14

Performance Trade-offs for a Multimedia Distributed Application

Kurt Maly, C.M. Overstreet, Hussein Abdel-Wahab, A.K. Gupta, Muthu Kumar and Rahul Srivastava

Old Dominion University
Department of Computer Science
Norfolk, Virginia 23529, U.S.A

Phone: 804-683-3915, Fax: 804-683-4900
E-Mail: (maly, cmo, wahab, ajay, rethi_m, sriva_r)@cs.odu.edu

Abstract

We have developed an Interactive Remote Instruction (IRI) system to support university-level instruction in a networked environment. The IRI system creates a virtual classroom where all participants have similar experiences independent of their location (separated by up to order of 100 km). Participants interact with audio, video, and collaborative tool operation using a multimedia workstation. We describe an IRI testbed consisting of Sun Sparcstations 5 running Solaris 2.4 implemented over a local area network. We discuss the performance trade-offs and the scalability of the IRI system by analyzing results obtained during the execution of a test suite. The most important conclusion of this work is that, though some improvements are required, we are able to achieve an acceptable performance within the goals we had established for IRI-required functionalities using moderately priced hardware. Feasibility means that response time for user commands is no more than twice in the worst case (Mosaic running on a single workstation by itself versus Mosaic running collaboratively on the testbed with multiple video streams and audio added). Results from this experimentation are being used to design dynamic resource management scheme which will regulate the demand on the IRI system resources to improve user perception of the quality of the virtual classroom.

Keywords

Networking, multimedia, remote instruction, performance evaluation

1 INTRODUCTION

While several multimedia teleconferencing systems are available which support collaborative work, none have the ability to support more than a few simultaneous users and none automate the handling of video and audio streams. We present a performance study on the tradeoffs in functionality and costs and user perceived quality using a system which we are building to support Interactive Remote Instruction (IRI). As we continue development of this IRI prototype, we are simultaneously using it to evaluate our current user-interface design, the performance achievable with our current software and hardware architectures, and the more subtle performance issues required to provide acceptable services. Our goal is to demonstrate a system which can be used to show the feasibility of supporting college-level education across spatial boundaries in a manner largely transparent to the students and faculty involved.

After a brief discussion of the current IRI architecture and functionality, the core of this paper is a presentation of performance results achieved to date. The purpose of this study is two-fold: to show that with current technology the use of multimedia for remote instruction is viable; second to obtain performance data which can guide us in developing a resource management subsystem to use for allocating shares of system resources to the critical functions affecting user response time.

Section 2 gives an overview of the functionality of the IRI system, section 3 describes our experimental testbed, analysis methodology and performance results. Section 4 presents our conclusions from this work and describes planned future activities.

2 ARCHITECTURE AND FUNCTIONALITY

IRI can facilitate both teacher/student and student/student interactions through two-way audio and video, tool sharing, electronic mail, and electronic conferencing capabilities [MAL94, MAL94].

In conferencing applications the network must provide a level of service give participants a "natural feel" about interactions. The time between a speaker moving or speaking and that motion or sound perceived by other participants is crucial. Emerging video compression standards indicate that one-way end-to-end delay should be less than 150 milliseconds and round-trip delay less than 300 milliseconds [RAD94] for conversational modes. The delay between audio and video transmission has to be limited to provide lip-synchronization of the audio and video signals.

IRI transparently integrates teaching tools with which instructors can communicate course content. While an energetic instructor today might use slides, overheads, paper handouts, photographs, graphics, video, and audio, each mode requires separate tools for delivery. IRI consolidates these resources at one workstation and places them at the instructor's fingertips.

