A Brief History of NTP Time: Confessions of an Internet Timekeeper [1,2]

David L. Mills, *Fellow ACM, Senior Member IEEE*[3]

Abstract

This paper traces the origins and evolution of the Network Time Protocol (NTP) over two decades of continuous operation. The technology has been continuously improved from hundreds of milliseconds in the rowdy Internet of the early 1980s to tens of nanoseconds in the Internet of the new century. It includes a blend of history lesson and technology reprise, but with overtones of amateur radio when a new country shows up on the Internet with NTP running.

This narrative is decidedly personal, since the job description for an Internet timekeeper is highly individualized and invites very few applicants. There is no attempt here to present a comprehensive tutorial, only a almanac of personal observations, eclectic minutae and fireside chat. Many souls have contributed to the technology, some of which are individually acknowledged in this paper, the rest too numerous left to write their own memoirs.

Keywords: computer network, time synchronization, technical history, algorithmic memoirs

1. Introduction

An argument can be made that the Network Time Protocol (NTP) is the longest running, continuously operating, distributed application in the Internet. As NTP is approaching its third decade, it is of historic interest to document the origins and evolution of the architecture, protocol and algorithms. Not incidentally, NTP was an active participant in the early development of the Internet technology and its timestamps recorded many milestones in measurement and prototyping programs.

This paper documents significant milestones in the evolution of computer network timekeeping technology over four generations of NTP to the present. The NTP software distributions for Unix, Windows and VMS has been maintained by a corps of almost four dozen volunteers at various times. There are too many to list here, but the major contributors are revealed in the discussion to follow. The current NTP software distribution, documentation and related materials, newsgroups and links are on the web at www.ntp.org. In addition, all papers and reports cited in this paper (except [23]) are in PostScript and PDF at www.eecis.udel.edu/~mills. Further information, including executive summaries, project reports and briefing slide presentations are at www.eecis.udel.edu/~mills/ntp.htm.

There are three main threads interwoven in the following. First is a history lesson on significant milestones for the specifications, implementations and coming-out parties. These milestones calibrate and are calibrated by developments elsewhere in the Internet community. Second is a chronology of the algorithmic refinements leading to better and better accuracy, stability and robustness that continue to the present. These algorithms represent the technical contributions as documented in the references. Third is a discussion of the various proof-of-performance demonstrations and surveys conducted over the years, each attempting to outdo the previous in calibrating the performance of NTP in the Internet of the epoch. Each of these three threads winds through the remainder of this narrative.

2. On the Antiquity of NTP

NTP's roots can be traced back to a demonstration at NCC 79 believed to be the first public coming-out party of the Internet operating over a transatlantic satellite network. However, it was not until 1981 that the synchronization technology was documented in the now historic Internet Engineering Note series as IEN-173 [35]. The first specification of a public protocol developed from it

---

1. Sponsored by: DARPA Information Technology Office Order G409/J175, Contract F30602-98-1-0225, and Digital Equipment Corporation Research Agreement 1417.

2. This document has been submitted for publication. It is provided as a convenience to the technical community and should not be cited or redistributed.

3. Author's address: Electrical and Computer Engineering Department, University of Delaware, Newark, DE 19716, mills@udel.edu, http://www.eecis.udel.edu/~mills.

appeared in RFC-778 [34]. The first deployment of the technology in a local network was as an integral function of the Hello routing protocol documented in RFC-891 [32], which survived for many years in a network prototyping and testbed operating system called the Fuzzball [24].

What later became known as NTPVersion 0 was implemented in 1985, both in Fuzzball by this author and in Unix by Louis Mamakos and Michael Petry at U Maryland. Fragments of their code survive in the software running today. RFC-958 contains the first formal specification of this version [29], but it did little more than document the NTP packet header and offset/delay calculations still used today. Considering the modest speeds of networks and computers of the era, the nominal accuracy that could be achieved on an Ethernet was in the low tens of milliseconds. Even on paths spanning the Atlantic, where the jitter could reach over one second, the accuracy was generally better than 100 ms.

Version 1 of the NTP specification was documented three years later in RFC-1059 [27]. It contained the first comprehensive specification of the protocol and algorithms, including primitive versions of the clock filter, selection and discipline algorithms. The design of these algorithms was guided largely by a series of experiments, documented in RFC-956 [31], in which the basic theory of the clock filter algorithm was developed and refined. This was the first version which defined the use of client/server and symmetric modes and, of course, the first version to make use of the version field in the header.

A transactions paper on NTP Version 1 appeared in 1991 [21]. This was the first paper that exposed the NTP model, including the architecture, protocol and algorithms, to the technical engineering community. While this model is generally applicable today, there have been a continuing series of enhancements and new features introduced over years, some of which are described in following sections.

