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 1991

UCSD Technical Report CS91-181

Abstract

At the Computer Systems Laboratory at UCSD, we conduct research in distributed systems, operating systems, network protocols, and performance evaluation. This document reviews the current research within the Computer Systems Laboratory. There are four sections which describe the work of the four groups within the laboratory: Decentralized Control Research Group; Operating Systems Research Group; Communication Networks Research Group; Performance Evaluation Research Group. The final section contains descriptions of research projects undertaken by our graduate students.

This research is supported in part by the National Science Foundation, the Powell Foundation, IBM Corporation, NCR Corporation, Digital Equipment Corporation, and the UC Micro Program. For more information, send mail to the authors at *Computer Systems Laboratory Department of Computer Science and Engineering, University of California at San Diego, La Jolla, CA 92093-0114.* 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@cs.ucsd.edu* or by telephone at (619) 534-3508.

1. Introduction

At the Computer Systems Laboratory at UCSD, we conduct research in distributed systems, operating systems, communication networks, and performance evaluation. In distributed systems, we are investigating mechanisms for decentralized control for effective group decision-making and distributed resource allocation. 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 the area of performance evaluation, we are interested in characterizing supercomputer workloads, such as that of the Cray Y-MP at the San Diego Supercomputer Center.

This document reviews the current research within the Computer Systems Laboratory. In the next four sections, we describe an overview of the work of the four research groups within the laboratory: the Decentralized Control Research Group, the Operating Systems Research Group, the Communication Networks Research Group, and the Performance Evaluation Research Group. The last section contains research summaries of the graduate students in each group.

2. Decentralized Control Research Group

In the Decentralized Control Research Group, our goal is to understand the fundamental problems of decentralized control, and develop techniques for overcoming them. 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: updating the global state; multi-agent coordination; and group formation. Our approach to global state updating 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. We are working on extending this idea to large numbers of communicating agents.

We are investigating whether randomization can be used as an effective mechanism for multi-agent coordination. Consider a system of m managers and r resources, where each manager's goal is to allocate one of the r resources. An instance of the coordination problem is to determine how to allocate the resources so that each resource is allocated by exactly one manager (if no manager allocates the resource it is wasted, and if more than one try to allocate, conflicts occur). We are studying the effectiveness of a technique we call time-space randomization to reduce the occurrence of mutually conflicting decisions. The idea is for each manager to make two decisions: (1) should I allocate a resource now; (2) if ‘‘yes,’’ which resource should be allocated. The first decision is decided by flipping a biased coin, where the probability of ‘‘yes’’ is some value ρ. The second decision is decided by choosing one of the r resources with equal likelihood. If an agent decides ‘‘no’’ for (1), all this means is that it will reconsider the question at a later time (e.g., the next time step). Thus, decision (1) allows agents to randomize their allocations over time, and decision (2) allows agents to randomize over the space of resources. We have determined an optimal value for ρ given m managers and r resources for a variety of gain functions describing system performance.

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).

Graduate students working in the area of decentralized control are Ted Billard (uncertainty due to communication delays, distributed game automata), Alex Glockner (co-adaptive behavior based on learning automata).

3. Operating Systems Research Group

The Operating Systems Research Group is currently focusing on investigating system software structures which support I/O intensive applications. 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. This includes work on kernel design based on fast device communication, window systems generalized to include audio and video, and file system design based on disk arrays.

Consider a program which emits motion video at a rate of 30 frames/sec in a window with resolution 640 x 480 pixels. If each pixel requires three bytes for describing color, the total output bandwidth required by this program is just over 220 megabits/sec. Furthermore, there is a real-time delay constraint which must be met: each frame must be delivered to the video interface every 40 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 quickly become overwhelming.

In the area of operating systems, we are investigating real-time resource scheduling (so that threads processing continuous media are able to meet deadlines), intermedia synchronization (how to synchronize related audio and video streams), and fast communication between processes and devices (by limiting memory copies, transferring compressed data when possible, and constructing in-kernel data paths for device to device communication).

In additions, we are studying how to organize the operating system's I/O software such that information movement between devices is minimized. 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.

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). A reorganization of the data flow between stream modules is required so that memory is copied at most once between the source and destination (or no copy at all if memory can be shared by the source and destination).

Scheduling and synchronization is 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 tranmission 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.

In the area of window systems, we are investigating how to extend the user window interface to include video as well as audio. Video windows are simply windows containing motion video. Audio windows are spatially located sound sources. Often, a video and audio window are co-located, so that sound emanates from the location of the video window. This is particularly effective in audio/video conferencing, where the location of sound gives the user a cue as to who is talking, and where to look.

We have already 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. 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, our 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.

Finally, we are considering file system designs based on disk arrays which provide high throughput for large files, and yet low delays for small files. This is based on a combined log-based storage and striped storage design, and carefully separating file control information and data. Our initial results indicate order-of-magnitude improvements in throughput without increasing delay.

The graduate students in the Operating Systems Research Group are Eric Anderson (graphics and video device support, multicast channel design), Kevin Fall (I/O interfaces, device models), Scott McMullan (multicast channel design), Keith Muller (multistructured file systems), Dipti Ranganathan (multiuser programming abstractions), and Robert Terek (window systems supporting audio and video).

