Scalable, High Speed, Internet Time Synchronization

Defense Advanced Research Projects Agency
Contract DABT 63-95-C-0046

FInal Report
12 April 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, graduate students Qoing Li and Robert Redwinski, and undergraduate student Douglas Miller. The project continues previous research in network time synchronization technology jointly funded by DARPA, US Navy and US Army. 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.

Recent quarterly reports have been 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. The following sections of this report are summarized from the project status pages on the web at www.eecis.udel.edu/~mills/status.htm These describe the importance of the problem, approach and current status. Where available, links to further information and briefing slides will be cited. A bibliography of all papers, reports and internet drafts appears at the end of this report. In the case of papers, the paper abstract is included.

2. Autonomous Configuration and Authentication in Very Large Networks

The Network Time Protocol (NTP) is widely used in the Internet to synchronize computer clocks. It is supported by over 230 primary time sources located in several countries, together with a hierarchical network of over 100,000 servers and clients scattered all over the globe. Management and configuration of this network has become almost unworkable. As the network is still growing rapidly, a means is required to automatically configure and authenticate the hierarchy in response to changing topology and available server and network resources.

There are two broad areas of effort on this project, each supporting the other, but funded as a separate DARPA contract. Each has its own project description, briefing slides and status reports. The Autonomous Configuration *autoconfigure* effort explores how a broadly distributed service such as the Network Time Protocol (NTP) can be deployed automatically in a very large network without manual configuration engineering. The Autonomous Authentication *autokey* effort explores how a suitably configured NTP subnet can be cryptographically authenticated relative to

published public values. The technology developed should be applicable to many other protocols and services to be deployed in a very large, ubiquitous network.

A separate but related effort involves the simulation of very large networks. The motivation for this project is past evidence of obscure, unexplained routing instability or *churn* observed over several years in the developing Internet. The goal in this effort is to construct experiments in the order of 10,000 nodes operating with embedded routing algorithms for unicast and multicast paradigms. We intend to develop and test the autokey and autoconfigure technology in this simulator.

2.1 Autonomous Authentication

A reliable distributed network service requires provisions to prevent accidental or malicious attacks on the servers and clients in the network. Reliability requires that clients can determine that received messages are authentic; that is, were actually sent by the intended server and not manufactured or modified by an intruder. Ubiquity requires that any client can verify the authenticity of any server using only public information. This is especially important in such ubiquitous network services such as directory services, cryptographic key management and time synchronization.

2.1.1 Brief Description of Work and Results

The Network Time Protocol (NTP) contains provisions to cryptographically authenticate individual servers as described in the protocol specification; however, the existing protocol model does not provide a scheme for the distribution of cryptographic keys, nor does it provide for the retrieval of cryptographic certificates that reliably bind the intended server identification data with the associated keys and related public values.

However, using conventional key agreement and digital signatures with time-critical applications and large client populations such as NTP can cause significant performance degradations. In addition, there are problems unique to NTP in the interaction between the authentication and synchronization functions, since reliable key distribution requires reliable lifetime control and good timekeeping, while secure timekeeping requires reliable key distribution.

Our current work is designed to produce a cryptographically sound and efficient methodology for use in NTP and similar distributed protocols. A detailed assessment of existing schemes proposed by various IETF task forces, specifically the IPSEC community, has been completed. Preliminary evaluation of SKIP, Photuris and ISAKMP schemes suggest that a scheme which is cryptographically sound, efficient and interoperable with existing and proposed directory and certificate services is a challenging task. The primary difficulty with this approach is the requirement for per-association state variables, such as shared Diffie-Hellman keys. This contradicts the principles of the remote procedure call (RPC) paradigm in which servers keep no state for their clients.

A literature review of cryptographic techniques and related mathematics has been completed. As a direct result of this research, a syllabus has been constructed for an advanced graduate level course in cryptographic theory and practice. This course has been offered on a regular basis for electrical engineering and computer science students. A review of the RSA Public-Key Infrastructure (PKI) model and algorithms has been completed. From experience with the algorithms implemented in the rsaref20 package distributed by RSA, we conclude that any scheme requiring every

NTP message to be encrypted with a RSA key would result in unacceptable time synchronization performance.

