# Computer Systems Laboratory Research Review

*Joseph Pasquale*
*George Polyzos*

Computer Systems Laboratory
Department of Computer Science and Engineering
University of California, San Diego
La Jolla, CA 92093–0114

April 1993

UCSD Technical Report CS93-284

*Abstract*

At the Computer Systems Laboratory at UCSD, we conduct research in operating systems, communication networks, performance evaluation, and distributed systems. This document reviews the current research within the Computer Systems Laboratory. We describe our involvement in the Sequoia 2000 Project, and we describe our work in the areas listed above. The final section contains descriptions of research projects undertaken by our graduate students.

---

This research is supported in part by Digital Equipment Corporation, the National Science Foundation, the Powell Foundation, NCR Corporation, IBM Corporation, the UC Micro Program, and TRW Corporation. For more information, contact the authors at the above address, or by FAX at (619) 534-7029. Joseph Pasquale may be contacted by e-mail at *pasquale@ucsd.edu* or by telephone at (619) 534-2673. George Polyzos may be contacted by e-mail at *polyzos@ucsd.edu* or by telephone at (619) 534-3508.

# 1 Introduction

At the UCSD Computer Systems Laboratory (CSL), we conduct research in operating systems, communication networks, performance evaluation, and distributed systems. In operating systems, we are focusing on high performance I/O software support for a variety of I/O intensive applications, such as multimedia applications and scientific applications. In communication networks, we concentrate on problems associated with supporting digital continuous media, such as audio and video, over high-speed networks. In performance evaluation, we are interested in characterizing the I/O time-behavior of scientific applications running on supercomputers, such as the Cray Y-MP at the San Diego Supercomputer Center (SDSC). In distributed systems, we are investigating mechanisms for decentralized control for effective group decision-making and distributed resource allocation.

CSL members include two assistant professors (the authors), and twelve graduate students, most of whom are doctoral students. CSL is equipped with a variety of workstations, including three DEC 3000 Model 500 AXP Alpha deskside workstations, five DEC 3000 Model 400 AXP Alpha workstations, seven DECstation 5000/200, four DECstation 5000/240, two Sun SPARCstation 2, two Sun 3/60, and one Apple Macintosh IIci. Networks within the laboratory include Ethernet and FDDI local area networks. CSL is connected to various UCSD campus networks, as well as a private campus-wide FDDI network which is part of the Sequoia 2000 Network. This connects CSL directly with SDSC and the Scripps Institution of Oceanography, both members of the Sequoia 2000 project. SDSC provides CSL with access to its Cray Y-MP as well as various multiprocessor supercomputers. SDSC is also a node on the NSFNET backbone, providing CSL with excellent connectivity to the outside world.

This document reviews the current research within CSL. Section 2 contains a description of Sequoia 2000, a project in which CSL is participating. In the next four sections, we describe an overview of the work of the four research areas in the laboratory: operating systems, communication networks, performance evaluation, and distributed systems. The following section contains abstracts of past CSL doctoral dissertations. The last section contains research summaries of the graduate students.

# 2 Sequoia 2000

CSL is part of Sequoia 2000, a University of California multi-campus project which addresses the massive data storage, network, and visualization requirements of Global Change researchers.[1] Faculty investigators include computer scientists from UC Berkeley and UC San Diego, and environmental scientists from UC Santa Barbara, UCLA, UC San Diego, and UC Davis. The project was initiated by DEC with initial seed funding totaling $15 million for 3 years; additional sponsors are now being sought.

CSL, in collaboration with the Tenet Group at UC Berkeley, is responsible for designing the Sequoia 2000 Network (S2Knet) [FPP92]. The S2Knet is a private, wide area packet switched network providing high throughput for distributed scientific applications, and real-time services for multimedia collaboration tools such as video-conferencing. It consists of 45 Mb/s T3 backbone connecting the Sequoia 2000 campuses, and 100 Mb/s FDDI local area networks for local distribution at each campus. The actual construction of the S2Knet is a large distributed engineering effort involving all the Sequoia sites; a "handbook" serves as an implementation guide as new sites are added, and is continually updated to reflect the current status of the network [PFF92].

The S2Knet became operational in 1992, and now provides the experimental infrastructure for us to carry out network research (e.g., to evaluate new network protocols), while at the same time providing the project's earth scientists with a fast production network (during normal working hours) that is rarely down and is well

[1] M. Stonebraker, "Overview of the Sequoia 2000 Project," *Proceedings 1992 IEEE Computer Conference (COMPCON'92)*, San Francisco, CA, February 24-28, 1992.

utilized. This network provides a significant improvement in performance over what was previously available through the Internet [PPFK91].

We are investigating fast integrated I/O system and network software architectures to support the real-time transfer of massive data objects between a terabyte database being designed at UC Berkeley and any number of scientists' workstations located throughout the S2Knet. Much of this data is in the form of a time-sequence of highly detailed images, collected by satellites and remote sensors located on the ocean or on land, which scientists wish to process, visualize, browse and manipulate interactively. Consequently, the network must support high-bandwidth transmission of these images and meet various delay constraints to simulate visual continuity.

Some of the issues we are working on include integration of I/O system software and network protocol software to allow dynamic construction of in-kernel datapaths between the network device(s) and I/O source/sink devices (e.g., the frame buffer), determination of where to locate compression and decompression processes, and determination of how to take advantage of prioritized packets containing hierarchically coded data so that the network has the flexibility of selectively dropping packets if necessary (e.g., to prevent congestion).

To begin the process of improving network performance, we analyzed the network I/O software bottleneck in [KP93]. On the high performance DECstation 5000 RISC workstations which are used as routers in the Sequoia network, we categorized the major kinds of work (i.e., protocol processing, checksum computation, data copying, memory allocation, and lesser time-consuming functions) done by the most common Internet protocol, UDP/IP, on a per-layer basis (i.e., socket, transport, network, data-link), for a wide range of different message sizes. We improved the checksum computation, and proposed a scheme for checksum elimination in the case of a sender and receiver located on the same LAN to take advantage of the LAN hardware checksum which makes the software checksum redundant, achieving overall throughput improvements up to 90%. We are in the process of submitting this algorithm (along with working software) for experimental use by the Internet community. A summary of related work which analyzed TCP/IP can be found in [KP93a].

To understand the demands of the network, we are studying the I/O time behavior of Sequoia 2000 applications such as climate and ocean general circulation models. In addition, Sequoia 2000 requires tools which allow the geographically distributed faculty investigators to effectively collaborate: CSL is developing workstation videoconferencing and distributed whiteboard tools, supported by high performance multicasting communication protocols which minimize utilized network bandwidth while bounding delay.

# 3 Operating Systems

We are currently focusing on investigating system software structures which support I/O intensive applications. Current I/O system software found in most operating systems is ill-suited for the high throughput requirements of multimedia and scientific applications which have I/O components consisting of video, high quality audio, high resolution images, and massive scientific data sets [PP90, P+91]. We believe that operating systems design must shift from the traditional focus of managing CPU cycles to that of managing I/O channel bandwidth which is increasingly becoming the scarce resource [P92]. We are investigating I/O system software architectures which support fast I/O switching of communication resources (e.g., the system bus) to support multiple I/O streams with time-varying bandwidth requirements and other real-time constraints [P93a].

Consider a program which emits motion video at a rate of 30 frames/sec in a window with resolution 1280 $\times$ 1024 pixels. If each pixel requires three bytes for describing color, the total output bandwidth required by this program is approximately 118 megabytes/sec. Even after significant compression, a large fraction of the bandwidth of current workstation busses is consumed given existing I/O system software which inefficiently copies data between domains. Furthermore, there is a real-time delay constraint which must be met: each frame must be delivered to the video interface every 33 milliseconds, otherwise, the continuity of the video may be destroyed. If the program were a distributed multi-media application, where multiple nodes are generating video (and audio) which must be distributed to all the participants, one can easily see that the bandwidth and delay requirements become difficult to meet.

We are considering stream I/O architectures, specifically efficient methods of data movement between stream endpoints, which is important for high-bandwidth delay-sensitive continuous media. These endpoints may be a device's buffers and a process's address space (or vice-versa), or directly between the buffers of two devices (one being a source, e.g., a microphone, and one being a destination, e.g., a loudspeaker). Programmers are given the ability to create fast "in-kernel datapaths," which minimize inefficient data copying between memory buffers. Data is transferred between devices by going directly over the system bus or through a single kernel memory buffer while retaining control in user processes. In addition, we are providing better integration of I/O system and network software, optimizing the performance of the latter by using the same minimal data copying techniques. A design and implementation of these some of ideas [FP93] demonstrates the feasibility and performance gains of this approach.

We are considering distributing the kernel so that parts not only reside in main memory, but also, parts providing major functionality (not just device driver code), reside on I/O devices with powerful processors so that a great deal of processing can occur right on the I/O device board. For instance, when a video frame is captured from a camera into a video frame buffer, assuming there is a fast processor on the video board with the video buffer, it may be desirable to compress (or do other sorts of processing tasks) the video frame within the video board itself. Then the frame can be moved about, within the distributed kernel itself, or over a network, more efficiently since it is smaller.

Scheduling and synchronization are less understood, and depend greatly on the control flow organization of stream modules. For example, if each stream module is a coroutine or process, these can be scheduled and synchronized using already developed real-time scheduling and process synchronization techniques (if they are coroutine-based, synchronization methods are more limited, and coroutines cannot be preempted). However, data transmission along streams is inefficient if each stream module is a process, because of potential physical memory copying between processes. We are considering buffer preallocation strategies which help solve this problem. If, however, stream modules are simply procedures, and a single process describes the flow of control along a string of stream modules connecting source and destination endpoints, different problems arise. Scheduling these processes is more coarse-grained, and therefore less flexible, than in the previous approach. To avoid missing deadlines (or recover from missed deadlines), preemption may be required; therefore, this approach may be less efficient than having separately schedulable stream modules.