4. Communication Networks Research Group

In the Communication Networks Research Group, we investigate network architectures and protocols supporting *Digital Continuous Media* (DCM), i.e., digital audio, video, and other digital media in which data is produced or consumed in a periodic manner. Audio and video are often associated with real-time applications such as teleconferencing. The interactive nature of this application imposes stringent delay requirements on the audio and video flows. Various guidelines set the end-to-end delay tolerance to tens or a few hundreds of milliseconds. Meeting these delay requirements may require dropping packets when they accumulate between the sender and the receiver. Fortunately, humans receiving real-time audio and video have some tolerance to transmission errors. For example, for telephone quality audio, a 1-2% data loss during talkspurts is usually acceptable. 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 for acceptable behavior, but has some tolerance to data loss when the requirements cannot be met. New hierarchical (or component) coding techniques split DCM signals into subcomponents of varying importance. Less important 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 drop in service quality. This is important, as it gives the system software the capability of deciding how to gracefully degrade transmission dynamically when resources are limited.

Another important characteristic of DCM is their very high traffic volume. For example, motion video (640×480 pixel-frames transmitted at 30 frames per second), requires about 220 Mb/s of bandwidth. This bandwidth can be significantly reduced with efficient coding schemes and compression techniques. Thus, for most broadcast quality video, 30-45 Mb/s is sufficient. However, throughput requirements such as these are beyond the capabilities of most current networks and protocols, even though link transmission speeds can support them. The reason is that fiber-optic technology has moved the bottleneck of networking from the link transmission speed to the speed of switching and protocol processing. Furthermore, unfortunately, this problem is deeply rooted into the networking architectures, which do not seem to scale-up well to satisfy the requirements of DCM. New high-speed network architectures try to address this new state of affairs by making every effort to minimize the processing required in each protocol layer and eliminate the unnecessary copying of information between layers. However, even though many designs consider high-speed networks and the problems associated with the very large amounts of information ‘‘stored’’ in transit (due to the large bandwidth-propagation delay product), there has been little attention paid to the specific characteristics and requirements of real-time DCM.

We are investigating network and transport layer protocols that provide various levels of quality of service guarantees by exploiting connection-based resource reservation mechanisms which take advantage of the periodicity and the specific statistical properties of each DCM flow. We are in the process of experimentally evaluating the performance of current protocols and protocol implementations, specifically those of the DoD Internet protocol suite, in conjunction with high-speed LANs, such as FDDI, and DCM. In addition, we plan to modify these protocols, by eliminating some functions which are of limited use to DCM, and repeat the performance evaluation in order to determine the degree to which current protocols could be useful for supporting DCM traffic.

Due to the high data volume of DCM it is important to provide efficient data transfer mechanisms even when high-speed fiber-optic networks are available. 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 the network may be unduly overloaded. Broadcasting sends the data to many more nodes than required. Multiple unicasts could use the same link several times to carry the same data, and the sender 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 reducing the number of copies of data sent over the network. This is achieved by delaying duplication of data until the routes to different destinations diverge. In addition to reducing bandwidth requirements for multimedia communication, routes that have low end-to-end delay from the source to possibly multiple destinations must be found. We would like to have data paths that avoid crossing too many network relay nodes and many abstraction layers (e.g., protocol layers) because they introduce delay in data delivery. This delay may be due to switching, queuing, protocol conversion, etc. Propagation delay must also be added to the equation. However, taking into consideration the fact that delay is tolerable until it exceeds some threshold level, a good routing algorithm may not have to minimize the delay along the route; bounding the delay should prove 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 some heuristics for this problem which seem very promising. The heuristics are based on an abstraction of the problem from a full network to that of a complete graph with vertices the members of the multicast group (only), and edges the shortest paths (with respect to bandwidth — or link count) between any two such members. Thus, trees on this much smaller graph are much easier to track. We are investigating the performance of these schemes both empirically and analytically, on random graphs and graphs resulting from specific network topologies.

Finally, we consider 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 to justify exponential interarrival times assumptions, i.e., the Poisson assumption, or even makes it completely unusable. 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.

Graduate students working in the area of communication networks are Mark Bubien (multicast channel implementation), Kimberly Claffy (support for high demand digital continuous media), Jonathan Kay (protocol extensions for audio and video communication), and Vachaspathi Kompella (multicast routing algorithms).

5. Performance Evaluation Research Group

In the Performance Evaluation Research Group, our goal is to understand the performance limitations of supercomputers, and to develop tools for their measurement and evaluation. 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.

We have conducted a preliminary workload characterization study of the SDSC Cray X-MP (the predecessor to the Cray Y-MP), running the CTSS operating system. This workload characterization was in terms of the types of jobs executed (i.e., functional level characterization), and what resources those jobs required (i.e., resource or system level characterization). We used existing measurement probes in the operating system, which generated negligible interference, to obtain a trace of all jobs executed during a two month interval, along with their resource consumptions. We then clustered jobs according to their type, and resource usage, using a clustering algorithm we developed, based on the classical *k-means* clustering method.

From this study, we learned that most of the jobs submitted, approximately 95%, are executions of public utilities and various interactive programs, which consumed a small percentage, approximately 10%, of the total CPU time. The remaining 5% of jobs were were user codes consuming the remaining 90% of total CPU time; these jobs are the "scientific applications" which we expected to see on this supercomputer. Although the public utilities and interactive jobs make relatively small demands on the CPU, we observed that these jobs consume 22% of the total memory space-time product, and 75% of the total I/O channel time, the latter being a very significant resource demand made by the non-scientific application component of the workload. We have been investigating the impact this has on the scientific applications.

We are conducting a similar workload characterization study for the Cray Y-MP running UNICOS, the Cray version of the Unix operating system. This system provides more detailed and more accurate measurements of resource demands made by individual user jobs, as well as more global measures of system activity. This study will tell us whether the Y-MP workload is any different from that of the X-MP, and one of our goals will be to determine whether this is due to the differences in the machine, or the operating system, or whether the user community has somehow changed. We are also analyzing how system behavior changes over time, to develop an understanding of how the workload changes as a function of time for each type of resource, such as the CPU, memory, and disk I/O. With this understanding, our goal is to build prediction models which the operating system may use to dynamically anticipate resource demands based on workload behavior.