A report detailing analysis of the NTP security model and authentication scheme has been completed. The report concludes that, while there may be a place for IPSEC-style authentication, as well as PKI-style authentication, neither of these schemes alone satisfy the needs of the NTP service community. A revised security model and authentication scheme is proposed in this report. Called *autokey*, it is based on a combination of PKI and a scheme involving a pseudo-random sequence, where the generating function uses the MD5 algorithm. A prototype implementation of this scheme has been completed for the NTP Version 4 distribution and is now in test.

2.1.2 Future Plans

The analysis and design of the autokey scheme has evolved to a stable state; however, There are two remaining implementation issues. One has to do with the Unix sockets semantics used for multicast. The problem is how to set the source address when more than one interface is present. Since the autokey scheme hashes the IP addresses, as well as the NTP header, it is necessary that the correct address be known before the hash is completed. The present socket interface does not permit this. Workarounds are available and are being implemented.

The other issue is implementation of the PKI functions which bind the plaintext hash to the encrypted hash. The NTP algorithms have been modified to use the rsaref20 interface, so the various algorithms implemented in that package can be swapped in and out as required. The cryptographic routines originally implemented for NTP have been abandoned. These functions require directory services as provided by Secure DNS. Integration with these services will happen as the services mature.

2.2 Autonomous Configuration

The Network Time Protocol (NTP) is widely used in the Internet to synchronize computer clocks. It is supported by over 230 primary time sources located in several countries, together with a hierarchical network of over 100,000 servers and clients scattered all over the globe. Management and configuration of this network have become almost unworkable. As the network is still growing rapidly, a means is required to automatically configure the hierarchy in response to changing-topology and available server and network resources.

2.2.1 Brief Description of Work and Results

The approach to this work involves distributed algorithms that collect information on the current timekeeping network topology using a combination of multicast and anycast messaging, directory services and network management resources. Existing and proposed service location protocols are expected to play a part as well. Distributed algorithms are used to process these data and construct a forest of spanning trees rooted on the primary servers, themselves synchronized to radio clocks or modem services. These algorithms attempt to minimize a metric corresponding to the most accurate time, subject to constraints designed to protect the server and network resources.

Present progress in previous projects includes refinements to client and server software to distribute time information via multicast and anycast messaging. An authentication scheme autokey has

been developed to insure certifiable traceability to national standard time sources. A preliminary version of the new multicast/anycast and authentication schemes has been implemented in NTP Version 4 and is now in test and refinement.

We have developed a candidate heuristic algorithm which optimizes the NTP synchronization hierarchy subject to resource constraints. To support this work, our graduate students have designed and implemented enhancements to current Internet multicasting support and distributed these enhancements to the Internet research community. These developments have been reported at a recent IETF meeting.

To support the new algorithm, changes to the current NTP specification and implementation are required. Changes to several algorithms used to associate clients and servers have been designed and implemented. The new anycast mode, similar to that used to discover nearby servers in other similar protocols, has been incorporated in the design. A new distributed mode of operation, with objectives similar to the matrix time synchronization method described in the literature, has been incorporated in the design.

2.2.2 Future Plans

Development and refinement of the heuristic algorithms will continue. These algorithms will be implemented in NTP Version 4 and tested using the NTP simulator ntpsim, then in our local research network DCnet, then in the CAIRN research network and finally released to the general user population.

2.3 Simulation of Very Large Networks

Routing instability has always been the most destructive hazard to high survival internetwork services. If a critical network link carrying routing information fails, large portions of the network may be isolated and the entire network may become unstable. Lessons of the early ARPAnet years demonstrated that the design of the routing algorithm itself could contribute to unexpected congestive failures. While modern routing algorithm designs are highly resistive to such failures, there remain significant gaps in understanding the survivability issues when the number of network elements spanned by the routing system becomes very large.