Over the last few years, we have obtained a great deal of experience in building multimedia applications. The goal in building these applications was not only to gain experience in developing multimedia software, but more importantly to push the performance of the underlying system software, and observe what problems would arise.

First, we built a simple audio conferencing system using standard system software, specifically the X Window System, SunOS, and TCP/IP, using Sun Sparcstations connected by Ethernet [TP91]. This study demonstrated the simplicity and naturalness of using the window system interface for audio, and how to support a form of continuous media without having operating system support for real-time guarantees. This was only possible because voice-quality audio does not make high demands on bandwidth, and delays introduced by the system are imperceptible by humans. To accomplish this, we implemented a simple strategy for recovering from missed deadlines by discarding "late" audio data.

For higher quality audio, and especially video, such simple strategies will not be appropriate. The software structure of the X Window System must be redesigned for supporting continuous media. To support requests which actually contain audio or video data, these must be processed within real-time delay bounds. A multi-threaded design is needed so that requests do not have to wait for previous requests to be completely processed. On a uniprocessor, it it important to allow for the preemption of a thread if a higher priority request (e.g., one containing a audio or video request) is received. This is support that must be provided by the operating system.

More recently, we constructed a video conferencing system using DecStation 5000 workstations connected by an FDDI network. The application also supported playback of video digitally stored on local magnetic disks. The application allowed two users to communicate by audio and video completely through their workstations. Each user sees two live video windows (one in color and one in grayscale) of size 160x120 pixels, while simultaneously viewing a 640x480 color window of video retrieved from disk, all updated approximately 5 to 10 frames per second, depending on compression ratios. Since the video hardware had no support for compression, all compression was done in software. Compression was required to reduce the cost of memory-to-memory copies, which dominated all other costs. The result of our experiences in building these applications is that existing I/O and network software is not adequate for supporting the high bandwidth requirements for multimedia computing. The major problem is that existing I/O software is not well suited to moving large quantities of data between address spaces. Other problems, such as lack of support for real-time synchronization, are also important, but are secondary to the data movement problem.

We have also investigated file system design for very large file I/O such as that required for scientific applications or multimedia (e.g. audio and video) applications [MP91]. The design was driven by results of a file workload characterization study we carried out on a heavily utilized file system at UCSD. File requests at the system call level were characterized as either control (e.g. open, close, chmod), data read, or data write. At the disk level, requests were characterized as either data reads, data writes, control reads, or control writes. Measurements showed that disk level requests were broken down as follows: 15% data reads, 21% data writes, 8% control reads and 56% control writes. Although control requests dominate data requests, the average amount of data transferred by control requests was 250 bytes whereas for data requests it was 4500 bytes. The main observation was that file data and file control requests have very different patterns for accessing disk storage: file control requests arrive at a much higher rate than file data requests, but transfer much smaller amounts of information. This has a significant effect on performance when accessing large sequential files, which is common in scientific or multimedia applications. To increase throughput, the blocks of these large files are often stored contiguously on disk, so that minimal disk head movement is required to access them. This is important because disk head movement is the major cause of performance degradation. However, given the high fraction of control requests to disk, there is a high likelihood that the accesses to sequential file blocks will be interspersed by accesses to other parts of the disk because of control requests.

Our design was based on using separate storage structures to contain control information and file data. These storage structures are constructed using disk arrays, configured in different ways (e.g., mirrored or striped) so that performance for the corresponding control or data request pattern is optimized. A further improvement was obtained by optimizing file cache performance. Disk requests for cache backups (for reliability) were separated from those for cache flushes (for managing the cache storage). The third (and final) storage structure, a disk array configured as a sequential write-only log, is used for cache backups, so that only flushes need to go to the control or data storage. Our *multi-structured* file system can support much higher data rates than existing file systems, especially when the workload consists of very large file requests.

# 4 Communication Networks

In the area of Communication Networks we explore issues related to (a) network traffic characterization and modeling, (b) performance evaluation of local, metropolitan, and wide area networks through measurement, analysis, and simulation, and (c) the design of network architectures for high-speed real-time transport of Digital Continuous Media.

We have undertaken an NSF-sponsored effort to **characterize and model the traffic** on the NSFNET backbone and other wide area network environments. Unlike other similar projects, our goal is to model traffic and performance for a wide area network as a whole, rather than characterize network performance as perceived by an application or traffic characterization within the confines of a local area network. This approach focuses on supporting the integrity of the network, and on developing tools and methodologies to facilitate its graceful evolution. Our research results should allow more accurate capacity planning in high-speed realms and network tuning, such as of routing mechanisms, at optimal levels of granularity. This research is of particular importance now since new services not only dramatically increase network traffic, but can also completely change traffic profiles.

Our initial traffic characterization results, focusing on the T1 NSFNET backbone, appeared in [CPB93]. We are now studying the accuracy of various traffic sampling mechanisms for data collection; this is very important for high-speed networks where the amount of traffic makes it almost impossible to capture complete traces. Other directions pursued include characterization of international traffic flows, studies of correlation between traffic volume and performance at coarse granularities and other performance studies. For example, our study of Internet latencies has shown that the widely used approximation of unidirectional latencies by half the round-trip "ping" delay can be considerably inaccurate [CPB93b].

Since FDDI is becoming the high-speed Local Area Network of choice and since the Sequoia 2000 network consists of interconnected FDDI rings, we have investigated techniques to extend to interconnected FDDI rings the performance guarantees offered by the **FDDI synchronous mode** [B92]. Furthermore, although the synchronous mode is an integral part of the FDDI standard, as well as a distinguishing characteristic of the FDDI medium access protocol, appropriate system software for synchronous mode is unfortunately not typically implemented. Therefore, we have made modifications to Ultrix device drivers and have extended network protocols in order to enable applications with real-time requirements to take advantage of the synchronous mode delay guarantees [J92, E93].

In the area of **Metropolitan Area Networks** (MANs) we have focused on a key problem in the DQDB protocol. The Distributed Queue Dual Bus (DQDB) protocol is the channel access mechanism of the proposed IEEE MAN standard (IEEE 802.6). This is a very interesting system because its frame structure is compatible with ATM/B-ISDN and is believed to be the first form of high-speed connectivity to be offered for the local loop. We have developed the Rotating Slot Generator (RSG) scheme, which overcomes the well known unfairness of DQDB through architectural modifications that allow the system to operate essentially as a dual ring rather than as a dual bus. The basic idea is to use a folded dual bus, with the original slot generators embodied into a 'slot generator' station which is rotating with a fixed period around the 'ring.' Geometry then takes care of any inherent unfairness. Of course, the difficult part is to make the scheme operate without undue complexity and external intelligence or intervention, and also to guarantee that no traffic is lost during the transients. We present this design in [KPP92] together with some initial performance results which support our expectations for fairness, in terms of both throughput and delay. We further investigate the performance of the RSG scheme in [KPP92a] where we show that not only does the approach provide fair service, but it is also more efficient than other mechanisms proposed in order to improve DQDB fairness. RSG provides higher throughputs and lower delays, and also improves other system characteristics, such as the degree of discrimination between priority classes for multiple priority systems.

We have also contributed to the performance analysis of **Conflict Resolution Algorithms** for broadcast communication channels. These protocols are applicable to satellite channels and terrestrial packet radio networks. We have developed a queueing theoretic methodology for the analysis of these protocols which allows us to obtain exact steady state distribution results for the packet delay for many interesting configurations [PMV87]. In [PM93] we provide a complete delay analysis for the celebrated FCFS 0.487 Algorithm.

Another problem of increasing importance with the proliferation of portable computers and the integration of computing and communications is the provision of **location transparency** to users and applications. Providing location transparent services across a network is a problem which can be addressed at multiple levels, from the data-link layer to the application layer. In [CP92] we provide a survey of issues related to this problem as well as a set of existing and proposed solutions.

One of our main areas of concentration is currently the investigation of network architectures and protocols supporting **Digital Continuous Media** (DCM), i.e., digital audio, video, and other media in which data is produced or consumed in a periodic manner (e.g., scientific visualization). Audio and video are often associated with real-time applications such as teleconferencing. The interactive nature of these applications imposes stringent delay requirements on the audio and video flows. Various guidelines set the end-to-end delay tolerance to tens or at most a few hundreds of milliseconds. Meeting these delay requirements may require dropping packets when they accumulate between the sender and the receiver because of transient congestion. Fortunately, humans receiving real-time audio and video have some tolerance to transmission errors. For example, telephone quality audio can usually withstand a 1-2% data loss during talkspurts. The perceived error may vary, however, depending on the particular encoding schemes used and the level of compression. These two observations lead to a characterization of DCM as *soft real-time traffic*. This traffic requires specific performance levels, but has some tolerance to data loss when the requirements cannot be met. **Hierarchical coding** (also referred to layered coding) techniques can split DCM signals into subcomponents of varying importance. Therefore, less important signal components can be dropped without causing an interruption of service; communication can continue in the presence of delays, errors, or network congestion, with only a moderate drop in service quality [P93b].

We also address the analytical performance evaluation of systems supporting DCM flows. For traditional data traffic, Poisson arrivals and either exponential or constant packet sizes (leading to M/M/1 or M/D/1 queueing systems, respectively), have been shown in the past to provide acceptable analytic approximations. DCM sources, on the other hand, generate traffic at regular intervals. Coding and compression of the information generated at these instants results in a variable data rate which depends on the actual information content. Still, data are produced at times that are multiples of an easily identifiable basic time interval. From an analytic point of view, it then becomes more difficult, or even impossible, to justify exponential interarrival times assumptions, i.e., the Poisson assumption. Therefore, a more appropriate model of the output of DCM sources should have constant interarrival times and general "packet" length distribution (which can be determined experimentally). Therefore, a discrete time model with periodic arrivals and general service time distribution (such as a discrete time D/G/1 queue), naturally represents a single DCM flow going through a server. Analytical solutions for discrete time D/G/1 queues, i.e., a single DCM flow, are known. We are currently extending this model to represent multiple DCM flows.

