept a spoofed response. For example, the client could limit the accepted minimum interval to 2^15 (about 9 hours) if authentication is disabled, or 2^18 (about 3 days) if authentication is enabled.

Each successful measurement provides the client with an offset, delay and dispersion. When combined with the server’s root delay and dispersion, it gives the client an estimate of the maximum error.

On each poll, the client:

1. Generates a new random cookie.

2. Formats a request with necessary extension fields and the fields in the header all zero except:

   * Version is set to 5.

   * Mode is set to 3.

   * Timescale is set to the timescale in which the client wants to operate.

   * Flags are set according to the requested mode. The interleaved mode flag requests the server to save the transmit timestamp of the response and provide the transmit timestamp of a previous response corresponding to the server cookie (if not zero).

   * Server cookie is set only in the interleaved mode. It is set to the server cookie from the last valid response, or zero if no such response was received yet or the transmit timestamp of that response would no longer be useful to the client (e.g. after missing too many responses).

   * Client cookie is set to the newly generated cookie.

3. Sends the request to the server to the UDP port 123 and captures a transmit timestamp for the packet.

4. Waits for a valid response from the server and captures a receive timestamp. A valid response has version 5, mode 4, client cookie equal to the cookie from the request, and passes authentication if enabled. The client MUST ignore all invalid responses and accept at most one valid response.

5. Checks whether the response is usable for synchronization of the clock. Such a response has the Synchronized flag set, stratum between 0 and 16, root delay and dispersion both smaller than a specific value, e.g. 16 seconds, and timescale equal to the requested timescale. If the response is in a different timescale, the client can switch to the provided timescale, convert the timestamps if the offset between the timescales is known from the Secondary Receive Timestamp Extension Field or other sources, or ignore the response.

6. Saves the server’s receive and transmit timestamps. If the client internally counts seconds using a type wider than 32 bits, it SHOULD expand the timestamps with the provided NTP era.

7. Calculates the offset, delay, and dispersion as specified in Measurement Modes (Section 8).

A client which operates as a server for other clients MUST include the Reference IDs Request Extension Field in its requests in order to track reference IDs of its sources. If the server’s set of reference IDs contains the client’s own reference ID, it SHOULD not select the server for synchronization to avoid a synchronization loop. If the client is requesting the reference IDs in multiple chunks, it SHOULD NOT select the server for synchronization until it received the whole set.

A client which uses multiple servers MUST be able to handle servers providing timestamps in different timescales. It can ignore servers not using the most common or preferred timescale, or convert them to a common timescale if it knows the offsets between them.

If the client synchronizes its clock to a leap-smeared timescale, it MUST NOT apply leap seconds and it SHOULD provide the same timescale to its own clients if it is a server.

The client SHOULD periodically (e.g. every two weeks) refresh IP addresses of all servers specified by hostname to limit the duration when an IP address of a migrated or decommissioned server will still be receiving NTP traffic from long-running clients. The client SHOULD ignore the TTL value of the DNS record that provided the IP address to avoid excessive DNS traffic for pools of NTP servers using a short TTL for better load balancing.

10. Server Operation

A server receives requests on the UDP port 123. The server MUST support measurements in the basic mode. It MAY support the interleaved mode.

For the basic mode the server does not need to keep any client-specific state. For the interleaved mode it needs to save transmit timestamps and be able to identify them by a cookie.

The server maintains its leap indicator, stratum, root delay, and root dispersion:

* Leap indicator MUST be 3 if the clock is not synchronized or its maximum error cannot be estimated by root delay and dispersion. Otherwise, it MUST be 0, 1, 2, depending on whether a leap second is pending in the next 14 days and, if it is, whether it will be inserted or deleted.

* Stratum SHOULD be 1 if the server is synchronized to a local reference clock, or be one larger than the stratum of the NTP server it selected as best for synchronization. It MUST be greater than the minimum stratum of all servers selected for synchronization, as they provided in their last accepted response, or 0 to indicate infinity.

* Root delay MUST be an estimate of the double of the maximum error of the server's clock due to asymmetries in the network delay and other delays (e.g. timestamping) on the path to the primary time source. For a server that is synchronized to other NTP servers, one possibility is to set it to the latest measured delay to the server it considers best plus the root delay provided in that response.

* Root dispersion MUST be an estimate of the maximum error of the server's clock from other causes than those covered by root delay, e.g. instability of the clock and measurements by which it is synchronized. If the server is synchronized to other NTP servers, it MUST include root dispersion from one of the servers' responses.

The server has a randomly generated 120-bit reference ID. It MUST track reference IDs of its servers in order to be able to respond with a Reference IDs Response Extension Field.

