Survivable, Real Time Network Services

Defense Advanced Research Projects Agency
Contract F30602-98-1-0225, DARPA Order G409

Quarterly Progress Report
1 January 1999 - 31 March 1999

David L. Mills
Electrical Engineering Department
University of Delaware

1. Introduction

This report covers the work done in support of the DARPA Information Technology Office program in computer networking. Contributors to this effort include Prof. David L. Mills and graduate students Qoing Li and Robert Redwinski. The project continues previous research in network time synchronization technology jointly funded by DARPA and US Navy. The technology makes use of the Network Time Protocol (NTP), widely used in the Internet, together with engineered modifications designed to improve accuracy in high speed networks. Specific applications benefiting from this research include multicast topologies, multimedia, real-time conferencing, cryptographic systems, and management of distributed, real-time systems.

This quarterly report is submitted in traditional report form on paper. As the transition to web-based information dissemination of research results continues, almost all status information and progress reporting is now on the web, either on pages belonging to the principal investigator or to his students. Accordingly, this and future progress reports will contain primarily schedule and milestone data; current status and research results are reported on web pages at www.eecis.udel.edu/~mills in the form of papers, technical reports and specific briefings.

2. NTP Version 4

Work continues on the Network Time Protocol Version 4. The principal areas of activity include the run-time and compile-time autoconfigure scheme, clock discipline algorithm and nanokernel project.

2.1 Autokey

A paper describing the autokey algorithms as implemented in NTP Version 4 has been published in a DIMACS journal [1].

2.2 Run-time Autoconfigure

The manycast/anycast scheme proposed in previous report [11] has been implemented in part. The schemes work well when both the client and server are homed to only a single IP address. Extension to multiple addresses, which are common in many environments, including ours, remains to be completed.

Ajit Thyagarajan has completed his dissertation and is preparing for his dissertation defense. His work on autoconfiguration involves the study of algorithms to form optimal and quasi-optimal spanning trees subject to degree and total metric constraints in dynamic network topologies. He has developed heuristic algorithms that can generate trees with defined upper bounds when compared to an optimal algorithm. He has verified the correctness of the algorithms and these bounds using a simulation approach. Additional details will be available when the dissertation is published.

2.3 Clock Discipline Algorithm

In the initial testing done in our laboratory, the NTP Version 4 clock discipline performed well as predicted in simulation [2]. However, according to several reports received, there are some scenarios not foreseen in the testing program and which lead to some suboptimal results. Some observers noticed a subtle instability in the form of a 1-ms oscillation with period ranging from minutes to hours. Another observer reported wild swings in frequency and unexpected incidence of step corrections. A couple of problems were due to minor bugs which were quickly corrected. However, the suspicions remained that there might be some subtle interaction overlooked in the design.

After some analysis and experiment in real world meltdown conditions, at least one of the causes for instability was the computations which established the weighting functions to use for the phase-lock loop (PLL) and frequency-lock loop (FLL) contributions to the frequency adjustment term. The weighting functions were computed from past prediction errors, which would seem to be a rational approach to the problem. This worked well in simulation and testing in practice. However, in chaotic conditions when servers are wandering in and out of the correctness interval, clock hopping from one source to another made the predictions ineffective.

From this evidence, the original design based on predictive algorithms was abandoned in favor of one based solely on the interval between updates and the measured RMS error of the discipline loop. From prior experience, it is known that the PLL works better at relatively short intervals, since the FLL can be easily spooked by large delay jitter, and that FLL works better at relatively long intervals, since the frequency gain of the PLL is too small. What is needed then was an algorithm that can automatically weight PLL contributions more heavily at short intervals and FLL contributions more heavily at long intervals. Also from analysis and anecdotal experience, the averaging time for FLL contributions should track the Allan variance as the network jitter increases.