Finally, we have built a general-purpose queueing network simulator which allows users to quickly construct a network of user-definable servers and obtain standard results such as average number of jobs, utilization, throughput, and average system time. This simulator is written in the C programming language, runs on a wide variety of machines, and is available upon request.

The graduate students in the Performance Evaluation Research Group are Barbara Bittel (methodologies for supercomputer workload characterization), Daniel Kraiman (measurement techniques for multiprocessor supercomputers).

6. Graduate Student Research Summaries

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

System Software Support for Group Exchange of Live Motion Video

Eric Anderson

Operating Systems Research Group

My recent work has been devoted to examining various issues regarding the incorporation of motion video into computer workstations. I have looked at this problem at many levels, from where to place a video camera, down to the services required in the operating system kernel. In particular, we wish to determine the hardware and software support which is required to achieve the display of multiple real-time video streams which are received from a network. This requires the efficient and timely management of a large amount of data, so the problem is essentially one of allocating the available resources appropriately.

In addition to allocating resources, we would like to provide multiple levels of service quality, so that some amount of video service can be offered even when insufficient resources are available for full service. Service degradation can be achieved by reducing the frame rate, pixel depth, or frame resolution. Ultimately, we would like to decompose the video stream into packets of varying priority. High priority packets contain adequate information to provide reduced quality service, while lower priority packets contain extra information. When these packets are reassembled, full quality video is produced. If some low priority information is discarded because resources are not available to transmit it, then reduced quality video is still available at the receiving end.

In order to make this complex service available to users, we have designed a service entity called a Channel. The simple Channel is a virtual network shared by a group of processes. Each process can broadcast messages into this virtual network which are received by the other processes using the Channel. In an attempt to make the Channel more useful and efficient we have dressed it up with a variety of features, including name service, subgroups, selective control over the existence of streams, and multiple media streams. While these features add a great deal of flexibility to the channel, they also complicate the implementation immensely. My current goal is to segregate the channel into two or more layers, so that the underlying mechanism can be kept simple.

Must of the design for the channel has been motivated by the results of implementing an earlier design, which was based on Unix signals. When we attempted to use this early version to implement a conferencing program we discovered that it was somewhat inconvenient to use. Our current model eliminates signals in favor of a message based interface. By implementing it as several layers, I hope to determine where further improvements can be made to offer this service as conveniently as possible without complicating the kernel support level. If these levels work well together, I believe that they can eventually be collapsed into one integrated entity, most (or all) of which is contained in the kernel, which provides very efficient multimedia communication services.

Modeling Uncertainty in Distributed Systems

Ted Billard

Distributed Systems Research Group

My interest is in large distributed systems in which decision-makers or agents must make good, though not necessarily optimal, decisions. It is assumed that the costs of communication and the nature of the systems do not permit high bandwidth communication in making each decision. My work is influenced by the *common-knowledge* problem [1] which decision-makers cannot know the level of knowledge obtained by others. As a result, each decision-maker has its own *view* of the global state. Frequent communication decreases the uncertainty in global state but has a cost which can adversely affect the performance. The problem is to make good decisions with low communication of state information.

The problem is for each distributed decision-maker to make good decisions in three areas. First, the decision-maker must decide *with whom* to coordinate. This implies finding a subset of decision-makers, from a possibly large network, that have compatible strategies, resources and constraints. Second, the decision-maker must decide *what actions* to perform as a member of the selected subset. Third, the decision-maker must decide *how often* to communicate information concerning global state. This affects the communication overhead and the uncertainty in the global state. In all cases, the decisions affect the overall performance of the system and each decision-maker has an uncertainty about the decisions made by others.

I have created a model called distributed game automata which is similar to learning automata [2]. Each automaton uses a probabilistic strategy, e.g. a coin, to make decisions in a game. The game payoffs represent a performance measurement of the underlying distributed system. After making a move in the game, an automaton receives feedback on the quality of its decision. The feedback adjusts the probabilistic strategy by changing the weights of the coin. However, the feedback is based upon the view of the other automaton's probabilistic strategy. Since this view is maintained by communication, infrequent communication may distort the feedback.

I have created a simulator for this model to explore the trade-off between low communication rates and decision-quality, as measured by the resultant performance. In a load balancing game with two players, a severe drop in performance occurs when the communication rate decreases below an analytically predicted value. Another simple load balancing problem for three players shows the optimal group structures. There are configurations where a third player should be excluded from the group in order to improve system performance.

In the future, uncertainty *with whom* to coordinate will be introduced to observe the effects of delayed communication on the resultant structures. In a uniform treatment of this decision-making, the model includes a probabilistic strategy to select group membership. This allows for spontaneous coordination out of necessity similar to biological species [3]. Again, I expect to observe, and predict analytically, the communication rates that permit good decisions.

References

[1] J. Halpern and Y. Moses, ''Knowledge and Common Knowledge in a Distributed Environment,'' *PODC,* Vol. 3, pp. 50-61 (1984).

[2] K.S. Narenda and M.A.L. Thathachar, *Learning Automata: An Introduction,* Prentice-Hall, Inc., Englewood Cliffs, New Jersey (1989).

[3] R. Axelrod and W. Hamilton, ''The Evolution of Cooperation,'' *Science,* Vol. 211, pp. 1390-1396 (March 1981).