The NTP Version 2 specification followed as RFC-1119 in 1989 [25]. A completely new implementation slavish to the specification was built by Dennis Fergusson at U Toronto. This was the first RFC in PostScript and as such the single most historically unpopular document in the RFC publishing process. This document was the first to include a formal model and state machine describing the protocol and pseudo-code defining the operations. It introduced the NTP Control Message Protocol for use in managing NTP servers and clients, and the cryptographic authentication scheme based on symmetric-key cryptography, both of which survive to the present day.

There was considerable discussion during 1989 about the newly announced Digital Time Synchronization Service (DTSS) [23], which was adopted for the Enterprise network. The DTSS and NTP communities had much the same goals, but somewhat different strategies for achieving them. One problem with DTSS, as viewed by the NTP community, was a possibly serious loss of accuracy, since the DTSS design did not discipline the clock frequency. The problem with the NTP design, as viewed from the DTSS community, was the lack of formal correctness principles in the design process. A key component in the DTSS design upon which the correctness principles were based was an agreement algorithm invented by Keith Marzullo in his dissertation.

In the finest Internet tradition of stealing good ideas, the Marzullo algorithm was integrated with the existing suite of NTP mitigation algorithms, including the filtering, clustering and combining algorithms, which the DTSS design lacked. However, the Marzullo algorithm in its original form produced excessive jitter and seriously degraded timekeeping quality over typical Internet paths. The algorithm, now called the intersection algorithm, was modified to avoid this problem. The resulting suite of algorithms has survived substantially intact to the present day, although many modifications and improvements have been made over the years.

In 1992 the NTP Version 3 specification appeared [18], again in PostScript and now running some 113 pages. The specification included an appendix describing a formal error analysis and an intricate error budget including all error contributions between the primary reference source over intervening servers to the eventual client. This provided the basis to support maximum error and estimated error statistics, which provide a reliable characterization of timekeeping quality, as well as a reliable metric for selecting the best from among a population of available servers. As in the Version 2 specification, the model was described using a formal state machine and pseudo code. This version also introduced broadcast mode and included reference clock drivers in the state machine.

Lars Mathiesen at U Copenhagen carefully revised the version 2 implementation to comply with the version 3 specification. There was considerable give and take between the specification and implementation and some changes were made in each to reach consensus, so that the implementation was aligned precisely with the specification. This was a major effort which lasted over a year during which the specification and implementation converged to a consistent formal model.

In the years since the version 3 specification, NTP has evolved in various ways adding new features and algorithm revisions while still preserving interoperability with older versions. Somewhere along the line, it became clear that a new version number was needed, since the state machine and pseudo code had evolved somewhat from the version 3 specification, so it became NTP Version 4. The evolution process was begun with a number of white papers, including [14] and [11].

Subsequently, a simplified version 4 protocol model was developed for the Simple Network Protocol (SNTP) version 4 in RFC-2030 [9]. SNTP is compatible with NTP as implemented for the IPv4, IPv6 and OSI protocol stacks, but does not include the crafted mitigation and discipline algorithms. These algorithms are unnecessary for an implementation intended solely as a server. SNTP version 4 has been used in several standalone NTP servers integrated with GPS receivers.

There is a certain sense of the radio amateur in the deployment of NTP around the globe. Certainly, each new country found running NTP was a new notch for the belt. A particularly satisfying conquest was when the national standards laboratory of a new country came up an NTP primary server connected directly to the national time and frequency ensemble. Internet timekeepers Judah Levine at NIST and Richard Schmidt at USNO deployed public NTP primary time servers at several locations in the US and overseas. There was a period where NTP was well lit in the US and Europe but dark elsewhere in South America, Africa and the Pacific Rim. Today, the Sun never sets or even gets close to the horizon on NTP. The most rapidly growing populations are in Eastern Europe and South America, but the real prize is a new one found in Antarctica. Experience in global timekeeping is documented in [8].

One of the real problems in fielding a large, complex software distribution is porting to idiosyncratic hardware and operating systems. There are now over two dozen ports of the distribution for just about every hardware platform running Unix, Windows and VMS marketed over the last twenty years, some of them truly historic in their own terms. Various distributions have run on everything from embedded controllers to supercomputers. Maintaining the configuration scripts and patch library is a truly thankless job and getting good at it may not be a career enhancer. Volunteer Harlan Stenn currently manages this process using modern autoconfigure tools. New versions are tested first in our research net DCnet, then in bigger sandboxes like CAIRN and finally put up for public release at www.ntp.org. The bug stream arrives at bugs@mail.ntp.org.

At this point the history lesson is substantially complete. However, along the way several specific advancements need to be identified. The remaining sections of this paper discuss a number of them in detail.