2.1 The IRI Software Architecture

The software architecture for IRI is built, whenever possible, on existing software components, many in the public domain. Several components are derived from XTV, a general purpose collaborative system developed jointly by Old Dominion University and the University of North Carolina at Chapel-Hill [ABD90, ABD94] collaborative use of tools. All traffic generated by an X tool is transmitted reliablely using traditional TCP. A resource_management module is the arbiter on tradeoffs among conflicting demands for bandwidth and will, if necessary, decrease the bandwidth on a video stream to give more bandwidth to a reliable

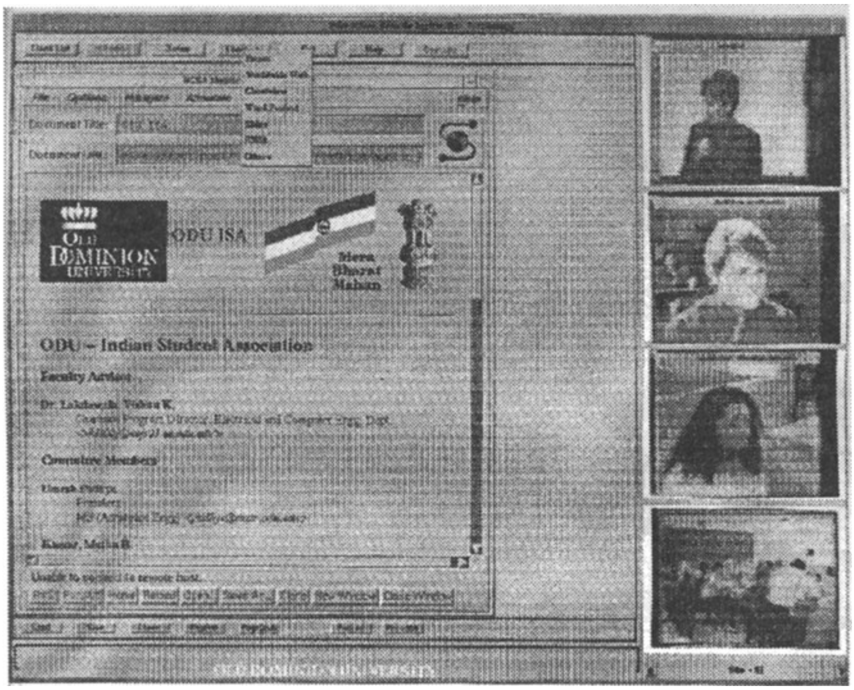

Figure 1: User Interface with Mosaic running

connection.

We believe that a successful IRI system must have the following characteristics:

• The *use of the system should be comfortable to all users*, both instructors and students. It should support modes of instruction that faculty believe are most effective for their subject matter and should provide students useful choices.

• *Both faculty and students should see personal benefit from its use*; it should not be perceived as a "cheap second choice" motivated by cost-cutting.

• *The implementation should be flexible and scalable* due to rapid evolution of supporting technology. The capabilities of the hardware we use for the initial experimental system will be very different from that of the hardware we would choose in year five. But the system we build must still function effectively on that new hardware.

• *It should support multiple modes of interaction.* For example, students must be able to use the system for team projects with participants distributed across the state. Students must be able to discuss problems with instructors on an individual level.

IRI Functionality Description: We have demonstrated a prototype of the IRI system called 'tv-learn.' Though with limited capabilities (e.g., all facilities are within one building, TV signals use coaxial cable between workstations, data are transferred with a standard

10Mb/second Ethernet), it allowed us to illustrate many of the basic system capabilities and key features of the user interface.

The screen image in Figure 1 illustrates many of IRI's features described in [MAL94, MAL94]. When an instructor is actively presenting, the largest window contains a full-motion video similar to instructional television. However, this figure depicts a state in which an instructor might have invoked WorldWideWeb to bring on-line resource materials directly into the classroom. When the large window is occupied by a tool, a small full motion video image of the instructor appears in the upper right corner as depicted in Figure 1. The two top small images on the right below that are those of students engaged in a discussion with the teacher. Any tool which the instructor chooses appears in the large window. This window is also a transparent pane which allows the student to make notes and, if desired, these notes can be made public (see buttons on bottom of the window - these are active when the note button has been pressed). The instructor can use almost any type of computer tool (word processor, spreadsheet, simulation with animated output, graphical drawing software), and any student in the class will see the tool running as if it were running on her own workstation. In addition, any student at any site can take control of the tool and use it while all other students see the results of that student's efforts, The bottom image on the right is that of one of the remote class rooms.