Even with the availability of high-speed fiber-optic networks, the large data volume of DCM requires extremely efficient data transfer mechanisms. Therefore, support for efficient multicasting is an important requirement, well recognized, but poorly supported at the present time. **Multicasting** refers to one-to-many communication, in contrast to *unicasting* which is one-to-one communication, and *broadcasting* which is one-to-all communication. Multicast can be thought of as intermediate service between unicast and broadcast, and, unfortunately, many times it is implemented as either a broadcast (with filtering at higher protocol layers), or as a series of unicasts from the source. These naive approaches to multicasting are inefficient because they may place an unnecessary load on the network. Broadcasting sends the data to many more nodes than required. Multiple unicasts will likely use the same link several times to carry the same data, and the source must work harder to send the multiple copies. However, multicasting can be done efficiently using tree-based communication structures. These structures can reduce bandwidth utilization by delaying duplication of data until the routes to different destinations diverge, thus reducing the number of copies sent over the network. In addition to reducing bandwidth requirements for multimedia communication, it is important to determine routes which have low end-to-end delay from the source to possibly multiple destinations. In selecting data paths, one should minimize the crossing of either network relay nodes or abstraction layers (e.g., protocol layers), both of which contribute to delay in data delivery. Switching, queuing, protocol conversion, as well as propagation delay all contribute to total delay. However, since delay is often tolerable below a certain threshold level, a good routing algorithm does not necessarily have to minimize the delay along the route; bounding the delay is adequate.

The problem of finding optimal tree forwarding routes from a source to the members of a multicast group under given delay constraints is intractable. Therefore, we have developed heuristics which are both efficient and effective, typically providing feasible solutions that are only a few percent less than optimal [KPP92a]. Source routing heuristics are based on an abstraction of the problem from a full network to that of a complete graph, with the members of the multicast group (only) as vertices, and the shortest paths (with respect to bandwidth — or link count) between any two such members as edges. Trees on this much smaller graph are much easier to track. In particular, we show that for two metrics proposed as selection functions for the heuristics, one leads to better solutions in the source routing case and the other in the distributed routing case. We are investigating the performance of these schemes both empirically and analytically on random graphs and graphs resulting from specific network topologies.

In [PPAK92], we describe a new communication abstraction for dissemination-based communication of multimedia. It is based on a Cable-TV model: a source places media streams on a channel, without explicit interaction with receivers; receivers can dynamically "tune in" to channels of interest, without explicit interaction with the source. The use of loose-coupling between a source and the multiple receiver nodes promotes pro-active control mechanisms [PPK93]. In addition, heterogeneous receivers are supported by encouraging hierarchical coding of sourced media streams which receivers can filter arbitrarily. Finally, filter propagation promotes efficient use of network bandwidth.

# 5 Performance Evaluation

Based on the need to design well-balanced supercomputer systems that deliver the high performance expected of them, we are undertaking a comprehensive study of the relationship between the I/O requirements of scientific applications and the capabilities of the I/O system. We are working closely with the San Diego Supercomputer Center (SDSC) which is one of the four national supercomputer centers sponsored by the National Science Foundation. SDSC currently maintains a Cray Y-MP supercomputer (among other machines) which we have been measuring. SDSC offers Cray time to users all over the United States, who generally wish to run large scientific applications. These users comprise 2700 researchers from 170 institutions, spanning 44 states.

The goals of this research are:

1. to characterize the I/O behavior of a wide spectrum of scientific applications
2. to develop models for well-defined classes of scientific application I/O activity
3. to construct an executable I/O benchmark set for the purpose of locating existing or potential I/O bottlenecks
4. to recommend operating system software enhancements to improve supercomputer I/O performance.

The first goal addresses the problem that our understanding of scientific applications needs to be improved. Scientific applications form an interesting set of programs worthy of study because they absolutely need high performance supercomputers and therefore, systems should be designed to meet their requirements. Before we can do the latter, we must understand these requirements. This involves observing an environment supporting a variety of scientific applications (e.g., SDSC Cray Y-MP), and determining what to observe to deduce these requirements in a way that does not interfere with their execution.

The second goal involves summarizing these observations into a form that may be used for descriptive or predictive purposes. The intent of the third goal is to provide a comprehensive set of scientific I/O benchmarks so that users can select the I/O requirements that best represent their own workloads. Finally, the fourth goal is to put results into practice: to improve I/O performance on the supercomputer at our disposal which supports scientific workloads. This will be limited to operating system software enhancements, such as the design and implementation of I/O resource scheduling algorithms, or modification of software cache sizes. This may also involve suggestions to the designers of applications of how to improve their codes.

We are focusing mainly on application characterization rather than overall system behavior because our intent is to investigate not only current but also future potential problems which are not limited to what is observable in current systems. Specifically, we are interested in understanding the issues involved around a potential I/O bottleneck in future supercomputers and to provide tools to prevent it. This scenario is associated with supercomputers that are computationally much more powerful than those of today and also with workloads that are much more I/O intensive. The latter is driven by user expectations about performance of the future machines. Based on current application I/O characterizations one can extrapolate future I/O application requirements, and then through synthesis determine overall I/O workload intensities and patterns. Results of our analyses support our hypothesis that a significant proportion of I/O intensive, long running, frequently executed scientific applications have predictable I/O requirements [PP92].

Our emphasis on application I/O and what we learn will be applicable to and benefit a larger audience including disciplines outside of computer science. It is our experience that the scientists and engineers running these codes are very interested in learning about the performance of their applications.

We also performed a workload characterization study of the SDSC Cray X-MP (the predecessor to the Cray Y-MP), running the CTSS operating system [PBK91]. In this work, our underlying strategy has been to recognize the attributes of the workload being submitted by the national academic research community for execution on the supercomputers at SDSC. Many of the jobs comprising the workload require either large fractions of a given resource, or require support which traditionally has not been available on supercomputers.

The workload was characterized at the functional level, by grouping jobs according to the name of the command executed by the job. These commands were classified as either public or private, depending on whether the program was available to all users (e.g., system utilities, generally available scientific packages) or just a single user (e.g., user-developed codes not made generally available).

We found that 93% of all jobs were public jobs, yet 90% of all CPU time was consumed by private jobs. Although the public utilities make relatively small demands on the CPU, they consume 22% of the total memory space-time product, and 75% of the total I/O channel time. The latter indicates that a very significant I/O demand is being made by the public component of the workload.

The workload was also characterized at the resource level, by classifying jobs according to their resource consumption statistics, specifically CPU time, I/O channel time, and memory space-time product. The jobs were then clustered according to their command name and resource usage, using a variant of the *k-means* clustering algorithm.

We related the functional and resource level characterizations by (1) examining what command types made up each cluster and (2) seeing what percentage of system resources each cluster consumed. The cluster **Pub-Low** contained commands whose functions are listing, removing, displaying, and deleting files. **Pub-Medium** contained commands whose functions are interactively managing files, printing manual pages, editing text files, and searching files for specified text strings. **Pub-High-CPU** contained commands such as *g86lib*, which is a quantum chemistry application. **Pub-High-I/O** contained commands such as *ftp*, which performs file transfers.

Table 1 shows the seven clusters produced by this analysis, including the amounts and relative percentages of system resources consumed by each cluster.

Table 1: Individual Cluster Statistics

| Class | Number of Jobs | % | CPU (s) | % | I/O (s) | % | Mem-Time (Mw × s) | % |
|---|---|---|---|---|---|---|---|---|
| Pub-Low | 471392 | 46.60 | 5268 | 0.03 | 53870 | 1.72 | 2268 | 0.01 |
| Pub-Medium | 415267 | 41.05 | 254723 | 1.50 | 892470 | 28.50 | 128604 | 0.82 |
| Pub-High-I/O | 43975 | 4.35 | 242541 | 1.42 | 1291697 | 41.25 | 1581887 | 10.11 |
| Pub-High-CPU | 17038 | 1.68 | 1234901 | 7.24 | 99618 | 3.18 | 1782237 | 11.39 |
| Priv-Low | 17712 | 1.75 | 4836 | 0.03 | 5888 | 0.19 | 1019 | 0.01 |
| Priv-High | 27889 | 2.76 | 689063 | 4.04 | 111281 | 3.55 | 146545 | 0.94 |
| Priv-Massive | 18318 | 1.81 | 14631825 | 85.75 | 676757 | 21.61 | 12002096 | 76.72 |

The basic results were the following. 88% of all jobs are either in clusters **Pub-Low** or **Pub-Medium**, but these jobs used only 2% of the total CPU time. Only 2% of all jobs are in the cluster **Priv-Massive**, which used 86% of the total CPU time. **Priv-Massive** was also the largest user of memory, utilizing 77% of the total memory space-time product (over all jobs). Three clusters used 92% of the total I/O time: **Pub-Medium**, **Pub-High-I/O**, and **Priv-Massive**. One important observation was that even jobs that make insignificant demands on the CPU (e.g., interactive system utilities) can have a major effect on the I/O system.

# 6 Distributed Systems

Resource sharing is a major reason motivating the design of distributed computer systems. We are concerned with decentralized allocation of resources, its fundamental problems, and how to overcome these problems [P91]. Decentralized control offers many benefits over centralized control algorithms, such as speed (through parallel processing and keeping control near data), reliability (by avoiding single points of failure), scalability (because of symmetric control relationships), and autonomy (by allowing control over local resources to be kept locally).

However, these benefits have not been fully realized because of difficult problems associated with distributing control. There are two fundamental problems: (1) no decision-making agent can be certain of the global system state; (2) no decision-making agent can be certain of the global system action. The first problem exists because because communication cannot occur instantaneously (and we assume there is no shared memory in a distributed system with decentralized control). Thus, any communicated state information necessarily describes a past state, and at best, the current state may be inferred but not known with complete certainty.