3. Autonomous Deployment

It became clear as the NTP development continued that the most valuable enhancement would be the capability for a number of clients and servers to automatically configure and deploy in an NTP subnet delivering the best timekeeping quality, while conserving processor and network resources. Not only would this avoid the tedious chore of engineering specific configuration files for each server and client, but it would provide a robust response and reconfiguration scheme should components of the subnet fail. The DTSS model described in [23] goes a long way to achieve this goal, but has serious deficiencies, notably the lack of cryptographic authentication. The following discussion summarizes the progress toward that goal.

Some time around 1985 Project Athena at MIT was developing the Kerberos security model, which provides cryptographic authentication of users and services. Fundamental to the Kerberos design is the ticket used to access computer and network services. Tickets have a designated lifetime and must be securely revoked when their lifetime expires. Thus, all Kerberos facilities had to have secure time synchronization services. While the NTP protocol contains specific provisions to deflect bogus packets and replays, these provisions are inadequate to deflect more sophisticated attacks such as masquerade. To deflect these attacks NTP packets were protected by a cryptographic message digest and private key. This scheme used the Digital Encryption Standard operating in Cipher Block Chaining mode (DES-CBC).

Provision of DES-based source authentication created problems for the public software distribution. Due to the International Trade in Arms Regulations (ITAR) at the time, DES could not be included in NTP distributions exported outside the US and Canada. Initially, the way to deal with this was to provide two versions of DES in the source code, one operating as an empty stub and the other with the algorithm but encrypted with DES and a secret key. The idea was that, if a potential user could provide proof of residence, the key was revealed. Later, this awkward and cumbersome method was replaced simply by maintaining two distributions, one intended for domestic use and the other for export. Recipients were placed on their honor to fetch the politically correct version.

However, there was still the need to authenticate NTP packets in the export version. Louis Mamakos of U Maryland adapted the MD5 message digest algorithm for NTP. This algorithm is specifically designed for the same function as the DES-CBC algorithm, but is free of ITAR restrictions. In NTP Version 4 the export distribution has been discontinued and the DES source code deleted; however, the algorithm interface is compatible with widely available cryptographic libraries, such as rsaref2.0 from RSA Laboratories. If needed, there are numerous sources of the DES source code from foreign archive sites, so it is readily possible to obtain it and install in the standard distribution.

While MD5-based source authentication has worked well, it requires secret keys, which complicates key distribution and, especially for multicast-based modes, is vulnerable to compromise. Public-key cryptography simplifies key distribution, but can severely degrade timekeeping quality. The Internet Engineering Task Force (IETF) has defined several cryptographic algorithms and protocols, but these require persistent state, which is not possible in some NTP modes. Some appreciation of the problems is apparent from the observation that secure timekeeping requires secure cryptographic media, but secure media require reliable lifetime enforcement [4]. The implied circularity applies to any secure time synchronization service, including NTP.

These problems were addressed in NTP Version 4 with a new security model and protocol called Autokey. Autokey uses a combination of public-key cryptography and a pseudo-random keystream [1]. Since public-key cryptography hungers for large chunks of processor resources and can degrade timekeeping quality, the algorithms are used sparingly in an offline mode to sign and verify time values, while the much less expensive keystream is used to authenticate the packets relative to the signed values. Furthermore, Autokey is completely self-configuring, so that servers and clients can be deployed and redeployed in an arbitrary topology and automatically exchange signed values without manual intervention. Further information is available at www.eecid.udel.edu/~mills/autokey.htm.

The flip side of autonomous deployment is how a ragtag bunch of servers and clients randomly deployed in a network substrate can find each other and automatically configure which servers directly exchange time values and which depend on intervening servers. The technology which supports this feature is called Autoconfigure and has evolved as follows.

In the beginning, almost all NTP servers operated in client/server mode, where a client sends requests at intervals ranging from one minute to tens of minutes, depending on accuracy requirements. In this mode time values flow outward from the primary servers through possibly several layers of secondary servers to the clients. In some cases involving multiply redundant servers, peers operate in symmetric mode and values can flow from one peer to the other or vice versa, depending on which one is closest to the primary source according to a defined metric. Some institutions like U Delaware and GTE, for example, operate multiple primary servers, each connected to one or more redundant radio and satellite receivers using different dissemination services. This forms an exceptionally robust synchronization source for both on-campus and off-campus public access.

In NTP Version 3, configuration files had to be constructed manually using information found in the lists of public servers at www.ntp.org, although some sites partially automated the process using crafted DNS records. Where very large numbers of clients are involved, such as in large corporations with hundreds and thousands of personal computers and workstations, the method of choice is broadcast mode, which was added in NTP Version 3, or multicast mode, which was added in NTP Version 4.

