



Infrastructure for Research towards Ubiquitous Information Systems

Citation

Grosz, Barbara, H.T. Kung, Margo Seltzer, Stuart Shieber, and Michael Smith. 1994.
Infrastructure for Research towards Ubiquitous Information Systems. Harvard Computer
Science Group Technical Report TR-28-94.

Link

<http://nrs.harvard.edu/urn-3:HUL.InstRepos:26506441>

Terms of use

This article was downloaded from Harvard University's DASH repository, and is made
available under the terms and conditions applicable to Other Posted Material (LAA), as set
forth at

<https://harvardwiki.atlassian.net/wiki/external/NGY5NDE4ZjgzNTc5NDQzMGIzZWZhMGFIOWI2M2EwYTg>

Accessibility

<https://accessibility.huit.harvard.edu/digital-accessibility-policy>

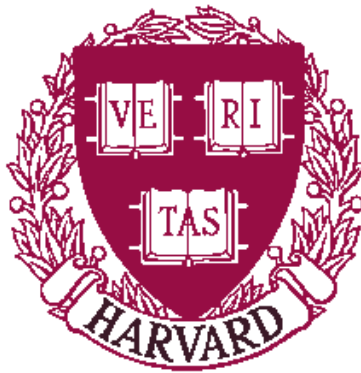
Share Your Story

The Harvard community has made this article openly available.
Please share how this access benefits you. [Submit a story](#)

Infrastructure for Research towards Ubiquitous Information Systems

Barbara Grosz
H. T. Kung
Margo Seltzer
Stuart Shieber
Michael Smith

TR-28-94



Center for Research in Computing Technology
Harvard University
Cambridge, Massachusetts

Contents

1	Executive Summary	3
2	Research Narrative	4
2.1	Overview of Research Projects	6
2.2	New Networks (H. T. Kung, M. D. Smith)	7
2.2.1	Network Architectures	8
2.2.2	Research in Host Interface Systems	11
2.2.3	Research in Wireless LANs	11
2.2.4	Research in Network Applications	12
2.3	Architectures for High-Performance Networked Hosts (M. D. Smith) .	13
2.3.1	The Potential of Compiled I/O	14
2.3.2	Architectural and Operating System Opportunities	15
2.4	High-Performance File and Disk Systems (Margo I. Seltzer)	18
2.4.1	High Performance File Systems	19
2.4.2	Disk Scheduling	23
2.5	Applications	25
2.5.1	Collaborative Planning Systems (B. J. Grosz)	25
2.5.2	Computer Communication with Users (B. J. Grosz, S. M. Shieber)	27
2.5.3	Large Databases of Text and Images (M. I. Seltzer, S. M. Shieber)	33

1 Executive Summary

The availability of fast, inexpensive computers and the growth of network technology have resulted in the proliferation of computing power and an enormous increase in information available in electronic form. However, most of the information stored on computers is extremely difficult for the common person to obtain. Thus, a central challenge for computer science and engineering in the next decade is to create the scientific and technological base for large-scale and easy-to-use information systems. These systems must work together in a coherent and cohesive manner, providing shared information easily for the general user. We refer to these systems as systems for *ubiquitous information*. The development of the National Information Infrastructure (NII) amplifies the urgent needs for research in this area.

We propose to develop a new generation computing facility to support experimental research in ubiquitous information systems. The research to be carried out using this facility spans from the development of new technologies that support the rapid transmission of large amounts of data between computer systems to the development of more flexible and adaptable systems for human-computer communication. The proposed infrastructure will include emerging equipment with new capabilities critical to this new research, such as Asynchronous Transfer Mode (ATM) networks capable of guaranteeing performance, file servers capable of handling video, and graphics workstations with advanced human interface capabilities. This equipment will supplement the basic computing and networking equipment typically found in computer science departments.

Participating Faculty: The Computer Science faculty at Harvard has been greatly strengthened by the recent arrival of H. T. Kung, a leading figure in parallel computing and computer networking, and two assistant professors, Margo Seltzer, an expert in computer systems design and implementation, and Michael Smith, an expert in high-performance computer architectures and optimizing compilers. They join in submitting this proposal Barbara Grosz and Stuart Shieber, who are prominent in the fields of computational linguistics and artificial intelligence, especially in areas relating to planning and communication. The proposal requests funding to establish the facilities needed for the coordinated experimental research of these faculty.

Areas of Research: Kung, Seltzer, and Smith will investigate the development of high-performance networks, host I/O architectures, and operating systems for ubiquitous information. Their research will provide the basis for the construction of distributed systems of heterogeneous machines that enable vast amounts of information to be available within short time frames to users who themselves may be

geographically distributed. Grosz, Seltzer, and Shieber will investigate techniques for accessing massive data repositories and develop systems for human-computer communication. Their research will make the power of ubiquitous information systems available to users without requiring the user to know details of the system structure.

The research on software applications will provide a crucial test of the network, computer architecture, and operating systems research. But the flow will also go in the reverse direction: networks of distributed systems will provide a crucial testbed for new database algorithms and for new techniques for human-computer communication.

Related Research Environment: At Harvard, faculty and research programs in Computer Science and Electrical Engineering are housed under the Division of Applied Sciences, a multidisciplinary department hosting researchers collaborating across related areas of science and engineering. The Division is part of the Faculty of Arts and Sciences (FAS) which contains all the departments not belonging to the professional schools at Harvard. The Division has about 24 faculty in Computer Science and Electrical, Computer, and Systems Engineering and ongoing research areas include the theory of computation, artificial intelligence, natural language processing, robotics, computer vision, software for parallel computers, computer architectures, networking, operating systems, database systems, VLSI, microelectronics, and systems performance and evaluation.

2 Research Narrative

Major computing applications are increasingly carried out on *distributed systems*, connected by *high-speed networks* over which many kinds of information are transmitted, among *collaborating groups* of people and machines. The extent of information exchange is exemplified by the NSFNET backbone of 10,000 networks connecting several million computers and more than 1,000 anonymous FTP servers indexed by ARCHIE. However, much of the information stored on computers is extremely difficult for most people to obtain. For example, users of the UUNET massive store of information typically must know exactly what they are looking for to be able to find it. The tools to make it possible to share information in a simple and coordinated manner are missing. Thus, a central challenge for computer science and engineering in the next decade is to create the scientific and technological base needed to construct information systems that work together in a coherent and cohesive manner, providing shared information easily for the naive user. We refer to these systems as systems for *ubiquitous information*.

Today's systems are limited by the ability to transmit and communicate information effectively, not by the ability of either computers or humans to absorb it. For example, specialized computers are now powerful enough to perform computations on images at full-motion video rates, yet transmission of digitized full-screen images over computer networks at video rates is not reliably attainable. Similarly, human beings can absorb information at video rates by looking at images, and can generate and absorb information very efficiently over the acoustic channel by using natural language; but computers are unable to take advantage of these human capabilities, and human-computer interaction is still a relatively low-bandwidth channel.

In the future, a wider variety and greater amount of information will be available on-line. Movies, records, news broadcasts, libraries, and banking services should be easily accessible using home or office computers. There will be real-time demands on scientific, industrial, and medical databases that are geographically distributed. In this ubiquitous information world, such information must be accessible through comfortable, familiar means and retrieval must be based on content and in terms familiar to the user (e.g., "books like this one") rather than by exact matching of formal specifications. Thus, computer systems increasingly need abilities to coordinate their activities with other systems and with people who may be widely distributed geographically. The participants — both human and computer — in these coordinated activities may be connected by communication links of varying types, speeds, and interface properties. It is this future world that our research will support.

One example of the kind of system we envisage is a sophisticated telephone switch maintenance system; a computer system that maintains information about the telephone system might be used for planning extensions and alterations to the system, for troubleshooting daily problems, and for providing information about congestion, scheduling, and repairs. Such a system would be geographically distributed in the sources and delivery points of information, would receive queries in natural language form that presume shared context and hence only include partial specifications of the information desired, and would deliver information in multiple media (including, for example, text, stored and synthesized graphics, and video). Another example is an image processing facility to be used for analyzing medical imaging data and processing the data into 3D reconstructions, so that medical image reconstruction can be used as a tool in planning surgery. Remote medical consultation and similar applications require vast computing resources, probably provided by one or more high-performance parallel computers, connected via very high-speed networks to image-capture and -synthesis equipment operating in real time.

2.1 Overview of Research Projects

Our research is directed at the development of effective distributed collaborative computing systems for ubiquitous information. The projects described below fall into the following categories: networking, host systems, file and disk systems, and applications and interfaces. Thus, our research spans from the development of new technologies that support the rapid transmission of large amounts of data between computer systems to the development of more flexible and adaptable systems for human-computer communication. The backbone of ubiquitous information will be provided by a variety of new hardware and software technologies that allow information distributed among heterogeneous machines over extensive networks to be available reliably and within sufficiently short time frames. To make this information accessible to users will require advances in a variety of applications.

At each of several levels, our research aims at insulating users from needing to know details of system organization that are irrelevant to their information needs. At the network level, users will be protected from needing to know the details of the routing of queries. At the operating system level, they will be protected from needing to know the specific location of the information they requested. At the application level, they will be relieved of the need to know details of database structure and underlying schemas. Finally, our research on collaborative planning, natural language and graphics will protect users from needing to know special purpose programming or query languages. Users will phrase requests and provide information in terms that are familiar and natural to them; systems will communicate information to users in ways that take into account the users' cognitive abilities and constraints, as well as the problem-solving context in which information needs arise. This insulation is crucial to the success of systems for ubiquitous information: without it, no ordinary user will be able to tap into the vast resources soon to be available on computer networks.

Each of the research problems we propose to address is central to some part of the technological base needed to construct systems for ubiquitous information. The diversity of these problems precludes a single attack on the entire problem. However, the challenge of ubiquitous information cannot be met by separate, specialized investigations in each area alone; it requires a unified approach that addresses the interactions between hardware and software as well as among different architectural levels, and that integrates the technologies developed for different system components. Such a unified approach enables tradeoffs made at one level of the design to take advantage of the capabilities of the other levels of the system, and ensures that they are constrained by the requirements of other levels. While other research projects are

investigating how to build storage repositories for massive quantities of data [102], the research for which we are requesting infrastructure support, focuses on providing ways to make the data available using high speed networks, sophisticated tools, and application systems that provide users easy access.

Thus, this proposal requests funding to support the infrastructure needed to coordinate the research activities across the different system levels. Research in the development of high-speed networks, host I/O architecture, and file and disk systems will provide the basis for the construction of distributed systems of heterogeneous machines that enable vast amounts of information to be available within short time frames to users who themselves may be geographically distributed. Research on techniques for accessing massive data repositories and on developing systems for human-computer communication will make the power of these systems available to users without requiring the user to know details of the system structure. But, the flow will also go in the reverse direction: networks of distributed systems will provide a crucial testbed for new database algorithms and for new techniques for human-computer communication. In the remaining sections we will discuss each level of the system in turn — starting with those that make distributed data available in reasonable times, and ending with those that provide easy and intuitive access.

2.2 New Networks (H. T. Kung, M. D. Smith)

The networking research group is developing a new generation of high-performance local area networks (LANs) with the following novel features: (1) guaranteed, real-time performance for priority, multimedia traffic; (2) maximum network utilization for “best effort” data traffic; (3) reliable and bandwidth-efficient multicasting; and (4) uniform circuit-oriented architecture for both wired and wireless LANs. These networks will be able to deliver a large amount of information to multiple sites in a short time and with a high degree of reliability and guaranteed performance in bandwidth and latency. As a result, they will greatly expand the application of ubiquitous information systems.

