

System Support for Integrated Desktop Video Conferencing

Amy Pearl

Conferencing and Collaboration (COCO) / Advanced Multimedia Platforms (AMP)

SMLI TR-92-4

December 1992

Abstract:

Desktop computers are increasingly used as communications devices. Advances in digital media are making the integration of video into desktop computers practical, both technically and economically. At the convergence of these technologies is computer-integrated desktop video conferencing. This paper discusses the requirements of integrated desktop video conferencing on a networked set of multimedia-capable workstations. Among these requirements are the following:

Media-intensive parts of applications should be distributed; a multimedia software platform should provide support for this.

Audio and video conferencing require network transparent location and reference of people, media devices, and conferences. The name and remote access reference for a conference must be exportable to client applications.

All group support applications that provide remote access require security services. True in any network application, this is more important with live communication streams, such as audio and video.

Low latency of an audio connection is more important than synchronization of audio with other timecritical communication, such as video conferencing and user gestures in shared interactive applications.

To efficiently support multi-person conferences, multicast networking protocols are essential.

A research prototype multimedia platform was evaluated, based on these requirements. This paper presents the lessons learned about system requirements for video conferencing that are not obviously required for single-user multimedia. It also discusses the motivation and requirements for extending the platform to support shared applications not previously considered to have constraints related to time.

 **Sun Microsystems**
Laboratories, Inc.

A Sun Microsystems, Inc. Business

M/S 29-01
2550 Garcia Avenue
Mountain View, CA 94043

email address:
amy.pearl@eng.sun.com

System Support for Integrated Desktop Video Conferencing

Amy Pearl

INTRODUCTION

In the last fifteen years, the ability of computers to capture, store, and manipulate information has been augmented by the ability to transmit that information to other computers. As computers become increasingly connected by various networks, they are increasingly used as communication tools. Much of that communication is in the form of forwarded and stored data such as electronic mail and bulletin boards [Byte 1985]. These tools support asynchronous communication between people, allowing communication to occur even if the recipient is absent. However, such tools have not replaced the need for interactive, or synchronous, communication, even in computers. For example, people have found simple text-based interactive “talk” programs useful for as long as computers have supported multiple users. Our research interest is in the new possibilities for interactive collaboration between remotely located workers who use networked workstations equipped with multimedia. One of the prototypes we have developed is a digital, integrated video conference application, called *videoconf*.

Video conferencing is a technology that has been emerging over the past thirty years, motivated by how much of our face-to-face communication is visual, or non-verbal. For example, there have been efforts to augment telephones to include transmission of visual images. Recently the Computer Supported Cooperative Work (CSCW) community has been exploring the use of computer controlled *analog* audio and video transmission to support group work [Root, 1988; Olson and Bly, 1991; Vin et. al., 1991]. Recent advances in media compression technology along with shrinking component size and cost are making digital video feasible. Unlike analog video, digital video can take advantage of the growing number of digital communication networks, including phone (e.g., ISDN) and institutional networks (e.g., LANs and WANs). It is also possible for programs to manipulate digital video, the effects of which are just now being explored [Liebhold and Hoffert 1991]. An example of this is the morphing of video images.

In Sun Microsystems Laboratories Inc. (SMLI), our conferencing and collaboration project is exploring issues in computer-assisted collaboration for geographically distributed small groups. *Videoconf* is part of a set of workstation tools built for user studies. The video conference application is built on the Advanced Multimedia

Platform (AMP) [Calnan 1991], also developed in SMLI. AMP was designed to provide system support to network multimedia applications. AMP defines multimedia as the combination of standard data types (text, graphics, images, etc.) with data types that have timeliness constraints (or *timecritical* data). It provides applications with support for managing the resources required for multimedia, including hardware devices. AMP uses a video board to capture, digitize, compress and display video in windows on the workstation. Our goals for developing the video conference application were to i) determine the feasibility of integrating digital audio/video conferencing into a standard networked UNIX desktop workstation; ii) to exercise the AMP multimedia platform; and iii) evaluate how much system support AMP provided for the development of multiuser, collaborative applications.

This paper describes what was learned about the requirements of video conferencing integrated into a networked workstation. The main body of this paper describes the system support that applications require to name, protect, share and optimize the resources needed for video conferencing. The paper covers features that were implemented and found useful as well as requirements discovered while testing the prototype. Then it briefly describes our application architecture in light of past CSCW literature. In the final section on future work, the paper describes features we expect to prototype and the further system capabilities required to support them. This includes an important addition to the class of timecritical data: general user actions in multiuser applications.

