The Design of a Digital Amphitheater *

Allison Mankin, Ladan Gharai, Ron Riley, Maryann Perez Maher, Jaroslav Flidr
USC / Information Sciences Institute
Arlington VA

October 10, 2000

Abstract

In this paper, we present the design of what we have termed a Digital Amphitheater (DA). The DA is a network teleconferencing architecture and application that aims to assemble together remote participants into a virtual lecture hall or amphitheater. One of the main uses that we envision for the DA is that of hosting a technical conference with hundreds of remote participants. (In fact we plan to use the DA to host a program meeting with roughly 200 remote attendees.) Although the design of the DA makes use of existing multicast conferencing technologies, a new active service architecture was developed to meet the challenge of working with hundreds of simultaneous video streams. Low rate video streams are coalesced in real-time so that each participant receives only a few video streams, instead of a separate stream from each participant. The active service approach includes that DA users dynamically locate instances of video merge service. The basic design and architecture of the DA are presented, along with measurements pertaining to the thesis that the DA will allows commodity PCs to participate in hundred sender videoconferences.

1 Motivation

The concept of a Digital Amphitheater arose out of a desire to support large virtual conference meetings, in order to use the network to bring an effect of mass telepresence. In a virtual meeting the goal is to have every participant be a **visible** attendee, who can be seen speaking, bored or even sleeping in his/her chair. Therefore the default is for all participants to be permanent video senders.

This may seem quixotic as it implies a very large scale many-to- many multicast, and the state for this in routers has been increasingly viewed as onerous [6]. However, with the DA, the hundreds of individual video sources are unicast. A smaller number of nodes perform the multicasting, after a novel merge of the audience video images into rows. Multicast announce-listen is used to enable the individual senders to dynamically find the fewer multicasters. None of this is incompatible with networks that might have only source-specific multicast IP [5] if PIM-SSM [9] becomes the dominant Internet multicast routing protocol. The DA design is for a few-to-many multicast session.

This may seem to be like relay or server-based (H.323-like) conferencing, but the similarity is misleading. The DA approach is not primarily motivated by the routing issues. The active services are provided to enable all commodity PCs to participate in hundreds of participant conferences. This motivation is distinct from those of router-based multicast assists (GRA) or other active router approaches as well as from unicast relays, all of which exist to compensate

*This paper is based upon work supported by the Defense Advanced Research Projects Agency Information Technology Office. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of DARPA. The authors may be contacted at {mankin,maher,rriley,ladan,jflidr@east.isi.edu}.

for limitations or absence of multicast routing.

As designed, the amphitheater video merge services (AVMSs) in the DA will dynamically load-balance DA users among themselves. They can provide some network performance benefits in doing so, but their primary role is to alleviate the end-system processing.

Video teleconferencing among small groups of people (2-5) is prevalent, but large structured meetings, such as research or professional conferences, have never been tried and are considered infeasible. Among the reasons why large scale conferencing, on the order of hundreds of participants, is hard, the most basic problem is overcoming end-system limitations. The packet processing needed, both at the network and application layer, to handle hundreds of individual video streams overwhelms most PC end-systems today, in bus access, interrupt processing, packet handling and demultiplexing, decoding, and display processing.

There's also the issue of physical screen space. How do we effectively display so many video images? Practical user applications for displaying and managing a large remote audience do not yet exist. The common problem of network bandwidth limitations which impede or disrupt the quality of communication is not the primary one addressed by the DA design; the majority of the hundreds of senders are in the audience and little resolution is needed for their images: think of the experience of a large conference; one recognizes many people and their reactions by a small amount of visual data per person.

Many of the current teleconferencing tools, especially the research-oriented ones such as the popular UCL MBONE tools[12] have been designed with scaling properties in mind. Nevertheless their scaling approaches don't fully address the challenges mentioned above. We are taking an approach of developing an active network service[1] that relieves end-systems from the system overhead of a hundred or several hundreds of video flows.

2 The DA User Application

The DA user application is an integrated teleconferencing tool. We have designed a JAVA-based graphical user interface (GUI) and a C-language multimedia library. The application is tailored to users with little experience in teleconferencing, attempting to give the illusion of looking across the auditorium at the other participants in the conference. The most novel feature of the display is the arraying of rows of attendees into "seats". The speaker or a panel of speakers is automatically positioned at the top (or in a central location if the user is for instance using multiple displays). The "audience" section is a scrollable panel that can essentially accommodate any number of attendees. Figure 1 shows an example DA user's display. The DA user application will be user-configurable, so that the arrangement of areas can be specified to suit.