The algorithm that emerged from the above observations is quite simple and easy to implement. The first assumption is that there is no need for a weight calculation for the PLL, since the frequency gain varies as the inverse square. By the time the update interval has increased to 1000 s or more, the PLL frequency contributions are negligible. The second assumption is that the Allan intercept depends only on the measured RMS error in a linear way. Therefore, the Allan intercept can be determined simply as a constant times the smoothed RMS error. The third assumption is that the FLL frequency gain can be determined directly from the update interval starting with zero at 256 s and rising linearly to one at 2048 s. This characteristic approximates the empirical characteristic found in simulations. The fourth and final assumption is that the FLL averaging time is linearly dependent on the update interval, but clamped not to decrease below an arbitrary minimum of 2048 s.

The implementation of the above algorithm was simple, more compact and more straightforward compared to the previous design. It has been exhaustedly tested under some very chaotic situations involving multiple servers, very large network delay jitter and some unusual network failure scenarios. Perhaps the most extreme test was with the Italian national time server time.ien.it, where the jitter reaches well over one second on occasion. The original algorithm became unstable and was unable to phase-lock on this source. The re-engineered algorithm not only locked on the source, but managed to increase the update interval to further stabilize the clock frequency.

In another test a machine was configured for the NIST primary server time.nist.gov and with a maximum poll interval of 17, which corresponds to 36.4 hours. The network path between Newark, DE, and Boulder, CO, can be quite treacherous at times, resulting in errors up to several tens of milliseconds or more. With the new algorithm operating in burst mode, where each update is computed from eight groomed roundtrip volleys, the error is rarely above ten milliseconds.

2.4 NTP Clock State Machine

The clock state machine is designed to handle extreme scenarios where the intrinsic frequency error is very large and where sudden large discontinuities occur due to reboot and drastic changes in ambient temperature. One problem occurs when the automatic power control (API) switches to a power conserving mode. While the CPU oscillator may continue to run, the ambient temperature can change radically. One report mentioned a particular motherboard where a Pentium CPU was located very near the quartz crystal that controls the system clock frequency. Apparently, the CPU heat sink is inadequate and the CPU temperature depends strongly on the instruction mix. The result is that a burst of floating point instructions creates a large blip on the oscillator frequency. While one application of this phenomenon might be to use NTP to monitor the instruction mix of a particular application, this would in most cases not be productive.

The clock state machine has been redesigned to be more robust in the face of scenarios like the above. While a detailed description is beyond the scope of this report, some idea of its operation will be evident from a description of its six states.

1. The clock frequency offset is unknown (the ntp.drift file has not been created and the intrinsic offset never recorded).

2. The clock frequency has been initialized from the ntp.drift file, but the time has not yet been set.

3. The clock time has been set, but no further update is yet available to estimate the frequency offset.

4. The frequency and/or time offsets are too large for the hybrid PLL/FLL to handle. Operation switches to a special mode designed to quickly compensate for the large errors.

5. The frequency and time offsets have converged to stable values and the poll interval allowed to vary as a function of the RMS error.

6. A large time spike has been detected and ignored. If the next update also shows a large offset, it will be believed and the clock will be stepped; if not, the discipline continues as usual.

The special frequency mode is used in two scenarios. In state 1 it is assumed that nothing is known about the intrinsic clock frequency offset, so the state machine is initialized to quickly learn the value. In state 3 the clock has been set; however, if the first offset following that is too large (over 128 ms), the frequency offset is beyond the capture range of the PLL and the coarse offset must be determined first.

A further sanity check implemented in the clock filter algorithm rejects outlyers more than a constant factor times the RMS error. This algorithm, called a popcorn spike suppressor, is similar to algorithms used in digital radios and called a noise blanker. Additional data grooming algorithms are used to clamp the values of some variables to prevent an unstable or runaway condition.

3. The Nanokernel Project

A project was initiated in 1993 to improve the timekeeping quality of typical workstations of that era. The goals of this project are summarized as follows:

1. Improve the time accuracy to the order of microseconds at the application interface. Previous accuracy expectations were in the order of milliseconds.