Supercomputer Workload Characterization

Barbara K. Bittel

Performance Evaluation Research Group

My current research pursuits are in the area of workload characterization for supercomputers. Supercomputers are expected to accelerate progress in many scientific disciplines because of the complex computations they can perform. Yet, little is known about the nature and composition of supercomputer workloads. The workload represents the demands generated by users in terms of hardware and software resources and must be accurately characterized for any meaningful performance evaluation study.

The goal of my workload characterization study of the San Diego Supercomputer Center's Cray XMP was to determine how a diverse community of users from industry, universities, and research institutions, actually use a supercomputer. By using accounting trace data collected over a 2-month period (10/1/89 - 11/20/89), I was able to characterize the overall structure of this period of the workload by combining a high-level functional description (jobs, programs) with a low-level resource usage description (CPU time, I/O time, memory usage over time).

This study involved 1) identifying each job in the trace period with a distinct command name; 2) classifying command names as "system" (system installed utility/application) or "user"; and 3) partitioning these command groups according to their resource usage profile. This resulted in command classes where members of a given class are either system or user programs, have the same command name, and have a similar resource-usage profile. Using the classes of each command as input to a k-means clustering algorithm, 7 characteristic groupings were found. The table below lists these 7 groups.

			Percentages of		
Cluster	Resource Usage	Frequency	CPU	I/O	Mem-Time
Sys-Low	low overall	46.60	.03	1.72	.01
Sys-Med	medium overall	41.05	1.50	28.50	.82
Sys-High-CPU	high CPU, other medium	1.68	7.24	3.18	11.39
Sys-High-IO	high I/O, other medium	4.35	1.42	41.25	10.11
		93%	10%	75%	22%
Usr-Low	low overall	1.75	.03	.19	.01
Usr-High	high overall	2.76	4.04	3.55	.94
Usr-Massive	extremely high overall	1.81	85.75	21.61	76.72
		7 %	90%	25%	78%

As this table indicated, 88% of all jobs are either in groups Sys-Low or Sys-Med, but these jobs use only 2% of the total CPU time. Only 2% of all jobs are in the group Usr-Massive, which uses 86% of the total CPU time. Usr-Massive was also the largest user of memory, utilizing 77% of the total memory space-time product (over all jobs). Three groups used 92% of the total I/O time: Sys-Med (29%), Sys-High-IO (41%) and Usr-Massive(22%). Comparing just system jobs to users jobs the results can be summarized as follows. 93% of all jobs are system jobs, but 90% of all CPU time is used by user jobs. However, this is not to say that system jobs have little effect on system resource usage: 75% of I/O time is due to system jobs and 22% of the total memory space-time product due to systems, which is not insignificant.

Based on this initial experience, I plan to continue studying and analyzing the nature and composition of supercomputer workloads for the purposes of creating representative benchmark sets. Supercomputers are capable of delivering the highest computational rates technology can achieve, but their measured operational values fall far below these peak computational rates. The discrepany between theoretical and actual computational rates is the direct result of the workload.

The Construction of a Multicast Communication Interface

Mark Bubien

Communication Networks Research Group

I am currently working with Vach Kompella and other members of the OSRG in implementing a multicast communication interface. True multicast allows communication between a group of computers while reducing both network traffic (as opposed to unicast) and the servicing of erroneous packets (as opposed to broadcast). Because we do not want to modify the UNIX kernel at this time, the interface will be built as a UNIX process. Currently we are working on an implementation for local area networks, and, once successful we will expand it to allow multicast in an internetwork.

At the process level, multicast can be simulated by using either multiple unicasts (send one packet per each receiver) or broadcast (send a single packet to every host on the LAN). Hughes [1] has shown that using broadcast to simulate multicast provides a more efficient response time than using multiple unicasts. Hughes' implementation, however, filters unwanted packets after they have been accepted by the Ethernet chip. Because multicast is provided at the Ethernet Data Link Layer [2], we would like to have the Ethernet chip to do the actual packet filtering.

To allow multicast packet filtering by the Ethernet chip, we attempted to use the functionality provided by TCP/IP. We created an IP address and used *arp* to change the Address Resolution Policy table to map that address to an Ethernet multicast address. Unfortunately, this method has not been successful since it does not provide us with enough control of the Ethernet Data Link Layer.

In order to gain to gain direct access to the Ethernet Data Link Layer we must use the Network Interface Tap (NIT). This will allow us to send and receive Ethernet multicast addresses explicitly.

Once the Ethernet multicast version is implemented, we will build an interface which functions in an internetwork. Vach Kompella is constructing a routing algorithm which will choose near-optimal paths for packets to travel between networks. We plan to implement this routing algorithm by expanding the completed multicast interface, allowing us to send packets to the chosen nodes and to use the LAN multicast once packets have reached the appropriate networks.

References

[1] Larry Hughes, ‘‘A Multicast Interface for UNIX 4.3’’, *Software — Practice and Experience*, Vol. 18, No. 1 (January 1988).

[2] J.F. Schoch, Y.K. Dalal, D.D. Redell and R.C. Crane, ‘‘The Ethernet’’, *Computer*, Vol. 15, No. 18 (August 1982).

Network Support for High Demand Digital Continuous Media

Kimberly Claffy

Communication Networks Research Group

My research currently focuses on network support for high-demand multimedia communication. I am interested in approaches to the design and implementation of networks in high-speed B-ISDN type realms. This includes both protocol design and evaluation, as well as network modelling, measurement, simulation, and performance evaluation.

My dissertation research objective will be to characterize traffic flows for various advanced applications across a high functionality network. Consideration of appropriate measurements, necessary features of existing and new tools, specific experiments, and algorithms and protocols to implement and account for different service qualities will play principal roles in my investigation.