Our research uses the Asynchronous Transfer Mode (ATM) network architecture [8]. ATM networks, which employ fixed-size small packets, called cells, to implement virtual circuits, represent one of the most significant developments in the networking industry in recent years. It is generally believed that in several years ATM networks will be widely used in both communication and computer industries. However, because they are new and their uses are highly flexible, many crucial issues such as network congestion control and application usage are not well understood currently.

It is critical that they be resolved before widespread services based on ATM networks are deployed.

Since June 1992 we have been designing an experimental ATM network at Harvard, in collaboration with Bell-Northern Research (BNR), for the purpose of studying these issues. At the time of writing of this proposal, the architectural design of an experiment ATM switch with 622 Mb/s ports has been complete, and the switch hardware design is near completion, with a couple of printed circuit boards already in fabrication. All the boards are expected to be built by early 1994. Intel, another industrial partner of our project, has been developing a 622 Mb/s ATM host adaptor card for Pentium-class PC platforms. Harvard has been interacting with Intel to ensure that this host adaptor will interoperate with the BNR/Harvard ATM switch mentioned above. This Intel host adaptor is expected to be operational second quarter of 1994. In addition, we have recently added a wireless component to our networking research. To foster collaboration, Motorola has provided us with a wireless LAN based on their ALTAIR system, capable of transmitting Ethernet packets at about 5.7 Mb/s. This wireless system currently supports both research and education activities at Harvard in the mobile networking area.

Using this networking infrastructure, our networking research in the next several years will focus in four areas: network architectures, host interface systems, wireless LANs, and network applications. Research topics include: mapping various service models into priority scheme for switching node, network resource management, LAN-interconnect router architectures for ATM wide area networks, network performance models, measurements and simulation, and fast virtual-circuit setup.

2.2.1 Network Architectures

Research in Flow Control The key architectural idea of our experimental ATM network is link-by-link flow control (**LLFC**) on a per virtual circuit (**VC**) basis [63, 65, 64]. In this architecture the receiver of each link of a **VC** will send its buffer status information to the sender. Using this feedback the sender ensures that it will never send more data than the receiver can hold.

A fundamental reason for per **VC LLFC** is to provide effective means of using filling-in traffic to maximize network utilization. Typically, best-effort traffic is used to fill in bandwidth slack left by scheduled traffic with guaranteed bandwidth and latency such as video and audio. To allow effective traffic fill in, fast congestion feedback for individual **VCs** is needed. Measurements have shown that data [32, 70] and video [36] traffic often exhibit large bandwidth variations even over time intervals as small

as 10 milliseconds. With the emergence of very high-bandwidth traffic sources such as high-speed host computers with 800-Mbps HIPPI network interfaces, networks will experience further increases in load fluctuations [63]. To utilize slack bandwidth in the presence of highly bursty traffic, fast congestion feedback is necessary. **LLFC** implements the required feedback at the fastest possible speed.

In addition, using **LLFC** a **VC** can be guaranteed not to lose cells due to congestion. When experiencing congestion, back-pressure will build up quickly along congested **VCs** spanning one or more hops. When encountering back-pressure, the traffic source of a congested **VC** can be throttled. Thus excessive traffic can be blocked at the boundary of the network, instead of being allowed to enter the network and cause congestion problems to other traffic.

The “per **VC**” **LLFC** allows multiple **VCs** over the same physical link to operate at different speeds, depending on their individual congestion status. In particular, congested **VCs** cannot block other **VCs** which are not congested.

The flow control described here would allow new services for hosts with high-speed network access links operating, for example, at 100 megabits per second. For instance, these hosts can be offered a new kind of data communications service, which may be called a “best-effort” service, where the network will accept as much traffic as possible at any instant from **VCs** under this service. Flow control can be used to throttle these **VCs** on a per **VC** basis when the network load becomes too high, and also to speed them up when the load clears. There will be no requirements for predefined service contract parameters, which are difficult to set dynamically. This “best-effort” service is expected to serve data traffic effectively and efficiently.

Over the past year we have made substantial progress in developing efficient and robust protocols to support per **VC** **LLFC** [65] and validating their performance by simulation [64]. The industrial partners involved in the Harvard ATM networking project (BNR and Intel) have committed to implementing our new flow control methods. Implementation details have been worked out, and hardware support has been provided in the BNR/Harvard switch. Furthermore, we have been an active participant in the ATM Forum, the most important industrial consortium in this area. The Forum has now accepted the notion of “best-effort” services for data traffic, and the proposal of developing standards on flow control in the coming year.

We expect credit-based per-**VC**, link-by-link flow control to provide high network utilization and low-cost implementation. We plan to demonstrate this in the systems context, using the experimental ATM network we have been developing with our industrial partners. In general, research is needed to understand new network phenomena that may be introduced by such flow control. For example, it is im-

portant to understand how the link-by-link flow control may shape or preserve load characteristics such as traffic burstiness as traffic passes through the network, and will interoperate with various traffic scheduling methods.

Research is also needed for a general network congestion control model, capable of modeling many load-sensitive, feedback control mechanisms which operate dynamically in the lifetime of a **VC**. **LLFC** is one example of such a mechanism; others include various fast buffer reservation schemes. The lack of such a model is a major deficiency of current ATM standards. We expect our research to help fix the problem.

Research in Multicasting Supporting multicasting has been a goal of network designers for many years. Multicasting is important for a large number of applications such as video-conferencing, distributed multimedia, and simulation over networks. However, efficient and reliable multicasting, which is able to guarantee message delivery to all the destinations without duplicating transmissions, has to our knowledge defied all previous implementation attempts. The difficulty arises from the fact that, in general, not all the destinations of a multicasting **VC** may be available to receive data at the same time and, as a result, some complicated management is required.

We are exploring two new techniques to overcome the difficulty. One is the use of **LLFC** to implement reliable multicasting. At a switch where a multicasting takes place, the multicasting **VC** will keep transmitting a cell until all the destination ports have received it. While waiting for some destination ports to become available to receive new cells, the **VC** throttles the cell transmission over the switch using the **LLFC** mechanism. Thus reliable multicasting can be achieved. Of course, in practice a “relatively” reliable multicast which allows some sort of time-out on blocked multicasting ports will be implemented so that an unreliable port will not hold up the whole multicast **VC** for an unbounded amount of time. How this time-out is to be implemented is an interesting research issue.

The second technique is the use of asynchronous multicasting to improve multicasting efficiency. Over a switching node, a multicasting **VC** can have its cells sent to different ports asynchronously. Therefore a congested port does not necessarily block transmission of cells to other ports as in the synchronous multicasting. Obviously, asynchronous multicasting is more costly to implement than synchronous multicasting. Research is required to determine the proper degree of asynchrony to maximize cost-performance.

2.2.2 Research in Host Interface Systems

The primary function of host interfaces is to move data between the attached network and the host memory accessible to the host application. It is well-known that most common transport protocols, in particular Transmission Control Protocol/Internet Protocol (TCP/IP), do not form performance bottlenecks, provided that the protocols are properly implemented. The host system architecture, which provides the environment in which protocols are executed, is really the key performance factor [22].

For instance, it is essential for efficient implementation of a protocol to minimize the number of accesses to the host memory. In traditional host interface implementations for Ethernet, there can be as many as six accesses to or from the host memory for each word received from or transmitted to the network. This excessive use of memory bandwidth will likely be a severe performance bottleneck for a high-speed network.

We are conducting research on the host system architecture that can support streamlined implementation of protocols. Our goals are to reduce copy and buffer operations, to pipeline data transfers, to provide high-bandwidth DMA, and to use efficient implementations of such procedures as checksumming, error handling, compression, and encryption.

2.2.3 Research in Wireless LANs

We have recently initiated a new project on wireless LANs in which we are studying ATM implementation over wireless LANs. We intend to provide quality-of-service guarantees, track mobile hosts, and re-route virtual circuits as the mobile hosts move among base stations.¹ This project, called SWAN (Switched Wireless ATM Networks), includes two industrial partners, BNR and Motorola. It aims to develop the following new capabilities for wireless mobile LANs: (1) support multimedia traffic, including data and video; (2) track mobile hosts and manage resources; (3) improve transmission reliability; (4) handle large bandwidth mismatch when interoperating with high-speed wired networks; and, (5) provide networking security.

Our approach of providing these capabilities is based on the use of virtual circuits over wireless networks. Specific ideas include the use of: (1) credit-based, per-

¹This research also has an educational component. H.T. Kung is teaching a new one-year project course at Harvard in which students will design and build a wireless LAN to carry both TCP/IP and ATM traffic.

VC Link-by-link flow control to maximize network and spectrum utilization, handle bandwidth mismatch, and provide protection against jammer; (2) VC scheduling to support priority, multimedia traffic, and allow bandwidth reservation, (3) multi-VC radio links to improve transmission reliability, shorten hand-off and reconfiguration time, and improve security using VC hopping.

A prototype wireless LAN, using the Motorola's ALTAIR, will be connected to the BNR/Harvard ATM switch to support this research.

2.2.4 Research in Network Applications

We have been actively pursuing high-speed network application research in several areas. These areas include surgical planning at Brigham and Women's Hospital (BWH), real-time imaging of human brain function at Massachusetts General Hospital (MGH), and electronic publishing over networks at Harvard Library. Because of space limitation of this proposal, we describe here only the network-based surgical planning application at BWH.

The Surgical Planning Laboratory of the Radiology Department of BWH (Dr. Ferenc A. Jolesz and Dr. Ron Kikinis) is developing and implementing an image processing environment for analyzing digital medical imaging data and processing these data into 3D reconstructions. The Laboratory specializes in computer-assisted, automated postprocessing of MRI and CT data sets. The results of this processing are used to generate 3D renderings for the planning, simulation, and monitoring of surgical procedures in neurosurgery, craniofacial surgery, orthopedic surgery, and abdominal surgery. Last year, the Laboratory generated over 50 plans for surgical procedures and the volume of generated plans is increasing steadily.

Since May 1992 the Laboratory has provided imaging capabilities to doctors during surgery using a cart-mounted computer that can be wheeled into the operating room (OR). In addition, they have extended the existing hospital network into the OR in order to allow compute server access for interactive rendering of 3D reconstructions during surgery. It is expected that this service will expand significantly in the near future, both from inside the hospital and from referrals from elsewhere.

When providing these services during surgery, the availability of sufficient computation capacity will be of great importance. BWH will need more computational power to support multiple parallel medical operating procedures, and would like to be able to tap into computer resources such as the parallel machines and other high-performance systems available at Harvard's Division of Applied Sciences. They must be able to do so assuming that high-speed networks make this process completely

transparent to the end-user at the BWH Laboratory.

Thus, this project will explore the development of network-based surgical planning applications. The goal is to demonstrate the feasibility of such a network service which exhibits total end-user transparency.

2.3 Architectures for High-Performance Networked Hosts (M. D. Smith)

Historically, architectural research for the workstation and personal computer class of machines has focused almost exclusively on improvements for the processor and memory system. Researchers expended very little effort in the area of the I/O system design. With the recent spread of networked and multimedia applications for this class of machines, there has been an explosion of activity in high-performance I/O design ranging from specialized hardware for video subsystems [21] to optimized kernel code for network protocol processing [19]. Though these investigations have improved the performance of specific I/O devices, the I/O design process is still largely driven by ad hoc approaches that rely on significant human guidance to optimize a small section of the I/O subsystem. As workstation-class machines include higher performance I/O subsystems and as the interactions between these high-performance I/O subsystems and the processor/memory subsystem become more complex, we will require a structured, quantitative approach to I/O system design in order to enable further improvements in I/O system (and ultimately the entire system) performance. We will investigate this type of quantitative approach through the novel application of compiler technology to I/O system design. Our goal is to systematically define an architecture that produces a closer coupling between the processor/memory subsystem and the high-performance I/O devices. This coupling will be driven by an understanding of the capabilities of the compiler, hardware, and operating system to support high-performance I/O (in the same way that RISC processor design was driven by an understanding of the capabilities of just the compiler and the hardware [48]). This research will become increasingly important as technology continues to increase the raw performance of processors and the raw bandwidth of memory and I/O subsystems since the overheads of inefficient I/O architectures will otherwise dominate the system performance.

