Network Working Group
Request for Comments: 1119
Obsoletes: RFC-1059, RFC-958

David L. Mills
University of Delaware
September 1989

Network Time Protocol (Version 2)
Specification and Implementation

Status of this Memo

This document describes the Network Time Protocol (NTP), specifies its formal structure and summarizes information useful for its implementation. NTP provides the mechanisms to synchronize time and coordinate time distribution in a large, diverse internet operating at rates from mundane to lightwave. It uses a returnable-time design in which a distributed subnet of time servers operating in a self-organizing, hierarchical-master-slave configuration synchronizes local clocks within the subnet and to national time standards via wire or radio. The servers can also redistribute reference time via local routing algorithms and time daemons.

This is an Internet Standard Recommended Protocol. Distribution of this memo is unlimited.

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

Table of Contents

1. Introduction . . . 1
1.1. Related Technology . . . 2
2. System Architecture . . . 3
2.1. Implementation Model . . . 4
2.2. Network Configurations . . . 5
2.3. The NTP Timescale . . . 7
2.4. The NTP Calendar . . . 8
2.5. Time and Frequency Dissemination . . . 10
3. Network Time Protocol . . . 11
3.1. Data Formats . . . 11
3.2. State Variables and Parameters . . . 12
3.2.1. Common Variables . . . 12
3.2.2. System Variables . . . 14
3.2.3. Peer Variables . . . 16
3.2.4. Packet Variables . . . 17
3.2.5. Clock Filter Variables . . . 17
3.2.6. Parameters . . . 18
3.3. Modes of Operation . . . 19
3.4. Event Processing . . . 21
3.4.1. Transmit Procedure . . . 21
3.4.2. Receive Procedure . . . 23
3.4.3. Packet Procedure . . . 24
3.4.4. Primary-clock procedure . . . 26
3.4.5. Clock-update procedure . . . 27
3.4.6. Initialization Procedure . . . 28
3.4.7. Clear Procedure . . . 29
3.4.8. Poll-update procedure . . . 30
3.5. Access Control Issues . . . 30
4. Filtering and Selection Algorithms . . . 31
4.1. Clock-filter procedure . . . 31
4.2. Clock-selection procedure . . . 32
5. Local Clocks . . . 34
5.1. Standard Oscillators . . . 34
5.2. Mathematical Model . . . 36
5.3. Fuzzball Implementation . . . 38
5.4. Uniform Phase Adjustments . . . 39
5.5. Nonuniform Phase Adjustments . . . 40
5.6. Maintaining Date and Time . . . 40
6. Acknowledgments . . . 41
7. References . . . 41
8. Appendix A. NTP Data Format - Version 2 . . . 45

9. Appendix B. NTP Control Messages 48
9.1. NTP Control Message Format 49
9.2. Status Words 51
9.2.1. System Status Word 51
9.2.2. Peer Status Word 52
9.2.3. Clock Status Word 53
9.2.4. Error Status Word 53
9.3. Commands 54
10. Appendix C. Authentication Issues 56
10.1. NTP Authentication Mechanism 57
10.2. NTP Authentication Procedures 58
11. Appendix D. Differences from Previous Versions. 60

List of Figures

Figure 1. Implementation Model 4
Figure 2. Calculating Delay and Offset 25
Figure 3. Phase-Lock Loop Model 35
Figure 4. Clock Registers 37
Figure 5. NTP Message Header 45
Figure 6. NTP Control Message Header 49
Figure 7. Status Word Formats 50
Figure 8. Authenticator Format 58

List of Tables

Table 1. Dates of Leap-Second Insertion 8
Table 2. System Variables 14
Table 3. Peer Variables 15
Table 4. Packet Variables 16
Table 5. Parameters 17
Table 6. Modes and Actions 24
Table 7. Characteristics of Standard Oscillators 34
Table 8. Clock Parameters 38

1. Introduction

This document constitutes a formal specification of the Network Time Protocol (NTP), which is used to synchronize timekeeping among a set of distributed time servers and clients. It defines the architectures, algorithms, entities and protocols used by NTP and is intended primarily for implementors. A companion document [44] summarizes the requirements, analytical models, algorithmic analysis and performance under typical Internet conditions. NTP was first described in RFC-958 [30], but has evolved in significant ways, culminating in the most recent NTP Version 1 described in RFC-1059 [42]. It is built on the Internet Protocol (IP) [14] and User Datagram Protocol (UDP) [9], which provide a connectionless transport mechanism; however, it is readily adaptable to other protocol suites. NTP is evolved from the Time Protocol [19] and the ICMP Timestamp message [15], but is specifically designed to maintain accuracy and robustness, even when used over typical Internet paths involving multiple gateways, highly dispersive delays and unreliable nets.

