NASA-CR-197772

INTERIM
IN-61-CR
OCIT
49071
P. 33

(NASA-CR-197772) PERFORMANCE AND POLICY DIMENSIONS IN INTERNET ROUTING (Delaware Univ.) 33 p

N95-26770

Unclas

G3/61 0049071

Technical Report

ud ee

University of Delaware NEWARK, DELAWARE 19716

Department of Electrical Engineering

Electrical Engineering Department University of Delaware
Technical Report 95-4-1 April 1995

Performance and Policy Dimensions in Internet Routing

David L. Mills, Charles G. Boncelet, John G. Elias,
Paul A. Schragger, Alden W. Jackson, Ajit Thyagarajan

Sponsored by: Advanced Research Projects Agency under NASA Ames Research Center contract NAG 2-638, National Science Foundation grant NCR-93-01002 and Northeastern Center for Engineering Education contract A30327-93.

Abstract

The Internet Routing Project, referred to in this report as the "Highball Project," has been investigating architectures suitable for networks spanning large geographic areas and capable of very high data rates. The Highball network architecture is based on a high speed crossbar switch and an adaptive, distributed, TDMA scheduling algorithm. The scheduling algorithm controls the instantaneous configuration and swell time of the switch, one of which is attached to each node. In order to send a single burst or a multi-burst packet, a reservation request is sent to all nodes. The scheduling algorith then configures the switches immediately prior to the arrival of each burst, so it can be relayed immediately without requiring local storage. Reservations and housekeeping information are sent using a special broadcast-spanning-tree schedule.

Progress to date in the Highball Project includes the design and testing of a suite of scheduling algorithms, construction of software reservation/scheduling simulators, and construction of a strawman hardware and software implementation. A prototype switch controller and timestamp generator have been completed and are in test. Detailed documentation on the algorithms, protocols and experiments conducted are given in various reports and papers published. Abstracts of this literature are included in the bibliography at the end of this report, which serves as an extended executive summary.

Keywords: supercomputer networks, gigabit networks, reservation-TDMA, Highball architecture.

Table of Contents

1. Introduction and Background

Local-area networks operating at speeds approaching or exceeding 1 Gbps have become common in supercomputer facilities and other institutions where very high speed communications are required. The bus architecture of the High Performance Parallel Interconnect (HPPI), widely deployed in supercomputer installations, operates at speeds close to 1 Gbps, as do modern workstation buses such as SBus and HYPERchannel. The fact that data paths are relatively wide (32 and 64 bits) and propagation delays small leads to designs with multi-wire, point-to-point transmission links and centralized bus or switch control.

Future common-carrier networks may employ the Synchronous Digital Hierarchy (SDH), which uses the Synchronous Optical Network (SONET) interface and can operate at speeds well in excess of 2.4 Gbps. It may well happen that high speed, wide area networks will be constructed using Synchronous Digital Hierarchy (SDH) transmission facilities, SONET interfaces, and Asynchronous Transfer Mode (ATM) packet switches co-located with the SONET termination equipment in the central office or user premises. SONET interfaces operate in a fixed configuration (i.e., not switched on either a circuit or a frame basis). At low and medium speeds, up to perhaps 622 Mbps for a single user, packet-switching access technologies, such as ATM or Switched Multi-Megabit Data Service (SMDS), may be appropriate for conventional interactive-access and bulk-transfer network services.

For unconventional applications at speeds much above 622 Mbps for a single user, packet-switching technologies may not be the best choice. This report presents the rationale and a strawman design for an alternative high speed network architecture, called Highball, capable of operation at end-to-end transmission rates of 1 Gbps or greater over significant geographical distances. It is intended for applications involving very high traffic rates typical of supercomputing, space science, and digital map retrieval applications.

Highball is a high speed, time division multiplexed computer network intended for a moderate number of users (10--100) spanning a relatively large area, such as the continental United States. The network uses crossbar switches connected together by unidirectional transmission links. In practice, these will likely be fiber optic connections. The switches are controlled locally by embedded computer controllers which are precisely time synchronized to within the order of a microsecond using a variant of the Network Time Protocol (NTP) together with precision timing receivers for the Global Positioning Service (GPS) or LORAN-C radionavigation system. Switching is done in a "just in time" fashion by reconfiguring the switches so that incoming bursts are routed to the correct outgoing links. Nowhere within the network are the data in transit stopped, queued, or otherwise altered. All queueing is done at the perimeter of the network.

Highball is designed for high demand, bursty users. Because of the high speeds involved, bursts can be much shorter than the link delays in the network. For instance, even a megabit burst at 1 Gbps is only a millisecond long and flies across the country in about 30 ms via landline and about 270 ms via satellite. Typical users include supercomputers, real-time disk-to-disk transfers, interactive video, and visualization on remote workstations. These users can individually demand 1 Gbps or more, but do so for brief, bursty periods. In contrast, conventional high speed networks are usually designed for highly multiplexed aggregates of low rate users. The statistics are very different for bursty users than for a multiplexed aggregate. A network designed for multiplexed, low rate users is likely to be ill suited for bursty, high rate users.

Examples in which Highball technology may be used effectively include intelligence, space science and astronomy. One candidate application is digital map retrieval, where tiles for a large composite, digitized map are retrieved in real time as a function of mouse movement on a synthetic lightbox. The Advanced Communications Technology Satellite (ACTS), which uses a satellite-switched time-division multiple access (SS-TDMA) protocol and transponders that operate up to several hundred Mbps, would appear to be an ideal candidate for this technology. Another candidate is the Space Station, which is to operate using a 300-Mbps space-ground link multiplexed for telemetry, video and voice. In these and other systems requiring high burst rates, but relatively low overall network utilization, the Highball approach would be equally effective.

The remainder of this report is as follows: Section 2 presents the design approach and discusses some of the motivation for Highball. Section 3 presents an overview of the Highball hardware developed for a prototype demonstration network. Section 4 outlines the current status of the project and presents initial results. Subsequent subsections cover scheduling, comparison with other scheduling algorithms, the Highball control channel, robustness against scheduling errors, time synchronization, and details of the Highball hardware and software. Section 5 discusses future work and Section 6 mentions some of the areas of technology transfer of the lessons learned from the project. The report concludes with a bibliography of publications arising from the project.

2. Design Approach

The architecture and design described herein has been specialized for high speed burst applications and a modest user population. A prototype Highball network might at deployment consist of from a few to a few tens of high speed links, either terrestrial or satellite, and include nodes at supercomputer centers and participating workstation sites. The primary service of this network would be to deliver very large quantities of data over very long distances as fast as possible and schedule network access on a fixed or dynamic basis. It would be capable of several reservation modes appropriate for bulk-data transfer and sustained motion-video refresh in both unicast and multicast modes.

The actual switching fabric takes the form of a crossbar switch driven by a system of counters which precisely determines the switch configurations and dwell times. The tables controlling these operations are computed using a synchronized, distributed algorithm which runs in a dedicated controller at each node. Since the nodes need not inspect the packet headers or data formats used for end-end traffic, the network can support a variety of protocols, packet formats and encoding methods. In addition, while other methods can be adapted for multicast delivery, in the externally switched crossbar approach a multicast capability is inherent.

The externally switched approach appears to be ideally suited for a high speed WAN. For reasons of switching speed, cost, and availability at the present state of development, electronic crossbar switches are more practical and represent the technology of choice for a prototype Highball network. However, the Highball approach may be particularly attractive should appropriate photonic switches become available, in which case the network speeds are limited only by the characteristics of the fiber and switch itself and no electronics need appear in the data paths.

The principal question in scheduling and routing for Highball is how and when to set the switches so that data successfully traverses the network. This is done using a multidestination control channel to send scheduling requests, timekeeping information, and network topology updates. The scheduling problem is perhaps best understood in terms of an analogy to a railroad of high speed trains.

The trains have finite size and are small compared with the distance across the network. We desire to dispatch trains without collisions and without stopping a train once started. Furthermore, unlike real train networks, we cannot signal ahead because our trains are traveling as fast as the signals. The scheduling problem is when to start the trains and how and when to set the switches along the way. One complication is that we desire to transmit many trains concurrently, each going in different directions. This scheduling analogy also gives the Highball Project its name. In the vernacular of trains, "highball" means that the train has priority and can travel as fast as the tracks allow. We are trying to allow our trains to travel as fast as their tracks allow.

Time synchronization is a significant hurdle for this network. Bursts propagate at about two-thirds of the speed of light and are switched on- the-fly. The entire network must agree on system timing to within about one microsecond, while delays between nodes must be known as accurately. We believe the network clocks can be made this accurate using the methodology of NTP, which has recently been extended for just this purpose.

Two paradigms have been proposed for burst scheduling: Reservation-TDMA (R-TDMA) and Adaptive Circular TDMA (AC-TDMA). In R-TDMA, when a node wishes to transmit a data burst, it sends a request on a control channel. This request is received by all other nodes, scheduled, and at the appropriate time, the burst is transmitted. Switches are set along the way as required. Since all nodes receive the same request and execute the same scheduling algorithm, they all arrive at the same schedule. Thus, there is no need to broadcast the schedule itself. To assure that all nodes process the requests in the same order, the requests are sorted on the basis of two fields. The primary field is a timestamp of when the request originated. The secondary field is the originating node number. These two fields uniquely define all requests within the network.

In AC-TDMA, a persistent circular schedule is set up when the network is initialized. This schedule allows each node to transmit to each other node once per rotation time. With 1 ms bursts, N nodes require a rotation time of at least N-1 ms. In practice, the time is slightly larger. (There are also advantages in having it slightly larger.) If a node needs more capacity in communicating with another, say for a video transfer lasting many rotation times, that node sends out a request on the control channel. This request is received and arbitrated, and all nodes adjust the schedule accordingly. This adjustment is akin to N people standing in a circle. In a person wants more space, he "elbows" his neighbors. If they are unwilling to adjust, they elbow their neighbors until either a gap is reached or some other person gives up some of his room. When the needs for increased capacity disappears, the schedule reverts back to the usual persistent one.