For each received request, the server:

1. Captures a receive timestamp.

2. Checks the version in the request. If it is not equal to 5, it MUST either drop the request, or handle it according to the specification corresponding to the protocol version.

3. Drops the request if the format is not valid, mode is not 3, or authentication fails with the Message Authentication Code Extension Field or another authenticator which does not have a specified response for failed authentication. The server MUST ignore unknown extension fields.

4. Server forms a response with requested extension fields and sets the fields in the header as follows:

* Leap Indicator, Stratum, Root delay, and Root dispersion, are set to the current server's values.

* Version is set to 5.

* Poll is set to the minimum allowed polling interval of the client (e.g. specified in the server configuration, using the same value for all clients).

* Precision is set to the expected precision of the receive and transmit timestamps included in the response (e.g. estimated on the start of the server or specified in its configuration).

* Timescale is set to the client's requested timescale if it is supported by the server. If not, the server SHOULD respond in any timescale it supports.

* The flags are set as follows:

  Synchronized is set if the server's clock is considered to be synchronized, and the receive and transmit timestamps provided in the response are usable for synchronization of the client.

  Interleaved mode is set if the interleaved mode is implemented, was requested, and a response in the interleaved mode is possible (i.e. a transmit timestamp is associated with the server cookie).

* Era is set to the NTP era of the receive timestamp.

* Server Cookie is set when the interleaved mode is requested and it is supported by the server, even if the response cannot be in the requested mode due to the request having an unknown or zero server cookie. The cookie identifies a more accurate transmit timestamp of the response, which can be retrieved by the client later with another request. The cookie MUST be unique in a sufficiently long interval to prevent a client from accepting a transmit timestamp that does not correspond to the previous response it received. The cookie generation is implementation-specific. For example, it can be a counter incremented on each received request, a randomly generated value, a timestamp, or an encrypted counter or timestamp making the value unpredictable.

* Client Cookie is set to the Client Cookie from the request.

* Receive Timestamp is set to the server's receive timestamp of the request.

* Transmit Timestamp is set to a value which depends on the measurement mode. In the basic mode it is the server's current time when the message is formed. In the interleaved mode it is the transmit timestamp of the previous response identified by the server cookie in the request, captured at some point after the message was formed.

5. The length of the response MUST be equal to the length of the request. The server adds the Padding Extension Field or increases its length if necessary to make the length of the response equal to the length of the request.

6. Drops the response if it is longer than the request to prevent traffic amplification.

7. Sends the response.

8. Saves the transmit timestamp and server cookie, if the interleaved mode was requested and is supported by the server.

11. Network Time Security with NTPv5

The Network Time Security [RFC8915] mechanism uses the NTS-KE protocol to establish keys and negotiate the next protocol. NTPv5 can be indicated as the next protocol with identifier [[TBD]] (draft use 0x8001). This can be used by clients and servers to negotiate NTPv5 for an NTS session.

No new NTS-KE records are specified for NTPv5. The records that were specified for NTPv4 (i.e. NTPv4 New Cookie, NTPv4 Server Negotiation, and NTPv4 Port Negotiation) are reused for NTPv5.

The NTS extension fields specified for NTPv4 are compatible with NTPv5. No new extension fields are specified.

(Note to editor: remove this paragraph before publishing.) Client implementations of a draft of this specification MUST provide the identity of the draft implemented as data in an NTS-KE record of type 0x4001, which does not have the critical bit set. The draft identity MUST be encoded as ascii and MUST not contain any trailing 0 bytes. Servers that implement a draft MUST not accept NTPv5 as an option unless they support the specific draft version identified.

12. NTPv5 Negotiation in previous NTP versions

Servers that support multiple versions of NTP MUST send a response in the same version as the request, to support backwards compatibility. This does not preclude servers from acting as a client in one version of NTP and a server in another.

NTPv5 messages are not compatible with NTPv4 and other previous versions of NTP, even if they do not contain any extension fields. Some widely used server NTPv4 implementations are known to accept requests indicating a higher version, interpreting them as NTPv4, and copying the version number from the request to its NTPv4 response. Responses to NTPv5 requests have a zero client cookie, which causes them to fail the NTPv5 validation and be ignored by the client.

The implementations are also known to not respond to requests with an unknown extension field, which prevents an NTPv4 extension field to be specified for NTPv5 negotiation. Instead, the negotiation reuses the reference timestamp field.