The service environment consists of the implementation model, service model and timescale described in Section 2. The implementation model is based on a multiple-process operating system architecture, although other architectures could be used as well. The service model is based on a returnable-time design which depends only on measured clock offsets, but does not require reliable message delivery. The synchronization subnet uses a self-organizing, hierarchical-master-slave configuration, with synchronization paths determined by a minimum-weight spanning tree. While multiple masters (primary servers) may exist, there is no requirement for an election protocol.

NTP itself is described in Section 3. It provides the protocol mechanisms to synchronize time in principle to precisions in the order of nanoseconds while preserving a non-ambiguous date well into the next century. The protocol includes provisions to specify the characteristics and estimate the error of the local clock and the time server to which it may be synchronized. It also includes provisions for operation with a number of mutually suspicious, hierarchically distributed primary reference sources such as radio clocks.

Section 4 describes algorithms useful for deglitching and smoothing clock-offset samples collected on a continuous basis. These algorithms evolved from with suggested in [28], were refined as the results of experiments described in [29] and further evolved under typical operating conditions over the last three years. In addition, as the result of experience in operating multiple-server subnets including radio-synchronized clocks at several sites in the U.S. and with clients in the U.S. and Europe, reliable algorithms for selecting good clocks from a population possibly including broken ones have been developed and are described in Section 4.

The accuracies achievable by NTP depend strongly on the precision of the local-clock hardware and stringent control of device and process latencies. Provisions must be included to adjust the software logical-clock time and frequency in response to corrections produced by NTP. Section 5 describes a local-clock design evolved from the Fuzzball implementation described in [21] and [43]. This design includes offset-slewing, drift-compensation and deglitching mechanisms capable of accuracies in the order of a millisecond, even after extended periods when synchronization to primary reference sources has been lost.

Details specific to NTP packet formats used with the Internet Protocol (IP) and User Datagram Protocol (UDP) are presented in Appendix A, while details of a suggested auxiliary NTP Control Message, which may be used when comprehensive network-monitoring facilities are not available, are presented in Appendix B. Appendix C contains specification and implementation details of an optional authentication mechanism which can be used to control access and prevent unauthorized data modification. Appendix D contains a listing of differences between Version 2 of NTP and previous versions.

1.1. Related Technology

Other mechanisms have been specified in the Internet protocol suite to record and transmit the time at which an event takes place, including the Daytime protocol [18], Time Protocol [19], ICMP Timestamp message [15] and IP Timestamp option [13]. Experimental results on measured times and roundtrip delays in the Internet are discussed in [20], [29], [41] and [42]. Other synchronization algorithms are discussed in [4], [22], [23], [24], [25], [27], [28], [29], [30], [31], [33], [35], [38], [39], [40], [42] and [44], while protocols based on them are described in [11], [12], [21], [26], [30], [35], [42] and [44]. NTP uses techniques evolved from them and both linear-systems and agreement methodologies. Linear methods for digital telephone network synchronization are summarized in [6], while agreement methods for clock synchronization are summarized in [25].

The Fuzzball routing protocol [21], sometimes called Hellospeak, incorporates time synchronization directly into the routing-protocol design. One or more processes synchronize to an external reference source, such as a radio clock or NTP daemon, and the routing algorithm constructs a minimum-weight spanning tree rooted on these processes. The clock offsets are then distributed along the arcs of the spanning tree to all processes in the system and the various process clocks corrected using the procedure described in Section 5 of this document. While it can be seen that the design of Hellospeak strongly influenced the design of NTP, Hellospeak itself is not an Internet protocol and is unsuited for use outside its local-net environment.

The Unix 4.3bsd time daemon *timed* [26] uses a single master-time daemon to measure offsets of a number of slave hosts and send periodic corrections to them. In this model the master is determined using an election algorithm [31] designed to avoid situations where either no master is elected or more than one master is elected. The election process requires a broadcast capability, which is not a ubiquitous feature of the Internet. While this model has been extended to support hierarchical configurations in which a slave on one network serves as a master on the other [35], the model requires handcrafted configuration tables in order to establish the hierarchy and avoid loops. In addition to the burdensome, but presumably infrequent, overheads of the election process, the offset measurement/correction process requires twice as many messages as NTP per update.

A scheme with features similar to NTP is described in [39]. This scheme is intended for multi-server LANs where each of a set of possibly many time servers determines its local-time offset relative to each of the other servers in the set using periodic timestamped messages, then determines the local-clock correction using the Fault-Tolerant Average (FTA) algorithm of [24]. The FTA algorithm, which is useful where up to k servers may be faulty, sorts the offsets, discards the k highest and lowest ones and averages the rest. The scheme, as described in [39], is most suitable to