3 EXPERIMENTAL TESTBED

Our approach to building the IRI system over a wide area network (WAN) is to implement an alpha version of the IRI system over a local area network testbed and study its performance before shifting the implementation to a WAN. The image of the teacher is distributed through a coaxial cable instead of satellite transmission. In this section we report on our measurements and their analysis from that testbed.

3.1 Bandwidth Limitation

For an initial estimate of network requirements, we developed a bandwidth budget based on the traffic required to provide acceptable functionality. In [MAL95] we reported on the details of that model. To estimate network needs, we assume a worst case scenario where all transmissions originate at the hub and one of the remote sites. We identified network traffic types, protocols used, features causing traffic, sources of transmissions, destinations, estimated bandwidths, and a priority for each traffic type. The priority is to be used to adjust bandwidth so that, for example, images of remote classrooms might be updated less frequently if the net or the CPU are heavily loaded.

IP multicasting is used to transmit multimedia data to avoid the overhead of establishing and maintaining TCP connections among all machines. All class members join a specific multicast group and send and receive date from this group. Each class in the tv-learn system maps to a unique class D IP address and port number; all participants joining this class get this mapping dynamically. Thus, participants can join and leave the class at any time without going through the overhead of any kind of joining or leaving protocol.

Because of the nature of video – occasional loss of a packet normally has little impact on the perceived image at receive sites – these images are sent using unreliable multicasting to avoid the overhead of reliable multicasting. However, crucial data such as that between the X server and X clients uses a reliable transport protocol.

The current version of the tool engine establishes a point-to-point (TCP) connection between the teacher and each student's program. This ensures the reliability needed to maintain synchronicity for X applications. However, it implies that data have to be duplicated and sent on each of these connections separately. The consequences are two-fold: the time spent in sending data to every student individually and the network traffic generated. As expected, the number of bytes transferred is a linear function of the number of participants. To reduce this traffic from O(n) to O(1), we will use reliable multicasting. The reliable multicast protocol (RMP) proposed by Brian Whetten [WHE] has been shown to be efficient and robust and we intend to replace TCP with RMP.

3.2 Performance Analysis Methodology

The testbed consists of three Sparc 5 workstations running under Solaris 2.4 connected by Ethernet. A workstation has two Sun video cards, one for the display of the teacher's image and one for handling the image of the student at that workstation. These video cards perform compression on the card but decompression is handled by the workstation's CPU. Each workstation has a local disk and 16 MB of memory. The workstation serving the teacher is also the file server for the net with 32 MB of memory. At the user level all but the note-taking and replay user functions described in the previous section have been implemented. At the system level, the major module not implemented is the resource-management module. The alpha version has stripped all the GUI from the video, audio and collaborative tool operation and automated the user interaction as specified for the IRI system. Multicasting and synchronization of video and audio have been implemented; reliable multicasting and pipelining for tool collaboration is being implemented but not part of the experimentation described in this paper. We reported in an earlier paper [MAL95] on the linear increase in network traffic due to using TCP/IP for tool collaboration (our solution being the use of RMP [WHE]) and the need for pipelining data traffic between the tool server and the clients.

Hardware costs is a major consideration in developing the IRI system, thus we employed standard, inexpensive technology in setting up our testbed. Clearly, using technology we could have off-loaded decompression but it would have increased the cost considerably.

The key question we wanted to answer in our experimentation is: Can we operate a tool, such as Mosaic or ghostview, in a distributed manner over several workstations and add multiple video and audio streams without unduly increasing response time of the tool, degrading the video and audio reception, or crashing the system? We had set as our goal that response time should not be more than double in the worst case, audio should essentially remain the same and the video frame rate not be reduced by more than half. We were hoping that the experimentation would give us enough insight to enable us to design an intelligent resource management module which would detect changes in load on the CPU and the network traffic and regulate the demand on these resources by intelligently scheduling the tasks.