There are a number of available simulators designed for network simulations with relatively small numbers of nodes, from tens to possibly hundreds of nodes. These simulators tend to focus on the detailed interactions between one node or protocol and another, with the remaining nodes acting as routers or generating noise. With very large networks, the interest is on macro behavior and global phenomena and closely watched local behavior is less important than global stability.

2.3.1 Brief Description of Work and Results

In order to explore the behavior of routing functions in very large networks, we have developed a discrete event simulator suitable for network simulations including many thousands of nodes. There are three components which together make up the simulator system. The first is a random topology generator which generates network graphs using a probabilistic algorithm modeled on principles developed by Waxman. Topology generation is an offline process and concluded with a data file that is input to the simulator itself.

The simulator itself uses a conventional deterministic, discrete event model, but with a very large virtual memory to hold the simulation entities and event queue. The simulator is equipped to generate random failures of one or more links or nodes according to a designated probability model. One of our workstations has been expanded to 640 megabytes of RAM, which at the present stage development supports a network of 3,500 nodes.

The third component of the system is a suite of routing algorithm, including the venerable Bellman-Ford node-state algorithm and the Distance Vector Multicast Routing (DVMRP) Algorithm. The simulator implements these algorithms in the same way as on an actual network, including the effects caused by routing tables changing while the routing updates themselves are travelling between nodes, as well as random failures of the nodes and links and routing packets.

2.3.2 Future Plans

The goal of the project is to provide useful data in networks of the order of 10,000 nodes and perhaps three times this many links. We plan to explore the behavior in response to various kinds of failure models and failure/recovery rates and determine the character of these failures and how they might be predicted and avoided. We plan to explore other routing algorithms as well, in order to conduct comparative studies of the strengths and weaknesses of the selected algorithms within our experimental framework. Finally, we expect to test the algorithms developed for the Autonomous Configuration project using the simulator.

3. Network Time Protocol Project

The Network Time Protocol (NTP) is widely used in the Internet to synchronize computer clocks to national standard time. The NTP architecture, protocol and algorithms have evolved over almost two decades to the NTP Version 3 specification and implementations for Unix, VMS and Windows, as well as the NTP Version 4 implementation now in development. The architecture and security models provide for operation in point-to-point (unicast) and point-to-multipoint (multicast) modes, and include provisions for cryptographic authentication. These features are described in the Autonomous Configuration and Authentication in Very Large Networks page.

Previous funded research has resulted in a continuous series of improvements in accuracy and reliability of the protocol and supporting algorithms. Used in the Internet of today with computers ranging from personal workstations to supercomputers, NTP provides accuracies generally in the range of a millisecond or two in LANs and up to a few tens of milliseconds in global WANs. When kernel support for precision timing signals, such as pulse-per-second (PPS) signal, is available the accuracy can be improved ultimately to the order of one nanosecond in time and one nanosecond per second in frequency.

The current research effort represents a significant enhancement to the existing protocol, architecture and algorithms of NTP Version 3 and an evolutionary step to the new NTP Version 4. Specifically, these involve provisions for an autonomous configuration capability and a revised security model and authentication scheme based on public key cryptography. Both of these enhancements are necessary in order for large, diversified synchronization subnets to survive electronic warfare attacks on the network routing functions or source selection and clock discipline algorithms used by the NTP server and client population. The design of robust protocols and algorithms which

survive such attacks presents a significant challenge, especially in networks with well over 100,000 servers, such as the existing Internet.

3.1 Brief Description of Work and Results

In order to provide specific accuracy and reliability requirements, NTP Version 3 requires configuration engineering specific to each time server and client site. However, in a tactical network subject to damage and repair, as well as a widely deployed real-time simulation network such as the Defense Simulation Internet (DSI), manual configuration engineering is not acceptable. Our research effort is designed to develop an autonomous configuration capability for NTP Version 4 using multicast methods to achieve diversity and redundancy, as well as directory services and service location protocols as available.