Applications using digital continuous media, supporting various levels of video and audio quality, real-time database interaction and other network intensive data exchanges will require various qualities of service from the network. Modes of network interaction will also likely include special forms of addressing, such as multicast/multidrop, which are not well supported right now. An additional challenge will be the implementation of accounting and charging services in such an environment.

Network and transport layer protocols that utilize flexible gateway algorithms, dynamic reconfiguration, and real-time traffic analysis will become critical for providing diverse service requirements. Of particular interest are measurements or estimates of the variance of the end-to-end delay (jitter) which is of critical importance for digital continuous media, like video and other networking intensive data flows. Priority-based queuing, compression mechanisms, and voluntary sacrifice of service quality will all be important components of traffic management on high demand networks. I hope to explore and analyze these network managment techniques via a study of traffic patterns across wide-area networks with diverse service capacities.

Uniform I/O Interfaces

Kevin Fall

Operating Systems Research Group

The flexibility of a highly general, untyped byte stream abstraction for device and network interfaces has been effectively illustrated by the proliferation of the UNIX operating system The traditional UNIX untyped byte-stream abstraction was augmented in 1984 by the creation of *streams* [1] Despite their flexibility, streams have shortcomings when faced with continuous data composed of temporally synchronized multimedia information. The goal of my research is to suggest an appropriate set of flexible operating system interconnection mechanisms which may be employed to route time-sensitive data between devices and between processes and devices.

Availability of extremely diverse peripheral equipment has stressed the utility of the UNIX untyped byte-stream abstraction. Efforts such as Cheriton's UIO interface [2] extend the basic UNIX interface, but do not explicitly deal with the problems of temporal synchronization found in most multimedia applications. For example, an audio codec may produce audio samples at a regular rate of 8,000 samples each second. To implement a digital video conference, the output of a video codec must be synchronized in time with respect to the audio codec in order to produce an acceptable conferencing system. While some variation is possible, these samples must be delivered within a small maximal latency to any human receiver to be considered meaningful and must also arrive at the appropriate output devices at the same time. A small number of samples may be replayed if others are lost in transit, making low-latency the paramount requirement of the data paths interconnecting the originator and recipient. Work in this area includes Leung's work at Bell Labs [3], including a system-wide *connector* object, which is used to splice UNIX file descriptors together. An attached processor running its own operating system is employed to actually affect interconnections. My own work shall implement the interconnections within the local operating system.

To support the abstractions provided by the uniform I/O interface, efficient buffer management must be developed. Traditional buffer management schemes have included FIFO queueing with flow control located close to user-level processes. For example, the Berkeley UNIX socket level is primarily responsible for the implementation of any queueing policy. Under a streams-based approach, each stream module will have minimal queueing; only those modules which must perform internal queuing will do so (e.g. TCP retransmit queueing, IP fragmentation or reassembly queueing). While simple FIFO queueing may be sufficient for traditional data applications, isochronous data may call for different queueing behaviour, most likely based on priorities which allow some loss to occur. The notion of a *buffering toolkit,* which may be employed to implement various queueing behaviours for different devices is suggested by Momtahan and Kamel [4]. For optimal ease of use and convenience of experimentation, a dynamically-loadable streams module architecture will be developed to obviate the need to initiate a reboot when switching between a potentially large number of queueing disciplines.

To date, I have undertaken an investigation of the recent developments in the areas listed above. In the near term, I will be undertaking a design of a uniform I/O interface and the underlying buffering mechanisms to support this interface. The interface will be targeted for the isochronous data environment, but will also include support for traditional files and data. To export the interface across the network, I hope to incorporate other work such as the IETF's ST-II protocol [5] or Lixia Zhang's Flow protocol [6]. Availability of networks able to support reservations appears promising, with the development of networks like FDDI and Broadband ISDN.

References

[1] D. M. Ritchie, ''A Stream Input-Output System,'' *AT&T Bell Laboratories Technical Journal,* Vol. 63, No. 8, pp. 1897-1910 (October 1984).

[2] D. R. Cheriton, ''UIO: A Uniform I/O System Interface for Distributed Systems,'' *TOCS,* Vol. 5, No. 1, pp. 12-46 (February 1987).

[3] W. H. Leung, T. J. Baumgartner, Y. H. Hwang, M. J. Morgan, and S.-C. Tu, ''A Software Architecture for Workstations Supporting Multimedia Conferencing in Packet Switching Networks'' *IEEE Journal on Selected Areas in Communications,* Vol. 8, No. 3, pp. 380-390 (April 1990).

[4] P. Momtahan and R. Kamel, ''PX Connection Architecture,'' *First International Workshop on Network and Operating System Support for Digital Audio and Video,* Berkeley, CA (November 1990).

[5] C. Topolcic, Ed., ''Experimental Internet Stream Protocol, Version 2 (ST-II),'' RFC 1190, Network Information Center, SRI International Menlo Park, CA (October 1990).

[6] L. Zhang, ''A New Architecture for Packet Switching Network Protocols,'' Ph.D. Thesis, MIT (July 1989).

Co-adaptive Resource Management In Distributed Systems

Alexander Glockner

Distributed Systems Research Group

Decision-making problems in distributed systems often require cooperation among the entities in the system. We are investigating agents (decision-makers) that coordinate their actions by independently adapting their decision functions in ways that increase the global effectiveness of decisions. As each agent's decision function changes, the global decision function changes; this is called co-adaptation, literally "adapting together". As each agent finds its appropriate individual decision function, the collection of agents co-adapts into a good, but not necessarily optimal, global decision process.