An NTP server which supports NTPv5 and also NTPv4, NTPv3, NTPv2, or NTPv1, SHOULD check the reference timestamp in received client-mode requests of the previous NTP versions. If the reference timestamp contains the value 0x4E5450354E545035 ("NTP5NTP5" in ASCII), it SHOULD respond with the same reference timestamp to indicate it supports NTPv5.

Note to the editor: Remove this paragraph before publication. Implementations of a draft of this specification SHOULD use 0x4E54503544524654 ("NTP5DRFT" in ASCII) instead of 0x4E5450354E545035.

When NTPv5 is not expected to be widely supported on servers yet, an NTP client which supports both NTPv5 and a previous NTP version, is not configured to use a specific NTP version, and does not use NTS or other mechanism including negotiation of the NTP version, SHOULD start with the previous version and set the reference timestamp to 0x4e5450354e545035. If the server responds with the same reference timestamp, the client SHOULD switch to NTPv5. If no valid response is received for a number of requests (e.g. 2), the client SHOULD switch back to the orignal NTP version and stick with it for a larger number of requests (e.g. 256) before trying NTPv5 again.

The special value of the reference timestamp corresponds to 1941-08-24T01:13:25Z in NTP era 0 and 2077-09-29T07:41:41Z in NTP era 1. The negotiation will probably not be needed at that time anymore. If NTPv5 servers and NTPv4-or-older-only clients are still in use and they send a request with the special value by chance, they will get

an acceptable response with no side effects. If an NTPv5 client gets the special value by chance from an NTPv4-or-older-only server, it will switch to NTPv5 and back after missing a valid response.

13. Acknowledgements

Some ideas were taken from a different NTPv5 design proposed by Daniel Franke.

The author would like to thank Doug Arnold and David Venhoek for their contributions and Dan Drown, Watson Ladd, Hal Murray, Kurt Roeckx, and Ulrich Windl for their suggestions and comments.

14. IANA Considerations

IANA is requested to allocate the following entries in the NTP Extension Field Types Registry [RFC9748]:

Field Type	Meaning	Reference
[[TBD]]	Padding	[[this memo]] (Section 7.2)
[[TBD]]	Message Authentication Code	[[this memo]] (Section 7.3)
[[TBD]]	Reference IDs Request	[[this memo]] (Section 7.4)
[[TBD]]	Reference IDs Response	[[this memo]] (Section 7.4)
[[TBD]]	Server Information	[[this memo]] (Section 7.5)
[[TBD]]	Correction	[[this memo]] (Section 7.6)
[[TBD]]	Reference Timestamp	[[this memo]] (Section 7.7)
[[TBD]]	Monotonic Receive Timestamp	[[this memo]] (Section 7.8)
[[TBD]]	Secondary Receive Timestamp	[[this memo]] (Section 7.9)

Table 1

IANA is requested to allocate the following entry in the Network Time Security Next Protocols Registry [RFC8915]:

Protocol ID	Protocol Name	Reference
[[TBD]], selected by IANA from the IETF Review range	Network Time Protocol version 5 (NTPv5)	[[this memo]] (Section 11)

Table 2

15. Security Considerations

The security considerations of time protocols in general are discussed in [RFC7384]. A successful attack on the time distribution can result in one of three outcomes: Denial of Service (DoS), reduced accuracy, or obtaining incorrect time. NTP can be compromised through various methods, such as altering or delaying NTP messages during transit, injecting malicious NTP messages or replaying valid ones, or impersonating an NTP server.

The Message Authentication Code Extension Field can be used to provide integrity protection, thus mitigating in-transit NTP message modification and malicious packet injection.

Using NTS with NTPv5 provides enhanced security properties, including server identity verification, improved replay protection, and secure key establishment.

NTPv5 was designed to minimize the necessary on-the-wire data that is included in the NTPv5 header in order to limit the amount of information that is exposed to the network and minimize the potential effect of network reconnaissance. For example, the client’s transmission timestamp is not included in the NTPv5 header. Extension fields are used for conveying optional information.

The protocol operates in a client-server mode. Other modes of operation, which were supported in the previous version of the protocol have known security vulnerabilities. The symmetric and broadcast modes are vulnerable to replay attacks. The control and private modes can be exploited for denial-of-service traffic amplification attacks. These modes are not supported in NTPv5.

The NTP response message has the same length as the corresponding request message. This symmetric approach reduces the potential of amplification attacks.