3. Network Architecture

A Highball network consists of a number of switching nodes each consisting of a crossbar switch and switch controller. Hosts are connected to a local node using Ethernet, FDDI, or HPPI technology, depending on operating speeds. Nodes can be interconnected using SONET technology over terrestrial or satellite transmission media. For the highest speeds supported by the crossbar switch and internode links, an appropriate hardware-blocking interface is required at the external processor supporting the application. Input/output transfers are enabled by the node controller, but the timing and formatting of the data burst itself is the responsibility of the external interface.

The switch controller runs a reservation-directed distributed scheduling algorithm which allocates link transmission times to avoid collisions at the switches, as well as to synchronize all timing and switching operations. The controller is designed as a programmable peripheral for a general-purpose

RISC workstation with a bus architecture capable of speeds at least that of the network links. In the prototype Highball design the controller is a Sun Microsystems SPARCstation and the peripheral is implemented as two SBus interface modules, one to control the switch and the other to generate and process timestamps.

Nodes communicate among themselves using a signalling protocol and control message bursts time-multiplexed with the data bursts. The data transceivers include the necessary encoding-decoding and clocking functions common to other systems such as FDDI, so that chipsets for these systems may be used. While the data transceivers are necessary for signalling between the nodes of the network, they can in principle also be used to carry ordinary end-end data that originates in the node itself. This would occur if the node were used as an Ethernet gateway or monitoring platform, for example. However, in the prototype Highball network it is intended that these data originate and terminate on application processors using HPPI.

The basic switching fabric is constructed of an N x N x W asynchronous crossbar switch in which each of N W-bit input lines can be connected simultaneously to any combination of N output lines or to none, but no output line can be connected to more than one input line. As an exercise, a prototype VLSI 4 x 4 x 1 crossbar switch has been implemented using MOSIS technology. Multiple switches can be operated in parallel for additional bandwidth. The switch configuration is described by a vector of eight four-bit components, one component associated with each output line, with the value of the component indicating which input line is connected to it. The prototype VLSI switch operates at a data rate of 25 Mbps with a configuration dwell time of at least 10 us and an interburst guard time at least 1 us. The chip includes a configuration register, along with buffer and counter circuitry which provides a rapid reconfiguration capability.

4. Project Status and Results

The Highball project has demonstrated that the assumptions made can be satisfied. Principal among these assumptions are that schedules and be found and computed quickly enough and that nodes can be accurately synchronized to within about 1 us, thus allowing for efficient implementation of TDMA protocols. The results specific to scheduling are discussed briefly in Subsections 1-4 below, the timing results in Subsection 5, and the hardware and software development in Subsections 6 and 7.

4.1. Highball Scheduling Issues

Considerable progress has been made on scheduling bursts for Highball. On the theoretical side, we have demonstrated that the scheduling problem is hard (NP-complete). However, we have results comparing the Highball scheduling problem to a class of well known NP-complete problems that seem to afford good algorithms in the average case. This class includes bin packing, knapsack, and partitioning problems. We have shown that heuristic algorithms for Highball work well in the average case and that exponential algorithms are not necessarily bad in this application.

On the practical side, we have proposed several scheduling algorithms for R-TDMA. Each has its pros and cons. Primarily the tradeoff is between computational complexity and scheduling efficiency. By scheduling efficiency, we mean how successful is the algorithm at finding schedules subject to a time constraint?

Scheduling algorithms developed in this research include a generic labeling algorithm and two specific algorithms, Pathfinder, and Trailblazer. The labeling algorithm is the leading candidate for actual deployment. It is reasonably fast and is very effective at finding routes. It can be used for both R-TDMA and AC-TDMA. The labeling algorithm works by marking or labeling each link that can be reached from the source. Then, each link that can be reached from each of those is marked. The process continues until the destination is reached or no more links can be marked. A full discussion can be found in [BON92]. Pathfinder is fast, but somewhat less effective than the labeling algorithm at finding paths. Trailblazer is optimal, but very slow. Its main use is to provide a standard against which to compare other algorithms. Both are described in [SCH91].

For R-TDMA, we have done fairly complete simulations on a network modeled after the NSF backbone. The trial network has 13 nodes, located where the nodes are for the NSF backbone, connected together by 19 pairs of unidirectional links. With Poisson arrivals, uniform sources and destinations, and an out of band control channel, the labeling algorithm can achieve scheduling efficiencies as high as 90%. In other words, each node sends and receives 90% of the time.

For AC-TDMA, we have described a variation of the basic algorithm and built a simulator. On the same NSF backbone model, with 1 ms bursts, we have achieved rotation times as small as 14 ms, where a rotation is the interval in which each node can send a 1-ms burst. We have posed the adjustment problem (the "elbowing" problem) as a linear program with various objective functions and are studying solutions.

4.2. Comparison With Other Scheduling Algorithms

In order to assess the performance of R-TDMA, we compared it to two other simple schemes, Random Access (RA) and Fixed Time Division Multiple Access (F-TDMA). RA and F-TDMA networks are alternative methods of network access that could be used instead of the proposed Highball network. The simulations provide evidence of the improved performance of R-TDMA or AC-TDMA versus RA or F-TDMA.

R-TDMA methods allocate network resources on demand. The links between the switches are the resources and are allocated to bursts. The allocation is the result of assigning a path and departure time to each burst and is constrained by the requirements of eliminating collisions at the links and minimizing burst delay.

The R-TDMA algorithm designed for the Highball network, called the Pathfinder algorithm, was used in comparative simulations. The Pathfinder algorithm finds the shortest-delay schedule for each burst without preempting previously scheduled bursts. To reduce the complexity of finding a feasible schedule for each burst, the number of paths that are searched is limited to the non-looping paths in the network. The examination of the applicable paths results in a selection of a single path and departure time that yields the shortest delay.

A Random Access (RA) network transmits a burst as soon as the burst's source transmitter is free. To simulate RA, the burst paths are precalculated for each source/destination pair. The simulated network switches use the precalculated route to switch the burst to the outgoing transmission line in a cut-through type manner. If a collision occurs at a switch, both bursts involved are presumed lost. In our simulations a dropped burst is not retransmitted. The loss of the burst is tallied and the rate of dropping is calculated. This scheme avoids the complexities of introducing retransmission strategies into the model. However, without retransmission of dropped bursts, the network is unconservative and cannot be directly compared with the other access schemes. The RA network model was simulated only to provide insight into the operational constraints of a wide-area interconnection network without queues.

For F-TDMA, network access times are determined by a slotted and cyclic schedule. The cyclic schedule establishes when a path from a source to a selected destination is available. To create a repeating schedule, the network topology must be known and a traffic flow assumed. For our simulations a uniform distribution of traffic from all sources to all destinations was assumed. Two methods for creating repeating schedules were examined. The first method treats the network as a broadcast channel and constructs the repeating schedules using broadcast spanning trees. A scheduling algorithm called the Repeating Broadcast Schedule Algorithm (RBSA) was used. The second F-TDMA scheduling method uses the shortest delay paths for each source/destination pair and schedules them to occur in a repeating pattern. This algorithm is called the Point-to-point Repeating Schedule Algorithm (P2PSA).

Network traffic was simulated with a special-purpose, discrete-event simulator written in the C language. The 1990 NSFNET was used as the test network configuration. We track two variables, source queue size and network utilization, as the normalized network load is varied. Normalized offered network load is calculated for each simulation run as the total load divided by the number of destinations. Source queue size reflects delays due to scheduling constraints, while network utilization measures the percent of time that the destination nodes are receiving bursts.

The simulations of the network access schemes have shown that although the R-TDMA method incurs high queuing delays due to the control channel, but it has the advantage of handling higher utilizations than the other access methods. The RA simulations confirmed that significant loss due to collisions would occur even at low loads. Simulation of both F-TDMA networks showed that F-TDMA works well for low loads. However, since the channels are statically assigned, F-TDMA networks saturate at much lower loads that R-TDMA. From these results we infer that future scheduling research may profitably examine the hybridization of the F-TDMA and R-TDMA. One hybrid method, AC-TDMA, combines the lower delay at low loads of F-TDMA and the capability of carrying higher loads through adaptive allocation like R-TDMA.

4.3. Control Channel Design

As mentioned earlier, reservations and housekeeping information are sent using a multicast tree schedule. This "control" channel delivers scheduling requests, timekeeping information, and network topology updates to all the nodes. The schedule for the control channel is reserved before any data bursts, ensuring that no node is denied access to the control channel. The multicast schedule repeats within an interval chosen to minimize the waiting time, allowing timely access to the control channel. The periodic multicast also should allow the multicast period to be much smaller than the time to complete a single multicast. When this case exists, not only are there multiple frames in transit in the network at any given time, but there may also be multiple frames in transit on any link. Each node is allowed to transmit one multi-destination burst per interval or frame. Any changes to the multicast topology will require recalculation of the schedule.

Since the control channel messages are of fixed size, the optimal multicast schedule for the control channel occurs when each node receives a series of multicast messages one after the other, with no time in between. A frame is considered optimal when the unreserved portion is contiguous. Frames having this property maximize burst throughput, while avoiding fragmentation overheads.

Let the frame begin with the arrival of the first message. The optimal contiguous portion of the frame not allocated by the multicast, with N nodes, will be:

$$\text{frame length} - (N-1)(\text{multicast message length}).$$

It can be shown that by using cycles and single spanning-tree multicast topologies, periodic multicast schedules can be found with a contiguous, unreserved, per-frame interval of

frame length - (N)(multicast message length),

which is almost optimal.

For example, this efficiency can be achieved for a ring topology by using a super-token in a Hamiltonian cycle. A Hamiltonian cycle is a one-way path that contains each node only once and ends at the beginning node. The super-token has a size equal to the length of N multicast messages. As the super-token passes a node on the cycle around the network, the node reads N-1 messages and writes its own. Although the super-token mechanism for periodic multicast schedules is straightforward once the cycle is found, finding the existence of a Hamiltonian cycle in a network with a general topology is NP-complete.

We have developed a scheduling algorithm called Root Multicast - Single Tree (RM-ST) that achieves the above efficiency by restricting the multicast to one spanning tree. All the sub-root nodes of the tree pass the multicast messages up to the root, which simultaneously multicasts the messages, as they arrive, down to all the nodes. Finding a spanning tree in a general connected network is guaranteed and inexpensive, especially when compared to finding Hamiltonian cycles.