To investigate co-adaptive decision-making, we are applying it to distributed scheduling of jobs in a distributed computer system. Jobs created at any machine can be executed on any other machine. Agents schedule these jobs among the machines in order to reduce the mean job turnaround time, how long a job needs to complete on average. Every machine has its own agent.

The effectiveness of co-adaptive decision-making with agents implemented as stochastic automata [3] has been established [1]. Simulations of machines in a distributed system were run with two-hour job traces taken from various machines, and it was demonstrated that stochastic automata quickly learn to outperform both a policy of only scheduling jobs on the local machine, i.e., the machine they were generated on, and a policy that scheduled jobs on other machines with equal probability.

To investigate the dynamic properties of the co-adaptive process, distributed systems with exponentially distributed job inter-arrival and service times have been simulated [2]. In simulations of two-machine systems, stochastic automata again quickly learned to perform better than the local-only and equal-probability policies.

Tradeoffs resulting from the adjustment of the automata's learning constants were demonstrated to occur between the frequency of offloading, i.e., of sending jobs to another machine, and the difficulty of adapting to long-term changes in job generation characteristics. As an example, an agent that adapted appropriately to long-term changes also tended to offload too many jobs when there were no changes.

Most importantly, it was shown that the general behavior of an automaton agent is solely determined by its own learning constants. The effectiveness of that behavior, however, is dependent on the characteristics of the other agents in the system as well as the environment.

When as many as six machines have been simulated, improvements in the mean response time consistenly occur, but agents spontaneously and unpredictably form groups; for instance, one group of agents would act as if they had agreed to schedule all their extra jobs to the other machines, and in some cases, the other machines would form a second group that reciprocates.

Future work will consist of refining the above results, showing the effects of different automata feedback metrics and of changes in network delay, and experimenting with heuristics to assist co-adaptation in distributed scheduling performance.

References

[1] A. Glockner, ''Co-adaptive Resource Management in Distributed Computing Systems,'' unpublished dissertation proposal, Department of Computer Science and Engineering, University of California, San Diego (1989).

[2] A. Glockner and J. Pasquale, ''Co-adaptive Decision-making in Distributed Systems,'' submitted for publication (1990).

[3] K. Narendra and T. Mandayam, ''Learning Automata: An Introduction,'' Englewood Cliffs, New Jersey: Prentice-Hall (1989).

High-Speed Transport and Internet Protocols for Multimedia

Jonathan Kay

Communication Networks Research Group

I am working on protocols able to send the amounts of data necessary for video, audio and similar media across packet-switched networks such the DARPA Internet. Video in particular is expected to be the largest consumer of bandwidth in the future, and researchers are looking at a whole spectrum of issues, from fiber optic hardware to bandwidth-reserving protocols. I am examining three parts of the problem: transport, internet, and network protocols.

During the next three quarters, I plan to to identify the worst protocol problems facing transmission of multimedia over TCP/IP networks. We will be carrying out experiments at the San Diego Supercomputing Center (SDSC). Sufficiently fast networking hardware is already available at SDSC in the form of 100Mbps FDDI, linking workstations used for visualization as well as the supercomputers themselves.

The biggest problem is that current software cannot put data on the net sufficiently quickly, whether for protocol reasons or because of inefficient coding. We intend to find the bottlenecks by instrumenting the kernels of two workstations and using the results to create a system model. That will enable us to find the bottlenecks and decide how to remove them. We think it is likely we will end up designing a new transport protocol to take advantage of special properties inherent in video and audio. If we do design such a protocol, we will attempt to make it independent of its underlying layer, so it can run in other environments such as broadband ISDN and the OSI standard.

A related problem I am working on is bandwidth reservation. Bandwidth reservation is a technique which allows one to guarantee that a certain number of bytes will take no longer than a given amount of time to reach their destination; video in particular will almost certainly require it. Unfortunately, it is as yet unimplemented in all internets and almost all networks. Furthermore, it is difficult to implement in TCP/IP because it would require Internet gateways to maintain state about each connection that was using bandwidth reservation; to date the Internet has been built around the assumption that gateways would route each packet without considering anything about the packet except its destination. We will investigate the question of how such state should be added to the TCP/IP protocol suite, especially that necessary for bandwidth reservation.

As a test case, we are considering implenting bandwidth reservation on Ethernets. An obvious method would use time division multiplexing; we feel we can implement a method which will permit packets without reservations to be transmitted when no packets with reservations are waiting to be transmitted. That would alleviate the potential for starvation inherent in bandwidth reservation.

I have only recently started work on networking issues. Previous to this I designed and wrote a fault injector in software [1], which used UNIX debugging system calls to implant cosmic-ray-like faults in running programs. I used that injector to verify the worth of a new fault detection technique. Also, in the fall of 1990, I improved a queueing network simulator written by Joseph Pasquale. The most important change I made was to change its user interface to add recursive procedural abstractions, iterative constructs, and arithmetic. I also reorganized the simulator's source code to make it consume less disk space and to make it easier to modify.

References

[1] J. Kay, *An Empirical Study of Using Certification Trails to Achieve Software Fault Tolerance*, MSE thesis, The Johns Hopkins University (1991).

Multicast Routing for High-Bandwidth Delay-Sensitive Applications

Vachaspathi Kompella

Communication Networks Research Group

The problem under investigation is how best to transport large volumes of data to multiple nodes in a network, given a delay constraint that represents the lifetime of the data. An example of such data is digitized voice/video information from a live conference. If the digitized data reaches late, it loses its value.

