Stream:	Internet Engineering Task Force (IETF)
RFC:	9769
Updates:	5905
Category:	Standards Track
Published:	April 2025
ISSN:	2070-1721
Authors:	M. Lichvar	A. Malhotra
---	---	---
	Red Hat	*Boston University*

RFC 9769
NTP Interleaved Modes

Abstract

This document specifies interleaved modes for the Network Time Protocol (NTP). These new modes improve the accuracy of time synchronization by enabling the use of more accurate transmit timestamps that are available only after the transmission of NTP messages. These enhancements are intended to improve timekeeping in environments where high accuracy is critical. This document updates RFC 5905 by defining these new operational modes.

Status of This Memo

This is an Internet Standards Track document.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Further information on Internet Standards is available in Section 2 of RFC 7841.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at https://www.rfc-editor.org/info/rfc9769.

Copyright Notice

Table of Contents

1. Introduction

RFC 5905 [RFC5905] describes the operations of NTPv4 in a client/server mode, symmetric mode, and broadcast mode. The transmit and receive timestamps are two of the four timestamps included in every NTPv4 packet used for time synchronization.

For a highly accurate and stable synchronization, the transmit and receive timestamps should be captured close to the beginning of the actual transmission and the end of the reception, respectively. An asymmetry in the timestamping causes the offset measured by NTP to have an error.

There are at least four options where a timestamp of an NTP packet may be captured with a software NTP implementation running on a general-purpose operating system:

1. User space (software)
2. Network device driver or kernel (software)
3. Data link layer (hardware - MAC chip)
4. Physical layer (hardware - PHY chip)

Software timestamps captured in the user space in the NTP implementation itself are the least accurate. They do not account for delays due to system calls used for sending and receiving packets, processing and queuing delays in the system, network device drivers, and hardware. Hardware timestamps captured at the physical layer are the most accurate.

A transmit timestamp captured in the driver or hardware is more accurate than the user-space timestamp, but it is available to the NTP implementation only after it sent the packet using a system call. The timestamp cannot be included in the packet itself unless the driver or hardware supports NTP and can modify the packet before or during the actual transmission.

The protocol described in RFC 5905 [RFC5905] does not specify any mechanism for a server to provide its clients and peers with a more accurate transmit timestamp that is known only after the transmission. A packet that strictly follows RFC 5905 [RFC5905], i.e., that contains a transmit timestamp corresponding to the packet itself, is said to be in the basic mode.

Different mechanisms could be used to exchange timestamps known after the transmission. The server could respond to each request with two packets. The second packet would contain the transmit timestamp corresponding to the first packet. However, such a protocol would enable a traffic amplification attack, or it would use packets with an asymmetric length, which would cause an asymmetry in the network delay and an error in the measured offset.

This document describes an interleaved client/server mode, interleaved symmetric mode, and interleaved broadcast mode. In these modes, the server sends a packet that contains a transmit timestamp corresponding to the transmission of the previous packet that was sent to the client or peer. This transmit timestamp can be captured in any software or hardware component involved in the transmission of the packet. Both servers and clients/peers are required to keep some state specific to the interleaved mode.

An NTPv4 implementation that supports the client/server and broadcast interleaved modes interoperates with NTPv4 implementations without this capability. A peer using the symmetric interleaved mode does not fully interoperate with a peer that does not support it. The mode needs to be configured specifically for each symmetric association.

The interleaved modes do not change the NTP packet header format and do not use new extension fields. The negotiation is implicit. The protocol is extended with new values that can be assigned to the origin and transmit timestamps. Servers and peers check the origin timestamp to detect requests conforming to the interleaved mode. A response can only be valid in one mode. If a client or peer that does not support the interleaved mode received a response conforming to the interleaved mode, it would be rejected as bogus.

An explicit negotiation would require a new extension field. RFC 5905 [RFC5905] does not specify how servers should handle requests with an unknown extension field. The original use of extension fields was authentication with Autokey [RFC5906], which cannot be negotiated. Some existing implementations do not respond to requests with unknown extension fields. This behavior would prevent clients from reliably detecting support for the interleaved mode.

Requests and responses cannot always be formed in the interleaved mode. It cannot be used exclusively. Servers, clients, and peers that support the interleaved mode need to also support the basic mode.

This document assumes familiarity with RFC 5905 [RFC5905].

1.1. 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.

2. Interleaved Client/Server Mode

The interleaved client/server mode is similar to the basic client/ server mode. The difference between the two modes is in the values saved to the origin and transmit timestamp fields.