2.3.1 The Potential of Compiled I/O

In most computer systems today, access to a high-performance I/O device is through a set of generic yet powerful system calls which hide the underlying implementation of the I/O device. For example in 4.3BSD UNIX [68], the user-level socket interface to the network (e.g. `connect()`, `send()`, `receive()`) is independent of the actual network medium. This means that a change of the network adapter board (e.g. from ethernet to ATM) requires a change only in the low-level operating system drivers that implement the network interface layer, not the application code or the other higher network protocol layers in the kernel. Protocol layering models like this one insulate user programs and most of the operating system from low-level changes in network adapter hardware, and reduce the design complexity of the network interface [104]. This insulation comes at the cost of performance, and thus most systems do not directly implement a cleanly layered protocol stack architecture. The overhead of many layers makes it difficult to finely share data, and this is one reason why message-passing multiprocessor systems design their own simple send and receive primitives that are closely coupled to the processor and memory subsystems (e.g. see von Eicken et al. [107]). Our goal is not just to define another set of specialized I/O primitives however, but rather to reach a point where future system designers implicitly consider the capabilities of compiler technology in the same way that today's processor architects implicitly assume that the compiler will perform parallelization for vector machines or instruction scheduling for pipelined processors.

To help understand the potential of compilers in I/O system optimization, we are presently investigating ways of automating and extending the "fast path" work of Clark et al. [19]. By moving the boundary for the network system calls from the top of the transport layer to the bottom of the network layer, our compiler potentially can perform many of the optimizations proposed by Clark et al. For example in most applications, the path through the network protocol stack is a run-time constant, and therefore the compiler can perform constant propagation and dead-code elimination to remove many of the conditional branches in the protocol code. This optimization is similar in approach to what Bershad et al. [9] did by hand for remote procedure calls. To further optimize the code on the frequent I/O path, we can apply the trace-based ideas of global instruction scheduling [15][26][31][66][101]. To apply this previous work to I/O paths, we are extending these scheduling algorithms with more inter-procedural analysis [91].

We are also investigating the applicability of loop fusion [110] and redundant load removal to the protocol stack code. In this way, the compiler could automatically find

optimizations such as Clark et al.’s suggestion that the packet checksum calculation be performed during the copy of the packet from the I/O device to memory [19]. Part (b) of Figure 1 illustrates that this optimization reduces the number of memory references and the amount of loop overhead. By performing these types of optimizations automatically in the compiler, protocol designers can again write their code in layers that are clear and easy to maintain, and yet, a networked application will still run quickly, efficiently, and correctly.

By exposing both the application code and the network protocol code to the compiler, the compiler can now better overlap communication overheads with application computations. To go beyond just simple instruction scheduling overlap, we are investigating ways to apply the idea of compiler-driven speculative execution [14][67][101] to the network interface. In the area of high-performance processor architectures, the compiler uses speculative execution techniques to improve the run-time performance of application codes by executing some of the instructions early, before we know whether that instruction execution will be useful. We can map this idea into networking by extending the example in Figure 1 so that we not only calculate the checksum during the memory move, but we also begin to work on the data (see Part (c)). The commit point for this speculative work will be the result of the checksum. We have looked at this optimization for *nv*, a network video tool distributed by Frederick at Xerox PARC [33]. In this application, we can fold the decoding of the video pixels into the calculation of the checksum on the receive side of the network connection to improve the performance of this entire application². Speculation, in general, involves buffering of the speculative results and recovery modes for incorrect speculation, and we are addressing these issues for these types of speculative applications.

2.3.2 Architectural and Operating System Opportunities

As Anderson et al. [2] point out for RISC processors, the performance improvement of operating system functions has not been increasing as quickly as the performance of computation-bound application codes. We can recover some of this performance lag through the application of our compile-time optimization techniques for I/O. To solve this problem though, we must design architectures that better couple the processor and the high-speed I/O devices, and design cooperative interfaces that better couple the operating system and the compiler. Through this three-pronged approach, we

²Notice that we can fold the encoding and checksum work on the send side of the connection too, but this does not require any speculation.

```

foreach byte in packet {
    load r <= host_adapter;
    store r => host_memory;
}
foreach byte in packet {
    load r <= host_memory;
    use r in checksum calculation;
}

```

(a) Before any optimizations.

```

foreach byte in packet {
    load r <= host_adapter;
    store r => host_memory;
    use r in checksum calculation;
}

```

(b) After loop fusion and redundant load optimizations.

```

foreach byte in packet {
    load r <= host_adapter;
    store r => host_memory;
    use r in checksum calculation;
    use r in speculative application ‘‘calculation’’;
}
if (checksum correct) commit speculative results;
else discard speculative results and packet;

```

(c) After speculative optimization across protocol stack and application code.

Figure 1: Application of compiler optimizations to part of the TCP/ IP receive protocol processing code.

can appropriately distribute among the compiler, operating system, and hardware the functionality that is needed to efficiently support I/O-bound applications.

One approach to a tighter coupling of the network and processor is to directly map the network ports into the processor's register file [10][49]. This approach makes it efficient for a compiler to send a message since the compiler simply needs to build a message in the processor's register file. A shortcoming of this approach is that it bypasses the operating system, and we want to take advantage of the operating system functions that implement the protection and sharing of system resources. As suggested in Anderson et al.'s paper on improving the performance and flexibility of a threads package [1], the operating system and the application should cooperate through the sharing of relevant information.

We are currently defining a new architecture that closely couples the processor and the network. In this architecture, we are investigating a set of hardware mechanisms that will assist the operating system or application in interfacing to the network. For instance, we are looking at the feasibility of programmable functional units within a general-purpose, superscalar microprocessor [90]. With these dynamically-programmable functional units, the compiler is able to identify and collapse a sequence of multiple instructions into a single complex operation that takes less than a single processor cycle to execute (e.g. hardware-assisted CRC generation). We are also looking at ways of incorporating the network into the virtual memory system. One approach in this area is to extend the work of Cheriton and Kutter [17] to distributed systems. Another approach is to build TLB-like structures for protocol header generation and fast buffer mapping which would support higher performance than the equivalent software structures. This type of caching is already proposed for some network protocol software [19]. Finally, we are considering the impact of a separate hardware context for the operating system. (See [23][87][108] for information on multiple context processors in the area of multiprocessing.) We want to see if this small amount of dedicated hardware could be an efficient way to reduce the system call overheads.

Though this work is appropriate for and targeted toward today's network subsystems, this type of research will be especially important in the design of new architectures that will support the real-time needs of multimedia applications. In this application domain, it is especially important to control overheads so that we limit overhead-induced jitter. Unlike the Desk Area Network project that proposes to replace the workstation's I/O bus with an ATM switch [47], our project addresses the problem of predictable I/O performance without destroying the memory architecture that effectively handles our data processing needs. Furthermore, by including the pro-

cessor in the I/O paths of our high-performance multimedia streams, we can better support new applications that analyze and manipulate these streams in real time.

We are currently in the process of building analysis tools to determine exactly where the bottlenecks are in multimedia applications. We are developing a software tracing tool that will allow the fine-grained tracing of both application and kernel code. This tool is based on Larus's QPT [50] and is similar in function to a tool built by Chen at CMU [16]. With the logic analyzer requested in this proposal, we would also be able to build a hardware tracing tool like the one built by the Monster project [84]. This hardware tool would enable the capture of bus events that do not go through the CPU, and would allow us to exactly determine the timing between two events (something that is very difficult to do with a software-only tracing system, but is very important in jitter-sensitive environments like video). We are planning on connecting our "monster" to one of the SGI Indy machines. Another of the Indy's would act as a load generator, and a third would be used as a compiler and operating system testbed. The requested Network Appliance box would hold network and multimedia traces, and the SGI Power Challenge would act as our compute server for large compilations and for architectural simulations.

In summary, compilation technology has changed the way people design processor architectures, and it can change the way we design I/O subsystems. We have described some preliminary investigations that demonstrate how the appropriate application of compilation technology can automatically decrease network processing overheads. Still, there are a number of important research questions that we need to answer, and this infrastructure grant would provide the ideal environment necessary to support our interdisciplinary research. The following are some of the questions that this infrastructure grant would enable research on: (1) How does the performance of our compiler-driven approach compare to the performance obtained through human-driven optimizations? (2) How much of the protocol stack can the compiler access before operating system protection/security becomes a serious issue? (3) How do we extend the performance guarantees of our ATM network all the way to the individual applications? (4) What are the best I/O architectures for effectively supporting both data processing (e.g. database) and multimedia (e.g. natural language processing and speech synthesis) applications?

2.4 High-Performance File and Disk Systems (Margo I. Seltzer)

We will investigate two of the essential base technologies crucial to the performance of large, distributed data repositories, (1) high-performance file systems; and, (2) so-

phisticated scheduling of I/O devices.

2.4.1 High Performance File Systems

As CPU speeds have increased dramatically over the past decade, I/O performance is becoming more and more of a system bottleneck [89]. Therefore, there has been an increased emphasis in recent years to build better performing file systems and better performing I/O subsystems. This emphasis has led to the development of many array-based disk systems both in the research community [55, 58] and the commercial sector [40]. On the software side, this emphasis has led to new file system designs.

File system performance is limited by the frequency with which the disk is accessed and the cost of those accesses. In order to reduce the frequency of access, systems today are built with large main memory caches. These caches are very effective at reducing read traffic, but do little to reduce write traffic, because data must ultimately be written to permanent storage [7]. Additionally, most of these systems are forced to support UNIX file system semantics which guarantee that certain operations (directory operations such as create, remove, and link) are synchronous. One way of reducing the number of writes to disk is to use non-volatile memory instead of disk as permanent storage. This technique is used by Legato on the PrestoServe product [82] and Network Appliance in a special-purpose NFS file server [53] and is analyzed for use in distributed environments [6].

An alternative approach is to reduce the cost of writing data to disk. In the database domain, this is traditionally accomplished by writing short log records that describe the operation being performed. To make the changes permanent, only the log record need be written to disk. The modified data may be written at some later, more convenient time. There are three potential benefits from this approach. First, if the same data are updated repeatedly the data are only written to disk once, thus reducing disk traffic. Secondly, if the data are deleted before being written to disk, then the I/O is completely avoided, also reducing the number of disk accesses. Third, the existence of a sequential log allows for efficient file system recovery; the log can be read and processed from its last checkpoint, avoiding the more conventional scanning of the entire disk. Several of today's file systems have adopted this logging technique in an attempt to improve performance. [13, 106].

There have been two main approaches to the use of logging in file systems. The first, called journaling file systems, uses an auxiliary data structure, or log, to keep a record of the file system activity. This log is an append-only structure which can be written to disk very efficiently. The actual data are written back to the file system at

some later point when the disk accesses will not impact performance. There have been several variants of the journaling file systems and they have become widely-accepted in the commercial marketplace [13, 59, 106]

The second class of file systems are called log-structured, and they replace the entire file system with a single log-based representation of the file system. While the logging file systems write small log records to disk when data in the cache is dirtied, the log-structured systems (LFS) accumulate a large quantity of cached dirty data and then write it out to the log in one long, contiguous write [92]. When parts of a file are modified, rather than overwriting the existing blocks, new copies of the block are appended to the log. As a result, the old copies of the blocks become obsolete and their space must be reclaimed. In Rosenblum's LFS, this process is called cleaning, but is an implementation of generational garbage collection.