REQUIREMENTS

Most of the difficulty in a video conference application is in handling aspects of the network: finding conferees and the necessary resources (such as video cameras and video displays), providing access control and security, sharing resources, and sharing network bandwidth fairly.

Resource Naming

One of the first problems a distributed environment must address is how applications locate and reference services or resources. For video conferencing, there are three things we need to be able to find and reference: people, devices and conferences.

When users start a conference they think in terms of the people to whom they want to talk, or the name of the meeting they want to join. Applications need system support to map the names of people (“Connect Amy’s camera to John’s and Dave’s video displays.”) and conferences (“Add me to the SMLI staff meeting.”) to the resources and attributes associated with them. AMP associates multimedia resources,

such as video and audio devices, with machine names, and provides references to those devices. In order to allow users to refer to people or conferences by name, an application needs to map those names to the relevant machines.



Figure 1. Identifying people prior to initiating a conference.

In part because of this limitation, *videoconf* provides a phone-like invocation model where users need to know the name of each person with whom they wish to confer (Figure 1). However, users have found *videoconf*'s phone call model very limiting. Individual applications can provide their own mapping of people and conference names to the appropriate machines, but this support should be provided by a collaboration platform.

Users want the ability to join a conference in progress (usually referred to as late joiners). For example, if two people are conferencing and a third contacts one of them, they may want to add the caller to the existing conference. Users of *videoconf* cannot do this for two reasons: AMP provides no support for finding a conference, and AMP references to devices cannot be shared between applications. Truly general purpose platforms for multimedia and multiuser applications should provide support for these capabilities. At a lower level, this means that the network protocol must support adding and deleting connections dynamically.

Security

Once there are cameras and microphones on individuals' desks that are connected to a network that allows remote access of these resources, it becomes possible to eavesdrop on conferences and monitor individuals. There are two areas of security: security of access, and security of transmitted data. There are two reasons to provide security mechanisms at the platform level:

- Security is a service that many multimedia applications require, especially those that provide group support, and shouldn't have to be replicated by each of them.
- It is critical that these mechanisms are robust and that users of these systems be able to trust the security of such a system. This requires that security implementation be embedded at a level out of reach of the end user.

Access

Users minimally want to know if someone wants or attempts to view or listen to them. They may also want to restrict access to their local multimedia resources in various ways. This is an issue in any distributed multimedia environment that has multimedia devices located on desktops that can be accessed remotely. Because security was not a focus of this research, AMP currently provides minimal security. Anytime a video camera is initially accessed on a machine, AMP audibly notifies the user on that machine and prints a status message in the console window (the global workstation status window). This mechanism does nothing to restrict access, relying on users' awareness to mediate security.

Videoconf provides additional security by imposing a more restrictive access policy, symmetry of functionality (if I can see/hear you, you can see/hear me) [Borning and Travers 1991]. While this meets the requirement of having a simple model and is simple to implement, it is not a very scalable, flexible or practical model.

In our experience with a system that provides very little security control, we saw that much of access control can be socially mediated. This may in part be true because our target users are those who work closely together. We would like to prototype communication applications with a less restrictive policy than symmetry of functionality. For example, many more of our workstations are equipped with audio than with video. We would like to be able to impose symmetry on audio conferencing only, and allow video to be used, if available. In such a case, the access policy may be either socially mediated, or the application may impose a policy of symmetry of security (I am unable to see you unless I have permission to see you *and you have permis-*

sion to see me). This would require that the system include the concept of user access permissions for each resource.

In the example of symmetry of permissions, the access control was on the basis of each resource. We may also want conferences to have permissions. In the example of locating and finding a *conference* (“Add me to the staff meeting.”), the policy may be that conferences that have a *privacy* attribute set have a more restrictive admission procedure. In this case, users may have to contact and obtain permission from a conference leader first, or no one may be allowed to join who is not already on the membership list.