However, since clients to not send to servers, there was no way to calibrate and correct for the server-client propagation delay in NTPVersion 3. This is provided in NTP Version 4 by a protocol modification in which the client, once receiving the first broadcast packet, executes a volley of client/server exchanges in order to calibrate the delay and then reverted to listen-only mode. Coincidentally, this initial exchange is used by the Autokey protocol to retrieve the server credentials and verify its authenticity.

Notwithstanding the progress toward a truly autonomous deployment capability described here, there still remains work to be done. The current research project funded by DARPA under the Next Generation Internet program is actively pursuing this goal, as discussed in a following section.

4. Radios, we have Radios

For as many years as NTP has ticked on this planet, the definitive source for public NTP servers has been a set of tables, one for primary servers and the other for secondary servers, maintained at www.ntp.org. Each server in those tables is operated as a public service and maintained by a volunteer staff. Primary (stratum 1) servers have up to several hundred clients and a few operated by NIST and USNO may have several times that number. A

stratum-1 server requires a primary reference source, usually a radio or satellite receiver or modem. Following is a history lesson on the development and deployment of NTP stratum-1 servers.

The first use of radios as a primary reference source was in 1981 when a Spectracom WWVB receiver was connected to a Fuzzball at COMSAT Laboratories in Clarksburg, MD [34]. This machine provided time synchronization for Fuzzball LANs in Washington, London, Oslo and later Munich. These LANs were used in the DARPA Atlantic Satellite program for satellite measurements and protocol development. Later, the LANs were used to watch the national power grids of the US, UK and Norway swish and sway over the heating and cooling seasons [30].

DARPA purchased four of the Spectracom WWVB receivers, which were hooked up to Fuzzballs at MIT Lincoln Laboratories, COMSAT Laboratories, USC Information Sciences Institute, and SRI International. The radios were redeployed in 1986 in the NSF Phase I backbone network, which used Fuzzball routers [26]. It is a tribute to the manufacturer that all four radios are serviceable today; two are in regular operation at U Delaware, a third serves as backup spare and the fourth is in the Boston Computer Museum.

These four radios, together with a Heath WWV receiver at COMSAT Laboratories and a pair of TrueTime GOES satellite receivers at Ford Motor Headquarters and later at Digital Western Research Laboratories, provided primary time synchronization services throughout the ARPANET, MILNET and dozens of college campuses, research institutions and military installations. By 1988 two Precision Standard Time WWV receivers joined the flock, but these along with the Heath WWV receiver are no longer available. From the early 1990s these nine pioneer radio-equipped Internet time servers were joined by an increasing number of volunteer radio-equipped servers now numbered over 100 in the public Internet.

As the cost of GPS receivers plummeted from the stratosphere (the first one this author bought cost $17,000), these receivers started popping up all over the place. In the US and Canada the longwave radio alternative to GPS is WWVB transmitting from Colorado, while in Europe it is DCF77 from Germany. However, shortwave radio WWV from Colorado, WWVH from Hawaii and CHU from Ottawa have been useful sources. While GOES satellite receivers are available, GPS receivers are much less expensive than GOES and provide better accuracy. Over the years some 37 clock driver modules supporting these and virtually every radio, satellite and modem national standard time service in the world have been written for NTP.

Recent additions to the driver library include drivers for the WWV, WWVH and CHU transmissions that work directly from an ordinary shortwave receiver and audio sound card or motherboard codec. Some of the more exotic drivers built in our laboratory include a computerized LORAN-C receiver with exceptional stability [19] and a DSP-based WWV demodulator/decodor using theoretically optimal algorithms [6].

5. Hunting the Nanoseconds

When the Internet first wound up the NTP clockspring, computers and networks were much, much slower than today. A typical WAN speed was 56 kb/s, about the speed of a telephone modem of today. A large timesharing computer of the day was the Digital Equipment TOPS-20, which wasn't a whole lot faster, but did run an awesome version of Zork. This was the heyday of the minicomputer, the most ubiquitous of which was the Digital Equipment PDP11 and its little brother the LSI-11. NTP was born on these machines and grew up with the Fuzzball operating system. There were about two dozen Fuzzballs scattered at Internet hotspots in the US and Europe. They functioned as hosts and gateways for network research and prototyping and so made good development platforms for NTP.

In the early days most computer hardware clocks were driven by the power grid as the primary timing source. Power grid clocks have a resolution of 16 or 20 ms, depending on country, and the uncorrected time can wander several seconds over the day and night, especially in summertime. While power grid clocks have rather dismal performance relative to accurate civil time, they do have an interesting characteristic, at least in areas of the country that are grid-synchronous. Early experiments in time synchronization and network measurement could assume the time offsets between grid-synchronized clocks was constant, since they all ran at the same Hertz, so all NTP had to do was calibrate the constant offsets.