Schedules on both cycles and single spanning trees share a common efficiency, since each node receives N, not N-1, multicast messages per frame. Optimum scheduling is possible only if every node does not have to both send and receive its own message. Broadcasts on multiple spanning trees rooted at each node active in the multicast would not require any node to listen to its own message. However, it is difficult to schedule many repeating spanning trees and the burst availability is very fragmented.

To show the fragmentation of the frame, we derived the length distribution of the fragments and found it to be beta distributed with parameters, a=1 and b=N-1. This result implies that as the number of spanning trees in the periodic multicast schedule increases, the smaller the contiguous intervals become and the less likely it is that a schedule can be found to satisfy the criterion that the unreserved portion of the frame be contiguous. The results of simulations using the NSFNET 1990 topology support the conclusions made from this analysis.

In conclusion, both super-tokens on Hamiltonian cycles and RM-ST schedules achieve close to the optimal efficiency. The single spanning tree is preferred, however, since it is easier to compute and has a smaller total delay.

4.4. Error Modeling and Robustness Issues

One potential problem of the Highball paradigm of sending requests and having each node separately compute schedules is that of robustness when the request mechanism fails. In particular, if a request fails to arrive at a node, how is its computed schedule affected and for how long? We refer to this error as an "error of omission." This subsection presents the results of both simulations and analysis on this effect. In summary, simulations of simple networks indicate that errors of omission often correct themselves in a short time without need for intervention. Analysis also supports this conclusion.

By taking two schedulers and the same input sequence, we monitored the output of each when one is given errors of omission in the input sequence. Examining the error process, an error of omission in the input sequence to a scheduler will cause its corresponding schedule to diverge from a schedule derived from the correct input. If no other input is offered to be scheduled after the omission, the incorrect schedule will converge immediately back to the correct state after the omitted burst is scheduled.

An interesting question is for how long will the schedule be incorrect once an error of omission occurs if the input continues. We propose a simple Markov model of this error process and derived the limiting probabilities for the occurrence of correct and diverged states, given a fixed probability of error. The numerical results are compared with simulation data derived from the Pathfinder scheduler for simple chain networks. A node was given errors of omission at varying rates and the amount of time it scheduled incorrectly was computed and simulated.

Numerical results from the above model consistently underestimate the length of time in the error state found in simulations. This behavior occurs because the model under-estimates the length of time in the diverged state. After an error of omission, it is likely that an incorrect scheduling decision will be made if an arrival occurs while the schedule is in the diverged state. Accordingly, we also proposed a more detailed state dependent Markov model to try to account for this effect.

The numerical results of the state dependent model, using the data from the network above, upper bound the simulation data for all but the highest loads at high error rates (Probability of error 1e-3). The model fails at high load/high error rate, because the model does not account for successive errors of omission while in a diverged state.

Work is continuing on modeling the error of omission process for more complex networks where the interaction mechanism is less straightforward. Applications of the research include recovery mechanisms of the global schedule state after topology updates and the early committal of resources before all the nodes have received all the requests. To support early commit, a scheduling algorithm for the control channel called Greedy Multicast, which has performance between RM-ST and multiple tree broadcasts, has been developed.

4.5. Highball Time Synchronization

Work on computer network time synchronization has been going on in various programs since the implementation of the Hello routing protocol for the Fuzzball operating system in 1980 [MIL83]. The Fuzzball itself, documented in [MIL88b], was first deployed in the DARPA SATNET Project [MIL81] and more recently in backbone routers in the NSFnet between 1986 and 1988 [MIL87]. Time synchronization is an integral part of the Fuzzball operating system and Hello routing protocol, which served as the workbench on which four versions of the Network Time Protocol (NTP) evolved: the base version (RFC-958 [MIL85c]), version 1 (RFC-1059 [MIL 88a]), version 2 (RFC-1119 [MIL89c]) and the current version 3 (RFC-1305 [MIL90b]). A subset of NTP, called the Simple Network Time Protocol (SNTP), is used by some embedded time servers (e.g., Bancomm) where the full functionality of the protocol is not justified. NTP has since been ported beyond the Fuzzball to over two dozen Unix platforms from microcomputers to supercomputers and an estimated total of 100,000 installations worldwide [MIL94].

NTP in its various versions has been used in numerous experiments and measurement programs scaling from 100-Mbps LANs to worldwide WANs spanning every continent except Antarctica (coming soon) [MIL83a], [MIL85b], [MIL87], [MIL88b], [MIL89b], [MIL93]. The engineering design and performance of the NTP algorithms have been analyzed in several reports [MIL85a], [MIL90b], [MIL91b], [MIL92b], [MIL93]. The performance of NTP has steadily improved over the versions to the present, where accuracies to the order of a few milliseconds can be achieved with modern Unix workstations and typical Internet paths [MIL93].

NTP is an integral component of our Highball network architecture [MIL90c]. A Highball network is built using fast, computer controlled crossbar switches. The switches are controlled so that a message burst can be propagated end to end without storage at intermediate nodes, so all switches must be precisely synchronized. A concomitance in this approach is that a sending host must know exactly when to launch the burst, so this requires that the host be precisely synchronized as well. In our original approach, host and switch synchronization is maintained by a special-purpose bus peripheral with onboard counters and a programmable oscillator. The oscillator is controlled by a software phase-lock loop, which in turn is driven by a modified NTP daemon. The daemons synchronize among themselves using messages transmitted in dedicated time-multiplexed slots.

It is important to the Highball Project in particular and to the general timekeeping community in general to push the performance envelope of NTP to the greatest degree possible without requiring special-purpose hardware or software. We have pursued this goal in the full understanding that it is unrealistic to expect performance conforming to Highball accuracy requirements (1-2 usec) using current software and hardware platforms; however, workstation of the future with single-digit nanosecond clocks may well have the speed necessary to achieve that goal. A good indicator of the future may be the DEC Alpha workstation, in which we have measured the time for an application to capture a timestamp at about 10 usec [MIL93].

A major research activity has been the refinement of computer network timekeeping technology suitable for use in high speed networks such as Highball. This involves the development of disciplined frequency sources and control algorithms in the time servers, as well as refinement of the synchronization algorithms and protocols used in the network. This activity has focussed on the design and implementation of the Network Time Protocol (NTP) Version 3. The formal specification of this protocol has been published as RFC-1305 and designated a Proposed Standard by the IAB; however. NTP Version 3 provides more accurate time, certifiable error bounds, a revised and improved clock-selection procedure, and an adaptive-parameter local-clock model which significantly reduces network overhead. An implementation of NTP Version 3 has been running in the Fuzzball time servers for the last two years and in Unix time servers in North America, Europe and Asia for over a year.

One of the concrete objectives at this stage of time-transfer research is that of security, including replay and modification attacks. In response to misgivings about security voiced by the Kerberos Project at MIT, a crypto-checksum procedure was implemented in NTP Version 2 and carried over to Version 3. The design of this procedure was described in a recent meeting of the Privacy and Security Research Group (PSRG), which took the issue on as a work item. Subsequently, Matt Bishop, a member of the PSRG, prepared a detailed report analyzing the security issues involved with NTP. This report was highly useful in crafting subtle modifications to the NTP Version 3 procedures in order to make them more robust and secure.

The issue of overseas distribution of NTP has always been a problem, since the authentication features use the Digital Encryption Standard (DES) algorithm, export of which is controlled by the US Government. Louis Mamakos of the University of Maryland adapted the RSA MD5 algorithm for use as an alternate and this is now supported along with DES in NTP Version 3. For export a special version of the NTP Version 3 Unix daemon is available with the DES code itself encrypted with DES and a secret key. The secret key is available only to those who can prove US or Canadian residence.

The evolved NTP architecture and algorithms of Version 3 are directly portable to the Highball technology; however, the protocol machine is not. The existing protocol machine is organized around a system of hierarchical, bilateral peers, where peer associations are maintained separately by each peer for a subset of peers elsewhere in the synchronization subnet. In this model, synchronization is "relayed" from the root of the subnet to the leaves. For Highball, it is necessary that each peer maintain precise measures of delay and time offset between all nodes in the network, not just a subset.

The needed functionality is provided using the control channel and broadcast spanning trees of the Highball design. For this purpose the spanning tree must be augmented to include all nodes in the network. Each node keeps a table of the most recent timestamps received from all other nodes. The data include the transmission and receiving times of the message itself, plus the table of timestamps received from other nodes. After a number of iterations the nodes accumulate enough timestamps to construct a full delay/offset matrix.

The remainder of this section describes recent progress in refining the NTP architecture and model, together with a number of experiments designed to calibrate performance and confirm correct operation. Mention is made often of the NTP Version 3 daemon for Unix systems. A distribution of this daemon and related components is available for anonymous FTP from louie.udel.edu in the pub/ntp directory as the compressed tar archive xntp3.4o.tar.Z. Note that in future the version identifier 3b may change.

4.5.1. Precision Time Calibration and Distribution

As workstations and networks increase in speed, it is important that the accuracy and stability of the time-synchronization service increase to match. Recent modifications in the NTP Unix daemon have resulted in accuracies better than a millisecond with modern workstations on local nets. Recent modifications in the SunOS kernel by Van Jacobson and Craig Leres of Lawrence Berkeley Laboratories (LBL) have resulted in accuracies of a few tens of microseconds using precision-time (pps) signals provided by some radio and satellite receivers. However, there are few available timing receivers with this kind of accuracy and required stability. Among the few choices are a very expensive, special-purpose GPS timing receiver, a moderately expensive, modified general-purpose GPS navigation receiver, and an inexpensive, computer-controlled LORAN-C timing receiver. All three of these approaches have been evaluated in our laboratory, while the modified GPS navigation receiver approach has also been evaluated at LBL.

Due to vagaries of the transmission medium and particular radio designs, errors of various kinds sometimes occur. Most receivers depending on low-frequency radio propagation (e.g., WWVB, MSF, and DCF77) can be relied only to the millisecond and sometimes produce much larger errors due to atmospheric electrical discharges and ionospheric variations. In addition to these errors, the timecode formats of most radios have a precision only to the millisecond. Even if these radios have pps outputs, the jitter on these outputs, 100 microseconds or so, considerably exceeds the jitter due to the interrupt latency of a typical workstation, 20-30 microseconds for a SPARCstation with modified kernel.