2. Provide frequency steering should the NTP daemon cease operation for one reason or another.

3. In some systems, such as those based on the Digital RISC and Alpha architectures, time can be resolved only to the tick interval, which in most workstations ranges from about 1 ms to 10 ms. In those systems which have an internal CPU cycle counter with resolution equal to or less than a microsecond, provide a means to interpolate between tick interrupts to yield a time resolution of one microsecond.

4. Provide an interface and algorithms supporting an external source of precision time, such as a pulse-per-second (PPS) signal from a GPS receiver or cesium clock.

5. Support a peripheral device, such as a precision oscillator, as the master clock source, in order to avoid the instability of the typical workstation clock.

6. In cases where a protocol other than NTP, such as the Digital Time Synchronization Service (DTSS), directly controls the system clock, provide means to synchronize other systems indirectly using NTP.

The project resulted in a package of Unix kernel modifications that have since been integrated in the kernel binaries shipped with Digital Unix, Ultrix, Solaris, FreeBSD and Linux. The package has been implemented in experimental kernels for SunOS and HP-UX kernels, but not included in the products as shipped. The implementation model, as well as the kernel and applications interfaces, are described in RFC-1589. Development versions support all the above features, but those versions currently shipped by vendors support only the first three.

Since 1993 typical workstation capabilities have dramatically improved; for instance, the time to read the system clock in a vintage Sun IPC is 42 µs, while this takes only 2 µs on an UltraSPARC 30 in 1999. Clearly, it should be possible to improve timekeeping accuracy by a factor of 1000, in effect, replacing the original microsecond clock by a nanosecond clock. However, we have found this is not as easy as it may at first appear. The goals of this project are summarized as follows:

1. All internal time and frequency variables must be represented as 64-bit fixed-point variables with resolution equal or better than one nanosecond in time and one nanosecond/second in frequency. Arithmetic and logical operations must not degrade this degree of resolution nor introduce statistical biases.

2. The implementation must operate transparently in either 32-bit or 64-bit architectures. This requires double-precision arithmetic in 32-bit architectures. All operations must be in fixed point arithmetic; floating point arithmetic is not available for kernel routines.

3. The application interface must accept and provide time values in seconds and nanoseconds (timespec format) or in seconds and microseconds (timeval format) for legacy purposes. Nanosecond time values must be rounded toward zero when converted to microsecond time values.

4. The implementation must operate over a wide range of at least 500 ms in time and 500 PPM in frequency.

5. The application interface must be backwards compatible with the previous interface specification.

6. The implementation must be largely portable, self contained and intrude only minimally on other kernel functions. The routines themselves should require only minor changes in calling and return linkages to function in various kernel architectures.

7. The PPS interface must be compatible with the application program interface proposed by the IETF working group [16].

8. It must be possible to change the timer interrupt increment (tick) while the system is running and without significant disruptions of other applications running on the same machine.

A package meeting the above goals has been implemented, tested and made available for developers, including the FreeBSD developers group. It has been integrated and tested in experimental kernels, including Digital Unix, SunOS and FreeBSD. It is available as the compressed tar archive called nanokernel.tar.Z via FTP and the web. Following is a brief overview of the nanokernel model and algorithms. A complete description will be in a report now in progress.

The nanokernel consists of a suite of algorithms that adjust or discipline the system clock as a function of time offset updates introduced by a synchronization protocol such as NTP. It is described as an adaptive parameter, hybrid phase/frequency-lock loop. A detailed description and analysis of its design and implementation is contained in a report now in progress.

As implemented in the kernel, the hybrid loop, or clock discipline, operates in one of two modes, phase-lock loop (PLL) or frequency-lock loop (FLL) mode. Mode selection is determined by the interval between updates and by a status bit that can be set by a system call. If the interval since the last update is less than 256 s, the loop operates in PLL mode; if greater than 2048 s, it operates in FLL mode. Between 256 s and 2048 s, the loop operates in the mode selected by the calling program. At present, the NTP daemon selects PLL mode for this range.