The performance benefits from log-structured file systems can be attributed to the accumulation of a large amount of data in the cache before it is written. This has two side effects. First, when data are written to disk, the writes are large and sequential, using the disk most efficiently. Second, operations that are usually synchronous in conventional file systems (e.g. the Fast File System [81]) occur asynchronously in LFS. It has been widely believed that the performance benefits of log-structured file systems were the result of the large contiguous writes, but [95] showed that, in fact, the asynchronous nature of directory operations was the major contribution to the observed improvements in performance.

The current analyses imply that the journaling file systems have the potential to offer the performance benefits of log-structured file systems without violating the synchronous semantics of conventional file systems and without the overhead of a garbage collector in log-structured file systems. Unfortunately, there is no existing empirical evidence to support this hypothesis.

We will test this hypothesis. One of the difficulties in making fair file system comparisons is that each file system runs in a different operating system. Therefore, when comparing performance, it is virtually impossible to isolate the components responsible for the differences in performance. This obstacle has made it extremely difficult to objectively evaluate these newer file system techniques. We will implement a general logging file system that can be easily integrated into any conventional file system to facilitate rapid performance comparison. We will use this general purpose logging file system to turn SGI's extent-based file system and the Berkeley Fast File System (with clustering extensions described in [94]) into journaling file systems. We will then perform extensive performance comparisons between the journaled systems, the unjournaled systems and the 4.4 BSD log-structured file system.

This research will address the following questions: (1) Will the use of logging permit a conventional file system to utilize 100% of the available disk bandwidth? (2) Is it essential to provide contiguous allocation beyond a few tracks in order to use the disk system efficiently and satisfy time-sensitive requests? (3) How much non-volatile ram is necessary to mask all disk latencies? (4) Can a general-purpose logging file system support the needs of both another file system and user-level applications (e.g. a database manager)? (5) How much of the benefit of logging file systems is due to avoiding synchronous writes and how much is due to more efficient use of the disk while writing?

The general logging file system mentioned provides write-ahead logging capabilities, via a standard, but simplified file system interface. This common framework enables us to isolate each component of the file system and determine exactly what aspects of each contribute to the resulting performance. We can then carefully isolate each file system bottleneck and, when possible, remove it (e.g. by logging different operations). When we have integrated the Fast File System with the Write-Ahead Logging File System (WAFS), we can replace the disk log with a small amount of non-volatile memory (less than one megabyte) and provide performance better than any of today's file systems.

Figure 2 shows how to integrate WAFS with an existing file system. In addition to providing a basis for file system comparison, this logging file system is a general-purpose tool useful for a wide range of applications. For example, a data manager can use standard read and write file system operations to implement its write-ahead logging protocol [39]. Not only does this alleviate the data manager designer from implementing another WAL protocol, but it allows multiple data managers to share a single log structure. This allows transactions that span multiple databases to be committed with a single log write rather than via an expensive two-phase commit protocol [72]. This is an extremely attractive feature in a multidatabase environment.

There are other applications which will benefit from a logging file system. To facilitate fast recovery, complex applications such as window managers can log key operations and restore the user's complete window system environment quickly after a system reboot. Word processing systems that typically provide their own logging facilities can use a standard file system rather than their own proprietary one. We will investigate these alternate applications as well as the basic file system research.

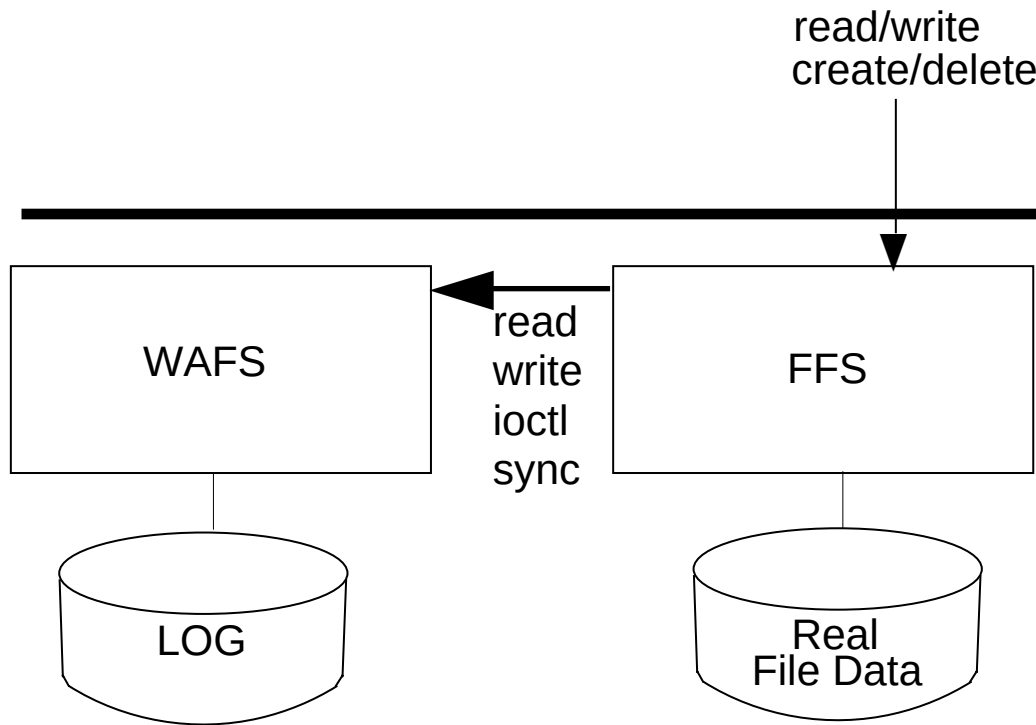


Figure 2: The WAFS architecture. Both WAFS and FFS function as independent file systems, but FFS makes calls into WAFS to perform logging operations.

2.4.2 Disk Scheduling

While the log-structured systems avoid all random writes, the journaling file systems merely delay random writes, and both systems will still be called upon to perform random reads. The result is that the I/Os fall in two general categories: asynchronous write requests and synchronous read requests. The expectation is that a large fraction of these reads are captured in the file system buffer cache, so that the majority of operations that actually become disk requests are these asynchronous reads. The data in [83] confirms this hypothesis, indicating that although the read to write ratio observed by the application is more than 3 to 1, the ratio observed by the disk is 1 to 3.6.

As nearly all the I/Os in a journaling system become asynchronous, it is increasingly important to schedule these writes efficiently so as to maximize the performance of the underlying disk system. Although disk scheduling has been extensively studied in the 70's and 80's [109, 105, 88, 54, 37, 20], changes in technology makes it necessary to address these issues once again. First, the early disk scheduling research examined very short queue lengths (averaging less than 50 requests). This limitation is due, in large part, to the absence of large main memories and the slower CPUs. With today's exponentially growing memory sizes, faster CPUs, and file systems that rely on asynchronous writing, more data will accumulate in the disk queues, which then grow to contain hundreds or even thousands of requests. Secondly, when the previous work was done, seek time dominated the total access time while in today's technology, seek and rotational positioning are nearly equal. Therefore, while earlier work focused on minimizing seeks, today's disk scheduling must focus on reducing both seek and rotation times.

Simulation results in [96] and [56] show that rotationally optimal algorithms can result in twice the performance of seek-only algorithms. Preliminary work on applying these rotationally optimal algorithms to actual trace data has been encouraging and our goal is to evaluate the feasibility of these algorithms and implement them in real systems. There are three main focuses to this work.

- Heuristic approximations to the rotationally optimal algorithms
- Heuristic techniques to reduce the computational overhead of the scheduling
- Dynamically adapting the scheduling algorithm to the workload presented.

Maintaining optimally sorted queues degenerates to solving the Traveling Salesman Problem, which is known to be NP-hard and obviously not viable for a disk

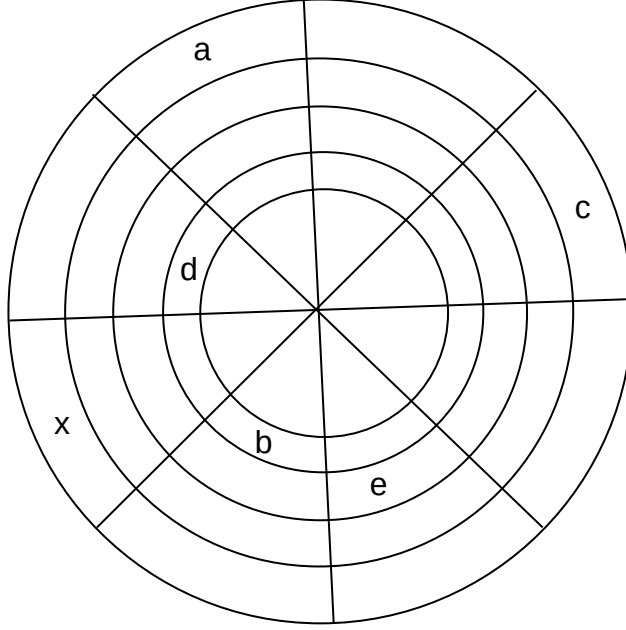


Figure 3: Disk Sorting Problem. Assume the characteristics of this disk are such that a single track seek takes longer than $1/8$ rotation, but less than $1/4$ rotation. Furthermore, the disk can seek 10s of tracks in less than $1/4$ rotation. The disk head is currently positioned at request a , and the queue is sorted in the greedy order of a , b , c , d , and e . Request x arrives. As x is in the same track as a , it can be serviced immediately after a . However, the new queue order once the head is at x is c , d , b , and e . Therefore, inserting x into the queue requires reordering the current entries.

sorting algorithm. The first simplification of this is to employ a greedy technique, selecting the minimal service time request. Variations of this technique were simulated in both [96] and [56] and found to behave extremely well.

However, even the greedy algorithm poses implementation difficulties. It requires that the queue be sorted by service time from the current position of the disk head. Each time a new request is added to the queue, the sort order of the other queued requests will change. Figure 3 depicts this phenomenon.

Preliminary simulation results in [71] show that selecting a desired I/O service time dependent upon current loading conditions and limiting the number of queued entries to search reduces the computational requirements of the disk scheduling algorithm to a practical value and obtains near optimal performance. These techniques,

called adaptive disk scheduling algorithms, will solve the disk scheduling problem until technology advances introduce new demands on the I/O subsystem (as is the case now that rotation times are close to seek times).

To date, we have investigated these techniques and implementation heuristics that reduce the computational demands in a very limited simulation environment. We will continue this work by first applying these algorithms to trace data and ultimately, by implementing them in a production system. Finally, we will investigate disk scheduling algorithms in the context of multidisk systems (RAIDS) found on most of today's high performance file servers.

Hewlett-Packard Corporation has supplied us with a large amount of trace data collected from their production machines. We will use this data to evaluate the algorithms and heuristics that we develop through simulation. We will then add the most promising algorithm(s) to our system and perform extensive performance evaluation to determine the benefit that can be achieved through careful disk scheduling. This work, in conjunction with the high performance file system activity described in the last section will give us a stable, high performance server with which to pursue the research in geographic caching.

2.5 Applications

2.5.1 Collaborative Planning Systems (B. J. Grosz)

The current dominant metaphor for human-computer interaction is a master-slave metaphor. Despite impressive advances in user-interface technology, users still tell computers explicitly what to do, and computers respond passively. For ubiquitous information systems, a new perspective is required: collaboration. In much the same way that people collaborate to solve problems and perform complex tasks, computers and people will also collaborate. The computer will move from being a tool to being a resource. The dialogue displayed in Figure 4, in which the participant NP is intended to be a computer system, illustrates the kind of collaborative problem solving and information sharing that future systems should have.