Data

The previous discussion addresses security of access. Users who transmit sensitive video data may want to prevent some people from monitoring it. Though this is an issue that also affects single-user multimedia applications, AMP does not address it. However, it is worth noting the fact that digital video opens up a wide array of data protection possibilities. For example, both analog and digital images can be altered, but it may be easier, through standard data integrity techniques such as checksums, to determine the authenticity of a digital image. We can protect digital images through powerful digital encryption techniques such as DES. However, while we must be concerned with issues of individual privacy when we have new ways to invade it, we should also consider just how private, or not, many of our current communication transmission mechanisms are. Conventional paper mail can be compromised in many ways. Electronic mail is often notoriously lax in its security. Telephone communication over lines can be tapped, and wireless communications are even more vulnerable. Many of our uses of communication and media rely on security mechanisms on which we have *socially* agreed, not purely technical ones.

No security system is invulnerable; the most important design goal is that the security model be one well understood by programmers and end users. The security requirements of collaborative applications vary widely. When applications are geared toward sharing big collections of information among large, loosely-connected groups of people, the security requirements are likely to be much greater than for sharing targeted collections of data among a closely connected work group. Security mechanisms should be flexible and sophisticated to accommodate a range of needs.

Shared Resources

Once applications have located their resources and the system has verified that they may access them, multiple clients may potentially share the same resources. Sharing resources is not unique to collaborative applications, but is inherently part of them.

There are many ways resources may be shared, and there are different requirements for each. Some resources are shared by partitioning them, so that users are protected from each other. For example, our digital video board supports up to four independent video displays. Other resources are shared outright, and the actions of one user may affect others. For example, a user may pause the transmission of his or her video camera for privacy during a conference; all display clients of that camera are then effectively halted.

Our video conference application requires three kinds of sharing: multiple receivers must share the data from a source, the display of multiple video images share a single frame buffer, and multiple users share control of video streams. First, transmitting each video camera output to multiple users' displays: each conferee's camera must transmit its data to a display on each conferee's machine. The displays share the camera datastream either by getting their own copy of the stream or by monitoring a single broadcasting broadcast stream (see discussion on optimization of video data transmission). AMP handles these connections and the choice of transmission.

Second, displaying video streams from multiple cameras on each user's workstation: the video board on each conferee's workstation must display the multiple streams it receives (Figure 2). Again, this is handled by AMP, since having multiple video displays on a user's workstation is not limited to collaborative applications.

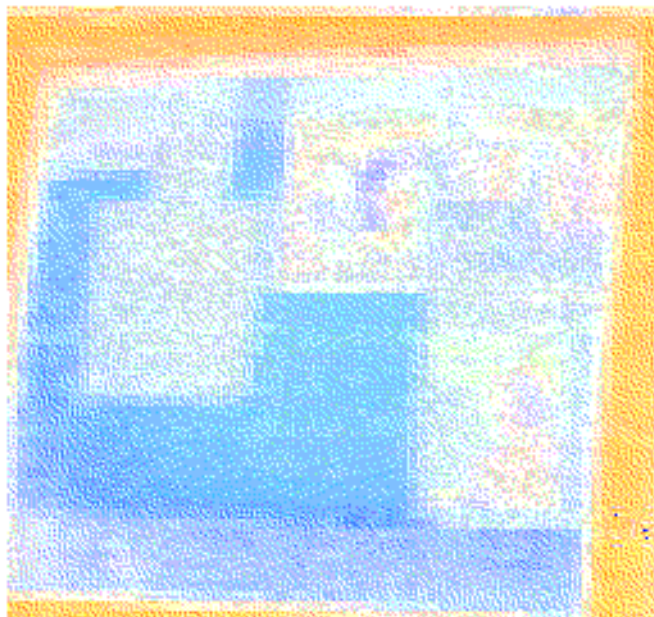


Figure 2. A three-site conference in progress.

Finally, allowing multiple users to share control of each video stream: in our video conference application, each of the users receiving the video data is able to control its behavior by pausing or resuming it. This can affect the view of others receiving that stream. Currently AMP provides no mechanisms to negotiate sharing control of resources. As a result, that capability is only provided in a very limited way by *videoconf*. Those resources may be shared by multiple applications that have no knowledge of each other. Any platform that manages shared resources must provide applications the ability to negotiate control of those resources. Because the AMP platform does not yet do so, *videoconf* does not provide users with the ability to manipulate many shared video characteristics such as frame rate and image resolution.

Users of a desktop video conferencing program don't really share data in the same sense as users of a multiuser editor. The data that video conference users share is only the multiplexing of the video sources to all the relevant displays. Unlike a shared editor, a video conference does not contain persistent data that is manipulated in different ways by the various users who receive it.