The implementation provides two system calls to read the system clock and to adjust its parameters. The ntp_gettime() call returns the time of day in either seconds and microseconds (timeval structure) or seconds and nanoseconds (timespec structure) and, in addition, certain statistical quantities useful to determine the accuracy and health of the timekeeping system. The ntp_adjtime() call is used to adjust the time and/or frequency of the clock, as well as set certain state variables which affect the operation of the discipline.

In order to achieve time accuracies in the order of nanoseconds, it is necessary to provide a precision means of outside synchronization. This can be done either with an external time source and interface to the system bus or with a pulse-per-second (PPS) signal and interface via a serial port modem control lead. A PPS source is the most common and readily available means, since PPS signal are generated by most radio and satellite timing receivers. The nanokernel implementation supports both an external clock and modem control lead.

In order to support the PPS signal, the nanokernel includes a special set of algorithms which groom the signal to remove noise components and to verify correct frequency and stability tolerances. In principle, if a PPS signal from a calibrated cesium clock is available, it is possible to set the computer time by some means, including manual, to the correct second, enable the PPS discipline and never have to rely on an external synchronization source. This may be useful for shipboard environments where low probability of intercept (LPI) and high jamming resistance is required.

4. Network Simulator

The network simulator described in previous reports has been largely completed. A description of its design, implementation and initial tests is in Robert Redwinski's Masters Thesis, which is now nearing completion. The protocols simulated include Bellman-Ford and DVMRP in networks with up to 3,500 nodes. Some interesting results were found, including a surprising observation about DVMRP operating over a Bellman-Ford substrate. Sometimes during a simulated outage as the spanning tree was under repair, DVMRP converged to a working, but sometimes suboptimal, multicast spanning tree. This observation needs to be explored further. It could be that some variant of a multicast routing algorithm might be an appropriate defense against an attack on the underlying unicast substrate.

5. Infrastructure

It is the announced CAIRN plan to upgrade the routers to FreeBSD 3.1 in the near future. At this time the FreeBSD developers have integrated the PPS API and nanokernel code in FreeBSD 4.0, which is a new version not scheduled for early release. The developers have agreed to provide sources and advice on how to integrate this code in the 3.1 version and claim this is essentially trivial. The three GPS receivers now deployed in CAIRN have PPS outputs which can be directly connected to a parallel port. From the current experience with the kernel modifications described above and now running on a 433-MHz Alpha, the accuracy expectations for the CAIRN primary servers should be within a microsecond or two.

5.1 FreeBSD Port

Poul-Henning Kamp, a self-employed consultant living in Denmark and a member of the FreeBSD developers group, has taken on the issue of NTP and the nanokernel in FreeBSD. He developed an API for FreeBSD that provides pulse-per-second (PPS) connections via both a serial port modem control lead, which is the method used in most systems, and a parallel port. The parallel port connection avoids the messy little construction project using level converters and pulse regenerators required for the serial port.

Mr. Kamp has a well equipped laboratory, including cesium and rubidium oscillators, oven-controlled quartz oscillators (OCXO) and various laboratory test equipment. A few years ago two LORAN-C receivers were constructed in our laboratory to function as a precision frequency source. One of those receivers is still in operation here; the other was sent to Mr. Kamp to see if he could duplicate the performance observed here with NTP Version 4 and the nanokernel code.

In order to evaluate the performance of various combinations of PPS interface hardware with the NTP daemon and nanokernel code, Mr. Kamp replaced a Pentium motherboard clock oscillator with an OCXO and built a FPGA counter-buffer which can generate timestamps with a resolution of 50 ns. The FPGA device is quite similar to the SBus peripheral built in our laboratory several years ago and used in the Highball Project. He performed a number of experiments to verify the performance of the Pentium with a PPS signal using GPS, LORAN-C and another OCXO. His results confirm ours, that timekeeping performance can be realized at least to precision of the PPS source itself and ultimately to one nanosecond.