The second problem is more subtle and exists because there is no common knowledge about actions. It is often the case that an agent makes a decision based on its view of the global system state, and even if this view is perfect (i.e., (1) is not a problem), its decision may conflict with decisions made by other agents at the same time. This is best illustrated through the following example. Consider a load balancing system, where every agent which makes decisions about offloading jobs knows perfectly the global system state. The agent on a heavily loaded machine will decide which machine is least-loaded, and offload there; that is the locally optimal decision for that agent. However, if there are many heavily loaded machines, those agents may all decide to make their locally optimal decision which is to offload to the same least-loaded machine, and overload it. Thus, these locally optimal decisions conflict, and result in a global disaster.

We are focusing on three aspects of these problems: the effects of global state uncertainty on decision-making; multi-agent coordination; and group formation. Our approach to the global state uncertainty problem is to characterize two loss functions: loss due to decision quality degradation, and loss due to communication overhead, both as a function of state update period. The loss due to decision quality degradation is low for small update periods (because the communicated state information is timely), increases as the period gets larger, and eventually levels off at some maximum value (because after some point, the information becomes worthless and remains worthless regardless of how much older it gets). The loss due to communication overhead is infinite for a period of zero, and rapidly decays as the period increases. Adding these loss functions gives the total loss, which generally has some minimum point, which is the optimal communication period. An analysis of this problem can be found in [BP92].

Decision-making problems in distributed systems generally require coordination among the entities in the system. In [GP93], we investigate a phenomenon we call "co-adaptation," whereby individual managers adapt their behavior in a complementary fashion to eventually achieve coordinated decision-making. Decision-makers receive input from and affect a shared environment, however they do not explicitly communicate with each other. They are given feedback (by some mechanism in the environment) to evaluate the success of their past decisions so that they may modify their decision functions to better perform in the future. We wished to observe whether any implicit coordination could be effected, i.e., coordination based on individual feedbacks from individual effects on a shared environment rather than explicit messages interchanged between managers to plan coordination.

As an experiment, we constructed a simple distributed job scheduling system consisting of two job schedulers and two processors; a stream of randomly sized jobs which appear at random times are given to the job schedulers, and they must determine which processor to execute the job to minimize turnaround time. To make load balancing decisions, each job scheduler uses a stochastic learning automaton, defined by a time-varying probability distribution over the processors. When a job needs to be assigned to a processor, a processor is randomly selected according to this distribution. If the job is assigned to the processor with the shortest queue, the probability for selecting this processor is raised, otherwise it is lowered. There are two parameters to "reward" and "penalize," quantifying the degree to which probabilities are raised and lowered.

Even in this simple system, we observed interesting forms of co-adaptive behavior. Where there are only rewards for good decisions and no penalties for bad decisions, the schedulers co-adapt by progressively favoring different processors while equalizing utilizations of both processors. This continues until an "absorbing state" is reached, i.e., a scheduler selects one processor with probability one, and since bad decisions are not punished, no further adaptation can occur. Schedulers that use both rewards and punishments co-adapt better by converging to the "equiprobable state," where the selection of either processor is equally likely, for both schedulers. However, the schedulers then fluctuate about the equiprobable state in different ways depending on the ratio of the reward and penalty parameters. The main contribution of this work is an analysis of these behaviors and a characterization of what aspects are emphasized by rewarding or penalizing.

Randomization is often used to coordinate decision-making agents so that decentralized access decisions over a shared space of resources can be made in a minimally-conflicting manner, with minimal communication. In [P93], we investigate the performance of an autonomous decentralized system based on a model which is limited to the simultaneous probabilistic access of resources by multiple managers: given m busy managers and r free resources, each manager decides to access a resource with probability a, and each resource is selected with equal likelihood. System performance is quantified by a gain G = U - bC, where random variables U and C are the number of utilized (i.e., accessed by exactly one manager) and congested (i.e., accessed by more than one manager) resources, respectively, and b is a design parameter which specifies the relative cost of congestion to utilization.

The presented results include the optimal access probability, the maximum expected gain, the optimal number of managers for a given number of resources, and the limiting maximum gain as the number of managers increases without bound. The optimal access probability is a simple function of r, m, and b; it is directly proportional to r, and inversely proportional to m. The maximum expected gain is relatively low when there is more than a single manager: while a perfect system would always access all r resources with no conflicts, a system based on this random mechanism will successfully access roughly r/e resources (for b=0, and less as b increases). Thus, the effectiveness of coordination based solely on randomization given only information on m, r, and b, is limited; however, it is better than if there were no coordination mechanism at all. Finally, we also show that, given m=kr managers and r resources, it is better to partition managers and resources so that each resource is accessed by a separate group of k managers rather than to allow all m managers to compete for all r resources.

Finally, we are considering how agents dynamically form groups such that these groups are more effective in solving problems than independent agents. A group is recognized by the amount of communication taking place between agents: agents within a group communicate at a much higher rate than agents in different groups. We wish to understand how groups form spontaneously, why they form, and whether such activity can be predicted. We are considering a number of prediction models based on game theory (agents within a group playing games, and groups playing games against each other) and learning automata (adaptive algorithms based on environmental feedback which reinforce or break down inter-agent relationships).

In [BP93], we investigate how delayed communication affects the dynamic formation of groups in distributed systems, where all decision-making agents join the same group because each expects to improve its own performance. For example, distributed job schedulers may form a group to utilize the idle resources of other members within the group. Forming a group is a search problem and we examine agents which use the feedback mechanism of stochastic learning automata to carry out this search. Although a group formation may have the potential for synergy, the agents must successfully coordinate their actions within the group relevant to the application. For example, job schedulers who form a group must still balance the load among the shared resources, and greedy schedulers who all pick the same processor may not be successful. Agents may find that working alone is more desirable since the collective actions need not be coordinated and the results of their own actions are more predictable. An additional challenge to the search problem is to cope with the delay in communication between the agents. The purpose of this study is to model systems where agents adaptively search for compatible co-workers, under the constraint of delayed communication. With insufficient communication, the agents decide to work alone (and receive a modest benefit) but, with sufficient communication, the agents make the more advantageous decision to work together.

In [BP93a], we consider load balancing over a network of processors given aged load information. We introduce a concept called "scope of control" of a scheduling agent, which determines the set of remote processors which the agent will consider for its load balancing decisions. We ask the general question: how large should the scope of control be, and what parameters affect the size? The larger the scope of control, the more state information that must be communicated; the smaller the scope of control, the less opportunities there are for offloading. Clearly, there is a tradeoff that must be evaluated. A basic result we report is that the size of the scope of control is directly proportional to the timeliness of information available on the state of the processors. In particular, we show that a load balancing system of processors optimally self organizes into small working groups, whose sizes depend directly on constraints imposed on the update rate at which state information about the group can be communicated (e.g. due to the desire to limit overhead).

# 7 CSL Doctoral Dissertations

This section contains abstracts of past doctoral disserations by CSL students.

**K. Muller: Multi-structured File System Design for High-performance Computing, 1990**

File system I/O is increasingly becoming a performance bottleneck in large distributed computer systems. This is due to the slow decline of disk access times (latency and seek) relative to the rapid increase in CPU speeds, memory size, and network bandwidth. Indeed, although disk storage capacity increases while the physical size and cost of disks decrease, file system performance remains limited by long disk access times and disk contention. The problem is further aggravated by new applications with intensive I/O components, such as multi-media or visualization programs.

Technology trends indicate that it is unlikely that disk access delays will significantly decline in the near future. Thus, solutions cannot be based on improvements in disk access time, and must somehow circumvent this problem. Until recently, the high cost of disk storage, limited the design of file systems to techniques which optimize storage efficiency. However, with the rapid decline in disk costs, solutions explicitly designed to optimize performance have become very attractive.

We present a design for a multi-structured file system. Our design is based on the coordinated and cooperative integration of disk caching with three different storage structures each implemented on independent and isolated disk systems. The individual storage structure were designed to optimize a different set of file system access characteristics such as cache writes, directory searches, file attribute requests or large sequential reads/writes. The isolated storage storage structures reduce contention and lower disk access delays, so high bandwidth I/O and decoupled concurrent access to multiple disks can be provided. Both file system performance and storage size can be scaled to meet the requirements of different work loads by adding disks to the appropriate storage structures. As part of our study, we analyze the performance of an existing file system using trace data from UNIX disk I/O-intensive workloads. In trace driven simulations, we show how performance improves using our technique of dividing file system requests based on access type to optimized storage structures, as implemented by our multi-structured file system.

**A. Glockner: Coadaptive Decision-making in Distributed Computing Systems, 1991**

Decision-making in distributed computing systems is complicated by the introduction of communication delays and the resultant uncertainty both in the global system state and in current actions in the system. This dissertation examines the effects of introducing *coadaptation* into distributed decision-making. By constructing the decision-making process as a set of autonomous agents that can individually learn from feedback, the set of agents adapt together, or *coadapt*, to form a good, though not necessarily optimal, global decision process.

These effects are demonstrated with simulation experiments in distributed computer job scheduling; the agents are implemented as learning automata. The resulting behavior is analyzed both in terms of a global performance metric — mean job turnaround time — and in terms of the states of the agents themselves. Several key elements in the behavior of coadaptive systems are illustrated. These include demonstrations of: situtions where apparently good individual decisions result in worse performance, correlations between automata type and decision-making policy of the agents, complementarity in agent behavior, differences in the relative performance of agents both with small changes in agent parameters and and with changes in the environment, and properties of stability in coadaptive systems. Simulations performed with changes to agents and to the distributed computing system model show further principles of coadaptive behavior. The dissertation concludes with a summary of principles of coadaptive behavior and a projection of future methods of coadaptive distributed job scheduling.