The key problem for this question was the relationship between the various functionalities and performance. To that end we designed a set of experiments to isolate the potential causes of performance degradation. To measure the effects we monitor CPU performance at two second intervals using mpstat and instrumented our code to monitor network traffic and frame rates at the same time intervals. In the figures that follow, CPU utilization is the sum of system and user time percentages, network traffic is in bits per second and frame rate in number of frames per second. We are not using routines such as netstat because we want to isolate the traffic generated by our system from the totality of traffic over the Ethernet. Response time is the elapsed wall-clock time from initiating a command until completion of that command. We report on the time to bring up a tool separately from the average of executing five typical commands for that tool.

Our test suite has three phases: a baseline of a single machine running a tool, machine(s) running a tool collaboratively, and machine(s) running the tool collaboratively and combinations of teacher and student images and voices. Individual tests were repeated and averaged, the number of times depending on the variation in the results. We selected tools according to their demands on CPU and network traffic generated: Mosaic, xv, ghostview, emacs, and xcalc. We provide figures in the result section for the worst (Mosaic) and best (xcalc) case and refer in the explanations to results from the other tool experiments.

Table 1 List of experimental operations

Mosaic	xcalc
Opening an URL	Log of a number
Flip back to prev. page	Square root of a number
Move Forward to next page	Sine of a number
Flip Back to prev page	Factorial of a number
Open a hyperlink	Inverse of a number

An experiment consisted of invoking a tool in tv-learn and after a 60 second delay executing the five actions of Table 1 in 30 second intervals. The html documents that were required for the operation of Mosaic were locally present. This was done to get a precise response time independent of network factors. Some minimal CPU activity was noted during the period of the end of an action and the beginning of the next action; these points are not plotted.

3.3 Experimental Results

3.3.1 Tool Performance

Comparisons of CPU load: Experiments were performed for several tools (emacs, an editor, ghostview, a postscript document viewer, and xv, an image handler, xcalc, and Mosaic). During tool use of a light-demand tool like xcalc, we see a series of CPU utilization spikes of under 20% for 3 to 5 seconds in response to each command operating the tool. Even for the worst case, although the CPU activity continues for a longer period, it is not necessarily in the high range. Worst case CPU utilization is close to 100% when using xcalc, but only for a few seconds in response to tool invocation. Mosaic was picked to illustrate the impact of high-demand tools.

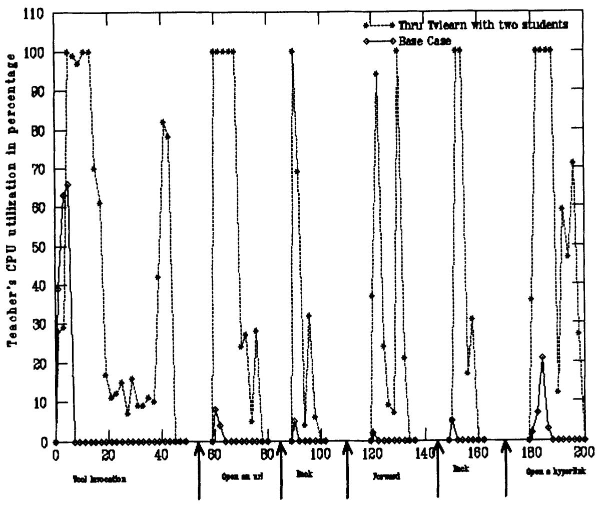

Figure 2: Comparison of CPU utilization for tool Mosaic (without video)

CPU utilization for the other Xclients mention fall within the envelope between xcalc and Mosaic; none have 100% CPU utilization in normal usage even for the worst case. With Mosaic, the worst case shows sporadic 100% CPU utilization but also show the feasibility of running all supporting tools in a workstation with a single processor. Figure 2 compares CPU utilization in response to a set of Mosaic commands for Mosaic running stand-alone (the base case) and under tv-learn with an instructor and two students participating. Under tv-learn, utilization would peak at 100% for as long as 10 seconds in response to these Mosaic commands.