In this discourse, the network manager (NM) and the network presenter (NP) work together to determine the type of maintenance to perform on a switching node that can no longer handle the amount of traffic flowing through it. NM begins by stating the problem and then proceeds to ask for information that would be useful in solving it. NP supplies that information, in a combination of natural language and graphical displays. Furthermore, NP does not provide only the explicitly requested

- (1) NM: It looks like we need to do some maintenance on node39.
- (2) What kind of switch is it?
- (3) NP: It's an XYZ, but it's at full capacity.
- (4) NM: OK, then let's replace it with an XYZ+.
- (5) First, we'll have to divert the traffic to another node.
- (6) Which nodes could be used?
- (7) NP: *[puts up diagram]*
- (8) Node41 looks like it could temporarily handle the extra load.
- (9) NM: I agree.
- (10) Let's go ahead and divert the traffic to node41
 and do the replacement.

Figure 4: A sample dialogue with a network maintenance system.

information, but also makes suggestions and provides additional information [e.g. in utterance (3)]. It does not only wait passively for requests, but [e.g. utterance (8)] actively offers suggestions and specific solutions.

An ability to plan and act collaboratively is also essential to providing computer systems with an ability to communicate naturally and fluently with their users. Dialogues are fundamentally instances of collaborative behavior among multiple agents [45]. This collaborative property affects communication in all modalities and thus is a factor that must be reckoned with in developing more advanced systems for human-machine communication.

We have developed a model of collaborative planning, called SharedPlans, that provides a basis for human-computer collaborative problem solving systems [45, 75, 73]. The formalization of SharedPlans [42] is being tested both as a component of dialogue systems [74] and in distributed AI systems [62].

SharedPlans were developed initially to provide the intentional component of the model of discourse structure proposed by Grosz and Sidner (henceforth, G&S) [44]. The intentional structure plays a central role in discourse structure: the determination of discourse segmentation depends on identification of discourse intentions and relationships between them. According to G&S, each segment of a discourse has an underlying purpose intended by the speaker/writer to be recognized by the listener/reader, the *Discourse Segment Purpose* (**DSP**). Each **DSP** contributes to the overall *Discourse Purpose* (**DP**) of the discourse. For example, a discourse might

have as its **DP** the intention that the listener be able to navigate from his current location to a final destination, and individual segments forming that discourse might have as their **DSPs** intentions that the listener be able to navigate to intermediate destinations along the way. **DSPs** may in turn comprise other intentions and relations between them.

The SharedPlans model differs significantly from previous work on planning and plan recognition for discourse in taking the collaborative aspect of dialogue seriously. It differs from other current work on collaboration in not requiring any notion of the joint intentions of multiple agents. Instead, this model grounds group activity in the individual intentions and actions of individual agents, while providing specific constraints on the intentions of different agents to ensure collaboration.

In the past year we have utilized this model of discourse structure in examining intonational variation in discourse (Section 2.5.2), and have extended the formalization to handle groups of agents and complex activities involving actions by multiple agents at all levels of recursion. Under continuing support from U S WEST Advanced Technologies, we have begun to apply SharedPlans to systems for collaboration in providing direction assistance and multimedia access to complex information systems. Infrastructure support will allow us to connect and test these applications in a distributed system environment.

2.5.2 Computer Communication with Users (B. J. Grosz, S. M. Shieber)

Collaborative systems pose a new challenge to the user-interface designer. Collaboration will be easiest and most effective if we can communicate on our terms — using drawings, charts, gestures, natural language. Human-to-computer communication may involve natural-language understanding and the use of sophisticated graphical user interfaces; computer-to-human communication may involve natural-language generation and the use of graphical displays. Our recent research has focused on computer-to-human communication. It is, of course, central to the effectiveness of a ubiquitous information system that information be delivered in a way most appropriate for the receiver. For a computer system to participate in fluent dialogues, it must be able not only to determine what to say, but also to determine an appropriate way in which to say it.

In this context, collaboration between the user and ubiquitous information system precludes the use of “canned” graphical displays and natural-language utterances. To cooperatively solve problems not previously encountered, a collaborative system must be capable of designing graphical displays and composing natural-language utterances

to communicate new and arbitrary information in a context-sensitive manner; it must be linguistically and graphically articulate.

Our research on automated graphic design centers on developing computationally tractable techniques for generating graphical output that meets the constraints of the human perceptual system and dialogue context. Our research in coordinating graphics and natural language focuses on problems of determining which medium is most appropriate for communicating a given piece of information in a given context, and coordinating the manner in which the two media express this information. Our research in speech synthesis is directed at specifying representations of those discourse features essential for producing appropriate intonational variation at the discourse level.

Automated Graphic Design (S. M. Shieber) An example may help explain what we mean by a graphically articulate computer. Imagine a computer system, as described in Section 2.5.1, that participates in a collaborative effort with a human in routing telephone calls over a dynamically changing network of telephone lines and switches. The user of such a system may request various types of information—concerning levels of usage of lines, capacities and expected time to failure of switches, and so forth. This information is best presented in ways that depend not only on the type of information but its quantity, its relation to previously displayed information, even the goal which its presentation is in service of. For instance, a presentation of the loads on the lines might be organized as a network diagram with the nodes situated in positions approximating their geographical position if geographical trends are of interest. The diagram of Figure 5 shows a network diagram of this sort, which serves to highlight the comparatively high West Coast usage. This information would be obscured by a scatter plot of usage load per line with lines ordered by their length, say. But this latter display might well demonstrate a relation between line distance and load obscured by the network diagram design. (See Figure 6.) A graphically articulate computer must be fluent in handling these distinctions.

The development of methods to enable a computer to be graphically articulate has only recently received attention in the research community. The VIEW system designs iconic displays to communicate the result of database queries [35, 34]. Mackinlay [76] has developed methods for the automated design of graphs for the presentation of quantitative information. Feiner [28, 93] has shown how to automate the design of graphics that depict physical objects and actions. Our own research has led to the ANDD system for automating the design of network diagrams [77, 78].

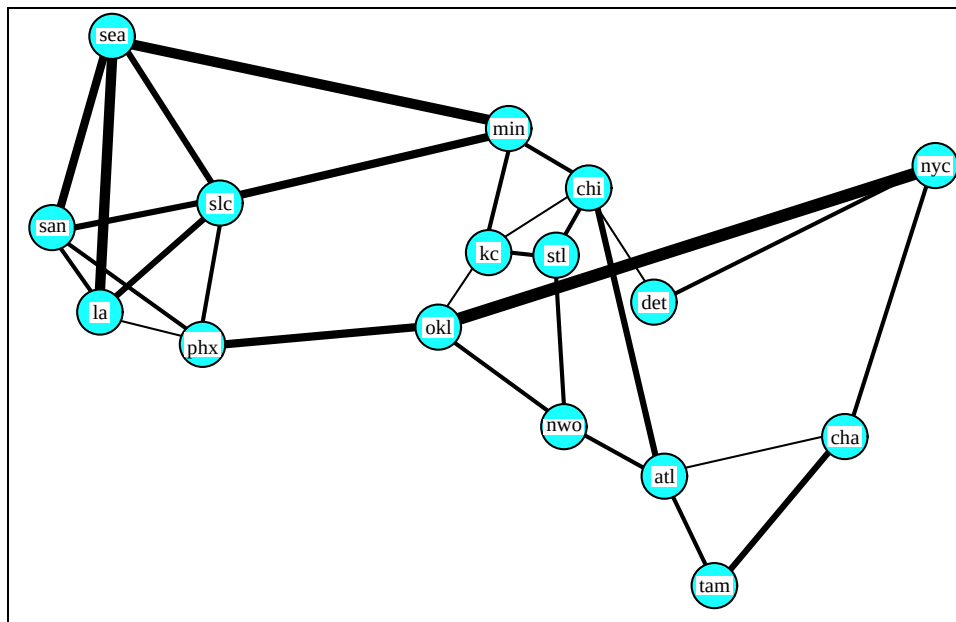


Figure 5: A network diagram showing line usage and using approximate geography for layout

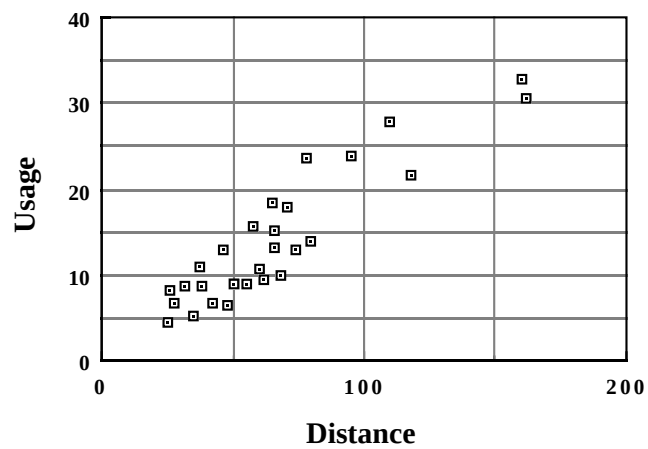


Figure 6: A scatter plot showing line usage versus line length

The design of many types of diagrammatic graphics—such as charts, graphs, network diagrams, maps, and so forth—can be thought of as the automating of *choices* while respecting *constraints*. From this perspective, automated graphic design is a problem of constrained combinatorial optimization. The constraints result from inherent properties of the graphic type, perceptual abilities of the viewer, aesthetic criteria, and the information to be conveyed. In the course of our research, methods for solving the constrained optimization problems implicit in graphical design problems have been developed. We have had especially good results using stochastic optimization methods.

In previous research (with continuing funding from U S WEST Advanced Technologies and NSF), we have concentrated on two testbed problems in automated graphic design. The first is the problem of network diagram layout. In particular, we have developed a robust system for the design of network diagrams incorporating a parallel genetic algorithm to handle the computationally difficult task of layout [79]. This system was incorporated into a software environment for scientific and program visualization [61].

Second, we have attacked the task of automating the process of designing and generating maps [80, 18]. A major factor affecting the clarity of graphical displays such as maps that include text labels is the degree to which labels obscure display features (including other labels) as a result of spatial overlap. Point-feature label placement (**PFLP**) is the problem of placing text labels adjacent to point features on a map or diagram in order to maximize legibility. This problem occurs frequently in the production of many types of informational graphics, though it arises most often in automated cartography. Complexity analysis reveals that the basic **PFLP** problem and most interesting variants of it are NP-hard. Not surprisingly, therefore, all previously reported algorithms for **PFLP** either have exponential time complexity or are incomplete. To solve the **PFLP** problem in practice, then, we must rely on good heuristic methods. We have proposed two new methods, one based on a discrete form of gradient descent, the other on simulated annealing, and have compared these and the other known algorithms for the problem. Our results indicate that the simulated annealing approach to **PFLP** is superior to all existing methods, regardless of label density.

For both of these computationally difficult problems, our stochastic methods are currently the best methods known for automating the graphic design process, yet they are computationally feasible on real problems. Nonetheless, as we come to understand the combinatorial optimization problems implicit in graphic design tasks better, we intend to develop more direct methods for these and other graphic design problems.

We have also investigated the coordination of graphics and natural language. Certain kinds of display may be able to convey only a subset of the information to be communicated, or may only be suitable for displaying certain types of information. Sometimes sequences of displays will be more appropriate than individual displays. Some information may not be suitable for graphical communication at all. The negotiation of these issues falls under the rubric of content apportionment, which is the allotment of information among various display and linguistic media. Closely related to the content-apportionment problem is the media-coordination problem: having apportioned the information to be communicated among various display and linguistic media, there remains the problem of coordinating these media. Finally, the impact of discourse context on graphic design must be considered. Preliminary research in this area led to a prototype implementation of a system for describing causal models through coordinated network diagrams and natural language. The range of expressivity of the system was quite limited, but it served as a proof of concept for the research program.