Performance Characteristics

Digital video conferencing has very specific performance requirements. This is due to the need for interactivity, the need for multiple video displays on each workstation, and the large volume of data in a digital video stream. People perceive audio delay greater than 200 milliseconds. Our video board allows conferencing between up to five people at a time. Like several existing digital video standards, the compression scheme AMP uses generates over one Mbits/second for each video datastream at full motion video rates.

Prior to the development of our prototype system, we conducted a study of dedicated video conference rooms at Sun between the east and west coasts of the United States. The vendor-supplied systems enforced audio and video synchronization. Users found that the significant transmission delay (over 500 milliseconds) interfered with natural verbal communication. As a result, users frequently switched to using the telephone for an audio channel. This was because the absence of delay made conversation more natural and productive, even though the audio and video were then out of synch. Our multimedia platform was designed to support media stream synchronization, which is achieved in part by liberal use of buffering that introduces delay. Our video conference room study showed that this approach is antithetical to the needs of low-latency conferencing. AMP was implemented to take this into consideration. Because a multimedia platform hides details such as buffer size from applications, it must carefully consider the requirements of *all* multimedia appli-

cations. The importance of low audio delay may also imply the need for some audio data to be treated at a higher priority than other timecritical data.

The volume of data is an important consideration for both data storage and transmission. Generally, the higher the quality of compression technique used, the more computation required to achieve the results. For single user multimedia applications it makes sense to use whatever computational resources are available for decompression of that user's datastreams. In conferencing, however, we want to display multiple streams at once. In addition, the number of streams is variable. This means that, unless the system can dynamically scale the complexity of the decode algorithm appropriately, it is desirable to use a less computationally-intensive decode algorithm, even at the cost of lower quality or higher data rate. The current AMP platform provides a very simple algorithm that does not result in a very great compression ratio (about 4x). It also requires relatively little computation. Our system had usable results with this algorithm, and it looks likely that as better compression techniques become available, either image quality will improve, or the data rate will get lower, or both.

The goals of integrating video into a desktop collaboration environment must include sensitivity to the use of the shared network resources. A single desktop video datastream will often have multiple destinations. Examples are three-way conferencing and broadcasting a lecture. There are three ways to have multiple destinations receive the data: point-to-point addressing, broadcast, and multicast. Point-to-point addressing requires sending each of the destinations a copy of the data. Replicating video data for each destination adds significant load to the shared network bandwidth. A broadcast mechanism sends a single stream of the data out on the network, and all machines on that network receive it. Broadcast has two disadvantages: first, by convention, broadcast messages are limited to a local network in order to prevent wide area network overload; second, every machine on the network receives the video data, whether it is interested in it or not. This puts significant extra load on processors not even involved with the conference. Multicast, or group addressing, sends a single stream out on the network, but only machines with users interested in the data can receive it [Deering 1990]. AMP uses multicast connections when possible, minimizing applications' impact on the network. While multicast transport is not explicitly a requirement of interactive video, network performance optimization is.

APPLICATION ARCHITECTURE

There are two general approaches to designing multiuser applications: a centralized approach in which a single copy of the application is run on one machine, and is explicitly or implicitly connected to user-interface code on each participant's

machine; and a distributed approach in which each user interacts with a separate copy of the application, and some mechanism is used to keep all copies of the application synchronized. The advantages and disadvantages of both schemes have been well documented for the case of applications employing text and graphics [Lantz 1986]. In applications that make heavy use of audio, video, or high-resolution still images, performance considerations will often dictate that at least the media-intensive parts of the application should be distributed.

In our case, one of the most useful functions provided by AMP was the management of distributed multimedia. Applications can use AMP to allocate resources and hook them up in the desired way. This took care of distributing the flow of media. We then had to decide whether to make the video conference application itself centralized or decentralized. Beyond having AMP and the network window system allocate and connect resources, the application's primary function is mediating user requests to pause, resume, and exit. Because of the CSCW literature and the fact that we were building a conferencing application, we expected that it would be preferable to write *videoconf* as a distributed application. However, because the application was a prototype, it was expedient to write it as a centralized application. It was much simpler to write because all the communication was within one process. Much to our surprise, as we analyzed what functionality was *not* provided by our centralized implementation, the only one we established was reliability. Reliability in the face of machine failure, though a classic advantage of distributed architectures, is not a major issue for the small group collaboration addressed by our application.