As mentioned, during this experimentation video traffic accounted for most network traffic; even a large tool like Mosaic generated only around 50k bps, with relatively rare peaks of 150k bps. Video traffic is consistently around 650k bps. While the total number of bits transferred is directly proportional to the number of students, at any one instance, the bits transferred are not necessarily higher for two students than one student. This is due to the processing time involved for sending the packets to two students. The total number of bits transferred with two student participants (5,827,544) is more than twice that for one participant (2,711,752). We are now working to incorporate the reliable multicasting protocol to keep the network traffic independent of the number of students.

No significant amount of video and audio traffic was dropped by using UDP. The sum of all the data rates is well within the bandwidth budget for IRI using duplex T1 connections between sites (and Ethernet within a site) if we assume we can hold the rate for data traffic (collaborative tool operation) constant by using a reliable multicast protocol.

3.3.2 Video Performance

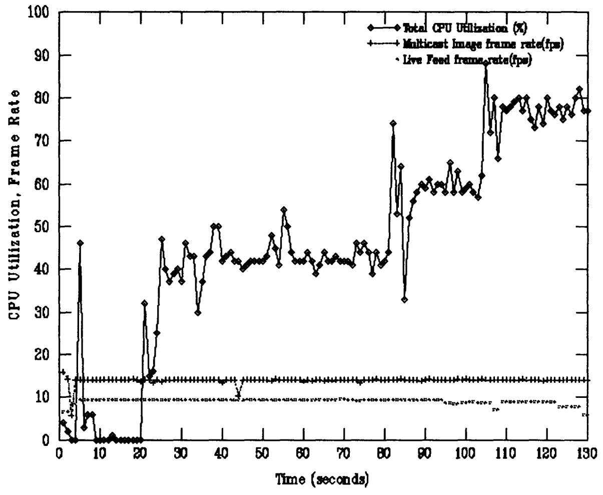

Figure 3: CPU load and Frame Rates for Instructor with 2 students, no tool

Baseline CPU load: Figure 3 represents the load on the CPU when running the baseline case of 'tv_learn' with the Instructor and 2 student participants. This scenario represents running the 'tv_learn' code, a live feed video capture and display of 2 student images, multicast from the other 2 student workstations, on the instructor's workstation.

Table 2 shows the times when each new action is initiated. Figure 3 shows corresponding jumps in the CPU load at these points. These CPU utilization levels are the baseline for subsequent discussions and figures showing loads for the operation of Mosaic and xcalc.

Table 2 Activity Log

time	activity
4s	tv_teach code started
20s	live feed capture started
80s	1 multicast image started
102s	2nd multicast image started

The frame rates for the live feed and the multicast image about 9 and 14 frames/sec respectively. The lower rate for the teacher's image is due to the size of the display; we expect both to increase with the next generation of video cards. We see that different activities (e.g., live feed, multicast image) do not produce a significant effect on other frame rates.

Comparisons of CPU loads for Mosaic and xcalc:

No significant CPU load variations result from xcalc operations after the activities described in Table 2 initiated the video and audio streams. In the case of Mosaic, in Figure

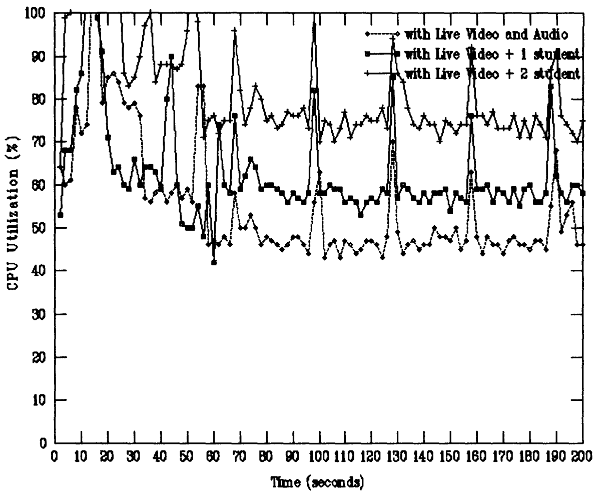

Figure 4: Comparison of CPU Utilization for tool Mosaic (with video)

2, we can clearly discern spurts of CPU activity corresponding to the actions performed. This shows that IRI running on this platform can handle multiple video and audio streams in conjunction with collaborative tool operation. Since all these additions are multicasting streams they are independent of the number of students participating in a class. Handling of the audio had no significant effect on either CPU load or network traffic but user perceived audio quality was degraded. This is due to the simple algorithm used to synchronize video and audio. Seeing that we have CPU and net resources available we will now improve the algorithm.