The origin timestamp is a cookie that is used to detect that a received packet is a response to the last packet sent in the other direction of the association. It is a copy of one of the timestamps from the packet to which it is responding, or zero if it is not a response. Servers following RFC 5905 [RFC5905] ignore the origin timestamp in client requests. A server response that does not have a matching origin timestamp is considered bogus.

A client request in the basic mode has an origin timestamp equal to the transmit timestamp from the last valid server response, or the origin timestamp is zero (which indicates the first request of the association). A server response in the basic mode has an origin timestamp equal to the transmit timestamp from the client request. The transmit timestamp in the response corresponds to the transmission of the response in which the timestamp is contained.

A client request in the interleaved mode has an origin timestamp equal to the receive timestamp from the last valid server response. A server response in the interleaved mode has an origin timestamp equal to the receive timestamp from the client request. The transmit timestamp in the response corresponds to the transmission of the previous response that had the receive timestamp equal to the origin timestamp from the request.

A server that supports the interleaved mode needs to save pairs of local receive and transmit timestamps. The server **SHOULD** discard old timestamps to limit the amount of memory used for the interleaved mode, e.g., by using a fixed-length queue and dropping old timestamps as new timestamps are saved. The server **MAY** separate the timestamps by IP addresses, but it **SHOULD NOT** separate them by port numbers to support clients that change their port between requests, as recommended in RFC 9109 [RFC9109].

The server **MAY** restrict the interleaved mode to specific IP addresses and/or authenticated clients.

Both servers and clients that support the interleaved mode **MUST NOT** send a packet that has a transmit timestamp equal to the receive timestamp in order to reliably detect whether received packets conform to the interleaved mode. One way to ensure this behavior is to increment the transmit timestamp by 1 unit (i.e., about 1/4 of a nanosecond) if the two timestamps are equal, or a new timestamp can be generated.

The transmit and receive timestamps in server responses need to be unique to prevent two different clients from sending requests with the same origin timestamp and the server responding in the interleaved mode with an incorrect transmit timestamp. If the timestamps are not guaranteed to be monotonically increasing, the server **SHOULD** check that the transmit and receive timestamps are not already saved as a receive timestamp of a previous request (from the same IP address if the server separates timestamps by addresses), and generate a new timestamp if necessary, to prevent an incorrect interleaved response later.

When the server receives a request from a client, it **MUST NOT** respond in the interleaved mode unless the following two conditions are met:

1. The request does not have a receive timestamp equal to the transmit timestamp.
2. The origin timestamp from the request matches the local receive timestamp of a previous request that the server has saved (for the IP address if it separates timestamps by addresses).

A response in the interleaved mode **MUST** contain the transmit timestamp of the response that contained the receive timestamp matching the origin timestamp from the request. The server can drop the timestamps after sending the response. The receive timestamp **MUST NOT** be used again to detect a request conforming to the interleaved mode.

If the conditions are not met (i.e., the request is not detected to conform to the interleaved mode), the server **MUST NOT** respond in the interleaved mode. If it responds, it **MUST** be in the basic mode. In any case, the server **SHOULD** save the new receive and transmit timestamps to be able to respond in the interleaved mode to the next request from the client.

The first request from a client is always in the basic mode, and so is the server response. It has a zero origin timestamp and zero receive timestamp. Only when the client receives a valid response from the server will it be able to send a request in the interleaved mode.

The protocol recovers from packet loss. When a client request or server response is lost, the client will use the same origin timestamp in the next request. The server can respond in the interleaved mode if it still has the timestamps corresponding to the origin timestamp. If the server already responded to the timestamp in the interleaved mode or it had to drop the timestamps for other reasons, it will respond in the basic mode and save new timestamps, which will enable an interleaved response to the subsequent request. The client **SHOULD** limit the number of requests in the interleaved mode between server responses to prevent the processing of very old timestamps in cases where a large number of consecutive requests are lost.

An example of packets in a client/server exchange using the interleaved mode is shown in Figure 1. The packets in the basic and interleaved modes are indicated with B and I, respectively. The timestamps t1~, t3~, and t11~ point to the same transmissions as t1, t3, and t11, but they may be

less accurate. 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 did not have the pair of timestamps t6 and t7 (e.g., they were dropped to save timestamps for other clients using the interleaved mode).