Notwithstanding the accuracy problems due to propagation variations and radio timing jitter, the greatest threat to accuracy is oscillator frequency stability. For reasons mentioned above, it is desirable to use an external frequency source, either in the form of a temperature-compensated crystal oscillator and bus interface card or an external frequency standard. However, as shown in recent experiments, it is not practical to expect frequency stabilities better than 1 ppm in the case of onboard uncompensated oscillators or 0.1 ppm in the case of commonly available and relatively inexpensive compensated oscillators. In order to realize an accuracy of ten microseconds using SPARCstations with measured intrinsic frequency errors in the range 100 ppm, for example, the interval between oscillator corrections would have to be not greater than 100 ms, which is probably not practical for most applications.

It would be highly desirable to reduce or eliminate the requirement for frequent interaction between the NTP server and its peers in order to achieve the highest accuracy. One way to do this is to employ NTP or a relatively imprecise timecode receiver to supply the seconds numbering and resolve the precision on-time seconds epoch using a precision frequency source and counter. We have in fact implemented this using a cesium clock, a GPS timing receiver, and a commercial LORAN-C timing receiver. Each of these devices furnishes a precise pps signal; however, they are frightfully expensive and not likely to find wide acceptance in a Highball network.

We have experimented with the construction of a relatively inexpensive LORAN-C receiver with some success [MIL92a]. It uses a single printed-circuit board containing both the analog and digital components and plugs directly into an IBM-compatible computer. With a good oven-controlled crystal oscillator this equipment produces pps signal with a precision of a few hundred nanoseconds and an oscillator stability of a few parts in 10^10, which is comparable to a GPS timing receiver and commercial LORAN-C timing receiver, but costs far less. As time and priorities permit, we intend to work on a next-generation receiver using surface-mount technology implemented as a drop-in Sbus peripheral for a Sun workstation.

4.5.2. Kernel Modifications for Improved Performance

Our more recent timekeeping improvements require some surgery of the Unix kernel. All of these require only dropping in a couple of modified binary routines, editing configuration files and recompiling the kernel. These modifications have been successfully implemented in the SUnOS 4.1.x, Ultrix 4.x, DEC OSF/1 V3, and HP-UX 9.x kernels; however, with information detailed in [MIL93], similar surgery on other BSD-derived Unix kernels is considered not difficult. Central to the success of this work were hardware grants from Sun Microsystems, DEC, HP and Cisco.

4.5.2.1. Improved Precision for Unix Kernels

Most Unix systems can maintain time with a precision limited by the resolution of the timer-interrupt system, usually in the range 1-10 ms. Some systems, including recent Sun Microsystems products (SPARCstation running SunOS 4.1.1 or Solaris 2.2 or later) include hardware registers and operating system support for microsecond resolution. However, certain other systems include the hardware capability, but the stock operating systems do not exploit it. We have modified Unix kernels for the DECstation 5000/240 running Ultrix 4.2 or later and DEC 3000 AXP Alpha running OSF/1 V1.2 or later to provide microsecond resolution for all kernel and application programs using the standard gettimeofday() system call.

The main feature of the kernel modifications is in moving the intricate phase-lock loop from the daemon to the kernel. This provides a much cleaner interface, more accurate timekeeping and frequency correction between daemon updates. In addition, we have incorporated a 1-PPS signal into the kernel functions which provides stability to a degree normally associated only with precision oscillators. These modifications have been submitted to Sun, DEC and HP, who have distributed them to interested customers, including Lawrence Berkeley Laboratories (LBL) and National Institutes of Standards and Technology (NIST) and others. The modifications can be distributed in either source or binary form to customers with the requisite licenses with permission of the licensor.

4.5.2.2. New Operating System Interface

It is generally agreed by the implementors of the three time-synchronization protocols proposed to the standards bodies, Digital Time Synchronization Service (DTSS), Probabilistic Clock Synchronization (PCS) and NTP, that some form of error estimate/correctness interval should be available at the user interface. NTP provides two error statistics, one representing the worst-case error (called synchronization distance) and the other the estimated error (called dispersion). While the detail requirements for an appropriate user interface have only recently been addressed as a work item by ANSI, an interim interface has been designed for NTP. This takes the form of two new Unix kernel operating system calls, one to read the system clock and error quantities (ntp_gettime()) and another to adjust the clock time and frequency (ntp_adjtime()).

The new system interface, described in [MIL93] has been implemented for SunOS 4, Ultrix 4 and DEC OSF/1 kernels and is supported in the current NTP Version 3 implementation. It has been installed on all of our research machines, the DARTnet routers and additional sites. For information about the availability of the Sun version, contact Bob Hinden (hinden@eng.sun.com). For information about the availability of the DEC versions, contact Jeff Mogul (mogul@pa.dec.com).

4.5.2.3. Software Timekeeping Improvements

In order to improve the timekeeping accuracy to the submillisecond regime, a primary synchronization source of equal or better accuracy is required. In addition to the errors introduced by the hardware, which are usually in the order of a millisecond, there can be substantial additional errors occasionally reaching tens of milliseconds in the operating system itself. A recent report [MIL93] discusses the sources of error and presents countermeasures with the conclusion that timekeeping can in general be maintained to the order less than a millisecond on LANs and WANs operating at 1.544-Mbps or faster.

There are two sources of software error, one due to system queueing and signalling jitter and the other due to the inherent limitations of the Unix timekeeping model. We have developed several schemes to reduce software jitter through the use of Unix system priorities, interprocess communication and various types of signalling methods and have in general been able to reduce these errors to well below a millisecond on modern workstations made by Sun, DEC, HP, SGI and others. The reductions have been realized primarily by improved software in the NTP Version 3 daemon and are described in [MIL93].

As explained in [MIL93], additional improvement is possible only with some kernel surgery, which we have performed on stock Sun and DEC kernels. The surgery amounts to moving the NTP phase-lock loop from the daemon to the kernel. This allows the jitter due to incremental, periodic clock phase adjustments to be reduced from the order of 100 usec to the order of 1 usec, or about two orders of magnitude. The improvements also provide clock status information so that, with the new user interface, application programs can learn the status of the timekeeping system.

The improved code has been supplied to Sun and DEC for distribution to qualified customers and has been implemented as an inherent feature of the LINUX operating system for the Intel 386-class machines. Modified kernels are currently deployed in our research machines, DARTnet routers and some test hosts, NIST Boulder and at several sites in Europe. Information about the availability of sources and binaries can be obtained from the above contacts.

4.5.2.4. Interfaces for Special Signals

In general, submillisecond accuracies cannot be achieved through use only of the ASCII timecode usually used in conjunction with a serial port. We have investigated several methods in which precision signals generated by some radio clocks can be used to augment the ASCII timecode. There are two kinds of signals that can be used for this purpose, a one-pulse-per-second (PPS) signal generated by most radio clocks and precision timekeeping equipment, such as the cesium clock in our laboratory, and the other the IRIG-B signal generated by some radio clocks and synchronization equipment.

In the case of the PPS signal, we use a software feature designed by Van Jacobsen at LBL which captures a timestamp at the transition of a control lead of the serial port used for the ASCII timecode. This reduces jitter at the hardware interrupt level relative to the PPS signal to the order of 6 usec (estimated) on a Sun IPC. The feature is supported in the NTP Version 3 daemon in conjunction with the ASCII timecode to resolve the seconds numbering. Jitter and incidental timing noise are reduced to the order of 20 usec using a trimmed-mean filter. The PPS timestamp feature is implemented as a streams driver for the SunOS 4.1.1 kernel, but has not been tested with later SunOS versions. The software is included in the NTP Version 3 distribution along with a fast microtime() routine that reduces the running time to capture a timestamp from 42-85 usec to 3 usec.

In the case of the IRIG-B signal, we have obtained a commercial interface (KSI/Odetics TPRO IRIG-B interface for the SBus) and implemented an NTP driver for it. Using this approach the NTP daemon disciplines the system clock to the interface as if it were a radio clock, but ordinary applications still read the stock kernel system clock. While this provides performance comparable to the PPS signal, it does not take full advantage of the precision oscillator which is a feature of the interface.

Perhaps the most interesting approach we have taken is interfacing the IRIG-B signal by means of the audio codec that is a feature of many modern workstations. We have modified an audio driver for the Sun workstation to include facilities to demodulate the IRIG-B signal and extract a precision timestamp at the designated zero crossing to within 125 usec, which is the limitation imposed by the codec sample clock. The modified driver is supported by the NTP Version 3 implementation and in use in several of our research machines. Work is under way to port these modifications to our DECstation 5000/240 and 3000 AXP Alpha workstations using the DECaudio lofi TurboChannel audio device as well. The new audio driver is available by anonymous FTP from louie.udel.edu in the directory pub/ntp as the compressed tar archive bsd_audio.tar.Z.

4.5.3. Reducing Frequency Wander of the System Clock

Systematic assault on various identified sources of jitter have essentially eliminated this as a factor in determining timekeeping accuracy. What remains, according to [MIL93], is the inherent instability of the local clock oscillator itself, usually a garden-variety uncompensated quartz oscillator. Measurements on typical workstations suggest a frequency wander of a couple of parts-per-million (PPM) can be expected due to normal room temperature variations. while the frequency discipline imposed by the ordinary NTP operation can compensate for a constant frequency offset, the agility in following temperature-induced frequency wander is limited. In typical cases temperature fluctuations can contribute a millisecond or more in timing error, unless compensated by some means.

4.5.3.1. PPS Signal Discipline

Analysis suggests two approaches to minimizing wander. In one the time constant of the NTP phase-lock loop is reduced so that the agility of the loop to follow rapid frequency changes is improved. The problem with this is that it requires a corresponding reduction in the poll interval, which could overload the network. The other involves a different kind of loop suggested by experience with the NIST Automated Computer Time Service (ACTS) designed by Judah Levine and collaborators at NIST Boulder. This design is based on a frequency-lock loop in which the engineering parameters are determined from an analytic process involving the first-order time differences recorded at intervals varying from tens of seconds to hundreds of hours. These data are used to determine the Allan-deviation characteristic over this range of intervals and from this characteristic the optimum poll interval and averaging time.