5.2 New Domain ntp.org

In order to regularize archive and distribution functions for the NTP community, we have registered the domain name ntp.org with the InterNIC and installed a SPARC 1 to serve as a web server and home directories for the volunteer code developers and testers. This development environment, which was borrowed from the FreeBSD developers community, allows the security policies of the Department and Internet research activities to be isolated from the policies more appropriate to the volunteer corps.

5.3 Miscellany

We have so far been unable to bring up the full DARPA teleconferencing applications suite in our three UltraSPARC machines which use PCI video support. For some reason not yet explained, the stock vic video application does not work with the PCI cameras and the xil library. We have now found a version of vic that works in this configuration. All of our personal workstations now have a complete working suite of teleconferencing applications.

6. Plans for the Next Quarter

Our plans for the next quarter include continued testing and refinement of the NTP Version 4 protocol model, specification and implementation. Specifically, we plan to resolve the problems with the Unix socket interface mentioned in the previous report, so that the NTP autoconfigure feature is really useful. In addition, we plan to continue the collaboration with Coastek InfoSystems in the design and implementation of the cryptographic certification algorithm. The daemon is to be tested first in the research net, then the DARTnet/CAIRN community. As the extensions are backwards compatible, the new features can be activated and tested in regular operation without impacting current users.

7. Publications

All publications, including journal articles, symposium papers, technical reports and memoranda are now on the web at www.eecis.udel.edu/~mills. Links to the several publication lists are available on that page, as well as links to all project descriptions, status reports and briefings. All publications are available in PostScript and PDF formats. Briefings are available in HTML, PostScript, PDF and Proponent. The project descriptions are cross-indexed so that the various interrelationships are clearly evident. Links to other related projects at Delaware and elsewhere are also included on the various pages. Hopefully, the organization of these pages, which amount to a total of about 300 megabytes of information pages and reference documents, will allow quick access to the latest results and project status in a timely way.

Following is a retrospective list of papers and reports supported wholly or in part on this project and the immediately preceeding project "Scalable, High Speed, Internet Time Synchronization," DARPA Order D012. The complete text of all papers and reports, as well as project briefings, status reports and supporting materials is at www.eecis.udel.edu/~mills.

7.1 Papers

1. 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.

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

3. Li, Qiong, and D.L. Mills. On the long-range dependence of packet round-trip delays in Internet. *Proc. IEEE International Conference on Communications* (Atlanta GA, June 1998), 1185-1191.

4. 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.

5. Mills, D.L. Authentication scheme for distributed, ubiquitous, real-time protocols. *Proc. Advanced Telecommunications/Information Distribution Research Program (ATIRP) Conference* (College Park MD, January 1997), 293-298.

6. 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.

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

7.2 Technical Reports

8. Sethi, A.S., H. Gao, and D.L. Mills. Management of the Network Time Protocol (NTP) with SNMP. Computer and Information Sciences Report 98-09, University of Delaware, November 1997, 32 pp.

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

10. 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.

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. 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.

13. Mills, D.L. Simple Network Time Protocol (SNTP). Network Working Group Report RFC-1769, University of Delaware, March 1995, 14 pp.

14. Mills, D.L. Simple Network Time Protocol (SNTP). Network Working Group Report RFC-1769, University of Delaware, March 1995, 14 pp.

7.3 Internet Drafts

15. Mills, D. L., T.S. Glassey and M.E. McNeill. Authentication Scheme Extensions to NTP. Internet Draft draft-mills-ntp-auth-coexist-01.txt, IETF, September, 1998.

16. Mogul, J., D.L. Mills, J. Brittenson, J. Stone, P.-H. Kamp and U. Windl. Pulse-per-Second API for UNIX-like Operating Systems, Version 1.0. Internet Draft draft-mogul-pps-api-02.txt, IETF, June 1998, 25 pp.