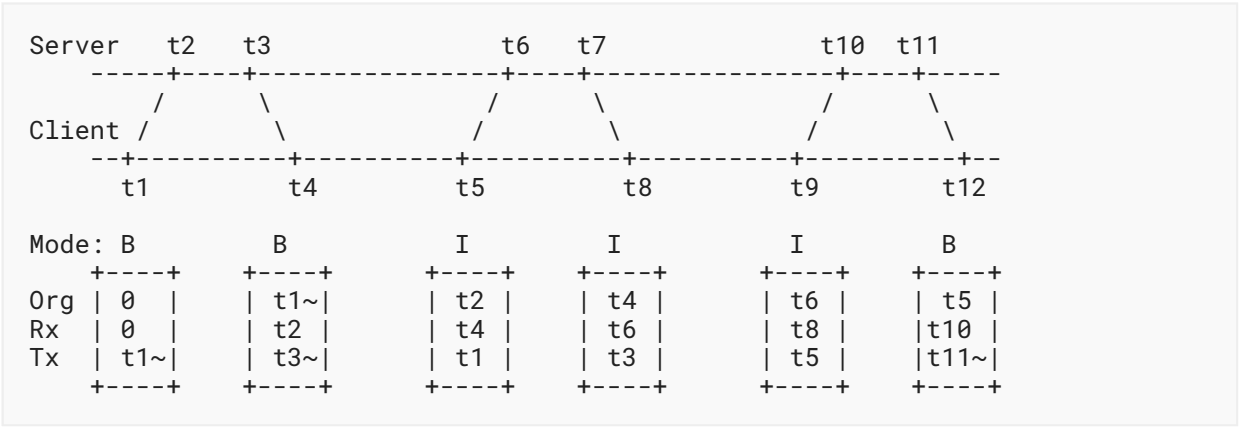

Figure 1: Packet Timestamps in Interleaved Client/Server Mode

When the client receives a response from the server, it performs the tests described in RFC 5905 [RFC5905]. Two of the tests are modified for the interleaved mode:

1. The check for duplicate packets compares both receive and transmit timestamps in order to not drop a valid response in the interleaved mode if it follows a response in the basic mode and they contain the same transmit timestamp.
2. The check for bogus packets compares the origin timestamp with both transmit and receive timestamps from the request. If the origin timestamp is equal to the transmit timestamp, the response is in the basic mode. If the origin timestamp is equal to the receive timestamp, the response is in the interleaved mode.

The client **SHOULD NOT** update its NTP state when an invalid response is received, so that the timestamps that will be needed to complete a measurement when the subsequent response in the interleaved mode is received will not be lost.

If the packet passed the tests and conforms to the interleaved mode, the client can compute the offset and delay using the formulas from RFC 5905 [RFC5905] and one of two different sets of timestamps. The first set is **RECOMMENDED** for clients that filter measurements based on the delay. The corresponding timestamps from Figure 1 are written in parentheses.

- T1 - local transmit timestamp of the previous request (t1)
- T2 - remote receive timestamp from the previous response (t2)
- T3 - remote transmit timestamp from the latest response (t3)
- T4 - local receive timestamp of the previous response (t4)

The second set gives a more accurate measurement of the current offset, but the delay is much more sensitive to a frequency error between the server and client due to a much longer interval between T1 and T4.

- T1 - local transmit timestamp of the latest request (t5)
- T2 - remote receive timestamp from the latest response (t6)
- T3 - remote transmit timestamp from the latest response (t3)
- T4 - local receive timestamp of the previous response (t4)

Clients **MAY** filter measurements based on the mode. The maximum number of dropped measurements in the basic mode **SHOULD** be limited in cases where the server does not support, or is not able to respond in, the interleaved mode. Clients that filter measurements based on the delay will implicitly prefer measurements in the interleaved mode over the basic mode, because they have a shorter delay due to a more accurate transmit timestamp (T3).

The server **MAY** limit the saving of the receive and transmit timestamps to requests that have an origin timestamp specific to the interleaved mode in order to not waste resources on clients using the basic mode. Such an optimization will delay the first interleaved response of the server to a client by one exchange.

A check for a non-zero origin timestamp works with NTP clients that always set the timestamp to zero. From the server's point of view, such clients start a new association with each request.

To avoid searching the saved receive timestamps for non-zero origin timestamps from requests conforming to the basic mode, the server can encode in low-order bits of the receive and transmit timestamps below the precision of the clock a flag indicating whether the timestamp is a receive timestamp. If the server receives a request with a non-zero origin timestamp that does not indicate that it is a receive timestamp of the server, the request does not conform to the interleaved mode, and it is not necessary to perform the search and/or save the new receive and transmit timestamps.