Multicasting is a convenient mechanism for such communication. In addition, it can also make efficient utilization of network bandwidth. This is possible when the same data has to be transported to several nodes in a network. If we were to send the data to each node individually, it is likely that a link may be used several times, once to carry the data to one node, a second time to carry the data to another node, and so on. However, if we build a tree structure and inform the relaying nodes to route the data along the links of the tree, then the duplication of data will be minimized. Fig. 1 below shows such a tree, in which A is the source node, and B and E are destination nodes. The tree is in thick lines.

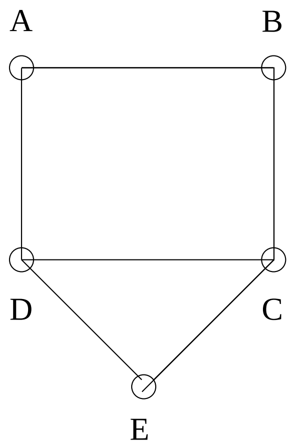

Fig. 1 Multicast from A to B and E.

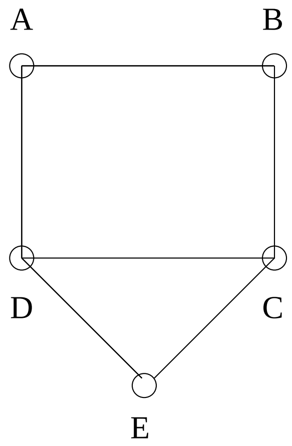

Fig.2 Multicast from A to B and E (bounded delay).

A problem with the tree of Fig. 1 is that, given equal delay on any link, the delay from node A to node E may be excessive. In particular, in a conference, the participant at node E may always be the last to hear what is being discussed. This would effectively leave E out of the conversation. The problem we are looking into attempts to add a delay constraint to multicast route determination, so that no path from one participant to another will have too large a delay. If the delay had to be under 3 units of time, then a tree as shown in Fig. 2 would be constructed.

The ''vanilla'' multicast route problem is NP-complete, and with the additional delay constraint problem, it is still as hard. We have a heuristic that performs very well on randomly generated networks.

Clustering Algorithms for Workload Characterization

Daniel Kraiman

Performance Evaluation Research Group

I have been investigating the performance of nonhierarchical clustering algorithms on trace data from a Cray X-MP supercomputer. Nonhierarchical clustering algorithms are important because they are the standard method for data analysis used in workload characterization studies. They attempt to construct an optimal partition of the data into a number of groups, called clusters. The clusters are then used to represent classes of jobs with similar resource usage characteristics. The most popular clustering algorithm is called *k-means.* However, some consider it to be a non-robust method, since it is sensitive to outliers in the data. An alternative, more robust method is called *k-medoids.* I am in the process of comparing the performance of several variations of the *k-means* and *k-medoids* algorithms on the trace data to determine which is the best.

For the past 6 months, our research group has been analyzing trace data from a Cray X-MP supercomputer which was located at the San Diego Supercomputer Center [5]. This study is important because very few workload characterization studies have been done on supercomputers, and thus less is known about them. The data consists of 1.2 million jobs and their resource usage statistics (CPU time, I/O time, memory).

One of the most critical factors in any data analysis are the statistical techniques used to do the analysis. In this study, these techniques are the clustering algorithms. Nonhierarchical clustering algorithms choose a set of k representative points as the centers of the clusters and then assign points in the data set to the clusters, using a distance criterion to find the optimal point assignments. This optimum assignment minimizes some distance criterion. The goal is to find the optimum partition of the data set into k clusters.

The *k-means* method [4] assigns a particular point to the cluster that minimizes the squared euclidean distance between the point and the cluster mean (centroid). The version of this algorithm found in [2] was used. [1] modified the *k-means* algorithm to allow the clusters to vary in size according to their variance, by using a weighting factor in the distance calculations, which is inversely proportional to each cluster's variance. However, [3] claims that variance minimization methods such as *k-means* are not robust in the presence of outliers. They use a version of the *k-medoids* method of [6], which uses the median instead of the mean as the center of a cluster and assigns points which minimize average dissimilarities (a nonparametric measure of distance based on correlation coefficients). They also provide a method for clustering large data sets. The k cluster center points are chosen by finding the k most central points (minimizing average dissimilarities) in a small random sample of the data set.

I am in the process of running these three clustering algorithms on random samples of the 1.2 million data points to see which is the best. So far, I have found that the algorithms of [2] and [1] work equally well on a random sample of 10,000 points, with the [1] algorithm having 40 fewer clusters than [2], but having a 35 greater average cluster standard deviation.

Another area of interest of mine is the performance evaluation of parallel processing computer systems. With parallel processors coming into increased use today, it is important to ascertain whether performance evaluation techniques developed for uniprocessors can be used or whether new methods and models of performance evaluation need to be developed. During fall quarter 1989 I wrote a survey paper on this issue, mainly concentrating on analytic techniques such as queueing networks and Petri nets. In preparation for my research exam, as well as the selection of my dissertation topic, I plan to continue this research in winter and spring quarters of 1991. I will be changing my emphasis to performance measurement techniques and concentrating on the statistical techniques used to analyze the measurement data. My thesis is that nonparametric statistical techniques are underutilized in this area, mainly because they are less well understood than the more traditional parametric techniques and many researchers in this area do not have deep statistical backgrounds. Nonparametric statistical techniques are more robust because they do not make as many underlying assumptions about the probability distribution of the data.

References

[1] Agrawala, A.K., J.M. Mohr, and R.M. Bryant, ‘‘An Approach to the Workload Characterization Problem,’’ *IEEE Computer*, pp. 18-32 (June 1976).

[2] Anderburg, M.R., *Cluster Analysis for Applications,* Academic Press, New York (1973).

[3] Kaufman, L., and P.J. Rousseeuw, *Finding Groups in Data,* John Wiley & Sons, New York (1990).