Our research in this area will concentrate on the following problems: (1) development and exploration of alternative optimization methods for the diagram layout and label placement problems; (2) extension of algorithms to more general problems in automated graphic design, especially problems in automated cartography and page layout problems; and, (3) coordination of natural language and graphics through integrated methods of solving linguistic and graphical design problems.

The computational facilities requested in this proposal are central to research on these problems. As an example, a single diagram from a recent paper on our label placement algorithm summarizing several thousand experiments required several weeks of computer time to generate. The requested facilities will have a clear effect on the rapidity with which such empirical studies can be completed.

Speech Synthesis (B. J. Grosz) In many collaborative work situations, speech is a preferred mode of communication. For a computer system to participate in dialogues using speech, it must be able not only to determine what to say, but also to determine an appropriate way in which to say it. Each individual utterance must be produced in a way that fits with the surrounding discourse context: following from the preceding utterance and providing the right base for the subsequent utterance.

A fundamental problem in speech synthesis is the absence of algorithms for automatic intonational variation [60]. This lack stems not from the inability of synthesis systems to produce natural-sounding intonation; systems such as DEC-Talk and the

AT&T synthesizer can be hand-tweaked to sound quite natural. However, no system currently represents the discourse-level information necessary to assign appropriate intonational features automatically, either from text analysis (text-to-speech) or from an abstract representation of the message to be conveyed (message-to-speech). Nor do algorithms exist which could make use of such information to associate intonational features such as pitch range, pulse duration, and speaking rate appropriately. In fact, the fundamental research necessary to determine the relationship between intonational variation and aspects of local and global discourse structure remains largely to be done.

Davis and Hirschberg [25] were able to produce improved prosodic variation in directions given by a “direction assistance” system, using a model of discourse structure based on Grosz and Sidner’s model of discourse structure [44] [G&S] (Section 2.5.1). Although this system demonstrated that G&S could be used to improve speech synthesis, each component of the discourse model was simplified in this system, many facets of discourse structure were hand-coded, and variation in intonational features was limited.

In research with Julia Hirschberg at AT&T Bell Laboratories, we performed a pilot study of discourse structure and intonation based on G&S³ [52, 41]. This study, supported in part by a grant from NSF, examined both the global level (discourse segments and relations between them) and the local level (features of utterances within a discourse segment) of discourse. We found statistically significant correlations of aspects of pitch range, amplitude, and timing with features of global and local structure both for labelings from text alone and for labelings from speech. We further found that global and local structures can be reliably identified from acoustic and prosodic features with (cross-validated) success rates of 86-97%.

Our studies identified a range of acoustic-prosodic features and discourse elements that are candidates for further testing in a larger corpus. The initial experiments also provided some indications that more complex models than previously thought may be needed to model the relationships between intonational and discourse features. For example, parentheticals seem to engender several patterns of associated acoustic-prosodic events: one subgroup was uttered very rapidly, while another was uttered

³The hypothesis that discourse structure is signaled by variation in intonational features such as pitch range, timing, and amplitude has been examined in several studies [69, 11, 100, 4]. However, a significant weakness of these studies is the lack of independent analyses of the structure of the discourses under consideration. As Brown and her colleagues state, [11, p. 27]: “...until an independent theory of topic-structure is formulated, much of our argument in this area is in danger of circularity.” The G&S model is such an independent model.

more slowly but in a lower pitch range. We are currently engaged in a more complete study (funded by NSF) based on the results of this pilot. The new study will require analysis of a large database of spoken language using graphics-based software for speech analysis. The results of this study will be used to specify representations of those discourse features essential for producing appropriate intonational variation at the discourse level, and to develop algorithms utilizing these representations to produce such intonational variation in speech synthesis systems.

In a separate pilot study of spontaneous narrative speech [85], we have shown that pitch accent on subject pronouns interacts with various attentional state factors at what G&S term the local level [44, 43]. A more extensive study of the interactions among intonational prominence, form of referring expression, and attentional state is underway. We expect to integrate these results as well into a text-to-speech synthesis system that uses discourse structure information to assign intonational features.

Thus, the focus of our research in speech synthesis will be on (1) incorporating the full G&S model, (2) specifying those properties of intentional structure and attentional state to which a speech synthesis system must have recourse, and (3) providing a more complete model of the relationships between intonational variation and discourse structure. Infrastructure support will provide systems for speech analysis and the disk storage for the data files needed for this analysis. The collaborative research setting will provide an environment in which to test the resulting speech synthesis systems.

2.5.3 Large Databases of Text and Images (M. I. Seltzer, S. M. Shieber)

Robust Natural Language Processing (S. M. Shieber) Our research in natural language processing is centered around efforts at better understanding of the structure and behavior of human language in the long term, and for the short term, around developing statistical models of language that allow useful approximations to the actual structure. The availability of large databases of free text presents both an opportunity and a problem for ubiquitous information systems. The opportunity of making use of such text databases by forwarding pertinent portions to users on the basis of their needs is obvious. But to enable such a system, the ubiquitous information system must be able to analyze the text robustly for effective retrieval of the pertinent passages. The natural-language-processing technology to allow such retrieval is still in its infancy, but the need for the technology is increasingly evident.

Three central problems — robustness, fluency, and modularity — confront researchers addressing the natural-language processing problems necessary for ubiq-

uitous information. These issues can be addressed through the use of systems of constraints as the underlying method for encoding the structures of natural language [98].

- *Robustness:* Natural-language-processing systems tend to be fragile, especially in the face of novel or unknown aspects of language. Existing grammatical formalisms assume that all constraints are categorical as opposed to defeasible in some sense, leading to fragile behavior. Recently, there has been increasing interest in statistical constraints as a basis for robust natural-language processing, but the statistical models have been too impoverished to scale appropriately to the rich structure of natural language. Our research [97] on combining statistical and grammatical approaches to language is an effort to combine the best aspects of grammatical and statistical sources of constraint.
- *Fluency:* Systems must become more fluent in reconstructing the ubiquitous implicit relationships that hold within and among sentences. These relationships are perhaps best exemplified by elliptical and coordinate constructions, which are found universally in language and which have been among the most intransigent problems in natural-language processing. We have begun studying the addition of higher-order constraints that may allow for solution of some of these problematic cases [24].
- *Modularity:* To make natural-language-processing systems easier to build and extend, they must be structured in a more modular fashion. For instance, the factoring of phrase-structure information postulated in the so-called tree-adjoining grammar formalism can eliminate the need for many of the constraints found in more traditional constraint-based formalisms [99].

Our research (funded in part by U S WEST Advanced Technologies and NSF) will make use of the computational infrastructure requested in this proposal in several ways.

- The computations necessary to enable statistical natural-language analysis require large quantities of data and tremendous computational resources, though they are quite amenable to parallel implementation. We will use the computational and storage resources to make possible exploration of a variety of these statistical approaches, especially our proposal based on stochastic tree-adjoining grammars.

- The study of natural language constructions such as elliptical and coordinate constructions is greatly furthered by the ability to analyze naturally occurring text in large computer-readable corpora for evidence of the underlying structure. Our analysis of ellipsis, for instance, was aided by small experiments along these lines. The availability of large text corpora that this proposal would make available makes possible the amplification of these efforts.
- The decomposition of linguistic structure can be studied through the use of large text corpora and related materials of the type now becoming available through the Linguistic Data Consortium. Membership in the Consortium will make these databases available, and the computational infrastructure requested here will allow for the storage, retrieval and use of such natural-language databases.

Searching Large Imagebases (M. I. Seltzer) Ubiquitous information systems must be able to provide storage and rapid retrieval of image data as well as textual data. Conventional indexing and retrieval techniques for text are difficult to apply to image bases because they rely on more exact matching than is possible with visual images. While keyword indices may be created for small image bases, doing so for large ones is practically impossible. For example, the Earth Observing System generates more than two gigabytes of image data per day. Of course, some attributes of this data can be automatically indexed, (e.g. longitude, latitude, and time of day), but that information alone is not sufficient to answer content-based queries such as, “Select images with more than 70% cloud cover.” Our goal is to provide mechanisms for content-based browsing of image data using interactive genetic algorithms.

Several techniques have been applied to the problem of content-based retrieval of images. Fine et al. [29] use compressed images and conventional database retrieval techniques to access hundreds of gigabytes of satellite data. They rely on sensor data collected with the images to select representative miniatures to display in response to content-based queries. Although their techniques decrease retrieval time and enhance the usefulness of the data, they are application specific and are practical only when the associated sensor data is available.

More general-purpose content-based retrieval systems have been constructed by adopting image processing techniques. For example, the FINDIT system [103] performs retrieval based on color. Images are preprocessed to create color histograms on both the entire image and smaller regions of the image. The system responds to queries by presenting the user images that are “similar” to previously selected images where the similarity is defined by the color histogram. The QBIC (Query by Image

Content) system [27] uses similar techniques to provide content-based retrieval using indices for color, texture, and shape. The color indices are constructed similarly to those in FINDIT. The texture indices have three components: contrast, coarseness, and directionality. Contrast is expressed in terms of the variance of gray-level histograms. Coarseness is a measure of feature size, computed by analyzing various scales of the image, and directionality is expressed as the jaggedness of the distribution of the gradient directions. Shape indices are based upon the work of Hirata [51, 57], matching low-resolution edge maps of the database images to a query image. The user may request similarity along one of the provided indices (color, texture, or shape).

The type of interaction provide by FINDIT and QBIC requires that the user have a preconceived notion of the target of the query and know which aspects of the query image are similar. However, in many applications (e.g. face recognition), users cannot precisely articulate what feature or aspect in an image is similar, only that the entire image is somehow related [12]. For such applications, a search procedure is needed that can use all available indices simultaneously.

As a first step toward providing this kind of search capability, we will investigate the use of interactive genetic algorithms. A genetic algorithm is a search procedure based on the mechanics of natural selection. An object is described by a set of parameters (genes), and each unique set of parameter values forms the genotype or genetic map for an object. In a conventional genetic algorithm, a static selection criteria (a function over the genotype) is computed and the most fit objects survive to produce the next generation of candidate objects. In an interactive genetic algorithm, the user performs the selection step. This approach is particularly useful when the selection criteria are difficult to define, as is the case when evaluating the qualities or attributes of an image. In both cases, the process is repeated to model the evolution of superior objects.

An alternative interpretation of this procedure is that each object represents a point in N -space, where N is determined by the number of parameters in the genotype. The evolution of a specific object is a directed walk through N -space. Typically, in a genetic algorithm, every point in N -space represents an object. However, when we apply the genetic algorithm to searching a finite database, this is no longer true; only those points corresponding to images in the database are occupied. Therefore, a conventional genetic algorithm must be modified to produce only those progeny that are actually present in the database.

As in a conventional interactive genetic algorithm, an interactive genetic algorithm begins the search with a collection of images chosen from the database. This choice

may be random, or broadly directed by the user. Then, for each generation of the algorithm, the user selects images that are similar or in some way related to the images of interest. The evolution step is a two-phased process. In the first phase, the genotypes of the selected images are mated to produce prototype progeny. In the second phase, we find the nearest neighbors in N-space for each prototype and use the images represented by those points as the next-generation progeny.

This approach raises several research questions. The most general is whether this approach is effective for database browsing. The work of Caldwell and Johnston [12] suggests that with a finite database and constrained mutation and mating operators, it is possible to locate specific objects in the space. On the other hand, in the less constrained Drawing Evolver [5], it was difficult to find a specific image in the nearly infinite space; this results suggests that the challenge lies in the ability to constrain the search space. Therefore, much of our initial work is focused on developing proper mutation and mating operators.