Creating the seated audience requires image processing both at send and merge time. At send time, the DA application does the needed processing to insert a uniform background and the back of an auditorium chair into the video stream behind its own sending user's image. An image processing algorithm has been designed to find the outline of the person in each frame, even as he/she moves. It is feasible because the audience member video uses a low frame rate (on the order of five fps), based on the intuition that one cannot watch the people across an auditorium extremely closely, so the video flow needs to be fresh enough not to annoy [11] the viewers but need not be updated at a full-motion rate.

The amphitheater video merge service (AVMS) takes the incoming images and tiles them into a long single row, which the user display will show as multiple, staggered and scaled rows. It is important to understand that the video merging here is application-specific (not a general transcoder or the like).

The AVMS is responsible not only for creating merged audience rows but for compiling an identity matrix for the rows. It identifies the arriving video uniquely by its source and RTP Synchronization Source (SSRC) Identifier [10]. It packages audience-identifying information from the sources' RTCP messages (user name mandatorily, along with

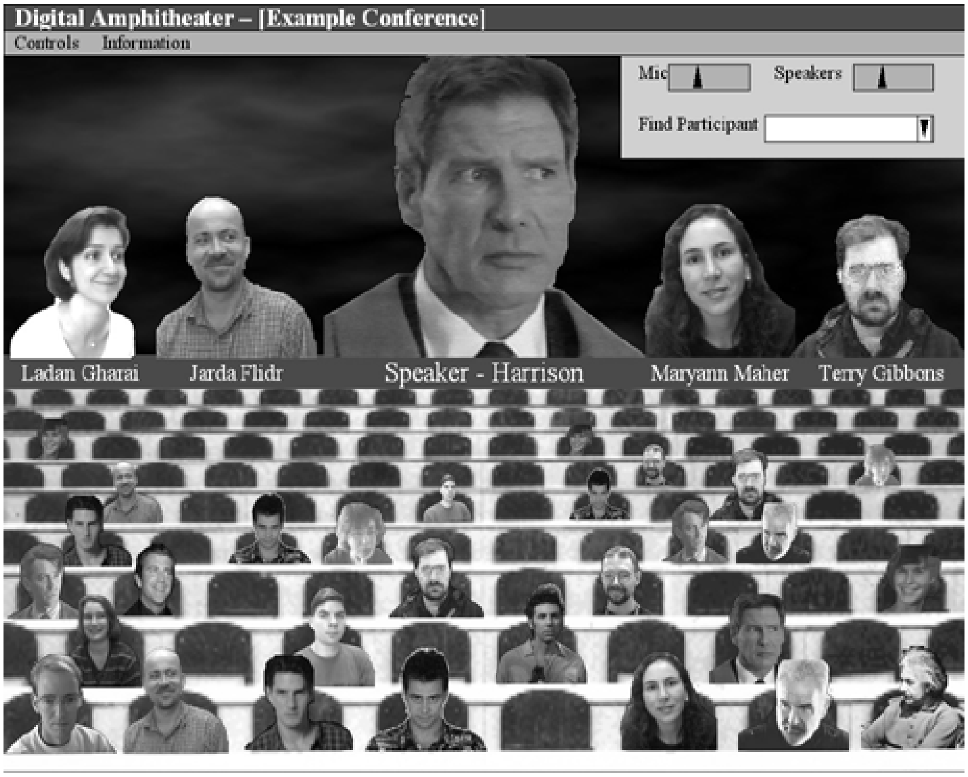

Figure 1: The Digital Amphitheatre Display

other information that may be collected for a particular DA meeting, such as project affiliation) in a data format that the DA receiver side can use to allow meeting enhancement; unlike in current physical meeting, in a DA meeting, an attendee can use a mouse click to learn who a fellow attendee is.

The AVMS also uses the unique per-source identity to maintain the integrity of rows, leaving an empty seat if someone "leaves". If we allowed shifting to fill the empty spaces, there would be a disturbing musical chairs effect. Our "seating" algorithm strives to visually smooth over such changes; new arrivals are at first placed in a trailing "reserved section"; members returning within some time window take back their original seat; eventually open seats are refilled with newer arrivals. The seating function is hierarchical: the receiving DA (or a recursively merging second level of AVMS) preserves the order of audience row sections from multiple AVMS's.