Our approach uses a suite of distributed algorithms to provide a completely automatic, dynamic server discovery and configuration capability as a generic feature of the NTP architecture and protocol. The *autoconfigure* algorithm automatically discovers available servers and organizes the synchronization subnets in response to server and network outages or attacks on the security infrastructure. The distributed algorithm, which is the subject of a pending dissertation, operates on a database constructed by the enhanced NTP protocol and selects the best subnet topology, subject to specified accuracy and reliability constraints. A related algorithm, now under development, calculates clock offsets between each pair of servers in the local neighborhood, then distributes the data to all other servers. Other algorithms already implemented filter and combine the data from all subnet members in the local neighborhood to provide the best accuracy and reliability. None of these algorithms require advance information of any kind, other than that collected in real time by the enhanced NTP protocol.

A robust security model has long been an intrinsic feature in the current and previous NTP versions. However, this model does not scale well to very large networks which may fragment and reform frequently due to attack and repair. The current model, which is based on private key cryptography with predistributed keys, does not work well in multicasting modes and imposes an excessive burden on the key management and distribution system in cases where keys can be compromised and countermeasures are required. These problems are exacerbated by the need to coordinate key management and time synchronization, since each of these services depends on the other.

Our approach involves the use of public key cryptography and crafted algorithms which provide reliable key distribution and management, while avoiding excessive processing and memory resources. The algorithms use shared keys for mutually redundant symmetric server modes (creche servers), dynamically computed keys for traditional client/server modes, and backwards computable hash functions for multicast modes. When necessary, private data are exchanged using RSA encryption with certificates, but this is done infrequently in a manner that does not degrade synchronization quality. We expect to make use of secure RPC services and secure DNS services as they become available.

3.2 Future Plans

We expect to combine the algorithmic mechanisms for autonomous configuration with traditional means involving multicast and directory services, as well as service location protocols now under

development by IETF task forces. We expect also to incorporate the ongoing work of the IETF IPSEC community as appropriate to the specific NTP protocol and security models. The combined mechanisms are to be implemented as extensions to the new NTP Version 4 protocol and implementation for Unix, VMS and Windows and made available to the research community at large. We expect to deploy the new implementation in the CAIRN/DARTnet research community for distributed testing with other CAIRN/DARTnet applications, such as multimedia conferencing. Finally, we expect to develop and publish a definitive protocol specification and vulnerability analysis.

4. Precision Time Synchronization

As generally available computers and networks are becoming increasing fast, accuracy expectations are extending from milliseconds to nanoseconds. The performance of existing computer hardware over extended periods where outside synchronization sources are lost is in some cases unacceptable. Recent surveys have shown that the intrinsic frequency offset of typical workstations and servers is from a few to a few hundred parts per million, which can result in timekeeping errors from a few to a few hundred microseconds, even when the clock is disciplined by NTP. Measurements made using fast, modern workstations demonstrate that the kernel time can be reliably disciplined to the order of a few tens of nanoseconds when sufficiently accurate and stable external reference sources are available.

However, most current Unix workstation kernel implementations do not provide precision time and frequency synchronization to the degree believed necessary for future real-time applications, such as distributed conferencing control and resource management. This project designs, implements, tests and refines the hardware and software algorithms to provide precision time synchronization to the degree possible with available computing equipment and synchronization sources.

4.1 Brief Description of Work and Results

Late models of workstations manufactured by Digital, Hewlett Packard and Sun Microsystems have been obtained under equipment grants. In addition, Global Positioning Service (GPS) receivers and cesium oscillators have been donated by several manufacturers and government agencies. We have pursued the analysis, simulation and synthesis of clock discipline algorithms that has resulted in a continuing refinement of the algorithms incorporated in the NTP software implementations over the last two decades. In addition, we have explored the application of external timekeeping devices and signals to improve timekeeping accuracy to the order of nanoseconds in modern workstations.