While in principal the Allan-deviation characteristic is unique to each individual oscillator, we have found the differences in characteristics to be unimportant for our purposes and have adopted a generic approach. Experimental loop designs were evaluated using two special-purpose NTP simulators designed to test and evaluate the NTP algorithms and clock-discipline code in both the daemon and kernel implementations. While the redesigned loop did considerably improve the frequency adaptation, the phase noise was high and the implementation requires floating point arithmetic, which is considered quite rude in a kernel implementation.

However, another approach has proven quite useful. In those cases where a PPS or IRIG signal is available, it is possible to discipline the frequency of an ordinary quartz oscillator using that signal. In this scheme a sample of the hardware microsecond counter is captured at each PPS transition and the difference between two samples computed at about 30-second intervals. From this the intrinsic local clock offset frequency is computed and used to adjust the frequency estimate ordinarily calculated by NTP. In our kernel implementation a 3-stage median filter is used to suppress incidental jitter and noise at the PPS input. On the assumption that NTP updates are determined from a source of high stability, like a GPS receiver or cesium clock, this scheme reduces residual wander by a factor equal to the interval between samples, in this case a factor of 30.

4.5.3.2. External Hardware Clock

Notwithstanding the various schemes to reduce incidental errors due to CPU oscillator wander, the best solution would be to replace the CPU oscillator with a precision unit. However, modifying the actual motherboard circuitry is not considered an option; therefore, some sort of bus peripheral is required along with kernel modifications to use it instead of the CPU oscillator. We have taken two different approaches to this, one involving a commercial SBus peripheral and the other an SBus peripheral designed and constructed in our laboratory.

The IRIG-B SBus interface mentioned previously includes a medium quality temperature-compensated oscillator with a rated stability of 0.2 PPM. We have designed and tested a Unix kernel driver for this interface and can use it instead of the original Unix clock as the system clock. In this design the Unix system calls to read the system clock (gettimeofday(), ntp_gettime()) return the IRIG-B time directly from the interface board. There is no provision to set the IRIG-B time or to adjust its rate; therefore, system calls ordinarily used to set and adjust the system clock (settimeofday(), adjtime()) and ntp_adjtime()) are inoperable as long as the IRIG-B signal is present, the interface has synchronized to the signal and the IRIG-B data are correctly formatted. If not, or if the IRIG-B clock is more than 1000 s from the Unix clock, the interface is disabled and system time reverts to the Unix clock.

The laboratory interface board was designed for our Highball project. It includes a pair of large Xilinx gate arrays, a precision oscillator and a pair of TAXI chips with optical transceivers. For the purpose of time synchronization, the gate arrays implement a 32-bit seconds counter and 18-bit microseconds counter in Unix timeval format, together with a 16-bit latch and programmable prescaler driven by the precision oscillator. The oscillator itself is oven compensated with a rated stability of .005 PPM. We have implemented and are now testing a Unix kernel driver for this board so that it can function as the system clock in a manner similar to the IRIG-B SBus interface.

In this approach the settimeofday() Unix system call is used to set an initial offset for the hardware counters, which then run at the intrinsic rate of the precision oscillator. Like every other, this oscillator must be disciplined relative to UTC as disseminated. For instance, at the moment the precision oscillator frequency error is about 0.16 PPM, which results in a time error of about 14 ms per day. Thus, the ordinary NTP phase-lock loop continues to provide incremental phase and frequency corrections relative to a selected radio clock or NTP peer selected as the synchronization source, so that the 0.16 PPM is cancelled out. In the present experiment setup involving a Sun IPC synchronized via NTP to a GPS-synchronized primary server, the residual time error or jitter is a few tens of microseconds and frequency error a few parts in 10^8. This results in a time uncertainty of about 400 usec ms after a whole day during which contact with all synchronization sources has been lost.

Both hardware approaches to provide a precision system clock require some provision for the original Unix clock, which is used to implement programmed delays for various housekeeping functions. If fact, both external-clock approaches involve two disciplined oscillators, one incorporated in the Unix clock, the other in the external clock. However, there is only one phase-lock loop (PLL) implemented in the kernel. In the case of the IRIG-B interface, the PLL is used to discipline the Unix clock in the usual way. If the IRIG signal is lost or the interface become defective, the synchronization source automatically devolves to the radio timecode or an NTP peer as configured. In the case of the laboratory interface, the PLL is used to discipline the counter registers. A simplified PLL is used in which corrections are calculated every minute or so and used directly to adjust the Unix clock. In neither case is the Unix clock discipline visible by applications which request system time using the ordinary kernel system calls.

4.5.3.3. Future Improvements for NTP Version 4

As part of the continuing development of timekeeping technology, we have proposed several changes to the basic NTP protocol and algorithms. The proposals, summarized in [MIL94e], include several improvements to existing algorithms, together with new features designed to support multicast modes of operation, fully distributed peer subnets and automated NTP peer discovery and subnet configuration.

4.6. Hardware Development

The Highball node hardware consists of the crossbar switch, switch controller, control transceiver and node controller. It supports some number of local supercomputers or workstations which are directly connected to the crossbar switch. In the prototype design described in [MIL90c], the switch controller is a bus peripheral of the node controller and data are transferred between the application hosts and the crossbar switch using coaxial or ribbon cable and HPPI or similar protocol. Input/output transfers are enabled by the switch controller, using a dedicated wire or timeslot in the cable, but the timing and formatting of the data burst itself is the responsibility of the application hosts and their data-transceiver peripherals. There is no intrinsic requirement to operate at the full network data rate, only an agreement to clock the transfers to and from the network and to agree with the receiving end on protocol and data rate.

All node controllers contain at least one transceiver, which provides the control-channel data path via the network links to other nodes, and the switch controller itself. In the prototype design, the crossbar switch is contained in a separate enclosure and the node controller is a Sun Microsystems SPARCstation. The SPARCstation includes the SBus, a fast backplane interconnect that operates at burst rates up to 800 Mbps. This workstation serves multiple purposes as node controller, development platform, and application-testing environment. The switch controller and transceiver are programmable SBus devices implemented using both multilayer PC board and wire-wrap technologies.

The basic switching fabric is constructed of an *N*x*N* asynchronous crosspoint switch in which each of the outputs can be connected to any one of the inputs. In the prototype design, a 32x32x1 chip made by AMCC was proposed. This device includes input and output latches, a set of configuration registers and control circuitry. It is rated at a 800-Mbps data rate and can be reconfigured in several nanoseconds. For the prototype Highball network, this device can be operated in a 4x4x8 configuration for convenient implementation of a 100-Mbps byte switch.

The crossbar switch configuration registers are loaded directly by the switch controller, which includes a slot counter used to time the configuration dwell time, together with a FIFO to buffer the configuration vectors. A master clock, consisting of a 5-MHz temperature-compensated quartz oscillator, controls all switch controller functions, except the transfer of data within a burst. In order to maintain overall network synchronization, it is necessary that the master clock frequency be controlled over a small range by the host processor. This is done by using a voltage-controlled oscillator and digital/analog converter.

The design is such that a given switch configuration can be held during a single dwell while the next configuration vector is shifted into the switch configuration registers. The dwell time is controlled by the slot counter. Upon reaching zero the counter generates a pulse which loads the configuration registers and requests the next configuration vector to be loaded into the buffer. The node processor transfers the switch schedule computed by the scheduling algorithm to the onboard FIFO. Control bits which are part of each entry determine the transmit, receive, and interrupt bits which are interpreted by the switch controller.

Three prototype switch controllers have been fabricated and tested, each of greater capability and complexity. The latest design uses both surface mount and through-hole components on a single-size SBus board to implement the various functions of the switch controller. This device contains all the features necessary to implement a bidirectional optical link between a host and node controller. It includes two Xilinx gate arrays, a set of optical transceivers, a set of TAXI encoders with FIFO buffers, and a temperature compensated quartz crystal oscillator.

In this implementation, designed and built by graduate student Donald Nelson, received and transmitted data frames generate events that are prioritized and associated with a timestamp that indicates their precise arrival time at the transceiver. An onboard FIFO holds the data and timestamps until the host is able to read the information. One of the Xilinx gate arrays controls the SBus interface, while the other controls the remaining functions. The program that drives the programmable gate arrays is downloaded from the host.

Two interface circuit boards of the latest design have been fabricated and tested. Most of the parts are on hand to fabricate two additional boards when the need arises. The boards are completely self contained and occupy one SBus slot in the host computer. The boards have been programmed and tested using an implementation of a programmable, precision clock for use with modified kernels and NTP as describedin [MIL94d]. Graduate student Tyrone Thompson is developing firmware to support a bidirectional data transport using the TAXI chips, FIFO buffers and optical transceivers as a project for his Masters Thesis.

4.7. Software Development

The Highball node software design presents several challenges due to real-time constraints and delicate synchronization requirements. The design utilizes a multiple-process, message-passing, object-oriented paradigm. The need for modularity and flexibility suggested a design consisting of several modules organized into six functions. Four of these operate on reservation requests, one maintains time synchronization, and one handles the frame interrupts.

At the start of each frame the software reads in any timestamp and reservation request information that arrived since the previous frame. The program sorts these incoming requests and incorporates them into a reservation list ordered according to the time each request becomes valid. Following this, the program calls the scheduling routine, which schedules valid requests in the order they appear on the reservation list. The schedule consists of configurations for the local crossbar switch and corresponding dwell times for each configuration.

Another routine processes reservation requests that come from the local node controller, queueing them for distribution to other nodes. This information is transmitted along with timekeeping information during the node's next scheduled "hello burst." The timekeeping routine processes the received timestamps to develop the node delay/offset matrix.

We have tried to make the software simple and straightforward, while still allowing future changes to be made easily. The Highball program-development methodology makes extensive use of C++ and the AT&T C++ Task Library. Each task runs a bounded number of short iterations over some procedure and suspends itself until the next frame. Each of the next waiting tasks runs until all finish executing. We carefully bound the maximum execution time of each task so that all finish before the interrupt signalling the beginning of the next frame arrives.