3.3.3 Response Time

From the user perspective, with the CPU utilization and network loads not directly observable, the response time to a command is the key performance metric. As our experiments were done on Xclients, we classify response time into two types. The invocation of the tool usually takes longer but is a one time action which can be overlapped with other activities by the teacher (or even be done in advance of class). The response time for subsequent operations of the tool is the prime concern.

In Figure 5 we have listed response times for all experiments ranging from the base case with no IRI system to the case of Mosaic running collaboratively and three video and an audio stream running concurrently. Tool invocation does indeed take up to 9 times more for the worst case (the solid bar). We expect to improve this through pipelining and multicasting the collaboration communication system. Thus, the processing time involved in distributing the packets to n students would be reduced to that of one student resulting in lowering the response time for bringing up a tool for the cases of more than 1 student. Typical operations,

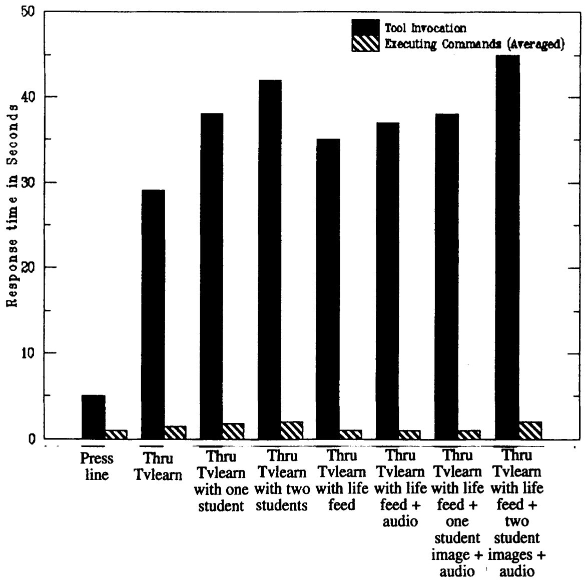

Figure 5: Response Time for Mosaic

however, take at most about twice the time of that of the base case. Commonly used tools like xcalc, emacs and xedit show a still better result: the response time of the worst case is close to that of the best case.

4 CONCLUSIONS

This paper reports on further results as we continue development of our Interactive Remote Instruction (IR) system. In [MAL94] we focused on implementation details; in [MAL95] on user-interface, ease-of-use issues. Here we report on performance issues and relate loads (CPU and network) to major functional areas, for example, support of audio/video and tool operation.

Our current version addresses audio/video synchronization only to a limited degre; this will be addressed in future versions of the software. In addition, we will implement a distributed resource management routine which will reallocate based on priority of the application/service. We will also incorporate RMP (reliable multcast protocol) to reduce network traffic when reliable transport is appropriate.

A major goal was to determine if we could obtain acceptable performance using standard workstation platforms when operating a tool such as Mosaic or ghostview in a distributed fashion across these workstations when audio/video functionality was also included. This prototype demonstrates that we can handle multiple video images (both instructor and student images) along with simultaneous distributed management of collaborative tools (such as Mosaic); existing low-cost technology can yield acceptable performance in a LAN environment, but with some aspects which need improvement. For example, the frame rate (number of frames of video image presented per second) was limited by the video boards we used to between 9 and 14 frames per second depending on application and image size. Use of better video boards will improve the frame rate and off-load the CPU by doing compression and decompression on the video board. The time required to initiation tools was high, through after initiation, performance was acceptable. We are studying the use of pipelining techniques for collaborative tools to decrease the time required to initiate a tool; this may also further improve response time on tool operation.