An important aspect in this investigation is the continued refinement of the algorithms used to distribute and maintain precision time in the Internet. Recent work demonstrates that the accuracy and stability of the system clock can be maintained with NTP message intervals much longer than previously possible. At the current level of refinement, accuracies of a millisecond or two can be maintained with update intervals up to several hours and at somewhat reduced accuracies to well over one day. This is an important feature when per-call toll charges are involved and in cases where low probability of intercept is required. This work is conducted in collaboration with NIST Boulder (Judah Levine) and USNO Washington (Richard Schmidt) and uses hybrid algorithms developed jointly by David Mills and Judah Levine.

Previously, we designed, built and tested generic Unix operating system kernel modifications for precision timekeeping. These modifications, described in RFC-1589, have been incorporated in experimental kernels for Digital, Sun and Hewlett Packard workstations. They are now incorporated in the production kernels for Digital and Sun workstations and in Linux and FreeBSD software distributions.

The kernel modifications have been rewritten to improve the resolution to the order of one nanosecond in time and one nanosecond per second in frequency. The new modifications have been incorporated in experimental kernels for Digital and Sun workstations and in FreeBSD software distributions. The software can be compiled for 64-bit machines, in which some variables occupy the full 64-bit word, or for 32-bit machines, where these variables are implemented using a macro package for double precision arithmetic. The software can be compiled for kernels where the time variable is represented in seconds and nanoseconds and for kernels in which this variable is represented in seconds and microseconds. In either case, and when the requisite hardware counter is available, the resolution of the system clock is to the nanosecond.

The new modifications also support a pulse-per-second (PPS) signal source using an external routine that captures transitions of this signal via a serial port device. The routines have been implemented and tested in Digital Unix, Sun SunOS and FreeBSD kernels. We expect the modifications will be integrated in Sun Solaris and Linux in the normal course of development.

The new NTP version includes several new reference drivers for new clock devices and operating modes. Two of these drivers use the audio hardware and software of Sun Microsystems' workstations and servers, together with advanced digital signal processing algorithms for filtering, demodulation and decoding the audio signals. One of these drivers is designed for the Canadian time/frequency station CHU and is used in conjunction with an ordinary shortwave radio. The other is designed for the IRIG-B and IRIG-E signals generated by some receivers and laboratory equipment. Performance with this driver and the IRIG-B signal of a GPS receiver provides reliable synchronization within a few microseconds on a Sun UltraSPARC. This is the best performance achieved without kernel modifications.

The new NTP Version 4 software daemon for Unix, VMS and Windows provides the capability to discipline the system clock well below the microsecond resolution formerly attainable. With sufficiently stable reference sources and the nanokernel modifications described above, the long term accuracy and stability can be improved to the order of one nanosecond in time and one nanosecond per second in frequency.

Past work involved the design and fabrication of workstation bus peripherals for precision time. An oven-stabilized precision clock oscillator using FPGA technology has been designed and implemented for the Sun Microsystems SBus. Hardware and software interfaces have been developed for the PPS and IRIG signals generated by some radio clocks and laboratory instruments. With these enhancements, residual jitter in computer time can be reduced to a few microseconds and stability to a few parts in 10^9.

An inexpensive LORAN-C precision timing receiver has been designed, built and tested. It is intended both as a backup for the GPS receivers and as a precision frequency stabilized source for the GPS receivers to reduce the effect of the intentional dither in the civil broadcast signal. It consists of three components, an ISA bus peripheral for a standard PC, an external temperature-stabi-

lized quartz crystal oscillator, and a C-language control program. Unlike our much more expensive Austron LORAN-C receiver, this receiver is completely automatic as it acquires up to four stations in a LORAN-C chain and disciplines the oscillator to within a few parts in 10^{10}. Two receivers of this type have been built, one operating in our laboratory and the other operating at a site in Europe.