3. Interleaved Symmetric Mode

The interleaved symmetric mode uses the same principles as the interleaved client/server mode. A packet in the interleaved symmetric mode has a transmit timestamp that corresponds to the transmission of the previous packet sent to the peer and an origin timestamp equal to the receive timestamp from the last packet received from the peer.

To enable synchronization in both directions of a symmetric association, both peers need to support the interleaved mode. For this reason, it **SHOULD** be disabled by default and enabled with an option in the configuration of the active side of the association.

In order to prevent the peer from matching the transmit timestamp with an incorrect packet when the peers' transmissions do not alternate (e.g., they use different polling intervals) and a previous packet was lost, the use of the interleaved mode in symmetric associations requires additional restrictions.

Peers that have an association need to count valid packets received between their transmissions to determine in which mode a packet should be formed. A valid packet in this context is a packet that passed all NTP tests for duplicate, replayed, bogus, and unauthenticated packets. Other received packets may update the NTP state to allow the (re)initialization of the association, but they do not change the selection of the mode.

A Peer A **MUST NOT** send a Peer B a packet in the interleaved mode unless all of the following conditions are met:

1. Peer A has an active association with Peer B that was specified with the option enabling the interleaved mode, OR Peer A received at least one valid packet in the interleaved mode from Peer B.
2. Peer A did not send a packet to Peer B since the time that it received the last valid packet from Peer B.
3. The previous packet that Peer A sent to Peer B was the only response to a packet received from Peer B.

The first condition is needed for compatibility with implementations that do not support, or are not configured for, the interleaved mode. The other conditions prevent a missing response from causing a mismatch between the remote transmit timestamp (T2) and local receive timestamp (T3), which would cause a large error in the measured offset and delay.

An example of packets exchanged in a symmetric association is shown in Figure 2. The minimum polling interval of Peer A is twice as long as the maximum polling interval of Peer B. The first packet sent by each peer is in the basic mode. The second and third packets sent by Peer A are in the interleaved mode. The second packet sent by Peer B is in the interleaved mode, but subsequent packets sent by Peer B are in the basic mode, because multiple responses are sent for each request.

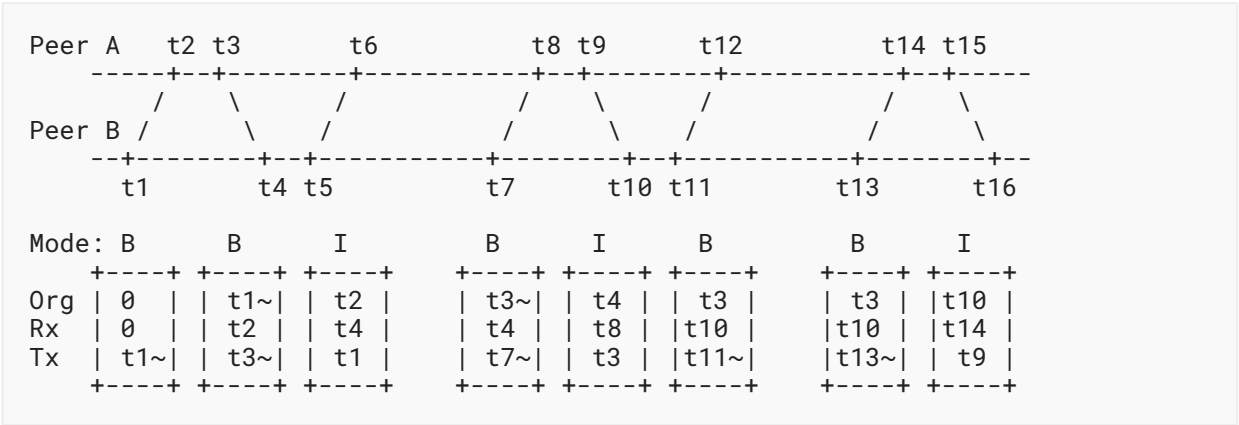

Figure 2: Packet Timestamps in Interleaved Symmetric Mode

If Peer A has no association with Peer B and it responds with symmetric passive packets, it does not need to count the packets in order to meet the restrictions, because each request has at most one response. The processing of the requests can be implemented in the same way as a server handling requests in the interleaved client/server mode.