To facilitate this investigation, we are using a constrained database, line drawings of faces produced by Baker's Drawing Evolver [5]. Because these drawings were all produced using an interactive genetic algorithm, we have the complete genotypic representation of each image, so that image-processing issues are reduced in this initial experimentation. It might be useful in addition to enhance the genotype with computed parameters such as distance between the eyes and the ratio of the vertical and horizontal dimensions of the face. Both the computed parameters and the simple genotypic parameters are used to create the N-space representation of each face. We are beginning the evaluation of this methodology using the same mutation and mating operators as used in the original system. We will augment the operators by retaining the history of user selections and applying weights to the possible transformations. In this way, we will be able to direct the search more efficiently in the directions that the user can be inferred to be pursuing.

The second important research question is finding efficient structures with which to represent and search N-space. The database literature suggests several multi-dimensional structures [46, 86, 30], but in practice, these become less effective at dimensions higher than three, and we are using dimensions of ten to twenty. Furthermore, for dimensions greater than or equal to three, there is no known algorithm for nearest neighbor searching that is both linear in space and polylogarithmic in complexity. As our algorithm does not depend on locating the exact nearest neighbor, we are investigating approximate nearest neighbor algorithms (e.g. [3]). Additionally, we are exploring modifications to the splitting algorithms in quad trees [30].

After our initial exploration of browsing the line drawings of faces, we will expand

our database complexity incrementally. First, we will move from line drawings to full images, using the face database collected by Gilbert [38]. Next we will expand our database to include a broader range of images. The Harvard University Art Museum is currently undertaking an effort to digitize its collection of drawings (numbering approximately 19,000) and this collection will be available on-line for use in research. Finally, we will investigate multi-attribute browsing in a database containing full color images using the indexing structures of Swain and Niblack et. al.

Our research in this area will thus concentrate on the following problems: (1) exploration of the application of interactive genetic algorithms for image browsing; (2) design, implementation, and evaluation of data structures to represent N-dimensional data facilitating nearest neighbor searching; and, (3) design and implementation of image browsing systems.

References

- [1] Thomas E. Anderson, Brian N. Bershad, Edward D. Lazowska, and Henry M. Levy. Scheduler activations: Effective kernel support for the user-level management of parallelism. In *Proceedings of 1991 Symposium on Operating Systems Principles*, pages 95–109, March 1991.
- [2] Thomas E. Anderson, Henry M. Levy, Brian N. Bershad, and Edward D. Lazowska. The interaction of architecture and operating system design. In *Proceedings of 4th International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 108–120, April 1991.
- [3] S. Arya and D. Mount. Approximate nearest neighbor queries in fixed dimensions. In *Proceedings of the 4th ACM-SIAM Symposium on Discrete Algorithms*, pages 271–280, 1993.
- [4] G. M. Ayers. Discourse functions of pitch range in spontaneous and read speech. In *Presented at the Linguistic Society of America Annual Meeting*, 1992.
- [5] E. Baker. Summary: Evolving line drawings. In *Proceedings of the Fifth Annual Conference on Genetic Algorithms*, July 1993.
- [6] M. Baker, S. Asami, E. Deprit, J. Ousterhout, and M. Seltzer. Nvram for fast, reliable file systems. In *Proceedings of the Fifth International Conference*

on Architectural Support for Programming Languages and Operating Systems, Boston MA, October 1992.

- [7] M. Baker and J. Hartman. Measurements of a distributed operating system. In *Proceedings of the Ninth Symposium on Operating System Principles*, Monterey, CA, November 1991.
- [8] Preliminary report on broadband isdn transfer protocols. Bellcore Special Report, SR-NWT-001763, Dec. 1990.
- [9] Brian N. Bershad, Thomas E. Anderson, Edward D. Lazowska, and Henry M. Levy. Lightweight remote procedure call. In *Proceedings of 1989 Symposium on Operating Systems Principles*, pages 102–113, March 1989.
- [10] Shekhar Borkar, Robert Cohn, George Cox, Thomas Gross, H.T. Kung, Monica Lam, Margie Levine, Brian Moore, Wire Moore, Craig Peterson, Jim Susman, Jim Sutton, John Urbanski, and Jon Webb. Supporting systolic and memory communication in iwarp. In *Proceedings of 17th Annual International Symposium on Computer Architecture*, pages 70–81, May 1990.
- [11] G. Brown, K. Currie, and J. Kenworthy. *Questions of Intonation*. University Park Press, Baltimore, 1980.
- [12] C. Caldwell and V. Johnston. Tracking a criminal suspect through face space with a genetic algorithm. In *Proceedings of the Fourth International Conference on Genetic Algorithms*, pages 416–421, 1991.
- [13] A. Chang, M. Mergen, R. Rader, J. Roberts, and S. Porter. Evolution of storage facilities in aix version 3 for risc system/6000 processors. *IBM Journal of Research and Development*, 34(1), January 1990.
- [14] Pohua P. Chang, Scott A. Mahlke, William Y. Chen, Nancy J. Warter, and Wen mei W. Hwu. Impact: An architectural framework for multiple-instruction-issue processors. In *Proceedings of 18th Annual International Symposium on Computer Architecture*, pages 266–275, May 1991.
- [15] Pohua P. Chang, Scott A. Mahlke, and Wen mei W. Hwu. Using profile information to assist classic code optimizations. Center for Reliable and High-Performance Computing Report CRHC-91-12, University of Illinois at Urbana-Champaign, Urbana, IL 61801, April 1991.

- [16] J. Bradley Chen. Software methods for system address tracing. In *Proceedings 4th Workshop on Workstation Operating Systems*, October 1993.
- [17] David R. Cheriton and Robert A. Kutter. Optimizing memory-based messaging for scalable shared memory multiprocessor architectures. In preparation.
- [18] Jon Christensen, Joseph Marks, and Stuart M. Shieber. Algorithms for cartographic label placement. In *Proceedings of the 1993 Annual Joint Convention of the ASPRS and ACSM*, 1993.
- [19] David D. Clark, Van Jacobson, John Romkey, and Howard Salwen. An analysis of tcp processing overhead. *IEEE Communications Magazine*, 27(6):23–29, 1989.
- [20] E. Coffman, L. Klimko, and B. Ryan. Analysis of scanning policies for reducing disk seek times. *SIAM Journal of Computing*, 1(3), September 1972.
- [21] Bernard Cole. The technology framework (part of special article on interactive multimedia). *IEEE Spectrum*, 30(3):32–39, 1993.
- [22] E. C. Cooper, P. A. Steenkiste, R. D. Sansom, and B. D. Zill. Protocol implementation on the nectar communication processor. In *Proc. SIGCOMM '90 Symposium on Communications Architectures and Protocols*, Philadelphia, Sep. 1990.
- [23] David E. Culler, Anurag Sah, Klaus Erik Schauser, Thorsten von Eicken, and John Wawrzynek. Fine-grain parallelism with minimal hardware support: A compiler-controlled threaded abstract machine. In *Proceedings of 4th International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 164–175, April 1991.
- [24] Mary Dalrymple, Stuart M. Shieber, and Fernando C. N. Pereira. Ellipsis and higher-order unification. *Linguistics and Philosophy*, 14:399–452, 1991.
- [25] James Raymond Davis and Julia Hirschberg. Assigning intonational features in synthesized spoken directions. In *Proceedings of the 26th Annual Meeting of the Association for Computational Linguistics*, pages 187–193. State University of New York at Buffalo, June 7-10 1988.

- [26] K. Ebcioglu and A. Nicolau. A global resource-constrained parallelization technique. In *Proceedings 3rd International Conference on Supercomputing*, pages 154–163, 1989.
- [27] Niblack et. al. The qbic project: Querying images by content using color, texture, and shape. Ibm research report rj9203 (81511), IBM Almaden Research Center, February 1993.
- [28] S. Feiner. Apex: An experiment in the automated creation of pictorial explanations. *IEEE CG&A*, pages 29–37, November 1985.
- [29] et al. Fine, J. Abstracts: A latency-hiding technique for high capacity mass-storage systems. Sequoia 2000 Technical Report 92/9, University of California, Berkeley, Computer Science Division, June 1992.
- [30] R. Finkel and J. Bentley. Quad trees – a data structure for retrieval of composite keys. *Acta informatica*, 4:1–9, 1974.
- [31] Joseph A. Fisher. Trace scheduling: A technique for global microcode compaction. *IEEE Transactions on Computers*, C-30(7):478–490, July 1981.
- [32] H. J. Fowler and W. E. Leland. Local area network traffic characteristics, with implications for broadband network congestion management. *IEEE J. on Selected Areas in Communications*, 9(7):1139–1149, September 1991.
- [33] Ron Frederick. nv — X11 videoconferencing tool. Available via anonymous ftp from ftp.ee.lbl.gov., March 1993.
- [34] M. Friedell. Automatic synthesis of graphical object descriptions. *Computer Graphics (Proc. SIGGRAPH '84)*, 18(3):53–62, July 1984.
- [35] M. Friedell, J. Barnett, and D. Kramlich. Context-sensitive, graphic presentation of information. *Computer Graphics (Proc. SIGGRAPH '82)*, 16(3):181–188, July 1982.
- [36] M. W. Garrett. *Statistical Analysis of a Long Trace of Variable Bit Rate Video Traffic*. PhD thesis, Columbia University, 1993. Chapter IV of Ph.D. Thesis.
- [37] R. Geist and S. Daniel. A continuum of disk scheduling algorithms. *ACM Transactions on Computer Systems*, 5(1), February 1987.

- [38] J. Gilbert. Jeff's thesis, April 1993. Senior Thesis, Harvard University.
- [39] J. Gray. Notes on database operating systems. In R. Bayer, R. M. Graham, and G. Seegmueller, editors, *In Lecture Notes on Computer Science*, volume 60. Springer-Verlag, New York, 1981.
- [40] J. Gray. A census of tandem system availability between 1985 and 1990. *IEEE Transactions on Reliability*, 39(4):409–418, October 1990.
- [41] Barbara Grosz and Julia Hirschberg. Some intonational characteristics of discourse structure. In *Proc. of International Conference on Spoken Language Processing*, Banff, Alberta, Canada, October 12-16 1992. to appear.
- [42] Barbara Grosz and Sarit Kraus. Collaborative plans for group activities. In *Proceedings of IJCAI-93*, Chambéry, France, 1993.
- [43] B.J. Grosz, A.K. Joshi, and S. Weinstein. Providing a unified account of definite noun phrase in discourse. In *Proceedings of the 21st Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, June 1983.
- [44] B.J. Grosz and C.L. Sidner. Attention, intentions, and the structure of discourse. *Computational Linguistics*, 12(3), 1986.
- [45] B.J. Grosz and C.L. Sidner. Plans for discourse. In P.R. Cohen, J.L. Morgan, and M.E. Pollack, editors, *Intentions in Communication*. Bradford Books at MIT Press, 1990.
- [46] A. Guttman. R-trees: A dynamic index structure for spatial searching. In *Proceedings of the 1984 International Conference on the Management of Data*, pages 47–57, August 1984.
- [47] Mark Hayter and Derek McAuley. The desk area network. *ACM Operating Systems Review*, 25(4):14–21, October 1991.
- [48] John Hennessy, John Hennessy, Norman Jouppi, Forest Baskett, Thomas Gross, John Gill, and Steven Przybylski. Hardware/software tradeoffs for increased performance. In *Proceedings of Symposium on Architectural Support for Programming Languages and Operating Systems*, pages 2–11, March 1982.