A special purpose DSP development unit has been built and populated with experimental algorithms. One of these is a demodulator/decoder for radiotelegraph transmissions used by fixed and mobile shortwave communication services. Another is a demodulator/decoder for the WWV time/frequency service operated by NIST. Both algorithms operate with an ordinary shortwave receiver and use matched filters, soft decision and maximum likelihood techniques to achieve theoretical optimum performance. The performance is so good that reliable operation can be achieved when the broadcast signals cannot be heard by ear.

4.2 Future Plans

We have been working with the kernel developers for Solaris, Digital Unix, Linux and FreeBSD on a generic API for the pulse-per-second (PPS) signal generated by some receivers and laboratory equipment. A preliminary design has been produced as an Internet Draft and is now being circulated in the IETF for review. An preliminary prototype has been implemented for the FreeBSD kernel. Using a new Alpha workstation donated by Digital, we are working on a similar implementation for the Digital Unix kernel.

As time permits, we plan to port the DSP version of the WWV demodulator/decoder as a driver for NTP Version 4 and the Sun audio system. In addition, we plan to adapt the audio interface to support generic codecs of other hardware and software architectures, in particular the generic PC used as the CAIRN/DARTnet routers.

Other areas that may be explored include clock discipline algorithms for use with precision oscillators, such as the existing FPGA-based precision oscillator. This device is ordinarily loosely disciplined by NTP; however, where accuracy requirements dictate, it can be stabilized using a LORAN-C receiver. As available resources permit, we plan to redesign the existing LORAN-C receiver as a PCI bus peripheral. We also plan to implement the discipline function itself, now performed by daemon or kernel emulation, entirely in the FPGA. This design provides an accurate, stable synchronization source independent of the daemon or kernel which, even if the daemon or network fails for extended periods, can continue to provide accurate time.

5. 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.

5.1 Autokey

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

5.2 Run-time Autoconfigure

The manycast/anycast scheme proposed in previous report [12] 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.

5.3 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.

5.4 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 discontinuity 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.

6. 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 inter-

faces, 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 [19].

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_adjtime() 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.

7. 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.

8. 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.

8.1 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.

8.2 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.

8.3 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.

9. 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. 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.

9.1 Papers with Abstracts

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.

   This paper describes a new security model and authentication scheme for distributed, real-time network protocols used in time synchronization and event scheduling applications. It outlines the design requirements of these protocols and why these requirements cannot be met using conventional cryptography and algorithms. It proposes a new design called *autokey*, which uses a combination of public-key cryptography and a psuedo-random sequence of one-way hash functions. Autokey has been implemented for the Network Time Protocol (NTP), but it can be adapted to other similar protocols. The paper describes the protocol operations, data structures and resources required for autokey, as well as a preliminary vulnerability assessment.

   This paper describes a new security model and authentication scheme for distributed, real-time network protocols used in time synchronization and event scheduling applications. It outlines the design requirements of these protocols and why these requirements cannot be met using conventional cryptography and algorithms. It proposes a new design called *autokey*, which uses a combination of public-key cryptography and a psuedo-random sequence of one-way hash functions. Autokey has been implemented for the Network Time Protocol (NTP), but it can be adapted to other similar protocols. The paper describes the protocol operations, data structures and resources required for autokey, as well as a preliminary vulnerability assessment.

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

   This paper describes the analysis, implementation and performance of a new algorithm engineered to discipline a computer clock to a source of standard time, such as a GPS receiver or another computer synchronized to such a source. The algorithm is intended for the Network Time Protocol (NTP), which is in widespread use to synchronize computer clocks in the global Internet, or with another functionally equivalent protocol such as DTSS or PCS. It con-

trols the computer clock time and frequency using an adaptive-parameter, hybrid phase/ frequency-lock feedback loop. Compared with the current NTP Version 3 algorithm, the new algorithm developed for NTP Version 4 provides improved accuracy and reduced network overhead, especially when per-packet or per-call charges are involved. The algorithm has been implemented in a special purpose NTP simulator, which also includes the entire suite of NTP algorithms. The performance has been verified using this simulator and both synthetic data and real data from Internet time servers in Europe, Asia and the Americas.