The most important conclusion of this work is that, though some improvements as outlined above are required, we are able to achieve an acceptable performance within the goals we had established for IRI-required functionalities using moderately price hardware based on current technologies.

References

[ABD90] H. Abdel-Wahab. Multiuser tools architecture for group collaboration in computer networks. *Journal of Computer Communications*, pages 165–169, 1990.

[ABD94] H. Abdel-Wahab and K. Jeffay. Issues, problems and solutions in sharing x clients on multiple displays. *Journal of Internet Practice and Experience*, Vol. 5, No. 1:1–15, March 1994.

[EGN93] M.W. Egan, M. Welch, B. Page, and J. Sebastian. Learner's perceptions of instructional delivery systems: conventional and television. *The American Journal of Distance Education*, 2(4):47–55, 1993.

[MAL95] K. Maly, H. Abdel-Wahab, C.M. Overstreet, Ajay Gupta, Muthu Kumar, and Rahul Srivatsava. Issues in scaling multimedia collaboration tools for remote instruction. *IEEE Symposium on Computers and Communications, Alexandria, Egypt, June 27-29, 1995*, Accepted for publication.

[MAL94] K. Maly, C. M. Overstreet, H. Abdel-Wahab, and A. K. Gupta. Melding television, networking, and computing for interactive remote instruction: Exploiting potential. *ED-MEDIA 94*, pages 367–372, 1994.

[MAL94] K. Maly and C.M. Overstreet. A new paradigm for distance learning: Interactive remote instruction. *13th World Computer Congress IFIP Congress '94*, pages 682–689, Hamburg, Germany, September 1994.

[RAD94] Radhika Roy. Networking constraints in multimedia conferencing and the role of atm networks. *AT&T Technical Journal*, pages 97–108, 1994.

[WHE] B. Whetten and Todd Montgomery. A high performance ordered multicast protocol. URL http://hopper.cs.wvu.edu/ mtodd/RMP.html.

Biographies

Kurt J. Maly received Ph.D. from the Courant Institute of Mathematical Sciences, New York University, New York, NY. He is Kaufman Professor and Chair of Computer Science at Old Dominion University, Norfolk, VA. His research interests include modeling and simulation, very high-performance networks protocols, reliability, interactive multimedia remote instruction, Internet resource access, and software maintenance. His research has been supported by DARPA, NSF, NASA, CIT, ARPA and the U.S. Navy among others. Dr. Maly is a member of the IEEE CS and ACM.

C. Michael Overstreet is an Associate Professor of Computer Science at Old Dominion University. He is currently chair of the Special Interest Group in Simulation (SIGSIM) of ACM. He received his Ph.D. from Virginia Polytechnic Institute and State University in 1982. His current research interests are in high performance networking, model specification and analysis, and static code analysis in support of software maintenance. He is currently a principal investigator in task funded by ARPA, ICASE at NASA Langley, and NSF. Dr. Overstreet is a member of ACM and IEEE CS.

Hussein Abdel-Wahab received the Ph.D. in 1976 from the University of Waterloo in Computer Communication Currently he is a full professor and the graduate program director of computer science at Old Dominion University. In addition he is an adjunct professor of computer science at the University of North Carolina at Chapel Hill. His main interests are collaborative engineering and desktop multimedia conferencing systems, real-time distributed information sharing, and mobile computing. His research has been supported by NSF, ONE, IBM, MCNC, MITRE among others. He is a senior member of IEEE CS and a member of ACM.

Ajay K. Gupta is the Director of Computer Resources in the Computer Science Department at Old Dominion University. He received his bachelors degree in Electronics Engineering from Bangalore University, Bangalore, India, and his masters in Computer Science from Old Dominion University, Norfolk, Virginia. His research interests include network performance evaluation, high speed networks, and interactive multimedia instruction. His research has been supported by NSF, NASA, and CIT. He is a member of ACM and IEEE.

Muthu Kumar received his BE from Bharathiar University in Coimbatore, India. He is currently a computer science graduate student at Old Dominion University.

Rahul Srivastava received his BE from Osmania University in Hyderabad, India. He is currently a computer science graduate student at Old Dominion University.