- [49] Dana S. Henry and Christopher F. Joerg. A tightly-coupled processor-network interface. In *Proceedings of 5th International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 111–122, October 1992.
- [50] Mark D. Hill, James R. Larus, Alvin R. Lebeck, Madhusudhan Talluri, and David A. Wood. Wisconsin architectural research tool set. *Computer Architecture News*, 21(4):8–10, September 1993.
- [51] K. Hirata and T. Kato. Query by visual example. In *Proceedings of Third International Conference on Extending Database Technology, EDBT '92*, pages 56–71, Vienna, Austria, March 1992.
- [52] J. Hirschberg and B. Grosz. Intonational features of local and global discourse structure. In *Working Notes of the DARPA Workshop on Speech and Natural Language*, Arden House, Harriman, New York, February 1992.
- [53] David Hitz. File system design for an nfs file server appliance. In *Proceedings of the 1994 Winter Usenix*, San Francisco, CA, January 1994. to appear.
- [54] M. Hofri. Disk scheduling: Fcfs vs. sstf revisited. *Communications of the ACM*, 23(11), November 1980.
- [55] M. Holland and G. Gibson. Parity declustering for continuous operation in redundant disk arrays. In *Proceedings of the Fifth International Conference on Architectural Support for Programming Languages and Operating Systems*, Boston, MA, October 1992.
- [56] D. Jacobson and J. Wilkes. Disk scheduling algorithms based on rotation position. Technical report hpl-csp-91-7, Hewlett Packard, Palo Alto, CA, February 1991.
- [57] T. Kato, T. Kurita, N. Otsu, and K. Kirata. A sketch retrieval method for full color image database. In *Proceedings of the International Conference on Pattern Recognition*, pages 530–533, The Hague, Netherlands, September 1992.
- [58] R. Katz, J. Ousterhout, D. Patterson, and M. Stonebraker. A project on high performance i/o subsystems. IEEE Computer Society Technical Committee on Database Engineering Newsletter, March 1988.

- [59] M. Kazar, B. Leverett, O. Anderson, A. Vasilis, B. Bottos, S. Chutani, C. Everhart, A. Mason, S. Tu, and E. Zayas. Decorum file system architectural overview. In *Proceedings of the 1990 Summer Usenix*, pages 151–164, Anaheim, CA, June 1990.
- [60] Dennis H. Klatt. Review of text-to-speech conversion for English. *JASA*, 82(3):737–793, 1987.
- [61] Sandeep Kochhar, Mark Friedell, Steve Sistare, Janusz Juda, Mark LaPolla, Joe Marks, Peter McMurry, Corey Kosak, and Stuart Shieber. A design environment for scientific and program visualizations. *Proceedings of Compugraphics '91, the First International Conference on Computational Graphics and Visualization Techniques*, pages 321–332, 1991. September 30 - October 2, Sesimbra, Portugal.
- [62] S. Kraus. Agents contracting tasks in non-collaborative environments. In *Proc. of AAAI-93*, Washington, D.C., 1993.
- [63] H. T. Kung. Gigabit local area networks: A systems perspective. *IEEE Communications Magazine*, 30:79–89, 1992.
- [64] H. T. Kung and A. Chapman. The fcvc (flow controlled virtual channels) proposal for atm networks, version 1.1, july 1993. In *Proceedings of 1993 International Conference on Network Protocols*, San Francisco, California, October 19-22 1993. A summary of the paper is to appear.
- [65] H.T. Kung, R. Morris, T. Charuhas, and D. Lin. Use of link-by-link flow control in maximizing atm networks performance: Simulation results. In *Proceedings of IEEE Hot Interconnects Symposium*, '93 Palo Alto, California, August 1993.
- [66] Monica S. Lam. Instruction scheduling for superscalar architectures. *Annual Review of Computer Science*, 4:173–201, 1990.
- [67] Monica S. Lam and Robert P. Wilson. Limits of control flow on parallelism. In *Proceedings of 19th Annual International Symposium on Computer Architecture*, pages 46–57, May 1992.
- [68] Samuel J. Leffler, Marshall Kirk McKusick, Michael J. Karels, and John S. Quarterman. *The Design and Implementation of the 4.3BSD UNIX Operating System*. Addison-Wesley Publishing Company, Inc., Reading, MA, 1989.

- [69] I. Lehisté. *Perception of sentence and paragraph boundaries*, pages 191–201. Academic Press, London, 1979.
- [70] W. E. Leland, M. S. Taqqu, W. Wilinger, and D. V. Wilson. On the self-similar nature of ethernet traffic. In *Proceedings of SIGCOMM '93 Symposium on Communications Architectures and Protocols*, 1993.
- [71] D. Lin and K. Chang. Adaptive disk scheduling algorithms. Unpublished report for the class Computer Science 262; Advanced Operating Systems, Harvard University Division of Applied Sciences, May 1993.
- [72] G. Lindsay, P. Selinger, C. Galtieri, J. Gray, R. Lorie, F. Putzolu, I. Traiger, and B. Wade. *Single and Multi-site Recovery Facilities, Distributed Data Bases*, chapter 10. Cambridge University Press, 1980.
- [73] K. E. Lochbaum. An algorithm for plan recognition in collaborative discourse. In *Proceedings of the 29th Annual Meeting of the ACL*, Berkeley, CA, 1991.
- [74] Karen E. Lochbaum. A collaborative planning approach to discourse understanding. Technical report, Harvard University, 1993. Forthcoming.
- [75] K.E. Lochbaum, B.J. Grosz, and C.L. Sidner. Models of plans to support communication: An initial report. In *Proceedings of the Eighth National Conference on Artificial Intelligence - AAAI-90*, Boston, MA, 1990.
- [76] Jock Mackinlay. Automating the design of graphical presentations of relational information. *ACM Transactions on Graphics*, 5(2):110–141, April 1986.
- [77] J. Marks. Automating the design of network diagrams. Technical Report 02-90, Center for Research in Computing Technology, Harvard University, Cambridge, MA, January 1990.
- [78] J. Marks. A syntax and semantics for network diagrams. In *Proceedings of the 1990 IEEE Workshop on Visual Languages*, pages 104–110, October 1990.
- [79] Joseph Marks and Stuart M. Shieber. The computational complexity of cartographic label placement. Technical Report Technical Note 05-91, Harvard University, Center for Research in Computing Technology, 1991.

- [80] Joseph Marks and Stuart M. Shieber. The computational complexity of cartographic label placement. Technical Report 05-91, Center for Research in Computing Technology, Harvard University, 1991.
- [81] M. McKusick, W. Joy, S. Leffler, and R. Fabry. A fast file system for unix. *Transactions on Computer Systems*, 2(3):181–197, August 1984.
- [82] J. Moran, R. Sandberg, D. Coleman, J. Kepecs, and B. Lyon. Breaking through the nfs performance barrier. In *Proceedings of the EUUG Spring 1990 Conference*, pages 199–206, Munich, Germany, April 1990.
- [83] K. Muller and J. Pasquale. A high performance multi-structured file system design. In *Proceedings of the Thirteenth Symposium on Operating Systems Principles*, pages 56–67, Monterey, CA, 1991.
- [84] David Nagle, Richard Uhlig, and Trevor Mudge. Monster: A tool for analyzing the interaction between operating systems and computer architectures. Technical Report CSE-TR-147-92, University of Michigan, Ann Arbor, MI, November 1992.
- [85] Christine Nakatani. Accenting on pronouns and proper names in spontaneous narrative. In *European Speech Communication Association Workshop on Prosody*, Lund, Sweden, September 27-29, 1993.
- [86] J. Nievergelt, J. Hinterberger, and K. Sevcik. The grid file: An adaptable symmetric multikey file structure. *ACM Transactions on Database Systems*, 9(1), 1984.
- [87] R.S. Nikhil, G.M. Papadopoulos, and Arvind. *t: A multithreaded massively parallel architecture. In *Proceedings of 19th Annual International Symposium on Computer Architecture*, pages 156–167, May 1992.
- [88] W. Oney. Queuing analysis of the scan policy for moving-head disks. *Journal of the ACM*, 22(3), July 1975.
- [89] D. Patterson, G. Gibson, and R. Katz. A case for redundant arrays of inexpensive disks (raid). In *Proceedings of the ACM SIGMOD Conference*, pages 109–116, Chicago, IL, June 1988.

- [90] Rahul Razdan and Mike Smith. Dynamically-programmable instruction sets. In preparation.
- [91] Stephen Richardson and Mahadevan Ganapathi. Code optimization across procedures. *Computer*, 22(2):42–50, February 1989.
- [92] M. Rosenblum. The design and implementation of a log-structured file system. *Transactions on Computer Systems*, 10(1):26–52, February 1992.
- [93] D.D. Seligmann and S. Feiner. Specifying composite illustrations with communicative goals. In *ACM SIGGRAPH Symposium on User Interface Software and Technology (UIST '89)*, pages 1–9, 1989. November 13-15, Williamsburg, VA.
- [94] M. Seltzer. The design and implementation of the 4.4bsd log-structured file system. In *Proceedings of the 1993 Winter Usenix*, San Diego, CA, January 1993.
- [95] M. Seltzer and M. Olson. Libtp: Portable, modular transactions for unix. In *Proceedings 1992 Winter Usenix Conference*, San Francisco, CA, January 1992.
- [96] M. Seltzer and M. Stonebraker. Transaction support in read optimized and write optimized file systems. In *Proceedings 16th International Conference on Very Large Data Bases*, Brisbane, Australia, 1990.
- [97] Stuart M. Shieber. Reconciling abstract structure and concrete data in statistical language processing. In *Proceedings of the 1991 IEEE Workshop on Automatic Speech Recognition*, pages 57–59, Arden House, Harriman, New York, December 15–18 1991.
- [98] Stuart M. Shieber. *Constraint-Based Grammar Formalisms*. MIT Press, Cambridge, Massachusetts, 1992.
- [99] Stuart M. Shieber and Yves Schabes. Synchronous tree-adjoining grammars. In *Proceedings of the 13th International Conference on Computational Linguistics*, pages 1–6, 1990.
- [100] K.E.A. Silverman. *The structure and processing of fundamental frequency contours*. PhD thesis, University of Cambridge, 1987. Unpublished doctoral dissertation.

- [101] Michael D. Smith, Mark A. Horowitz, and Monica S. Lam. Efficient superscalar performance through boosting. In *Proceedings of 5th International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 248–259, October 1992.
- [102] M. Stonebraker and J. Dozier. Sequoia 2000: Large capacity object servers to support global change research. Sequoia 200 Technical Report 91/1, University of California, Berkeley, Computer Science Division, July 1991.
- [103] Michael Swain. Interactive indexing into image database. *Image and Video Storage and Retrieval (SPIE)*, 19(8):95–102, 1993.
- [104] Andrew S. Tanenbaum. *Computer Networks*. Prentice-Hall, Inc., Englewood Cliffs, NJ, 1989.
- [105] T. Teorey and T. Pinkerton. A comparative analysis of disk scheduling policies. *Communications of the ACM*, 15(3), March 1972.
- [106] The vxfs file system type. from Advanced System Administration for UNIX SVR4.2, 1992. Unix System Laboratories.
- [107] Thorsten von Eicken, David E. Culler, Seth C. Goldstein, and Klaus E. Schauser. Active messages: a mechanism for integrated communication and computation. In *Proceedings of the 19th Annual International Symposium on Computer Architecture*, pages 256–267, May 1992.
- [108] Wolf-Dietrich Weber and Anoop Gupta. Exploring the benefits of multiple hardware contexts in a multiprocessor architecture: Preliminary results. In *Proceedings of the 16th Annual International Symposium on Computer Architecture*, pages 273–280, June 1989.
- [109] N. Wilhelm. An anomaly in disk scheduling. *Communications of the ACM*, 9(1), January 1976.
- [110] Hans Zima and Barbara Chapman. *Supercompilers for Parallel and Vector Computers*. ACM Press, 1990.