Later, clocks were driven by an oscillator stabilized by a quartz crystal resonator, which is much more stable than the power grid, but has the disadvantage that the intrinsic frequency offset between crystal clocks can reach several hundred parts-per-million (PPM) or several seconds per day. In fact, over the years only Digital has paid particular attention to the manufacturing tolerance of the clock oscillator, so their machines make the best timekeepers. In fact, this is one of the reasons why all

the primary time servers operated by NIST are Digital Alphas.

As crystal clocks came into widespread use, the NTP clock discipline algorithm was modified to adjust the frequency as well as the time. Thus, an intrinsic offset of several hundred PPM could be reduced to a residual in the order of 0.1 PPM and residual timekeeping errors to the order of a clock tick. Later designs decreased the tick from 16 or 20 ms to 4 ms and eventually to 1 ms in the Alpha. The Fuzzballs were equipped with a hardware counter/timer with 1-ms tick, which was considered heroic in those days.

To achieve resolutions better than one tick, some kind of auxiliary counter is required. Early Sun SPARC machines had a 1-MHz counter synchronized to the tick interrupt. In this design, the seconds are numbered by the tick interrupt and the microseconds within the second read directly from the counter. In principle, these machines could keep time to 1 μs, assuming that NTP could discipline the clocks between machines to this order. In point of fact, performance was limited to a few milliseconds, both because of network and operating system jitter and also because of small varying frequency excursions induced by ambient temperature variations.

Analysis, simulation and experiment led to continuing improvements in the NTP clock discipline algorithm, which adjusts the clock time and frequency in response to an external source, such as another NTP server or a local source such as a radio or satellite receiver or telephone modem [16]. As a practical matter, the best timekeeping requires a directly connected radio; however, the interconnection method, usually a serial port, itself has inherent jitter. In addition, the method implemented in the operating system kernel to adjust the time generally has limitations of its own [24].

In a project originally sponsored by Digital, components of the NTP clock discipline algorithm were implemented directly in the kernel. In addition, an otherwise unused counter was harnessed to interpolate the microseconds in much the same manner as in Sun machines. In addition to these improvements, a special clock discipline loop was implemented for the pulse-per-second (PPS) signal produced by some radio clocks and precision oscillators. The complete design and application interface was reported in [13], some sections of which appeared as RFC-1589 [15], produced in the first true microsecond clock that could be disciplined from an external source. Other issues related to precision Internet timekeeping were discussed in the paper [10].

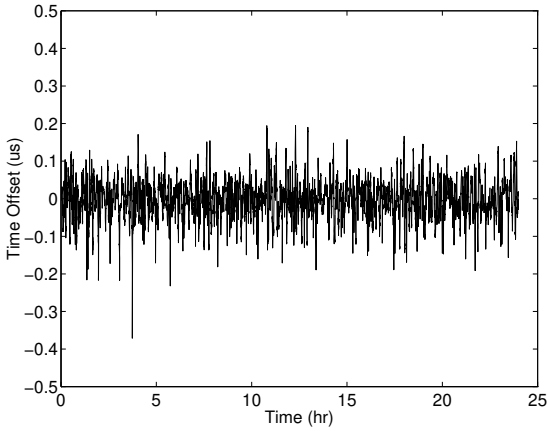

Figure 1. Nanokernel Clock Discipline with PPS Signal on Alpha 433au (from [MIL00b])

An interesting application of this technology was in Norway, where a Fuzzball NTP primary time server was connected to a cesium frequency standard with PPS output. In those days the Internet bridging the US and Europe had notoriously high jitter, in some cases peaks reaching over one second. The cesium standard and kernel discipline maintained constant frequency, but did not provide a way to number the seconds. NTP provided this function via the Internet and other primary servers. The experience with very high jitter resulted in special nonlinear signal processing code, called the popcorn spike suppressor, in the NTP clock discipline algorithm.

Still, network and computer speeds were reaching higher and higher. The time to cycle through the kernel and back, once 40 μs in a Sun SPARC IPC, was decreasing to a microsecond or two in a Digital Alpha. In order to insure a reliable ordering of events, the need was building to improve the clock resolution better than 1 μs and the nanosecond seemed a good target. Where the operating system and hardware justified it, NTP now disciplines the clock in nanoseconds. In addition, the NTP Version 4 implementation switched from integer arithmetic to floating double, which provides much more precise control over the clock discipline process.

For the ultimate accuracy, the original microsecond kernel was overhauled to support a nanosecond clock conforming to the PPS interface specified in RFC-2783 [3]. Nanosecond kernels have been built and tested for SunOS, Alpha, Linux and FreeBSD systems, the latter two of which include the code in current system versions. The results with the new kernel demonstrate that the residual RMS error with modern hardware and a precision PPS signal is in the order of 50 ns [1]. The skeptic should see Figure 1, although admittedly this shows the jitter and not the systematic offset, which must be calibrated out.