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.

   This paper describes a massive survey of Network Time Protocol (NTP) servers and clients in order to discover statistics of time and frequency errors, as well as determine the health and welfare of the NTP synchronization subnet operating in the global Internet. Among the conclusions of the survey are that most NTP clocks are within 21 ms of their synchronization sources and all are within 29 ms on average. However, additional errors up to one-half the roundtrip delay are possible, but relatively rare. There is, however, a disturbing incidence of improperly operating servers and misguided server configurations.

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.

   Cryptographic authentication methodology proposed for use in the Internet require substantial resources when very large client populations are involved. Resource provisioning becomes especially important when time-critical services are involved. In the cast of time-synchronization services, a special case exists, since cryptographic keys must enforce valid lifetimes, but validating key lifetimes requires cryptographic keys. This paper proposes a scheme which minimizes server resources while resolving the apparent circularity.

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.

   This paper describes algorithms to discipline a computer clock to a source of standard time, such as a GPS receiver or another computer synchronized to such a source. The algorithms are designed for use in the Network Time Protocol (NTP), which is used to synchronize computer clocks in the global Internet. They have been incorporated in the NTP software for Unix and Windows and, for the highest accuracy, in the operating system kernels for Sun, DEC and HP workstations. RMS errors on LANs are usually less than 10 μs and on global Internet paths usually less than 5 ms. However, rare disruptions of one kind or another can cause error spikes up to 100 μs on LANs and 100 ms on Internet paths.

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

The Network Time Protocol (NTP) is widely deployed in the Internet to synchronize computer clocks to each other and to international standards via telephone modem, radio and satellite. The protocols and algorithms have evolved over more than a decade to produce the present NTP Version 3 specification and implementations. Most of the estimated deployment of 100,000 NTP servers and clients enjoy synchronization to within a few tens of milliseconds in the Internet of today.

This paper describes specific improvements developed for NTP Version 3 which have resulted in increased accuracy, stability and reliability in both local-area and wide-area networks. These include engineered refinements of several algorithms used to measure time differences between a local clock and a number of peer clocks in the network, as well as to select the best ensemble from among a set of peer clocks and combine their differences to produce a clock accuracy better than any in the ensemble.

This paper also describes engineered refinements of the algorithms used to adjust the time and frequency of the local clock, which functions as a disciplined oscillator. The refinements provide automatic adjustment of message-exchange intervals in order to minimize network traffic between clients and busy servers while maintaining the best accuracy. Finally, this paper describes certain enhancements to the Unix operating system software in order to realize submillisecond accuracies with fast workstations and networks.

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

   This paper builds on previous work involving the Network Time Protocol, which is used to synchronize computer clocks in the Internet. It describes a series of incremental improvements in system hardware and software which result in significantly better accuracy and stability, especially in primary time servers directly synchronized to radio or satellite time services. These improvements include novel interfacing techniques and operating system features. The goal in this effort is to improve the synchronization accuracy for fast computers and networks from the tens of milliseconds regime of the present technology to the submillisecond regime of the future.

   In order to assess how well these improvements work, a series of experiments is described in which the error contributions of various modern Unix system hardware and software components are calibrated. These experiments define the accuracy and stability expectations of the computer clock and establish its design parameters with respect to time and frequency error tolerances. The paper concludes that submillisecond accuracies are indeed practical, but that further improvements will be possible only through the use of temperature-compensated local clock oscillators.

9.2 Technical Reports

9. 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.

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

11. 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.

12. 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.

13. 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.

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. Unix kernel modifications for precision time synchronization. Electrical Engineering Department Report 94-10-1, University of Delaware, October 1994, 24 pp.University of Delaware, March 1994. 31 pp.

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

17. Mills, D.L. Precision synchronization of computer network clocks. Electrical Engineering Department Report 93-11-1, University of Delaware, November 1993, 66 pp.

9.3 Internet Drafts

18. 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.

19. 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.