[4] MacQueen, J., ‘‘Some Methods for Classification and Analysis of Multivariate Observations,’’ 5th Berkeley Symp. Math. Statist. Prob., edited by L. Le Cam and J. Neyman, Vol. 1, pp. 281-297 (1967).

[5] Pasquale, J., B.K. Bittel, and D. Kraiman. ‘‘A Static and Dynamic Workload Characterization of the San Diego Supercomputer Center Cray X-MP,’’ to appear in *ACM Sigmetrics* (1991).

[6] Vinod, H., ‘‘Integer Programming and the Theory of Grouping,’’ *J. Amer. Statist. Assoc.*, Vol. 64, pp. 506-517 (1969).

A Prototype of the Multicast Channel

Scott McMullan

Operating Systems Research Group

I am investigating efficient kernel buffering mechanisms to facilitate an operating system's management of high bandwith continuous data. Briefly, we are developing an operating system abstraction – the *channel* – designed to handle large amounts of time sensitive data. A canonical application would be a real-time audio/video conferencing system, which would exercise both elements of time sensitivity (concerning frame syncing/picture quality) and high bandwith data (concerning the size and number of encoded "frames"). An integral part of the channel's success will be its ability to avoid unnecessary handling of data by the interested entities – the kernel, devices, and processes to name a few. This problem of data management within the kernel is the subject of my current research.

In the past sixth months we have implemented a prototype of the channel in user-space, primarily to determine if the abstractions presented to the user are useful for building distributed applications. Once successfully implemented as a set of library routines and servers using BSD sockets and TCP streams, we built a simple text conferencing program using X-windows and the channel. The channel proved extremely valuable, and we are continuing to refine the abstractions and semantics of the channel as a result of the experience. In addition, we encountered early on the difficulties in the handling of (relatively) high-bandwith data while maintaining connections between many distributed processes, re-enforcing that indeed the policies and mechanisms involved are of great importance.

The next stage of our research involves determining the proper operating system structures to support the channel abstraction. The handling and buffering of data is an important part of this, and will interact closely with resource reservation protocols, the connection of devices within the kernel and other fundamental, underlying services provided by the channel. I plan on aggressively investigating this topic, and hope to have a prototype implemented in a real operating system kernel by the end of the academic year.

Programming Support for Multi-User Applications

Dipti Ranganathan

Operating Systems Research Group

My work with the Operating Systems Research Group is focused on programming support for multi-user applications. As a group we have been concentrating on operating system support for distributed, multi-media applications. We have designed a multi-cast channel to support group communication. The channel allows many objects, which could be process or devices, to interact on a network transparent basis so members of the channel need not know the identities of the other members and yet can send information to all members of the channel.

To test the functionality of the channel we implemented it at the user-level upon which I built a simple text conferencing system. From this experience we were able to better understand what functionalities of multi-user application support should be in the kernel-level channel and what functionalities should be at a higher level.

To further pursue support for multi-user applications, I have been investigating multi-user, interactive, synchronous (MIS) applications and have found that there are some basic commonalities amongst MIS applications which tend to be implemented in an ad-hoc manner resulting in code redundancy and inefficiency. I believe that these fundamental functions should be supported by a programming language or toolkit.

The fundamental abstractions of MIS applications fall into two categories, control abstractions and data abstractions. Control abstractions encompass functions that handle movement of data, such as user synchronization (e.g. flow control) and media synchronization (e.g. synchronizing audio and video for viewing of a video tape). Placement of this information control greatly affects the performance of an MIS application. Centralized control, for instance, will improve consistency amongst participants, but creates a bottleneck which can increase response time.

Data abstractions create an interface for data objects. In MIS applications this involves both shared and public data objects as well as shared and private views. These abstractions allow a programmer to easily provide various views of the same shared object (e.g. different angles of a card table) as well as personal views of a shared object (e.g. a player's card hand). Such an interface to data objects should also allow a programmer to easily store state information. This simplifies the process of updating a new join to an already existing instance of an MIS application.

I have been studying various types of multi-user applications to understand the commonalities amongst these types of applications, including conferencing programs and multi-user games. From this study, I plan to design a canonical MIS application which include the basic functions that occur in all MIS applications. I will then implement these functions as a toolkit and then build this canonical application using this toolkit.

Window System Support for Multimedia Applications

Robert Terek

Operating Systems Research Group

I have been working with the Operating Systems Research Group since January of 1990. My primary goal is to determine how window systems should be extended to process so called "continuous media", such as audio and video. I call this extended window system a "Media System." The media system, along with the new operating system constructs and network protocols the group is designing, will provide the necessary support to build a conferencing system and other distributed multi-media applications.

My first investigative effort in this area has resulted in an extended X Windows server that processes audio as well as the usual graphics. The general structure of this server, and its relationship to an application, is depicted in Figure 1. The X server was modified to accept new audio requests through the X extension mechanism. The extension routines communicate audio data to a separate Audio Manager, a child process that manages the audio device, mixes multiple audio streams, and makes decisions to maintain the real-time quality of an audio stream. With this support in the X server, it was easy to modify existing X-based applications to include audio in their user interface. In particular, a simple text and audio conferencing system was built that supported a small number of users working cooperatively.

This work, although modest in nature, has raised many issues concerning the structure of a generalized media server. The most critical issue is the fact that the regular duties of the X server, which are non-realtime, must not be allowed to interfere with the processing of continuous media. This problem could be overcome by maintaining a separate connection to the Audio Manager, but this will complicate synchronization between different types of media. Instead, I plan to pursue a single server design with separate internal threads structured in a way that requests of different types are not able to interfere with one another.