Figure 1 shows three types of video images: speaker, first-row, and audience, decreasing in resolution and frame rate. The audience level in the preliminary code versions uses 80x64 images at five frames per second[8] and uses a non-traditional codec, YUVCR. We chose this codec to minimize AVMS latency and support the merge algorithm and because it has no set image sizes. More detail is provided in in Section 6.

A feature of the DA user application is the user's choice of seating preference. Before starting video transmission, the user requests first row seating if desired (to be seen better, though, rather than to see better) and also indicates if he/she is capable of sending speaker-quality video. Receivers have markers in their audience directory that show who is willing to be in the front row, and once one receiver has chosen a sender, that sender is signaled to transmit front row quality rather than just audience quality, and he or she "changes seats" for the choosing receiver.

Along with the GUI, the DA user application includes audio and video codecs, RTP/RTCP engines, and the client side of the DA merge service protocol (AVMCP, see Section 4). Where appropriate, we are reusing rather than re-implementing, by borrowing internal parts of the UCL RAT, VIC, and SDR [12], with thanks.

3 The DA Architecture

The Digital Amphitheater requires coordinating and making tractable up to several hundred individual multimedia sources. The keys to a scalable solution are the amphitheater service discovery and merge functions. The section entitled Merge Service describes and shows a figure of the spatial compositing of the individual images into larger, aligned frames by the merge function. We have already described how resultant frames are processed by the receiving user application to create the amphitheater layout.

The video merging is performed by AVMS daemons conceived along the lines of the "active service" model of Amir and McCanne, where an active service is a service inside the network implemented at the application layer and specific to an application. A protocol among the AS daemons will allow them to form a hierarchy so that they can use several stages of processing and avoid overload of flows on any one node.

Figure 2 illustrates the distribution of video data from four DA session participants. Participants A1 and A2 belong to the same site network A, while participants B and C are in distinct other networks. The traffic from A1 and A2 is merged first within their site at AVMS-A and then again in an AVMS discovered by AVMS-A outside their site, also discovered by participants B and C, which do not have local AVMSs.

The DA makes use of the Session Description and Session Announcement protocols (SDP and SAP)[2] [3]. We define a new SDP attribute to flag that a session is one requiring AVMS. A DA user is a normal SAP listener that filters for DA sessions only, ignoring others. Once the user decides to join the DA session, the application begins an autodiscovery process described in the next section.

4 The Video Merge Service Protocol

The amphitheater video merge service protocol (AVMSP) is derived from the active service control protocol (ASCP) [1]. ASCP protocols are based on the announce-listen communication model that is the core of the light-weight session architecture. The ba-

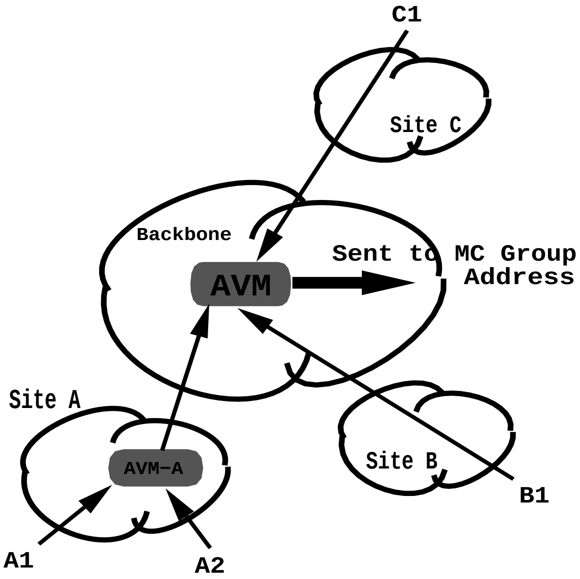

Figure 2: Merge Hierarchy

sic operation is as follows: A participant/client multicasts requests on a well-known AVMSP group with a limited scope (an inter-AVMS protocol handles reaching a distant AVMS, if that is needed). A listening AVM acts on these requests by creating a merge service instantiation and acknowledging the request with a reply. The reply is periodically announced on the AVMSP group so that other potential clients can automatically learn about the merge service instantiation. (Clients will also periodically refresh their desire for the service.)