**E. Billard: Delayed Communication and Dynamic Group Formation in Distributed Systems, 1993**

In a centralized computing system, the sole decision-maker, or autocrat, has a perfect view of the global state; it can claim that "it is the state." However, centralized computing is giving way to decentralized computing where the distributed decision-makers each have their own, but uncertain, view of the global state; the uncertainty is caused by the delay in communication of state information. There is a tradeoff in the frequency of communication: too much communication increases overhead and too little communication results in poor decision-making due to an inaccurate view of the global system state. We model this behavior with distributed game automata which are adaptive decision-makers with aged information concerning the strategies of each other. We focus on the performance of these decision-makers with respect to the amount of communication. A particular type of decision considers *with whom* to coordinate and the goal is to find optimal groups within a network of decision-makers. That is, agents may decide to work alone or to work together in groups to derive a mutual benefit. Simulation of a static load balancing application shows that, with sufficient communication, an adaptive strategy in dynamic group formation is globally optimal. Analysis indicates the potential for poor decisions when the agents communicate infrequently. The results illustrate the fundamental nature of the uncertainty present in any distributed system.

# 8 Graduate Student Research Summaries

Research summaries of the graduate students who are members of the Computer Systems Laboratory follow.

# I/O Interfaces for Optimal Performance

*Eric Anderson*

Experimental multimedia projects such as the distribution of realtime audio and video by an IP network are often hindered by the inefficient transfer of data between I/O devices on a workstation. Much of the inefficiency results from the design of the I/O interface which Unix presents to programmers. Design assumptions valid several years ago are not always suitable for high performance I/O.

The hardware arrangement of most modern workstations permits the bulk transfer of data from an I/O device into RAM, from RAM to a device, and sometimes directly from device to device. Experimental programs written at the user process level often inspect or modify data. Compression of video, silence detection in audio, and format conversion for either media typically require that the data is accessible by the user's program as the data moves between devices.

For this type of data-manipulating program, the optimal use of hardware occurs when the device-to-RAM transfer places data in the RAM which will be used by the user's program. Likewise, the RAM-to-device transfer should originate from RAM used by the user's program. Protocol processing and the traditional system call interface make this difficult or impossible to accomplish in current Unix systems.

I am investigating an alternate interface which will permit this optimal data transfer for certain types of I/O. The initial work will involve IP datagrams, because these require little protocol processing and are often used for multimedia applications. The interface will allow a user process to take ownership of a hardware page containing a received datagram, rather than having the datagram copied byte-for-byte into some other RAM which the user already owns. The process for sending datagrams will be similar.

This approach yields a variety of security, performance, memory management, and implementation problems. For example, the page inherited by the user might be larger than the received datagram. In this case, the part of the page not occupied by the datagram must not contain any kernel data to which the user would not normally have access. There are several possible solutions to this problem which are being studied.

It is expected that this new interface can be applied to many types of I/O, especially those which benefit the most from optimal transfer, and that the traditional interface can continue to be offered through a library placed on top of the new interface. Such an implementation could allow existing programs to operate unchanged, and possibly faster, while programmers who desire optimal transfer can obtain it by using the new interface.

Experiments will be conducted under the OSF/1 operating system, running on 64-bit Alpha AXP workstations connected to T1, Ethernet, T3, and FDDI networks. Raw and JPEG-compressed video provide an excellent source of high-volume data for exercising high-performance networks and operating systems.

# Traffic and Performance Characterization Methodologies for Wide-area Datagram Networks

*Kim Claffy*

My research focuses on establishing and improving the efficacy of traffic and performance characterization methodologies on wide-area data communication infrastructures. The core components of my research encompass three areas:

1. provision and refinement of empirical results on traffic and performance characteristics of the T1 and T3 NSFNET backbones;
2. analysis of the effects of sampling to characterize traffic and evaluate performance in a high-speed wide-area network environment;
3. evaluation and improvement of the data acquisition methodology of wide-area networks, with particular application to the NSFNET backbone.

I undertook a workload characterization and performance evaluation study of a working wide-area network environment, the 1992 T1 NSFNET backbone, using operationally available but previously uninvestigated data sets. I was successful in analyzing statistics on various areas of interest, including the drop in traffic volume toward the end of 1991, as the NSFNET project began to divert traffic to the newly deployed T3 backbone. Although the mean utilization for the month we analyzed was significantly lower than it had been before the T3 backbone existed, average utilizations for fifteen minute intervals were still as high as 80% for some links. As was true two decades ago in the ARPANET environment, traffic favoritism was high, e.g., 0.28% of the (customer/campus/site) network pairs generated 46.9% of the traffic. But perhaps the most valuable result of the study was a comprehensive understanding of the available data, including both the opportunities it offers for traffic characterization and performance analysis, as well as the difficulties with the completeness and accuracy of the statistics. Inattentive data collection methodologies prevented results at the level of desired detail, completeness and confidence. For the network service provider, operational services and stability clearly outweighed the special attention that long term statistics collection requires. The type, integrity and completeness of existing data sets can offer only a questionable contribution to analysis, modeling, and ultimately evolution of the network. This situation threatens to deteriorate further as network service providers try to respond to market demand in the face of technological advances in speed and ubiquity, with little resources left to devote to rational investigation of the infrastructure itself. As a case in point, the NSFNET/ANSnet backbone has recently integrated a sampling mechanism into their data collection methodology, without preparatory research or analysis into the impact of the sampling on the integrity of resulting data sets.

An important component of my thesis research will be an exploration of this current sampling mechanism used on the T3 backbone, specifically, to qualify the integrity of statistical objects derived via sampling. An investigation of the T3 backbone will not only require validation of the sampling mechanism, i.e., the microscope through which we examine the network, but also require qualification of the operationally gathered NSFNET statistic objects.

I will use this study to respond to a critically developing need in the field of wide-area networks, by devoting a component of my thesis to the development of a rational data collection methodology on a wide-area network. Starting from the existing status of the current T3 backbone implementations, I will offer:

1. additional and/or refined target metrics, with a rationale for each included metric;
2. improved procedures to obtain the new/existing obtained metrics;
3. error estimates on obtained data.

Improved statistics collection will help networking activities along various time horizons, from defining service quality patterns, to budget justification, to long-term capacity planning.

## System Software Support for I/O Intensive Applications

*Kevin Fall*

*I/O Intensive* applications move a large amount of data (on the order of hundreds or thousands of megabytes). Many applications, especially those manipulating multimedia information, require the movement of large volumes of data between devices or files in a timely fashion with minimal intermediate manipulation or processing.

The design of most conventional I/O systems dates back to the Multics system of the late 1960's and UNIX[2] system of the early 1970's. The UNIX I/O model provides a continuous untyped stream of bytes between applications and devices; little software support exists for the direct transfer of data between hardware devices. In contrast to the conventional I/O system, a *peer–to–peer* I/O system uses the computer as a "I/O director." User processes become the specifiers of the sources and sinks of data streams, leaving the job of data transfer up to the operating system.

The problems associated with the conventional I/O architecture are twofold. First, programming is made difficult because of the need to select appropriate buffer sizes and handle flow control. Second, the current I/O system interface degrades performance by requiring protection crossings, context switches, and unnecessary data copies. While the CPU spends time in overhead processing, it is made less available for running other user processes, resulting in longer execution times for other runnable processes. Moreover, the conventional I/O system prohibits enhancements such as shared buffering and direct hardware streaming because the standard **`read`** and **`write`** system call interfaces require I/O data to be made available to a user's address space.

My research investigates the following areas: the I/O interfaces at each system software layer ultimately responsible for the flow of I/O data, and the mechanism of flow control employed to assure data integrity (or at least predictable loss). The interface targets of investigation are: system call interface, device driver interface, and device interface in addition to flow control and buffering schemes supporting the peer–to–peer I/O architecture.

A prototype implementation of the peer–to–peer I/O architecture and including a new system call named *splice* has been implemented in UNIX. The implementation and its performance is described in Fall and Pasquale[1] and Fall[2]. The improvements achieved by splice result from two modifications to the I/O subsystem:

1. elimination of I/O data copies to and from user address space
2. elimination of need to run a user process to facilitate I/O data flow, delegating flow control and timing to the kernel

Elimination of data copies contributes primarily to the throughput and CPU availability metrics measured by the tests. The second issue contributes primarily to the ease of programming; a programmer need not be concerned with choices of buffer size, looping, etc.

Except for modifications to support non-blocking behavior in the file system implementation, splice does not require fundamental modifications to the buffering, scheduling, protocol, or block allocation strategies present in most UNIX systems, yet can achieve upwards of an 80% throughput improvement. The file system implementation uses the buffer cache interface present in most UNIX kernels, and the network implementation uses the Berkeley socket layer[3]. Future work will involve modification to other software layers, making the implementation of splice easier and more flexible. In addition, modifications various device drivers to support splice more efficiently will result in further performance improvement.

[2] UNIX is a trademark of USL.

**References**


[1] Fall, Kevin and Pasquale, Joseph "Exploiting In-Kernel Data Paths to Improve I/O Throughput and CPU Availability," *Proc. USENIX Winter Conference*, January 1993, pp. 327–333.

[2] Fall, Kevin "Improving I/O System Performance with In-Kernel Data Paths," *Proc. UniForum Winter Conference*, March 1993.

[3] Karels, Michael and McKusick, Kirk and Quarterman, John *The Design and Implementation of the 4.3BSD UNIX Operating System*, Addison-Wesley Publishing Company, 1989,

# Experiments with an Audio Conferencing System

*Norival R. Figueira*

The main objective of this work is to identify the problems and limitations of implementing a voice-quality audio conferencing system, based in system software and network lacking real-time guarantees. A secondary goal is to understand the problems inherent to an audio conferencing application.

The experiments led to the implementation of two versions of an audio conferencing application: one for the SPARCstation (SUN OS 4.1.2 of UNIX) and another for the DECstation 5000 (ULTRIX with LoFi library). The application's source code is basically the same for both versions, except for the control of the audio devices.