The peers can compute the offset and delay using one of the two sets of timestamps specified in Section 2. They can switch between the sets to minimize the interval between T1 and T4 in order to reduce the error in the measured delay.

4. Interleaved Broadcast Mode

A packet in the interleaved broadcast mode contains two transmit timestamps. One corresponds to the packet itself and is saved in the transmit timestamp field. The other corresponds to the previous packet and is saved in the origin timestamp field. The packet is compatible with the basic mode, which uses a zero origin timestamp.

An example of packets sent in the broadcast mode is shown in Figure 3.

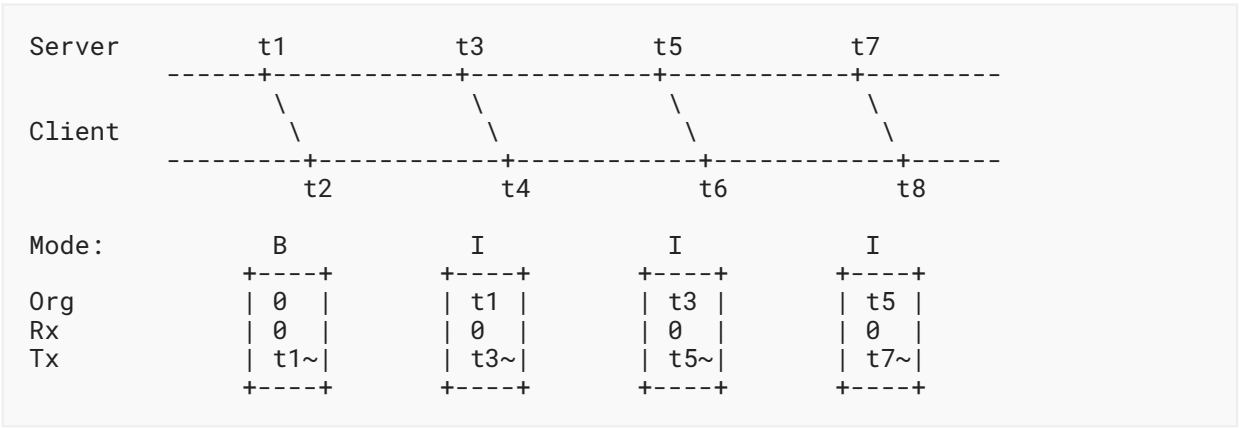

Figure 3: Packet Timestamps in Interleaved Broadcast Mode

A client that does not support the interleaved mode ignores the origin timestamp and processes all packets as if they were in the basic mode.

A client that supports the interleaved mode **MUST** check if the origin timestamp is not zero to detect packets conforming to the interleaved mode. The client **SHOULD** also compare the origin timestamp with the transmit timestamp from the previous packet to detect lost packets. If the difference is larger than a specified maximum (e.g., 1 second), the packet **SHOULD NOT** be used for synchronization in the interleaved mode to avoid a large error in the measurement.

The client computes the offset using the origin timestamp from the received packet and the local receive timestamp of the previous packet. If the client needs to measure the network delay, it **SHOULD** use the interleaved client/server mode. If it used the basic client/server mode or symmetric mode, the less accurate measurement of the delay would also impact the accuracy of the offset measured in the interleaved broadcast mode.

5. Impact of Implementation Errors

The interleaved modes make NTP more complex and more sensitive to implementation errors. Some errors that do not cause any problems between implementations supporting only the basic mode can cause an occasional missing or corrupted measurement when one or both sides support the interleaved mode. This section describes the impact of what could possibly be the most likely errors in the most commonly used mode -- client/server.

If a client that does not support the interleaved mode sets the origin timestamp to values other than the transmit timestamp from the last valid server response, or zero, the origin timestamp can match a receive timestamp of a previous server response (possibly to a different client) and cause an unexpected interleaved response. The client is expected to drop the response as bogus due to having a wrong origin timestamp. If it did not check for bogus responses, it would get a corrupted measurement, possibly with a large error in the offset and delay. It would also be vulnerable to off-path attacks.

The worst-case scenario for this failure would be a client that specifically sets the origin timestamp to the server's receive timestamp, i.e., the client accidentally implemented the interleaved mode, but it does not accept interleaved responses. This client would still be able to synchronize its clock. It would drop interleaved responses as bogus and set the origin timestamp to the receive timestamp from the last valid response in the basic mode. As servers are required to not respond twice to the same origin timestamp in the interleaved mode, at least every other response would be in the basic mode and accepted by the client.