The main advantage of using the Task Library is that each function is separate, yet all the execution, scheduling, and message passing is contained in one UNIX process. According to the developers of the library, switching between tasks is up to 10 times faster than switching between processes and message passing is more reliable. The Task Library also allows us to test our routines in an environment that simulates real-time execution. Therefore, we have built-in simulation capability that allows us to check if our routines meet their real-time requirements, and we do not have to change our code or environment to run them in real-time.

We have reusable C++ classes for developing drivers for SBus devices as well as for other applications involving the DAWN SBus development kit. In the process of debugging and testing the two Highball boards, we developed drivers that have been incorporated into C++ classes. This creates an easy and efficient interface to the boards. As an example, for the transceiver we have a class member function 'ReadFifo()' that returns the contents of the onboard FIFO. The user does not need to know the address of the FIFO registers or any other lower level details. Both of the boards have a programmer's guide that describes the function of the boards, the onboard registers that are accessible to the programmer and the C++ classes used that contain the driver code.

The Timestamp Generator driver waits for an SBus interrupt from the board to indicate that new events have occurred. The interrupt handler sends a SIGIO signal to the routine that reads the FIFO. Once the FIFO is empty, the driver awaits the next interrupt from the Timestamp Generator board. This driver uses asynchronous notification so that all reads are nonblocking and to avoid polling the FIFO's empty flag. The two main functions of the Switch Controller's driver are to write configuration vectors and dwell times to the board's FIFO and to control the VCXO by writing values to the DAC. The Switch Controller also generates signals that are sent to other machines to cause events to be recorded and timestamped there. The Switch Controller board generates no interrupts. For speed, the member functions that write to or read from hardware registers directly are coded as inline functions; that is, there is no function call or context switch when these functions are called.

We also designed diagnostic programs to test and debug the boards. We wrote three levels of test software: one that tests individual components on a board, another that tests if the components on a board worked together correctly, and a third that tests communication between boards.

5. Future Work

There are three distinct areas in which further work can be directed. The first area is in algorithmic improvements to the reservation, scheduling, and synchronization algorithms. This includes improving the speed of the scheduling algorithm, reducing delays in the reservation algorithm, and reducing the vulnerability to disruptions due to errors and node crashes. The second area is to implement the remaining components to achieve a working Highball demonstration with host-to-host data traffic. The third area is the application of Highball technology to advanced communication systems, in particular, the NASA/DARPA Advanced Communication Technology Satellite. Efforts in each of these areas are discussed below.

While the various scheduling algorithms we have developed are fast, they may not be fast enough, about 500 R-TDMA reservations or 500 AC-TDMA rate adjustments per second are needed to keep up with the larger or faster networks. Most of the time taken by these algorithms is in managing the data structures. We believe the algorithms can be improved by a factor of 5 to 10 by algorithmic adjustments and coding improvements in the data structures. In addition, we need to continue to study the adaptation problem in AC-TDMA. AC-TDMA has many advantages compared to R-TDMA. The computational cost is lower, the response time might be faster, and the robustness is better. One especially interesting application of AC-TDMA is to synchronize the rotation period to a (high definition) video frame rate, e.g. 16.67ms. Then video users can communicate synchronously with the video stream.

There is an inherent delay penalty in the Highball technology due to the time necessary to propagate R-TDMA reservations or AC-TDMA rate adjustments. In the present technology every node must wait until the reservation or adjustment has been received at all nodes. However, in geographically large networks, nodes tend to be clustered in relatively small areas. In such cases it should be possible to relax this requirement if both communicating endpoints are in the same area. We believe the field of Parallel Discrete Event Simulation (PDES) may provide a useful paradigm to study this problem. A PDES system uses real or logical clocks and a message-passing paradigm to support parallel execution of a distributed algorithm, so is very similar to the Highball scenario.

Much of our current work centers on verifying robustness and on recovering from disruptions that may occur due to transmission errors on the control channel or a node crash. Simulations with deliberately introduced errors show that the network usually resynchronizes state quickly after a disruption; however, the detailed behavior under various fault regimes has not been thoroughly explored. The methodology of PDES represents a point of departure for this investigation.

A particular case of interest in the application of Highball technology is the ACTS system. This system involves a geosynchronous satellite capable of 622 Mbps delivered to the users via three beams and a satellite-switched TDMA access mechanism. We have proposed a network of Highball nodes including a high speed HPPI/SONET switch and SPARCstation node controller. The existing Highball algorithms are directly transferrable for this purpose, as well as the DARTnet experiment-support software.

6. Technology Transfer

While the Highball Network postulates a specific set of hardware (point to point links connected by crossbar switches), much of the scheduling, routing, and robustness work will apply to tomorrow's multi-gigabit, all optical networks. The Highball paradigm of sending requests on a control channel and allocating network resources to particular communications transfers maps directly to many recently proposed wavelength-division multiplexed networks. The scheduling and routing technology apply directly. Another area of transferral is finding multicast routes and schedules. Multicasting is a special case of scheduling Highball's in-band control channel. We have studied this problem in the Highball context and are examining how to apply this knowledge to other networks.

We believe that the Highball technology may be most appropriate for future video transfer networks. Given the right switch hardware, time- compressed analog video can be transmitted just as easily as digital video. Highball's ability to synchronize the communications availability to the timing of video source make it ideal for transmitting isochronous traffic such as video. In effect, Highball could genlock computers and video workstations nationwide.

Our results in timekeeping apply directly and immediately to anyone wishing to synchronize computers over a large geographic area. They will allow researchers to more easily study synchrony and concurrency and to time geographically diverse events. Better timekeeping means better video distribution, easier coupling of supercomputers, and more accurate delay experiments, to name but a few applications.

7. Other Activities

It may be of some practical interest to briefly summarize some of our other activities in pursuit of precision time. We do admit to a mild degree of fanaticism in keeping our sometimes exotic time machinery precisely in tune.

We have amassed a considerable kit of timekeeping gadgetry, much of it donated or provided on indefinite loan, including a cesium clock, two GPS receivers, two WWVB receivers, three WWV/WWVH receivers, a CHU receiver and three LORAN-C receivers, one of which was designed and constructed in our laboratory. The configuration of this equipment and its numerous antennas, signal sources and computer interfaces is a complex operation in its own right, as described in detail in [MIL93].

Our master clock is an Austron 2201A GPS receiver with an integrated LORAN-C receiver (to help de-wiggle the intentional degradation inserted at the source). This device includes a number of options providing precision time generation and comparison with the cesium clock. All operations of this receiver and, for that matter, most of the other radio clocks as well, are intricately monitored on a continuous basis using facilities built into the NTP Version 3 implementation. The voluminous data collected are processed automatically once each day and comprehensive summaries generated using awk and shell scripts. The summaries demonstrate that the master clock is accurate to within about 50 ns RMS relative to UTC, which is not too shabby for even the most demanding applications. This has been confirmed using a portable clock initially synchronized to UTC at the U.S. Naval Observatory. The cesium clock has been a useful aid in keeping our menagerie of tickers in tune and to alarm various failures that have occurred.

The GPS receiver ASCII timecode, as well as PPS and IRIG-B signals, are bussed to three NTP primary servers, one dedicated for our research network, another to DARTnet and a third for public access. On our public primary server we use a hot-standby backup WWVB receiver in case the GPS receiver fails (which it has on several occasions). We also have an integrated GPS receiver/NTP server made by Bancomm for public access, but this device has only limited capability for monitoring and performance evaluation. In addition to the designated primary servers, we have a number of other primary and secondary NTP time servers used for testing and program development.

We have been trying for well over a year to identify the source of interference to WWVB radio reception in this area. It has been astonishingly hard to find the culprit, even with the combined resources of the local power company, FCC monitoring stations and other NTP sites with the same Spectracom radio. At one time the FCC monitoring stations in Maryland and Maine concluded the culprit was a powerful emitter in Quebec, but this has since been discredited. Since the problem occurs both at our campus laboratory in downtown Newark, DE, and at a suburban spot 2.3 km from town, but not at the place of manufacture in Rochester, NY, the emitter was thought not to be conductive interference from power or cable systems.

In cooperation with Spectracom, we have collected numerous internal statistics in our radios and then sent the radios back to Spectracom for similar measurements. However, we found the radios work well in Rochester, NY, but not in Newark, DE. In our latest adventure, we have obtained a communications radio that covers the 60-kHz signal frequency, built a preamplifier and borrowed a special antenna from Spectracom, then wandered the roads and byways of the local area sniffing for interference. We quickly found the source of at least some of the interference in the form of a touch-sensitive lamp (!) in one location, but the problem persists. After scouring the countryside, we found the source to be the local power grid itself. An antenna some distance from the nearest power line shows no evidence of the interference, while an antenna located within a few hundred yards of any power line pegs the meter. We are exploring with the power company how this can happen.

We have been supporting a set of experiments for Judah Levine at NIST Division of Time and Frequency at Boulder, CO. The experiments involve the ACTS and one of our hosts equipped with a conventional telephone modem. The results, reported in a paper submitted for publication, show that ACTS can achieve an accuracy comparable to NTP operating with an attached radio clock. As the results of these experiments, we made substantial modifications in the NTP algorithms which considerably improved the accuracy and stability when relatively long intervals are necessary between updates [MIL94e].

8. References and Bibligraphy Prior to 1990

Note: Many of these documents can be obtained via anonymous FTP from louie.udel.edu in the pub/ntp/doc directory. Most of these are in PostScript format, although a few are in cheapskate ASCII. An index with brief description of each document and file name of the distribution is in the README file in the pub/ntp directory.

[MIL81] Mills, D.L. Internetworking and the Atlantic SATNET. *Proc. National Electronics Conference* (Chicago, Illinois, October 1981), 378-383.

[MIL83a] Mills, D.L. Internet delay experiments. DARPA Network Working Group Report RFC-889, M/A-COM Linkabit, December 1983.

[MIL83b] Mills, D.L. DCN local-network protocols. DARPA Network Working Group Report RFC-891, M/A-COM Linkabit, December 1983.

[MIL85a] Mills, D.L. Algorithms for synchronizing network clocks. DARPA Network Working Group Report RFC-957, M/A-COM Linkabit, September 1985.

[MIL85b] Mills, D.L. Experiments in network clock synchronization. DARPA Network Working Group Report RFC-956, M/A-COM Linkabit, September 1985.