Some of the problems that were identified are:

1. As the use of CPU time by the application increases, the higher is the probability it will lose deadlines.

2. The variance of the delay, experienced by the packets sent through the network, varies widely with the time.

3. The network loses and reorders the packets (UDP).

4. TCP imposes too much delay for a real-time conversation.

Problem 1 was significantly reduced by the use of efficient algorithms and I/O blocking where possible. Unfortunately, this restricts the use of some complex algorithms like: complex silence detection, echo elimination, digital filtering, complex codecs, etc.

Problems 2 to 5 led to a dynamic playout delay determination algorithm. This algorithm finds dynamically a minimum playout delay, that is necessary to reduce the number of late packets and the size and amount of silence glitches.

The application was successfully tested in local and international communications and is being currently used to the evaluation of new algorithms.

**References**

[1] Terek, R. and Pasquale, J. "Experiments with Audio Conferencing Using the X Window System, UNIX, and TCP/IP," *U.C.San Diego Technical Report CS90-182*, December 1990.

## Traffic Characterization on the NSFNET

*Nancy Groschwitz*

The NSFNET is the cross-country backbone of the Internet. The purpose of my research is to examine the characteristics of the NSFNET traffic.

Over the past twenty years, as the original ARPA network evolved into the current Internet, it has expanded in size and complexity at a rapid rate. Accurate information about the behavior of the Internet is required to assess current performance and to plan for future growth. Such information could also be used to develop new models of the system, to verify and correct existing models, or to test alternative algorithms and protocols.

The available data includes 15-minute and monthly counts of total number of packets and bytes, number of errors (damaged packets), delay data, and service type. The delay data consists of round trip times between single-hop nodes, measured from an external address to an external address; thus, it is comparable to what might happen to a normal data packet.

Initially, I examined data for a single NSFNET node, the San Diego Supercomputer Center, which has single-hop links to three other NSFNET backbone sites: Houston, Palo Alto, and Seattle. Examination of packet and byte counts showed daily and weekly cycles that closely resembled traffic patterns that have been reported in the literature for many local area networks. I found that the autocorrelation between successive 15-minute samples is very high (above .90); there are also 24 hour oscillations in the autocorrelations. In other words, the correlation is highest for the immediately preceding 15-minute bin, but is also high for a 15-minute bin from exactly 24 hours earlier, and from successive bins set at 24-hour intervals.

I am currently analyzing delay data for all single-hop links on the backbone. There are 14 NSFNET nodes, and each one has single-hop links to an average of three other nodes. So far, I have found that there is a reliable correlation between variations in traffic and in delay on a given link, but no correlation between the average traffic on a link and average delay. In other words, when traffic on a particular link goes up, so will the delay, on the average; but the link with the highest average traffic flow may have the same average delay as a link with much less traffic. I am now attempting to determine whether there are links that habitually have longer delays than the network average.

The next direction for this research is to examine the changes in delay and traffic patterns over the last several years. Although the overwhelming trend in traffic through the years has been simply that it is getting larger, due to the ever-increasing number of sites connected to the Internet, I want to see if this trend is masking other, more cyclic effects. I also intend to extend the initial investigation of the San Diego Supercomputer Center traffic patterns to other NSFNET nodes.

# TCP/IP Optimization

*Jonathan Kay*

I am working on improving performance of TCP/IP implementation. I have performed a study of network software performance, and have already used the results of that study to design a checksum redundancy avoidance algorithm and an improved RISC checksum algorithm, which substantially raise network throughput [2] [3]. A big problem with modern workstations is that their networking software cannot keep up with network speeds, so these speedups are quite valuable. A sign of the interest in this work is that the improved RISC checksum algorithm is being incorporated into the Mach distribution.

The study of network software is of network software processing time breakdowns on Ultrix 4.2A running on a DECstation 5000/200 over FDDI. Many have suggested that network performance is largely held up by a few operating system primitives [1, 4], so I examined the network software with the object of deciding which functions consume the most time. Timings were collected by connecting a logic analyzer to the I/O bus of the DECstation and instrumenting the TCP/IP implementation in the kernel, including the network drivers and the sockets subsystem.

The most obvious result of the study is that checksumming is the most time-consuming activity — in a high-throughput mode, 40% of the time is spent checksumming packets. Thus, it was obvious that the first thing to do was spend time reducing the performance impact of checksumming as much as possible.

The best way of reducing the time spent checksumming is to avoid checksumming altogether. We devised an algorithm to do exactly that when it the checksum is completely redundant with a hardware checksum or CRC. It turns out that packets sent over Ethernets, FDDI, and most other LANs (which is the overwhelming majority of packets) are protected by a hardware CRC. Thus, we developed algorithms for TCP and UDP which avoid checksumming data in packets sent over single LANs. Checksumming is performed for packets sent over multiple hops since, in general, packets sent over multiple hops could be routed over networks which do not support checksumming, and in any case the chance of transmission without corruption at gateways decreases exponentially with the number of gateways passed through.

Since relatively few packets leave their originating LAN, the checksum redundancy avoidance algorithm nearly eliminates checksum overheads. Those that remain are drastically reduced by the improved RISC checksumming algorithm.

I plan on continuing to work on eliminating network software overheads. In particular, the next most important overhead shown by the study is data copying.

In addition to the incorporation under Mach, the entire suite of checksum improvements will shortly be deployed in Project Sequoia 2000. An implementation of the improvements is available for anonymous FTP in ucsd.edu:/pub/csl/fastnet/fastnet.tar.Z.

**References**


[1] D. D. Clark, V. Jacobson, J. Romkey, H. Salwen, "An Analysis of TCP Processing Overhead," *IEEE Communications Magazine*, 23-29, June 1989.

[2] J. Kay, J. Pasquale, "Measurement and Improvement of UDP/IP Network Software Throughput on the DECstation 5000/200," *Proceedings of the Winter 1993 Usenix Conference*, January 1993.

[3] J. Kay, J. Pasquale, "A Summary of TCP/IP Networking Software Performance for the DECstation 5000," abstract to appear, *Proceedings of the 1993 SIGMETRICS*, May 1993.

[4] R. W. Watson, S. A. Mamrak, "Gaining Efficiency in Transport Services by Appropriate Design and Implementation Choices," *ACM Transactions on Computer Systems*, 5(2), 97-120, May 1987.

## Multicasting for Multimedia Applications

*Vachaspathi P. Kompella*

Communications research is presently close to a major revolution with respect to the media being used over computer networks. While text and data were the media of yesterday, audio and video are hot media, both in research and in the computer industry. The most desirable aspect of audio and video is that they convey information so much more naturally than text. One notable use of multimedia is in multi-user applications, also known as groupware.

My research work is about how multimedia traffic can be routed cheaply over a packet-switched network to multiple destinations. The purpose of this research is to show that routing data as we know it now differs significantly from routing multimedia. The two dominant characteristics of audio and video are their volume, and their sensitivity to delay. I have been investigating routing algorithms that can address both these issues effectively. The two salient objectives of these routing algorithms are:

a) to minimize the cost of the multicast: assuming a cost assigned to each edge in the multicast tree (that could be a measure of the resources tied up in the communication over that edge), a good multicast tree should minimize the total cost of the tree

b) to bound the end-to-end path delay: since the media are delay-sensitive, it is important to ensure that packets are transported with bounded delay.

It must be emphasized that the problem of finding a minimum cost tree that spans a subset of nodes (also known as a Steiner Tree) in a graph is NP-complete. Introducing the additional constraint of delay bounded end-to-end paths does not simplify the problem. I have developed fast heuristics for the CST problem that have been empirically successful in generating near optimal CSTs. The Constrained Steiner Tree (CST) heuristics that I have been working on fall into two categories, i) the static multicast group, and ii) the dynamic multicast group.

With a static multicast group, the CST problem can again be attacked in two ways: i) the centralized algorithm, and ii) the distributed algorithm. I have worked on both centralized and distributed algorithms for the CST. In order to evaluate the heuristics, I compared the costs of the approximate CSTs generated by the heuristics against the cost of the optimal solution, found by enumeration. The result was that the approximate CSTs were, on the average, 5-10% more expensive than the optimal solution. Furthermore, I compared the approximate solutions against the "default" routing algorithm, namely the shortest delay tree. The shortest delay trees were as much as 70% more expensive than the approximate solutions, on the average. The distributed algorithms produce trees that are 15-20% costlier than the optimal solution.

I am currently working on the second problem, the dynamic multicast group. Here, the multicast group builds up over time, as members join the group or leave it. The tree has to be built up incrementally. As with other on-line algorithms, the approach is to compare heuristic tree costs with the optimal tree, assuming that the optimal solution knows beforehand who the members of the group are.

Another aspect of my research involves developing a multicast abstraction that would allow typical multimedia applications to be written naturally. The socket abstraction of UNIX is not sufficient to capture the generality of multimedia multicast communication. The "multimedia multicast channel" allows dynamic group formation, transmission of multiple data streams, synchronization points, and filters. Filters are procedures that apply to data on a stream, and can be used to tailor the data according to the host capabilities, e.g., applying a band pass filter to high fidelity sound because the host has low fidelity speakers. These filters may be allowed to propagate into the network to minimize network traffic, when possible.

The research group I work with is involved in a large collaborative effort called Project Sequoia 2000. It is aimed at providing Earth Scientists with adequate tools to tackle the grand challenge problems of Global Change. Sequoia 2000 is a joint venture of five University of California campuses, the San Diego Supercomputer Center, and Scripps Institute of Oceanography, and is funded extensively by Digital Equipment Corp. My contribution to Sequoia 2000 has been in helping set up a dedicated point-to-point network consisting of T1 (1.5Mbps) lines.