The protocol does not inherently mitigate delay attacks. An on-path attacker can delay NTP messages and compromise the accuracy. Delay attacks can be mitigated by limiting the maximum acceptable delay. In each request-response exchange the client computes the delay and can discard the response if the delay exceeds a maximum expected value. A more fine-grained mitigation approach is acquiring time from multiple sources and selecting the sources that are most likely to be accurate. A detailed scheme was defined in [RFC5905] and can be used in NTPv5. This scheme defines a selection algorithm, which identifies the most accurate and reliable time sources from a pool of candidates. These sources are then grouped by the clustering algorithm to minimize the impact of outliers. Finally, the combining algorithm averages the time data from these clustered sources to produce a final, stable time estimate. An enhancement to the selection algorithm was proposed in [RFC9523]. An alternative approach to using multiple time sources is using multiple diverse paths between the client and server [RFC8039], as it is assumed that only a subset of the paths is compromised by an on-path attacker.

The corrections provided by network devices in the Correction Extension Field are not authenticated. Man-in-the-middle attackers can modify the correction values. In order to mitigate such attacks, the client validates the correction values as described in Section 8. Thus, only corrections smaller than the measured delay are accepted by clients. The security impact is comparable to the impact of delaying unmodified NTP messages, as described above.

16. References

16.1. Normative References

[RFC1321] Rivest, R., "The MD5 Message-Digest Algorithm", RFC 1321, DOI 10.17487/RFC1321, April 1992, <https://www.rfc-editor.org/info/rfc1321>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8573] Malhotra, A. and S. Goldberg, "Message Authentication Code for the Network Time Protocol", RFC 8573, DOI 10.17487/RFC8573, June 2019, <https://www.rfc-editor.org/info/rfc8573>.

16.2. Informative References

[Bloom] Bloom, B. H., "Space/Time Trade-Offs in Hash Coding with Allowable Errors", June 1970, <https://doi.org/10.1145/362686.362692>.

[IEEE1588] Institute of Electrical and Electronics Engineers, "IEEE std. 1588-2019, "IEEE Standard for a Precision Clock Synchronization for Networked Measurement and Control Systems."", November 2019, <https://www.ieee.org>.

[RFC5905] Mills, D., Martin, J., Ed., Burbank, J., and W. Kasch, "Network Time Protocol Version 4: Protocol and Algorithms Specification", RFC 5905, DOI 10.17487/RFC5905, June 2010, <https://www.rfc-editor.org/info/rfc5905>.

[RFC7384] Mizrahi, T., "Security Requirements of Time Protocols in Packet Switched Networks", RFC 7384, DOI 10.17487/RFC7384, October 2014, <https://www.rfc-editor.org/info/rfc7384>.

[RFC7821] Mizrahi, T., "UDP Checksum Complement in the Network Time Protocol (NTP)", RFC 7821, DOI 10.17487/RFC7821, March 2016, <https://www.rfc-editor.org/info/rfc7821>.

[RFC7822] Mizrahi, T. and D. Mayer, "Network Time Protocol Version 4 (NTPv4) Extension Fields", RFC 7822, DOI 10.17487/RFC7822, March 2016, <https://www.rfc-editor.org/info/rfc7822>.

[RFC8039] Shpiner, A., Tse, R., Schelp, C., and T. Mizrahi, "Multipath Time Synchronization", RFC 8039, DOI 10.17487/RFC8039, December 2016, <https://www.rfc-editor.org/info/rfc8039>.

[RFC8877] Mizrahi, T., Fabini, J., and A. Morton, "Guidelines for Defining Packet Timestamps", RFC 8877, DOI 10.17487/RFC8877, September 2020, <https://www.rfc-editor.org/info/rfc8877>.

[RFC8915] Franke, D., Sibold, D., Teichel, K., Dansarie, M., and R. Sundblad, "Network Time Security for the Network Time Protocol", RFC 8915, DOI 10.17487/RFC8915, September 2020, <https://www.rfc-editor.org/info/rfc8915>.

[RFC9523] Rozen-Schiff, N., Dolev, D., Mizrahi, T., and M. Schapira, "A Secure Selection and Filtering Mechanism for the Network Time Protocol with Khronos", RFC 9523, DOI 10.17487/RFC9523, February 2024, <https://www.rfc-editor.org/info/rfc9523>.

[RFC9748] Salz, R., "Updating the NTP Registries", RFC 9748, DOI 10.17487/RFC9748, February 2025, <https://www.rfc-editor.org/info/rfc9748>.

[RFC9769] Lichvar, M. and A. Malhotra, "NTP Interleaved Modes", RFC 9769, DOI 10.17487/RFC9769, May 2025, <https://www.rfc-editor.org/info/rfc9769>.

Authors' Addresses

Miroslav Lichvar
Red Hat
Email: mlichvar@redhat.com

Tal Mizrahi
Huawei
Email: tal.mizrahi.phd@gmail.com