[MIL85c] Mills, D.L. Network Time Protocol (NTP). DARPA Network Working Group Report RFC-958, M/A-COM Linkabit, September 1985.

[MIL87] Mills, D.L., and H.-W. Braun. The NSFNET Backbone Network. *Proc. ACM SIGCOMM 87 Symposium* (Stoweflake Vermont, August 1987), 191-196.

[MIL88a] Mills, D.L. Network Time Protocol (Version 1) specification and implementation. DARPA Network Working Group Report RFC-1059, University of Delaware, July 1988. Also published as: Electrical Engineering Department Report 88-4-1, University of Delaware, May 1988.

[MIL88b] Mills, D.L. The Fuzzball. *Proc. ACM SIGCOMM 88 Symposium* (Palo Alto, California, August 1988), 115-122.

[MIL89a] Mills, D.L. Internet time synchronization: the Network Time Protocol. DARPA Network Working Group Report RFC-1129, University of Delaware, October 1989. Also published as: Electrical Engineering Department Report 89-9-1, University of Delaware, September 1989.

[MIL89b] Mills, D.L. Measured performance of the Network Time Protocol in the Internet system. DARPA Network Working Group Report RFC-1128, University of Delaware, October 1989. Also published as: Electrical Engineering Department Report 89-9-3, University of Delaware, September 1989.

[MIL89c] Mills, D.L. Network Time Protocol (Version 2) specification and implementation. DARPA Network Working Group Report RFC-1119, University of Delaware, September 1989. Also published as: Electrical Engineering Department Report 89-9-2, University of Delaware, September 1989.

9. Bibliography of Publications Supported by this Contract

9.1. Publications (with abstracts) in 1990

[MIL90a] 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.

This paper describes a series of experiments involving over 100,000 hosts of the Internet system and located in the US, Europe and the Pacific. The experiments are designed to evaluate the availability, accuracy and reliability of international standard time distribution using the Internet and the Network Time Protocol (NTP), which has been designated an Internet Standard protocol. NTP is designed specifically for use in a large, diverse internet system operating at speeds from mundane to lightwave. In NTP a distributed subnet of time servers operating in a self-organizing, hierarchical, master-slave configuration exchange precision timestamps in order to synchronize host clocks to each other and national time standards via wire or radio.

The experiments are designed to locate Internet hosts and gateways that provide time by one of three time distribution protocols and evaluate the accuracy of their indications. For those hosts that support NTP, the experiments determine the distribution of errors and other statistics over paths spanning major portions of the globe. Finally, the experiments evaluate the accuracy and reliability of precision timekeeping using NTP and typical Internet paths involving ARPANET, NSFnet and regional networks. The experiments demonstrate that timekeeping throughout most portions of the Internet can be maintained to an accuracy of a few tens of milliseconds and a stability of a few milliseconds per day, even in cases of failure or disruption of clocks, time servers or networks.

[MIL90b] Mills, D.L. Network Time Protocol (Version 3) specification, implementation and analysis. DARPA Network Working Group Report RFC-1305, University of Delaware, March 1992, 113 pp. Previous draft published as: Electrical Engineering Department Report 90-6-1, University of Delaware, June 1990.

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

[MIL90c] Mills, D.L., C.G. Boncelet, J.G. Elias, P.A. Schragger and A.W. Jackson. Highball: a High Speed, Reserved-Access, Wide-Area Network. Electrical Engineering Department Report 90-9-3, University of Delaware, September 1990.

This document describes a network architecture called Highball and a preliminary design for a prototype, wide-area data network designed to operate at speeds of 1 Gbps and beyond. It is intended for applications requiring high speed burst transmissions where some latency between requesting a transmission and granting the request can be anticipated and tolerated. Examples include real-time video and disk-disk transfers, national filestore access, remote sensing and similar applications. The network nodes include an intelligent crossbar switch, but have no buffering capabilities; thus, data must be queued at the end nodes. There are no restrictions on the network topology, link speeds or end-end protocols. The end systems, nodes and links can operate at any speed up to the limits imposed by the physical facilities.

This document presents an overview of an initial design approach and is intended as a benchmark upon which a detailed design can be developed. It describes the network architecture and proposed access protocols, as well as functional descriptions of the hardware and software components that could be used in a prototype implementation. It concludes with a discussion of additional issues to be resolve in continuing stages of this project.

[DAV90] Davis, M., K. Monington, E. Perkins. 8 x 8 scheduled crossbar switch. Electrical Engineering Department Report 90-10-1, University of Delaware, October 1990.

This report describes the design and testing of an 8 x 8 VLSI crossbar switch which operates at 25 MHz.

9.2. Publications (with abstracts) in 1991

[MIL91] Mills, D.L. Internet time synchronization: the Network Time Protocol. IEEE Trans. Communications 39, 10 (October 1991), 1482-1493.

This paper describes the Network Time Protocol (NTP), which is designed to distribute time information in a large, diverse internet system operating at speeds from mundane to lightwave. It uses a symmetric architecture in which a distributed subnet of time servers operating in a self-organizing, hierarchical configuration synchronizes local clocks within the subnet and to national time standards via wire, radio or calibrated atomic clock. The servers can also redistribute time information within a network via local routing algorithms and time daemons.

This paper also discusses the architecture, protocol and algorithms, which were developed over several years of implementation refinement and resulted in the designation of NTP as an Internet Standard protocol. The NTP synchronization system, which has been in regular operation in the Internet for the last several years, is described along with performance data which shows that timekeeping accuracy throughout most portions of the Internet can be ordinarily maintained to within a few milliseconds, even in cases of failure or disruption of clocks, time servers or networks.

[SCH91] Schragger, P.A. Scheduling algorithms for burst reservations on wide area high speed networks. Proc. IEEE Infocom 91 Symposium (Miami Florida, April 1991), 589-596.

This paper describes two scheduling algorithms for the Highball network, one called Trailblazer, the other Pathfinder. The Trailblazer algorithm is based on a strategically optimized heuristic and an exhaustive enumeration of all paths, while the Pathfinder algorithm is based on optimal schedule increment with a new request without preempting existing paths.

[MIL91] 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.

This paper summarizes issues in computer network timekeeping with respect to the Network Time Protocol, which is used to synchronize time in many of the hosts and gateways of the Internet. It describes the methods used to coordinate and disseminate international time services and how they are incorporated into NTP time servers. It discusses the hazards on reckoning NTP dates with the conventional civil calendar and a standard method for numbering the days. Finally, the paper describes the NTP timescale, its mapping to conventional civil time and date and its interpretation of leap seconds.

9.3. Publications (with abstracts) in 1992

[BON92a] Boncelet, C.G., and D.L. Mills. A labeling algorithm for just- in-time scheduling in TDMA networks. Proc. ACM SIGCOMM 92 Symposium (Baltimore Maryland, September 1992).

This paper describes a scheduling algorithm for high speed, TDMA communication networks. The problem is to find a route and determine switch schedules so that a burst of information can traverse the network from source to destination. We show that this problem is difficult, i.e., that several simple variations are NP- complete. We present an algorithm that finds schedules, and within a simplifying assumption, is guaranteed to find a route if one exists. Simulations indicate that this algorithm runs quickly in practice and overall scheduling efficiencies of 80-90% can be achieved. We also describe a variation in which the problem is to find a repeating schedule and we show how to adapt the basic algorithm for this variation.

[BON92b] Boncelet, C.G., Jr. On multi-access schemes for high speed broadcast channels with erasures. *Proc. of the 1992 Conference on Information Science and Systems* (March 1992).We present early results from a study of multiple access in high speed communication networks with erasures. The application arises in all-optical networks. In this paper, we assume the topology is a broadcast star. We assume that transmission is on a packet basis and that nodes can time synchronize to packet times. Time is slotted and groups of slots form frames. In each frame an active user tries to send one message. Each active user will (possibly) send his message many times per frame in hopes that at least one gets through. We consider two schemes: random and deterministic. In random transmission, each user randomly selects slots in which to communicate; in deterministic transmission, each user has a predetermined slot pattern.@INDENT HEAD = [MIL92a]

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.

This report describes the design and construction of a specialized radio timing receiver for the LORAN-C radionavigation system. The computer-controlled receiver provides a precision frequency source of 5 MHz and submultiples, together with a precision timing source of 1-pulse/second synchronized to Coordinated Universal Time UTC(LORAN). It is intended as a laboratory monitor and comparator for precision timekeeping equipment, but also has applications as a precision clock and timing source for computer networks and in other applications requiring precision timekeeping with respect to UTC.

[MIL92b] Mills, D.L. Network Time Protocol (Version 3) specification, implementation and analysis. DARPA Network Working Group Report RFC- 1305, University of Delaware, March 1992, 113 pp.

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

[MIL92c] Mills, D.L. Modelling and analysis of computer network clocks. Electrical Engineering Department Report 92-5-2, University of Delaware, May 1992, 29 pp.

This report describes analytical models and design methodologies for computer clocks synchronized to other clocks in a computer network or internetwork. This requires each clock to function as a disciplined oscillator which can be adjusted in both time and frequency to compensate for the intrinsic errors of the onboard frequency source. It also requires the use of a time- synchronization protocol such as the Network Time Protocol, summarized briefly in this report, or the Digital Time Synchronization Service, described elsewhere.

The goal in this effort is to improve the synchronization accuracy from the tens of milliseconds regime of present timekeeping models to the submillisecond regime. In this report analytical models are used to express the accuracy and stability of a computer clock and to establish its design parameters with respect to time and frequency error tolerances. These models are based on the theory of adaptive-parameter, phase-lock loops. Included is a discussion on methods to combine the timing information received from a number of synchronized peer clocks to improve the accuracy and reliability of the local clock. Finally, a detailed analysis of the maximum and expected errors accumulated from the primary source of time throughout the synchronization subnet is presented.

[MIL92d] Mills, D.L. Simple Network Time Protocol (SNTP). DARPA Network Working Group Report RFC-1361, University of Delaware, August 1992, 10 pp.