**References**


1. Pasquale, J.C., Polyzos, G.C., Anderson, E.W., Kompella, V.P., "The Multimedia Multicast Channel," *Third International Workshop on Network and Operating Systems Support for Digital Audio and Video*, San Diego, Nov. 1992.
2. Kompella, V.P., Pasquale, J.C., and Polyzos, G.C., "Multicast Routing for Multimedia Applications," *Proc. of '92 IEEE INFOCOM*, Florence, May 1992.
3. Kompella, V.P., Pasquale, J.C., and Polyzos, G.C., "Two Distributed Algorithms for Multicasting Multimedia Information," submitted to *International Conference on Computer Communications and Networking*, Mar. 1993.
4. Kompella, V.P., Pasquale, J.C., and Polyzos, G.C., "Multicast Routing Algorithms for Multimedia Communication," submitted to *IEEE Trans. on Comm.*, Jan. 1993.

## Low Bandwidth Video Conference System

*Tuong(Tom) Q. Nguyen*

Video requires a large amount of information to describe. This is one of the main reasons why video applications, especially those requires real-time processing such as videoconferencing, aren't widely used on workstations today. For instance, digitized video stored on the frame buffer usually requires 24 bits to describe one pixel. A modest video frame size of 256x256 pixels would require about 1.5 Mbits of data. To get good quality motion picture, one needs 30 frames/second, which totals 47 Mb/s.

One obvious problem with dealing with a large amount of data is transporting it over the network. Many of today's networks are of Ethernet technology, and have only 10 Mb/s maximum capacity. Another problem is transporting data within the system (e.g., moving from X Server to user's domain). It has been shown in our system that copying data takes up more than half the processing time.

I work with Dr. Pasquale and Jonathan Kay on a videoconference system that would have low bandwidth utilizing only standard hardware. Our first goal was to reduce the amount of data we need to process without losing vital information. The first step was to dither video from 24 bits to either 8 bits or 1 bit. This alone reduced the data size to 1/3 or 1/24, respectively, of its original size. The second step was compression, where we implemented time-domain compression and noise reduction. The third step was sampling, a technique which guesses at the location of important information on the frame, and only processes those.

Our current system can process 24-bit video to 8-bit or 1-bit. For the 1-bit receiver, the maximum frame rate is about nine frames/second with bandwidth of 68 Kb/s.

Our current bottleneck is the copying of data from the X Server's memory to the user's space. More than half of our processing time is spent there. Thus I am currently adjusting the X Server to get more direct access to the data. In the future, we will integrate the video with voice, and turn more attention to networking aspects.

# I/O Analysis of Scientific Applications

*Barbara K. Pasquale*

With the aid of high performance computing systems and sophisticated graphics devices, scientific research based on computational approaches has intensified. Current modes of research, involving complex, simulation-based modeling, require massive amounts of diverse data to be stored, organized, accessed, distributed, analyzed and visualized, and exert greater demands on existing I/O systems. In particular, Global Change research in fields such as oceanography, atmospheric science, and geology, relies on and produces environmental data sets ranging from hundreds of megabytes up to tens of gigabytes per execution. The combined effects of applications like these have the potential to create more I/O bound jobs which can become a bottleneck to system performance. However, we believe that knowledge of the regular and predictable pattern of scientific application I/O can be used to improve I/O system performance and to maintain well-balanced supercomputer systems.

We conducted a static I/O workload characterization study of the Cray YMP at the San Diego Supercomputer Center to determine the extent of scientific applications with large I/O demands and to study their characteristic I/O behavior[1]. The results of this study showed that 50 distinct scientific applications, from the YMP workload, were responsible for 71% of total system bytes transferred and only 9.5% of total system CPU time during a one-month period. The individual, average I/O rates for these applications ranged from 7MB/sec. to 132MB/sec. Detailed analysis of more than half of these I/O intensive applications, revealed a high degree of regularity in both the functional operation and resource usage of the individual applications. In particular, I/O demands, in terms of the total number of bytes transferred and the total number of logical I/O requests, showed considerable correlation with total CPU time consumed. We also observed that individual applications exhibited (1) both "short" (i.e., on the order of seconds) and "long" (i.e., on the order of minutes) CPU execution times, (2) fairly stable average logical I/O rates across different execution time periods, and (3) highly consistent in-core memory usage. Although these findings support our hypothesis that scientific applications have a high degree of regularity with respect to I/O resource usage, they cannot provide any insight as to how these resources are used over time. Determining the regularity or predictability of the behavior of application I/O can only be accomplished through dynamic monitoring of resource usage events during execution.

Using IPS, a newly released performance monitoring tool [2], we have begun analyzing the dynamic I/O behavior of Global Change applications. Collecting detailed information about each I/O event, allows us to construct run-time profiles depicting how an application utilizes its I/O resources over time. Based on the type of event, frequency, and intensity of I/O activity, we have been able to identify distinct computation phases within the application and to characterize, almost to the second, the recurring intervals of I/O activity.

**References**


[1] Pasquale, B. K. and Polyzos, G. C. "A Static Analysis of I/O Characteristics of a Broad Class of Scientific Applications," *UCSD Technical Report No. CS92-270*, November 1992.

[2] Miller, B. P., et. al. "IPS-2: The Second Generation of a Parallel Program Measurement System," *IEEE Transactions on Parallel and Distributed Systems*, Vol. 1, No. 2, April 1990, pp. 206–217.

## Performance Guarantees in Packet-Switched Networks

*Hanrijanto Sariowan*

My research interests are in the area of performance guarantees for high-speed packet-switched networks. As the Asynchronous Transfer Mode is slowly, but firmly, being accepted as the standard transport mechanism for future broadband networks, there has been an increasing interest on providing throughput, delay, and jitter guarantees to individual sessions in the network. These guarantees, which can be either statistical or deterministic, are of particular importance for applications which have stringent time requirements such as real-time audio and video communication.

I have analyzed a model for a multiplexer with two periodic inputs and general service time distributions, and obtained an exact expression for the transform of the waiting time distribution. The original intention was to use this multiplexer model as a source model in the analysis of a network consisting of the source, intermediate switching nodes, and the sink, with a final goal of characterizing performance guarantees of such a network.

Using the terminology of [Kurose, 1993], my approach can be categorized as an approximate approach to providing performance guarantees. Recently, there has been a greater interest in bounding approaches to network analysis because they seem to be more tractable for end-to-end performance analysis. Therefore, I am now exploring various avenues for extending the results of my research, including bounding approaches.

### References

**[1]** Kurose, Jim "Open Issues and Challenges in Providing Quality of Service Guarantees in High-Speed Networks," *Proceedings of the IEEE*, Vol. 23, No. 1, January 1993, pp. 6–15.

## 9 CSL Publications


**[B92]** Ben Taarit, R. "Extending FDDI Synchronous Mode Guarantees to Interconnected Rings," M.Sc. Thesis, Department of Computer Science and Engineering, University of California, San Diego, La Jolla, CA, September 1992.

**[BP92]** Billard, E. and Pasquale, J., "Effects of Periodic Communication on Distributed Decision-making," *Proc. IEEE International Conference on Systems, Man, and Cybernetics*, Chicago, October 1992, pp. 49-54.

**[BP93]** Billard, E. and Pasquale, J., "Effects of Delayed Communication in Dynamic Group Formation," *IEEE Trans. Systems, Man, and Cybernetics*, accepted for publication, January 1993.

**[BP93a]** Billard, E. and Pasquale, J., "Dynamic Scope of Control in Decentralized Job Scheduling," *Proc. 1st International Symposium on Autonomous Decentralized Systems*, Kawasaki, Japan, March 1993.

**[B+93]** Braun, H.W., Chinoy, B., Claffy, K., and Polyzos, G.C., "Analysis and Modeling of Wide-Area Networks: Annual Status Report," Technical Report GA-A21224, SDSC and UCSD, La Jolla, CA, February 1993.

**[B+92]** Braun, H.W., Claffy, K., Chinoy, B., and Polyzos, G.C. "Analysis and Modeling Tools for High-Speed Networks: Project Status Report," Technical Report CS92-237, Dept. of Computer Science and Engineering, University of California, San Diego, La Jolla, CA, April 1992.

**[CP92]** Claffy K., and Polyzos, G.C., "Location Transparent Connection Management: A Survey of Protocol Issues," *Proc. 11th Annual IEEE International Phoenix Conference on Computers and Communications*, Scottsdale, AZ, April 1-3, 1992.

**[CPB92]** Claffy, K., Polyzos, G.C., and Braun, H.W., "Traffic Characteristics of the T1 NSFNET Backbone," Technical Report CS 92-252, Dept. of Computer Science and Engineering, University of California, San Diego, La Jolla, CA, July 1992.

**[CPB93]** Claffy, K., Polyzos, G.C., and Braun, H.W., "Traffic Characteristics of the T1 NSFNET Backbone," *Proc. IEEE INFOCOM '93*, San Francisco, CA, March 28 - April 1, 1993.

**[CPB93b]** Claffy, K., Polyzos, G.C., and Braun, H.W., "Measurement Considerations for Assessing Unidirectional Latencies," *Internetworking: Research and Experience*, vol. 4, no. 2, 1993.

**[E93]** Etherton, D.C., "Approximating FDDI Synchronous Mode Guarantees Through Kernel Modification," M.Sc. Thesis, Department of Computer Science and Engineering, University of California, San Diego, La Jolla, CA, March 1993.

**[FP93]** Fall, K., and Pasquale, J., "Exploiting In-Kernel Data Paths to Improve I/O Throughput and CPU Availability," *Proc. USENIX Winter Conference*, San Diego, January 1993.

**[FPP92]** Ferrari, D., Pasquale, J., and Polyzos, G.C., "Network Issues for Sequoia 2000," *Proc. IEEE Compcon Spring '92*, San Francisco, CA, February 24-28, 1992.

**[G90]** Glockner, A. "Coadaptive Decision-making in Distributed Computer Systems," Ph.D. Thesis, Department of Computer Science and Engineering, University of California, San Diego, December 1990.