AVMSs communicate with each other on their own multicast group. The inter-AVMS protocol allows them to autoconfigure into a merge hierachy. The AVMSs need to run a positioning algorithm which aims to position the least loaded servers at the top of the hierarchy.

5 Background Processing

In order to engender the feeling of an audience in an auditorium, the local background of video supplied by audience participants is removed and replaced with a generic conference chair. The generic and static background creates the illusion of a mass audience as well as compressing the audience video aggregate to some degree. This section describes the processing and algorithms currently used in the DA for this background replacement. We plan experiments with audiences and audience evaluation and expect this processing to evolve.

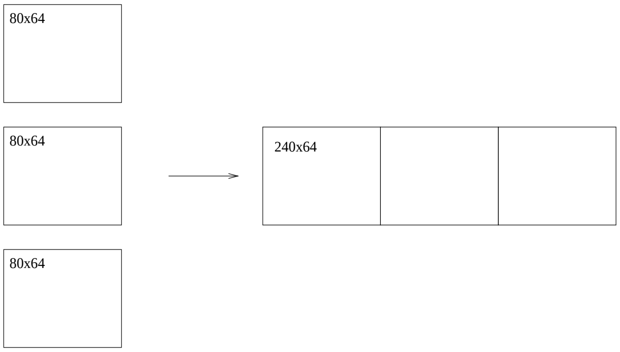

Figure 3: Merging three individual 80x64 frames, into a single 240x64 frame.

The modest spatial and temporal resolution of the video used for audience members will allow us to perform the background image processing on precoded data and include the task in the rest of those of the DA user application (that is, at the sender's workstation). The data is collected, preprocessed, and processed in standard 24-bit RGB format. It is converted to the DA's current YUVCR format just before transmitting.

There are a number of video preprocessing steps to facilitate segmenting the audience members from their background. First, the raw video is collected at the native resolution of the audience member's camera. Then, the odd rows of the image are removed to avoid interlace artifacts inherent in some video formats. The image is then downsampled horizontally by a factor of two to preserve the aspect ratio of the image, reduce the amount of data to be processed, and to improve signal to noise levels of the image intensities.

A feature of the Digital Amphitheater user application is that it enforces a brief training period in which the audience member must turn the camera on but stay out of the frame. During this training, the background reference frame is collected. A heuristic is used to detect if the audience member is not providing empty background for this training and the DA user application will not add the user to the audience without it.

The background reference frame is preprocessed as described above. There is not a large constraint on users other than the training, but we have yet to determine how well the DA will be able to compensate if there are either significant changes in lighting or moves of the camera after training.

The preprocessed video is displayed in a local mirror function to allow the audience member to check his or her position.

Frames collected after the background frame will undergo an additional preprocessing step to adjust for minor variations in camera exposure and lighting. An optimum scale is found between the current and background frame. Since we assume that the current frame will typically include the audience member, a simple least-squares error fit between the two frames would not produce the desired scale. Instead, we seek the statistical mode rather than the mean of the ration of the current frame to the background frame.

For each frame, we calculate the histogram of the ratio of pixels from the current frame to the background frame quantized into 256 bins over the range 0.5 to 2.0. Pixel pairs with ratios outside this range are not included in the histogram. We also exclude pixels with R, G, or B values of 255 as they may be saturated.

Once the optimum scale has been found, pixels from the current frame are compared to the scaled pixels from the background frame. A binary mask is created of the pixel pairs that are within a threshold distance. This mask will be filtered with morphological operators to remove isolated clusters of pixels [7]. The filtered mask will define the segmentation of the audience member from the background. The pixels in the current frame identified as background will be replaced with an image of an auditorium chair common to all audience members of the DA. Another simple algorithm will provide a base initial size of the person.

The processed image is further downsampled to the resolution required for displaying audience members on the DA, encoded, and sent to the amphitheater video merge server (possibly collocated).

Name	CPU	RAM	OS
chai	400 Mhz PPRO	64M	FBSD 2.2
sport	200 Mhz PPRO	96M	FBSD 2.2
hafez	550 Mhz PIII	256M	Linux 2.2

Table 1: Measured PC Hosts

6 The Amphitheater Video Merge Service

The amphitheater video merge service (AVMS) is responsible for merging multiple individual video streams into a single video stream. To create a merged video frame, individual video frames are tiled next to each other. The meta-data in the merged frame is adjusted to reflect the correct position of the video data within the new frame. Figure 3 displays an example of how three video frames are merged.