This memorandum describes the Simple Network Time Protocol (SNTP), which is an adaptation of the Network Time Protocol (NTP) used to synchronize computer clocks in the Internet. SNTP can be used when the ultimate performance of the full NTP implementation described in RFC-1305 is not needed or justified. It involves no change to the current or previous NTP specification versions or known implementations, but rather a clarification of certain design features of NTP which allow operation in a simple, stateless RPC mode with accuracy and reliability expectations similar to the UDP/TIME protocol described in RFC-868.

9.4. Publications (with abstracts) in 1993

[JAC93]

Jackson, A.W. Periodic multicast scheduling in reservation TDMA networks. PhD dissertation, 1993.

[MIL93] Mills, D.L. Precision synchronization of computer network clocks. Electrical Engineering Department Report 93-11-1, University of Delaware, November 1993, 66 pp.

This report 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 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 report concludes that submillisecond accuracies are indeed practical, but that further improvements will be possible only through the use of temperature-compensated clock oscillators.

[SCH93] Schragger, P.A. The job switching with no intermediate queues scheduling problem. PhD dissertation, 1993.

9.5. Publications (with abstracts) in 1994

[BON94] Boncelet, C.G., Jr, and D. Mills. Labeling Algorithms for Just-in-Time scheduling in TDMA networks. Submitted to: *IEEE/ACM Transactions on Networking*, 1994.We describe scheduling algorithms for high speed, TDMA communication networks. The problem is to find a route and determine switch schedules so that a burst of information can traverse the network from source to destination. We show that this problem is difficult, i.e., that several simple variations are NP-complete. We present an algorithm that finds schedules, and within a simplifying assumption, is guaranteed to find a route if one exists. Simulations indicate that this algorithm runs quickly in practice and overall scheduling efficiencies of 80-90% can be achieved. We present optimizations for sending messages from one source to many destinations, e.g., multicasting. We also describe a variation in which the problem is to find a repeating schedule and we show how to adapt the basic algorithm for this variation. This new algorithm is no longer guaranteed to find a solution, but is adequate in practice.@INDENT HEAD = [MIL94a]

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

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.

[MIL94b]Mills, D.L. A kernel model for precision timekeeping. ARPA Network Working Group Report RFC-1589, University of Delaware, March 1994. 31 pp.

This memorandum describes an engineering model which implements a precision time-of-day function for a generic operating system. The model is based on the principles of disciplined oscillators and phase-lock loops (PLL) often found in the engineering literature. It has been implemented in the Unix kernel for several workstations, including those made by Sun Microsystems and Digital Equipment. The model changes the way the system clock is adjusted in time and frequency, as well as provides mechanisms to discipline its frequency to an external precision timing source. The model incorporates a generic system-call interface for use with the Network Time Protocol (NTP) or similar time synchronization protocol. The NTP Version 3 daemon xntpd operates with this model to provide synchronization limited in principle only by the accuracy and stability of the external timing source.

This memorandum does not obsolete or update any RFC. It does not propose a standard protocol, specification or algorithm. It is intended to provoke comment, refinement and alternative implementations. While a working knowledge of NTP is not required for an understanding of the design principles or implementation of the model, it may be helpful in understanding how the model behaves in a fully functional timekeeping system. The architecture and design of NTP is described in [1] , while the current NTP Version 3 protocol specification is given in RFC-1305 [2] and a subset of the protocol, the Simple Network Time Protocol (SNTP), in RFC-1361 [4].

The model has been implemented in three Unix kernels for Sun Microsystems and Digital Equipment workstations. In addition, for the Digital machines the model provides improved precision to one microsecond (us). Since these specific implementations involve modifications to licensed code, they cannot be provided directly. Inquiries should be directed to the manufacturer's representatives. However, the engineering model for these implementations, including a

[MIL94c] Mills, D.L. Improved algorithms for synchronizing computer network clocks. *Proc. ACM SIGCOMM 94 Symposium* (London UK, September 1994), 317-327.

A later revision of this paper is to appear in *IEEE/ACM Trans. Networking*. See [MIL95b].

[MIL94d] Mills, D.L. Unix kernel modifications for precision time synchronization. Electrical Engineering Department Report 94-10-1, University of Delaware, October 1994, 24 pp. [Major revision and update of RFC-1589]

This report describes a model and programming interface for generic operating system software that manages the system clock and timer functions. The model provides improved accuracy and stability for most computers using the Network Time Protocol (NTP) or similar time synchronization protocol. This report describes the design principles and implementations of the model, while related technical reports discuss the design approach, engineering analysis and performance evaluation of the model as implemented in Unix kernels for modern workstations. The NTP Version 3 daemon *xntpd* operates with these implementations to provide improved accuracy and stability, together with diminished overhead in the operating system and network. In addition, the model supports the use of external timing sources, such as precision pulse-per-second (PPS) signals and the industry standard IRIG timing signals. The NTP daemon automatically detects the presence of the new features and utilizes them when available.

There are three prototype implementations of the model presented in this report, one each for the Sun Microsystems SPARCstation with the SunOS 4.1.x kernel, Digital Equipment DECstation 5000 with the Ultrix 4.x kernel, Digital Equipment 3000 AXP Alpha with the OSF/1 V1.x kernel and Hewlett Packard 9000 with the HP-UX 9.0x kernel. In addition, for the DECstation 5000/240 and 3000 AXP Alpha machines, a special feature provides improved precision to 1 μ s (stock Sun kernels already do provide this precision). Other than improving the system clock accuracy, stability and precision, these implementations do not change the operation of existing Unix system calls which manage the system clock, such as gettimeofday(), settimeofday() and adjtime(); however, if the new features are in use, the operations of gettimeofday() and adjtime() can be controlled instead by new system calls ntp_gettime() and ntp_adjtime() as described below.

A detailed description of the variables and algorithms that operate upon them is given in the hope that similar functionality can be incorporated in Unix kernels for other machines. The algorithms involve only minor changes to the system clock and interval timer routines and include interfaces for application programs to learn the system clock status and certain statistics of the time synchronization process. Detailed installation instructions are given in a specific README files included in the kernel distributions.

In this report, NTP Version 3 and the Unix implementation xntp3 are used as an example application of the new system calls for use by a synchronization daemon. In principle, these system calls can be used by other protocols and implementations as well. Even in cases where the local time is maintained by periodic exchanges of messages at relatively long intervals, such as using the NIST Automated Computer Time Service [LEV89], the ability to precisely adjust the system clock frequency simplifies the synchronization procedures and allows the telephone call frequency to be considerably reduced.

[MIL94e] 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, 52 pp.

This report describes several changes and additions proposed for the Network Time Protocol Version 4. Previous versions of this protocol have been widely deployed in the Internet and a considerable body of experience has accumulated in their use. The proposed new version contains several improvements to existing algorithms, together with new features designed to support multicast modes of operation, fully distributed peer subnets and automated NTP peer discovery and subnet configuration.

This report describes work in progress and is produced in order to stimulate discussion and refinement of the various algorithms. Some of the algorithms, in particular the work on refining the NTP local clock model, has already been implemented and now in test. Other work, including multicast support, is in the prototype phase, but needs further development and consensus on key management and distribution. Other work is still in the formative stage, including the fully distributed modes of operation and peer discovery and subnet configuration.

It is anticipated that further development in these areas will continue while discussion and consensus is reached on the final form of the protocol and its algorithms. When complete, the final design will be submitted for approval by the Internet standards apparatus.

[SHI94] Shin, J., and C.G. Boncelet, Jr. The UDel switch: an approach to very large fast packet switches. submitted to *IEEE/ACM Trans. Networking*, 1994.We propose a new architecture for a very large NxN, N=2^14, 18, self-routing, non-blocking, space-division fast packet switch for ATM B-ISDN. It is based on binary tree expansion, concentration, and combination of neighboring blocks of packets. This is an output queueing switch with the multi-casting capability even though only the point-to-point routing will be considered in this paper.@HEAD LEVEL 2 = Publications (with abstracts) in 1995

[MIL95a] Mills, D.L. Simple Network Time Protocol (SNTP). DARPA Network Working Group Report RFC-1769, University of Delaware, March 1995, 14 pp. [Major revision and update of RFC-1361]

This memorandum describes the Simple Network Time Protocol (SNTP), which is an adaptation of the Network Time Protocol (NTP) used to synchronize computer clocks in the Internet. SNTP can be used when the ultimate performance of the full NTP implementation described in RFC-1305 is not needed or justified. It can operate in both unicast modes (point to point) and broadcast modes (point to multipoint). It can also operate in IP multicast mode where this service is available. SNTP involves no change to the current or previous NTP specification versions or known implementations, but rather a clarification of certain design features of NTP which allow operation in a simple, stateless remote-procedure call (RPC) mode with accuracy and reliability expectations similar to the UDP/TIME protocol described in RFC-868.

This memorandum obsoletes RFC-1361 of the same title. Its purpose is to explain the protocol model for operation in broadcast mode, to provide additional clarification in some places and to correct a few typographical errors. A working knowledge of the NTP Version 3 specification RFC-1305 is not required for an implementation of SNTP. Distribution of this memorandum is unlimited.

[MIL95b] Mills, D.L. Improved algorithms for synchronizing computer network clocks. *IEEE/ACM Trans. Networking* (to apppear). [revision of ACM SIGCOMM 94 paper of the same name]

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 subset from among an ensemble of peer clocks and combine their differences to produce a local 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 algorithm parameters in response to prevailing network conditions, 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 kernel software in order to realize submillisecond accuracies with fast workstations and networks.

[SHI95] Shin, J., and C.G. Boncelet, Jr. Scalability Study of the UDel Swith for ATM B-ISDN Central Office. Submitted to *IEEE/ACM Trans. Networking*, 1995.We recently roposed a new architecture for a very large *NxN*, $N=2^{14}$, self-routing, memoryless, space-division fast packet switch for ATM B-ISDN based on binary expansion and unique spatial arrangement of blocks of packets. In this paper, we consider issues related to extending newly proposed UDel switch architecture in terms of hardware realizability and scalability for *N*x*N* swithches, $N=2^h$, $h= 14$, 15, 16, 17, and 18. The number of optical fiber cable interconnections, I, needed to interconnect 2-tier switch units increases linearly with the number of the input/output ports (*N*) so $I=aN$. The constant, *a*, varies from less than 2 for $N=2^{14}$ to 4.5 for $N=2^{18}$.