This represents the state of the art in current timekeeping practice. Having come this far, the machine used by this author now runs at 1 GHz and can chime with another across the country at 100 Mb/s, which raises the possibility of a picosecond clock. The inherent resolution of the NTP timestamp is about 232 picoseconds, which suggests we soon might approach that limit and require rethinking the NTP protocol design. At these speeds NTP could be used to synchronize the motherboard CPU and ASIC oscillators using optical interconnects.

6. Analysis and Experiment

Over the years a good deal of effort has gone into the analysis of computer clocks and methods to stabilize them in frequency and time. As networks and computers have become faster and faster, the characterization of computer clock oscillators and the evolution of synchronization technology has continuously evolved to match. Following is a technical timeline on the significant events in this progress.

When the ICMP protocol divorced from the first Internet routing protocol GGP, one of the first functions added to ICMP was the ICMP Timestamp message, which is similar to the ICMP Echo message, but carries timestamps with millisecond resolution. Experiments with these messages used Fuzzballs and the very first implementation of ICMP. In fact, the first use of the name PING (Packet InterNet Groper) can be found in RFC-889 [33]. While the hosts and gateways did not at first synchronize clocks, they did record timestamps with a granularity of 16 ms or 1 ms, which could be used to measure roundtrip times and synchronize experiments after the fact. Statistics collected this way were used for the analysis and refinement of early TCP algorithms, especially the parameter estimation schemes used by the retransmission timeout algorithm.

The first comprehensive survey of NTP operating in the Internet was published in 1985 [30]. Later surveys appeared in 1990 [24] and 1997 [8]. The latest survey was a profound undertaking. It attempted to find and expose every NTP server and client in the public Internet using data collected by the standard NTP monitoring tools. After filtering to remove duplicates and falsetickers, the survey found over 185,000 client/server associations in over 38,000 NTP servers and clients. The results reported in [8] actually represented only a fraction of the total number of NTP servers and clients. It is known from other sources that many thousands of NTP servers and clients lurk behind firewalls where the monitoring programs can't find them. Extrapolating from data provided about the estimated population in Norway, it is a fair statement that well over 100,000 NTP daemons are chiming the Internet and more likely several times that number. Recently, a NTP client was found hiding in a standalone print server. The next one may be found in an alarm clock.

The paper [20] is a slightly tongue-in-cheek survey of the timescale, calendar and metrology issues involved in computer network timekeeping. Of particular interest in that paper was how to deal with leap seconds in the UTC timescale. While provisions are available in NTP to disseminate leap seconds throughout the NTP timekeeping community, means to anticipate their scheduled occurrence was not implemented in radio, satellite and modem services until relatively recently and not all radios and only a handful of kernels support them. If fact, on the thirteen occasions since NTP began in the Internet the behavior of the NTP subnet on and shortly after each leap second could only be described in terms of a pinball machine.

The fundamentals of computer network time synchronization technology was presented in the report [17], which remains valid today. That report set forth mathematically precise models for error analysis, transient response and clock discipline principles. Various sections of that report were condensed and refined in the paper [16].

In a series of careful measurements over a period of two years with selected servers in the US, Australia and Europe, an analytical model of the idiosyncratic computer clock oscillator was developed and verified. While a considerable body of work on this subject has acreted in the literature, the object of study has invariably been precision oscillators of the highest quality used as time and frequency standards. Computer oscillators have no such pedigree, since there are generally no provisions to stabilize the ambient environment, in particular the crystal temperature.

The work reported in the paper [12] further extended and refined the model evolved from the [16] paper and its predecessors. It introduced the concept of Allan deviation, a statistic useful for the characterization of oscillator stability. A typical plot on log-log coordinates is shown in Figure 2. The paper also reported on the results of ongoing experiments to estimate this statistic using workstations and the Internet of that era. This work was further extended and quantified in the report [7], portions of which were condensed in the paper [5]. This paper presented the Allan intercept model which characterizes typical computer oscillators. The Allan intercept is the point (x, y) where the straight-line asymptotes for each NTP source shown intersect. This