**[GP92]** Glockner A., and Pasquale, J., "Co-adaptive Decision-making in Distributed Systems," *IEEE Transactions on Systems, Man, and Cybernetics*, accepted for publication.

**[GP93]** Glockner, A., and Pasquale, J., "Coadaptive Behavior in a Simple Distributed Job Scheduling System," *IEEE Transactions on Systems, Man, and Cybernetics*, March 1993.

**[J92]** Johnson, M.M., "FDDI Performance Guarantees for Real-Time Traffic in the Absence of a Synchronous Mode," M.Sc. Thesis, Department of Computer Science and Engineering, University of California, San Diego, La Jolla, CA, December 1992.

**[KPP92]** Karvelas, D., Papamichail, M., and Polyzos, G. "The Rotating Slot Generator Dual Bus," *Proc. of the 23rd Pittsburgh Conference on Modeling and Simulation*, Pittsburgh, PA, April 1992.

**[KPP92c]** Karvelas, D., Papamichail, M., and Polyzos, G.C., "Performance Analysis of the Rotating Slot Generator Scheme," *Proc. IEEE INFOCOM '92*, Florence, Italy, May 4-8, 1992.

**[KP92]** Kay, J., and Pasquale, J., "Network Sloth: Where the Time Goes in Network Software," Technical Report CS92-238, Dept. of Computer Science and Engineering, University of California, San Diego, La Jolla, CA, April 1992.

**[KP93]** Kay, J., and Pasquale, J., "Measurement, Analysis, and Improvement of UDP/IP Throughput for the DECstation 5000," *Proc. USENIX Winter Conference*, San Diego, January 1993.

**[KP93a]** Kay, J., and Pasquale, J., "A Summary of TCP/IP Networking Software Performance for the DECstation 5000," *ACM Conference on Measurement and Modeling (SIGMETRICS)*, (extended abstract), Santa Clara, CA, May 1993.

**[KPP91]** Kompella, V.P., Pasquale, J.C., and Polyzos, G.C., "Multicast Routing for High-Bandwidth Delay-Sensitive Applications," Technical Report CS91-185, Dept. of Computer Science and Engineering, University of California, San Diego, La Jolla, CA, March 1991.

**[KPP92a]** Kompella, V., Pasquale, J., and Polyzos, G., "Multicasting for Multimedia Applications," *Proc. IEEE INFOCOM Conference on Computer Communications*, Florence, Italy, May 1992, pp. 2078-2085.

**[M90]** Muller, K. "Multi-structured File System Design for High Performance Computing," Ph.D. Thesis, Department of Computer Science and Engineering, University of California, San Diego, December 1990.

**[MP91]** Muller K., and Pasquale, J., "A High-performance Multi-structured File System Design," *1991 ACM Symposium on Operating System Principles (SOSP) Proceedings*, Asilomar, CA, October 1991.

**[P88]** Pasquale, J., "Using Expert Systems for Managing Distributed Systems," *IEEE Network*, Vol. 2, No. 5, Sept. 1988.

**[P89]** Pasquale, J., "The Effects of Using Aging State Information in Decentralized Load Balancing," Technical Report CS89-161, Dept. of Computer Science and Engineering, University of California, San Diego, La Jolla, CA, October 1989.

**[P89b]** Pasquale, J., "Decentralized Resource Management Research in Large Distributed Computer Systems," Technical Report CS89-162, Dept. of Computer Science and Engineering, University of California, San Diego, La Jolla, CA, December 1989.

**[P90]** Pasquale, J., "Beyond Client Server Architectures," *NCR Perspective*, Vol. 4, No. 2, Spring 1990.

**[P91]** Pasquale, J., "Decentralized Control in Large Distributed Systems," *Coordination Theory and Collaboration Technology Workshop Proceedings*, National Science Foundation, Washington D.C., June 1991, pp. 9-20.

**[P91a]** Pasquale, J., "Decentralized Control in Large Distributed Systems," *Proc. NSF Coordination Theory and Collaboration Technology Workshop*, National Science Foundation, Washington D.C., June 1991.

**[P92b]** Pasquale, J., "I/O System Design for Intensive I/O," *IEEE Workshop on Workstation Operation Systems (WWOS) Proceedings*, Key Biscayne, FL, April 1992.

**[P93]** Pasquale, J., "Randomized Coordination in an Autonomous Decentralized System," *Proc. 1st International Symposium on Autonomous Decentralized Systems*, Kawasaki, Japan, March 1993.

**[P93a]** Pasquale, J., "System Software and Hardware Support Considerations for Digital Video and Audio Computing," *Proc. Hawaii International Conference on System Sciences (HICSS)*, Maui, Hawaii, January 1993.

**[P+90]** Pasquale, J., Anderson, E., Bubien, M., Fall, K, Kompella, V., McMullan, S., Ranganathan, D., and Terek, R. "Operating System and Window System Research for Multi-media Applications: A Status Report," Technical Report CS90-176, Dept. of Computer Science and Engineering, University of California, San Diego, La Jolla, CA, July 1990.

**[PBK91]** Pasquale, J., Bittel, B., and Kraiman, D. "A Workload Characterization Study of a Cray XMP Running CTSS," Technical Report CS91-222, Dept. of Computer Science and Engineering, University of California, San Diego, La Jolla, CA, April 1991.

**[PBK91b]** Pasquale, J., Bittel, B., and Kraiman, D., "A Static and Dynamic Workload Characterization of the SDSC Cray XMP," *1991 ACM SIGMETRICS Proceedings* (extended abstract), San Diego, May 1991.

**[PFF92]** Pasquale, J., Fall, K., and Forrest, J., "The Sequoia 2000 Network (S2Knet) Handbook," Sequoia 2000 Technical Report 92/5, U. C. Berkeley, November 1992.

**[PP90]** Pasquale, J. and Polyzos, G. C. "System Support for Multimedia Applications," *Proc. First International Workshop on Network and Operating System Support for Digital Audio and Video*, International Computer Science Institute, Berkeley, CA, November 8-9, 1990.

**[PP92]** Pasquale, B. K. and Polyzos, G. C., "A Static Analysis of I/O Characteristics of a Broad Class of Scientific Applications," Technical Report CS 92-270, Dept. of Computer Science and Engineering, University of California, San Diego, La Jolla, CA, November 1992. (Also, Sequoia 2000 Technical Report 92/19)

**[PPAK92]** Pasquale, J., Polyzos, G., Anderson, E., and Kompella, V., "The Multimedia Multicast Channel," *Proc. 3rd International Workshop on Network and Operating System Support for Digital Audio and Video*, San Diego, November 1992.

**[P+91]** Pasquale, J., Polyzos, G. C., Anderson, E. W., Fall, K. R., Kay, J. S. , Kompella, V. P., McMullan, S. R., and Ranganathan, D. "Network and Operating System Support for Multimedia Applications," Technical Report CS91-186, Dept. of Computer Science and Engineering, University of California, San Diego, La Jolla, CA, March 1991.

**[PPAK92]** Pasquale, J., Polyzos, G. C., Anderson, E. W., and Kompella, V. P., "The Multimedia Multicast Channel," *Proc. Third International Workshop on Network and Operating System Support for Digital Audio and Video*, La Jolla, CA, November 12-13, 1992.

**[P+91b]** Pasquale, J., Polyzos, G. C., Fall, K., Kompella, V. "Internet Throughput and Delay Measurements Between Sequoia 2000 Sites," Technical Report CS91-219, Dept. of Computer Science and Engineering, University of California, San Diego, La Jolla, CA, December 1991.

**[PPFK91]** Pasquale, J., Polyzos, G., Fall, K., and Kompella, V., "Internet Throughput and Delay Measurements Between Sequoia 2000 Sites," Sequoia 2000 Technical Report 91/7, U. C. Berkeley, December 1991 (also available as U. C. San Diego Technical Report CS91-219, December 1991).

**[PPK93]** Pasquale, J., Polyzos, G. C., and Kompella, V. P., "Real-time Dissemination of Continuous Media in Packet-Switched Networks," *Proc. IEEE Compcon Spring '93*, San Francisco, CA, February, 1993.

**[P92]** Polyzos, G. C., "Networking for Sequoia 2000," *Proc. 4th IEEE COMSOC International Workshop on Multimedia Communications*, MULTIMEDIA '92, Monterey, CA, April 1-4, 1992.

**[P93b]** Polyzos, G. C., "Hierarchical Coding of Images and Continuous Media for Transmission over Packet-Switching Networks," Technical Report CS 92-273, Dept. of Computer Science and Engineering, University of California, San Diego, La Jolla, CA, January 1993.

**[PMV87]** Polyzos, G.C., Molle, M.L., and Venetsanopoulos, A.N., "Performance Analysis of Finite Non-homogeneous Population Tree Conflict Resolution Algorithms Using Constant Size Window Access," *IEEE Transactions on Communications*, Vol. COM-35, No. 11, pp. 1124-1138, November 1987.

**[PM90]** Polyzos, G. C. and Molle, M. L. "Conflict Resolution Algorithms for High Error-rate Multi-access Channels," *IEEE International Symposium on Information Theory*, San Diego, CA, January 1990.

**[PM90a]** Polyzos, G.C. and Molle, M. L. "Delay Analysis for the FCFS 0.487 Conflict Resolution Algorithm," Technical Report CS90-179, Dept. of Computer Science and Engineering, University of California, San Diego, La Jolla, CA, October 1990.

**[PM93]** Polyzos, G. C. and Molle, M. L., "A Queueing Theoretic Approach to the Delay Analysis for the FCFS 0.487 Conflict Resolution Algorithm," *IEEE Transactions on Information Theory* (in press).

**[TP91]** Terek, R., and Pasquale, J., "Experiences with Audio Conferencing Using X, UNIX, and TCP/IP over an Ethernet," *1991 Summer Usenix Proceedings*, Nashville, June 1991.