It is worth noting that, while the sophistication of distributed computer systems is increasing, the number of applications taking explicit advantage of this new sophistication is not [Rodden and Blair 1991]. Our experience with AMP is that finally the environment is becoming rich enough to make distributed functionality easier. This is good news for collaborative applications.

FUTURE WORK

Extending the Platform for Conferencing Support

The application-specific data that is exchanged in a multiuser collaborative application is also timecritical, and is subject to exactly the same synchronization problems as the audio and video. Consider the example of an application that employs audio conferencing and provides a shared telepointer. A user who says, "Look at the picture I'm pointing to," wants some assurance that collaborators on remote machines receive the utterance and the pointer update at approximately the same time. This

observation suggests that the services required of multimedia by CSCW environments are not simply audio and video conferencing. Application-specific data must also be delivered in the same timely manner as the media.

Like most general purpose computer operating systems and networks, ours do not currently recognize data as having timecritical requirements. In an effort to minimize its impact on other network traffic and other processes on the workstation, like other current multimedia efforts, AMP makes three assumptions about the timecritical characteristics of multimedia: data is sampled at regular intervals; individual data samples decay quickly (if one arrives late it rapidly loses relevance); and synchronizing multiple streams of data (e.g., audio and video) is a priority.

These assumptions drive the design and implementation of multimedia toolkit support. However, these assumptions are not true for all timecritical domains.

		Interval	
		Predictable	Unpredictable
Delivery	Guaranteed	Traditional real-time events	e.g., Shared button events
	Not Guaranteed	e.g., Full-time digital	e.g., Shared drawing

Figure 3. Characterizing timecritical data.

In Figure 3, we characterize timecritical data samples on two axes: how regularly it is sampled and its need for guaranteed delivery. The top left quadrant contains traditional real-time events. These kinds of events, for example on a factory floor, occur at predictable intervals, and the servicing of those events must be guaranteed. Samples from a video compression algorithm that captured and transmitted a complete data sample thirty times every second would be in the “predictable interval, non-guaranteed” quadrant. That data requires regular sampling, delivery within the sampling interval, and each sample is independent of the others. Data samples that don’t correctly and completely arrive before the next interval can be discarded; they will be replaced by later samples with more current information. Some compression algorithms send full base frames at less frequent intervals, supplemented with frames containing just the difference information in the intervening intervals. Because the base frames require reliable deliv-

ery, and the difference frames do not, such algorithms are a hybrid on the full/incremental update axis. AMP supports exactly this class of events.

Events in multiuser applications occur at unpredictable intervals. Those events need to be shared in a timely manner, and some require guaranteed delivery, while others do not. In general, keyboard events and mouse events (such as mouse up, mouse down) must be reliably delivered, even if some delay is involved (top right quadrant). Events generated by drawing continuously using a mouse or stylus do not require guaranteed delivery. Delivering fewer events results only in a coarser representation, not an inaccurate one. This is the kind of data generated by shared drawing. A platform for collaboration must support this class of timecritical events.

Once we have a platform that supports a wider range of multiuser, collaborative applications, we can expand our concept of a conference on the workstation beyond that of a video conference. A conference can be an object that contains a set of applications, including a video conference application. That set can be dynamic, and actions in the different applications could now be synchronized, or be subject to shared attributes. We are interested in exploring richer models of conference naming, characterizing and browsing. We want to prototype other models of security, and we want to allow users to control other parameters during a conference, such as security attributes, recording function, video frame rate and image size and resolution. We want to be able to add late joiners to a conference.

SUMMARY

Our experience showed that it was feasible to build a video conferencing prototype on a standard UNIX workstation. The software platform on which we built the application included a platform to support networked multimedia applications and provided many of the services required for conferencing. It was relatively easy to build a simple video conference application for up to four conferees, running on a standard network.

While our distributed multimedia platform provided much of the system support needed to accomplish our goals, there are additional requirements needed for multiuser multimedia applications. For single-user applications the platform must provide a mechanism to name network entities such as devices. For multiuser applications such as video conferencing the platform must also provide ways to address people and conference. Single-user applications need to be able to associate attributes with their entities, e.g., access permissions and whether the data needs to be secure. Single-user multimedia applications need the platform to provide the resources for

multiple displays on each workstation, but multiuser applications need the platform to also provide the ability to deliver a source, such as the data from a camera, to multiple destinations. Single-user and multiuser applications require platform support to be “good network citizens.” Both can benefit from data compression, but multiuser applications also require intelligent connection management.