A missing or corrupted measurement can also be caused by problems on the server side. A server that does not ensure that the receive and transmit timestamps in its responses are unique in a sufficiently long interval can misinterpret requests formed correctly in the basic mode as interleaved and respond in the interleaved mode. The response would be dropped by the client as bogus.

A duplicated server receive timestamp can cause an expected interleaved response to contain a transmit timestamp that does not correspond to the transmission of the previous response from which the client copied the receive timestamp to the origin timestamp in the request, but a different response that contained the same receive timestamp. The response would be accepted by the client with a small error in the transmit timestamp equal to the difference between the transmit timestamps of the two different responses. As the two requests to which the responses responded were received at the same time (according to the server's clock), the two transmissions would be expected to be close to each other and the difference between them would be comparable to the error a basic response normally has in its transmit timestamp.

6. Security Considerations

The security considerations for time protocols in general are discussed in RFC 7384 [RFC7384]. Security considerations specific to NTP are discussed in RFC 5905 [RFC5905].

Security issues that apply to the basic modes also apply to the interleaved modes. They are described in "The Security of NTP's Datagram Protocol" [SECNTP].

Clients and peers **SHOULD NOT** leak the receive timestamp in packets sent to other peers or clients (e.g., as a reference timestamp) to prevent off-path attackers from easily getting the origin timestamp needed to make a valid response in the interleaved mode.

Clients using the interleaved mode **SHOULD** randomize all bits of receive and transmit timestamps in their requests (i.e., use a precision of 32) to make it more difficult for off-path attackers to guess the origin timestamp in the server response.

Unlike in the basic client/server mode, clients using the interleaved mode cannot set the origin timestamp in their requests to zero (i.e., reset the NTP association with every request) to make it more difficult to track them as they move between networks.

Attackers can force the server to drop its timestamps in order to prevent clients from getting an interleaved response. They can send a large number of requests, send requests with a spoofed source address, or replay an authenticated request if the interleaved mode is enabled only for authenticated clients. Clients **MUST NOT** rely on servers to be able to respond in the interleaved mode.

Protecting symmetric associations in the interleaved mode against replay attacks is even more difficult than in the basic mode. In both modes, the NTP state needs to be protected between the reception of the last non-replayed response and transmission of the next request in order for the request to contain the origin timestamp expected by the peer. The difference is in the timestamps needed to complete a measurement. In the basic mode, only one valid response is needed at a time and it is used as soon as it is received, but the interleaved mode needs two consecutive valid responses. The NTP state needs to be protected at all times, so that the timestamps that are needed to complete the measurement when the second response is received will not be lost.

7. IANA Considerations

This document has no IANA actions.

8. References

8.1. Normative References

[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>.

[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>.

[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>.

8.2. Informative References

[RFC5906] Haberman, B., Ed. and D. Mills, "Network Time Protocol Version 4: Autokey Specification", RFC 5906, DOI 10.17487/RFC5906, June 2010, <https://www.rfc-editor.org/info/rfc5906>.

[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>.

[RFC9109] Gont, F., Gont, G., and M. Lichvar, "Network Time Protocol Version 4: Port Randomization", RFC 9109, DOI 10.17487/RFC9109, August 2021, <https://www.rfc-editor.org/info/rfc9109>.

[SECNTP] Malhotra, A., Van Gundy, M., Varia, M., Kennedy, H., Gardner, J., and S. Goldberg, "The Security of NTP's Datagram Protocol", Cryptology ePrint Archive, Paper 2016/1006, 2016, <https://eprint.iacr.org/2016/1006>.

Acknowledgements

The interleaved modes described in this document are based on the implementation written by David Mills in the NTP project. The specification of the broadcast mode is based purely on this implementation. The specification of the symmetric mode has some modifications. The client/server mode is specified as a new mode compatible with the symmetric mode, similarly to the basic symmetric and client/server modes.

The authors would like to thank Doug Arnold, Roman Danyliw, Reese Enghardt, Daniel Franke, Benjamin Kaduk, Erik Kline, Catherine Meadows, Tal Mizrahi, Steven Sommars, Harlan Stenn, Kristof Teichel, and Gunter Van de Velde for their useful comments and suggestions.

Authors' Addresses

Miroslav Lichvar
Red Hat
Purkynova 115
612 00 Brno
Czech Republic
Email: mlichvar@redhat.com

Aanchal Malhotra
Boston University
111 Cummington St
Boston, MA 02215
United States of America
Email: aanchal4@bu.edu