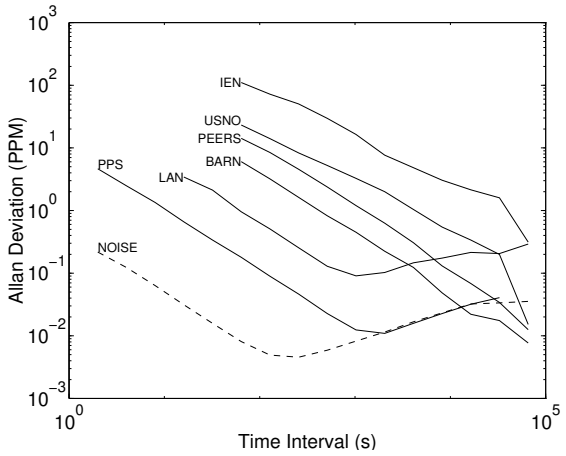

Figure 2. Allan Deviation Plot (from [MIL98])

work resulted in a hybrid algorithm, implemented in NTP Version 4, which both improves performance over typical Internet paths and allows the clock adjustment intervals to be substantially increased without degrading accuracy. A special purpose simulator including substantially all the NTP algorithms was used to verify predicted behavior with both simulated and actual data over the entire envelope of frenetic Internet behaviors.

7. As Time Goes By

At the beginning of the new century it is quite likely that precision timekeeping technology has evolved about as far as it can given the realities of available computer hardware and operating systems. Using specially modified kernels and available interface devices, Poul-Henning Kamp and this author have demonstrated that computer time in a modern workstation can be disciplined within some tens of nanoseconds relative to a precision source such as a cesium or rubidium frequency standard [1]. While not many computer applications would justify such heroic means, the demonstration suggests that the single most useful option for high performance timekeeping in a modern workstation may be a temperature compensated or stabilized oscillator.

In spite of the protocol modification, broadcast mode provides somewhat less accuracy than client/server mode, since it does not track variations due to routing changes or network loads. In addition, it is not easily adapted for autonomous deployment. In NTP Version 4 a new manycast mode was added where a client sends to an IP multicast group address and a server listening on this address responds with a unicast packet, which then mobilizes an association in the client. The client continues operation with the server in ordinary client/server mode. While manycast mode has been implemented and tested in NTP Version 4, further refinements are needed to avoid implosions, such as using an expanding-ring search, and to manage the population found, possibly using crafted scoping mechanisms.

Manycast mode has the potential to allow at least moderate numbers of servers and clients to nucleate about a number of primary servers, but the full potential for autonomous deployment can be realized only using symmetric mode, where the NTP subnet can grow and flex in fully distributed and dynamic ways. In his dissertation Ajit Thyagarajan examines a class of heuristic algorithms that may be useful management candidates. Meanwhile, the quest for new technology continues.

While almost all time dissemination means in the world are based on Coordinated Universal Time (UTC), some users have expressed the need for International Atomic Time (TAI), including means to metricate intervals that span multiple leap seconds. NTP Version 4 includes a primitive mechanism to retrieve a table of historic leap seconds from NIST servers and distribute it throughout the NTP subnet. However, at this writing a suitable API has yet to be designed and implemented, then navigate the IETF standards process. Refinements to the Autokey protocol are needed to insure only a single copy of this table, as well as cryptographic agreement parameters, is in use throughout the NTP subnet and can be refreshed in a timely way.

It is likely that future deployment of public NTP services might well involve an optional timestamping service, perhaps for-fee. This agenda is being pursued in a partnership with NIST and Certified Time, Inc. In fact, several NIST servers are now being equipped with timestamping services. This makes public-key authentication a vital component of such a service, especially if the Sun never sets on the service area.

8. Acknowledgements

Internet timekeeping is considered by many to be a hobby, and even this author has revealed a likeness to amateur radio. There seems no other explanation why the volunteer timekeeper corps has continued so long to improve the software quality, write clock drivers for every new radio that comes along and port the stuff to new hardware and operating systems. The generals in the army have been revealed in the narrative here, but the many soldiers of the trench must be thanked as well, especially when the hardest job is convincing the boss that time tinkering is good for business.

9. References (reverse chronological order)

Note: The following papers and reports, with the exception of [23] are available in PostScript and PDF at www.eecis.udel.edu/~mills.

1. Mills, D.L. The Nanokernel. Software and documentation, including test results, at www.ntp.org.

2. Mills, D.L. Public key cryptography for the Network Time Protocol. Electrical Engineering Report 00-5-1, University of Delaware, May 2000. 23 pp.

3. Mogul, J., D. Mills, J. Brittenson, J. Stone and U. Windl. Pulse-per-second API for Unix-like operating systems, version 1. Request for Comments RFC-2783, Internet Engineering Task Force, March 2000, 31 pp.

4. Mills, D.L. Cryptographic authentication for real-time network protocols. In: *AMS DIMACS Series in Discrete Mathematics and Theoretical Computer Science, Vol. 45* (1999), 135-144.

5. Mills, D.L. Adaptive hybrid clock discipline algorithm for the Network Time Protocol. *IEEE/ACM Trans. Networking 6, 5* (October 1998), 505-514.

6. Mills, D.L. A precision radio clock for WWV transmissions. Electrical Engineering Report 97-8-1, University of Delaware, August 1997, 25 pp.