Multiuser applications need the platform to provide some services not required by single-user multimedia applications. It must provide multiuser applications with support for negotiating sharing of control of shared devices. Multiuser applications may have strict timeliness requirements for some datastreams such as interactive audio. A multimedia platform supporting multiuser applications should also be able to support the application-specific data of these applications in a timecritical manner, just like video and audio.

REFERENCES

- Borning, A. and M. Travers, “Two Approaches to Casual Interaction Over Computer and Video Networks,” *SIGCHI 1991 Proceedings*, pp. 13-19.
- Byte Magazine*, December 1985 issue on Computer Conferencing, Volume 10, Number 13.
- Calnan, Roger S., “The Advanced Multimedia Platforms Project,” *2nd International Workshop on Network and Operating System Support for Digital Audio and Video*.
- Casner, S., K. Seo, W. Edmond, and C. Topolcic, “N-Way Conferencing with Packet Video,” Technical Report ISI/RS-90-252, USC/Information Sciences Institute, Marina del Rey, CA, 1990.
- Deering, Stephen E. and David R. Cheriton, “Multicast Routing in Datagram Internets and Extended LANs,” *ACM Transactions of Computer Systems*, Volume 8, Number 2, May 1990, pp. 85-110.
- Lantz, K. A., “An Experiment in Integrated Multimedia Conferencing,” *Proceedings of the ACM Conference on Computer Supported Cooperative Work*, ACM, New York, NY, 1986, pp. 267-275.
- Liebhold, Michael and Eric M. Hoffert, “Toward an Open Environment for Digital Video,” *Communications of the ACM*, Volume 34, Number 4, April 1991, pp. 103-112.
- Olson, Margrethe H. and Sara A. Bly, “The Portland Experience: A Report on a Distributed Research Group,” *International Journal of Man-Machine Systems*, Volume 34, Number 2, February 1991, pp. 211-228. Reprinted: *Computer-supported Cooperative Work and Groupware*, Saul Greenberg (Ed.), London: Academic Press, 1991, pp. 81-98.

Rodden, Tom and Gordon Blair, "CSCW and Distributed Systems: The Problem of Control," *Proceedings of the Second European Conference on Computer-Supported Cooperative Work - ECSCW '91*, Kluwer Academic Publishers, Bannan Robinson and Schmidt (editors), 1991, pp. 49-64.

Root, Robert W., "Design of a Multi-Media Vehicle for Social Browsing," *Proceedings of the Conference on Computer-Supported Cooperative Work*, Portland, OR, September 1988, pp. 25-38.

Vin, Harrick M., Polle T. Zellweger, Daniel C. Swinehart, and P. Venkat Rangan, "Multimedia Conferencing in the Etherphone Environment," *IEEE Computer*, Volume 24, Number 10, October 1991, pp. 69-79.

Acknowledgments

First and foremost, I must acknowledge the contributions of my two COCO project collaborators, David Gedye and John Tang. Their insight and contributions are innumerable and always invaluable. I am also grateful for the contributions, both in thought and practice, of our colleagues on the AMP team, led by Greg McLaughlin. In addition, I thank Roger Calnan, Trevor Morris, Alan Ruberg, Rab Hagy, and Brian Raymor for guidance and comments on this paper.

© Copyright 1992 Sun Microsystems, Inc. The SMLI Technical Report Series is published by Sun Microsystems Laboratories, Inc.
Printed in U.S.A.

Unlimited copying without fee is permitted provided that the copies are not made nor distributed for direct commercial advantage, and credit to the source is given. Otherwise, no part of this work covered by copyright hereon may be reproduced in any form or by any means graphic, electronic, or mechanical, including photocopying, recording, taping, or storage in an information retrieval system, without the prior written permission of the copyright owner.

TRADEMARKS

Sun, Sun Microsystems, and the Sun logo are trademarks or registered trademarks of Sun Microsystems, Inc. UNIX and OPEN LOOK are registered trademarks of UNIX System Laboratories, Inc. All SPARC trademarks, including the SCD Compliant Logo, are trademarks or registered trademarks of SPARC International, Inc. SPARCstation, SPARCserver, SPARCengine, SPARCworks, and SPARCcompiler are licensed exclusively to Sun Microsystems, Inc. All other product names mentioned herein are the trademarks of their respective owners.