7. Mills, D.L. Clock discipline algorithms for the Network Time Protocol Version 4. Electrical Engineering Report 97-3-3, University of Delaware, March 1997, 35 pp.

8. Mills, D.L., A. Thyagarajan and B.C. Huffman. Internet timekeeping around the globe. *Proc. Precision Time and Time Interval (PTTI) Applications and Planning Meeting* (Long Beach CA, December 1997), 365-371.

9. Mills, D.L. Simple network time protocol (SNTP) version 4 for IPv4, IPv6 and OSI. Network Working Group Report RFC-2030, University of Delaware, October 1996, 18 pp.

10. Mills, D.L. The network computer as precision timekeeper. *Proc. Precision Time and Time Interval (PTTI) Applications and Planning Meeting* (Reston VA, December 1996), 96-108.

11. Mills, D.L. Proposed authentication enhancements for the Network Time Protocol version 4. Electrical Engineering Report 96-10-3, University of Delaware, October 1996, 36 pp.

12. Mills, D.L. Improved algorithms for synchronizing computer network clocks. *IEEE/ACM Trans. Networks 3, 3* (June 1995), 245-254.

13. Mills, D.L. Unix kernel modifications for precision time synchronization. Electrical Engineering Department Report 94-10-1, University of Delaware, October 1994, 24 pp.

14. Mills, D.L, and A. Thyagarajan. Network time protocol version 4 proposed changes. Electrical Engineering Department Report 94-10-2, University of Delaware, October 1994, 32 pp.

15. Mills, D.L. A kernel model for precision timekeeping. Network Working Group Report RFC-1589, University of Delaware, March 1994. 31 pp.

16. Mills, D.L. Precision synchronization of computer network clocks. *ACM Computer Communication Review 24, 2* (April 1994). 28-43.

17. Mills, D.L. Modelling and analysis of computer network clocks. Electrical Engineering Department Report 92-5-2, University of Delaware, May 1992, 29 pp.

18. Mills, D.L. Network Time Protocol (Version 3) specification, implementation and analysis. Network Working Group Report RFC-1305, University of Delaware, March 1992, 113 pp.

19. Mills, D.L. A computer-controlled LORAN-C receiver for precision timekeeping. Electrical Engineering Department Report 92-3-1, University of Delaware, March 1992, 63 pp.

20. Mills, D.L. On the chronology and metrology of computer network timescales and their application to the Network Time Protocol. *ACM Computer Communications Review 21, 5* (October 1991), 8-17.

21. Mills, D.L. Internet time synchronization: the Network Time Protocol. *IEEE Trans. Communications COM-39, 10* (October 1991), 1482-1493.

22. Mills, D.L. On the accuracy and stability of clocks synchronized by the Network Time Protocol in the Internet system. *ACM Computer Communication Review 20, 1* (January 1990), 65-75.

23. Digital Time Service Functional Specification Version T.1.0.5. Digital Equipment Corporation, 1989.

24. Mills, D.L. Measured performance of the Network Time Protocol in the Internet system. Network

Working Group Report RFC-1128. University of Delaware, October 1989, 18 pp.

25. Mills, D.L. Network Time Protocol (Version 2) specification and implementation. Network Working Group Report RFC-1119, 61 pp. University October 1989, 27 pp.

26. Mills, D.L. The Fuzzball. Proc. *ACM SIGCOMM 88 Symposium* (Palo Alto CA, August 1988), 115-122.

27. Mills, D.L. Network Time Protocol (Version 1) specification and implementation. Network Working Group Report RFC-1059. University of Delaware, July 1988.

28. Mills, D.L., and H.-W. Braun. The NSFNET Backbone Network. *Proc. ACM SIGCOMM 87 Symposium* (Stoweflake VT, August 1987), 191-196.

29. Mills, D.L. Network Time Protocol (NTP). Network Working Group Report RFC-958, M/A-COM Linkabit, September 1985.

30. Mills, D.L. Experiments in network clock synchronization. Network Working Group Report RFC-957, M/A-COM Linkabit, September 1985.

31. Mills, D.L. Algorithms for synchronizing network clocks. Network Working Group Report RFC-956, M/A-COM Linkabit, September 1985.

32. Mills, D.L. DCN local-network protocols. Network Working Group Report RFC-891, M/A-COM Linkabit, December 1983.

33. Mills, D.L. Internet delay experiments. Network Working Group Report RFC-889, M/A-COM Linkabit, December 1983.

34. Mills, D.L. DCNET internet clock service. Network Working Group Report RFC-778, COMSAT Laboratories, April 1981.

35. Mills, D.L. Time synchronization in DCNET hosts. Internet Project Report IEN-173, COMSAT Laboratories, February 1981.