In our current implementation of the video merge service, we have used a non-traditional video codec, YUVCR, designed to exploit high bandwidth testbed links, that does some compression based on conditional replenishment [4]. In YUVCR, the video data is represented in planar YUV format, where the chrominance and luma values are represented separately. Compared with an RGB representation, this can provide for some reduction in data; for example with a 4:2:0 color subsampling, the video data is reduced by half compared with RGB.

The conditional replenishment also results in rate reduction. It operates in the following manner: Each video frame is divided into blocks of 16x16. Before transmission, each block is compared with the corresponding block from the previous frame and only if the content has changed the block is transmitted. The conditional replenishment algorithm also includes an aging process which will cause unchanging blocks to be transferred after a certain period.

Since video frames are transmitted in block units and only changed blocks are transmitted, when pack-

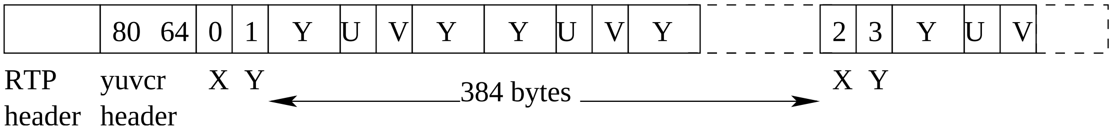

Figure 4: The RTP payload for a 80x64 video frame with two new blocks, (0,1) and (2,3).

etizing a video frame, each block must be preceded by its coordinates in the video frame. Figure 4 displays the RTP header and payload for a video frame of which blocks (0,1) and (2,3) are being transmitted. The figure displays the standard RTP header followed by the payload header, which for YUVCR contains the height and width of the video frame (80x64). Next, are the coordinates of the (0,1) block, followed by its data and then coordinates of the next updated block and its data. Each block is 384 bytes long and an RTP packet must always carry an integral number of blocks.

YUVCR lends itself well to our notion of merging with its block based structure, because blocks carry their coordinates with them and this reduces our merge algorithm to the manipulation of these coordinates. Because an AVMS is also likely to be a commodity PC, the processing it has to do must be of as low overhead as possible.

The operation of the actual merge algorithm is as follows: First the arriving video streams are sorted out, and partially reconstructed, while a boolean matrix keeps track of new incoming blocks. As soon as a frame is completed (the RTP M-bit is received), it is placed in the output data.

After a set time interval, (the output frame rate of the merge services is a parameter and later will be agreed upon among the AVMSs), an RTP stream is generated from the output buffers. This video stream represents the merged video frame, therefore all of the x and y coordinates in this stream are adjusted so as to reflect the correct position of the given block in the merged video frame. And of course, only updated blocks since the last transmission, are transmitted.

The separation of the incoming and outgoing video streams via the input and output buffers, allows for independent frame rates between the incoming video streams and the outgoing video stream. Therefore, the AVMS can adjust the outgoing frame rate according to current bandwidth availability.

7 Measurements

We conducted measurements pertaining to the central premise, that a commodity desktop PC can sustain the images of a large meeting better if most of the video flows are merged into a high-bandwidth but single flow. An experiment with a hundred individual senders daunted our ability to coordinate senders (still lacking the carrot of a new application for them to try at this point), so we conducted the measurements with all the senders we could muster, maximally thirteen. We contrasted performance measurements of all sending their individual multicasts with measurements of a flow from a prototype AVMS, merging the individuals. The systems used in the measurement are listed in Table 1.

A separate PC collected full trace data using tcpdump, so that the throughput, frame rate and so on of merged and unmerged flows could be measured.

For measuring the internal behavior (of the three systems in Table 1), we extended the vmstat tool as it is found in FreeBSD (calling it xvmstat) so that it included interrupt and load average statistics, and simply collected this at one-second intervals. The video senders, whether they were multicasting individual flows or unicasting to the AVMS, were a large range of Pentium PCs running a YUVCR source with the 80x64, five frames per second, audience parameter as their only possible output. The software sourcing YUVCR source was not the DA user application yet, but was a UCL vic [12] we modified to send and receive YUVCR. The DA user application's Java GUI needs more optimization than we were able to achieve for the timing of the NOSSDAV workshop.

Flow	Kbps	PPS	FPS
ToMerge	788.98	134.17	102.16
FromMerge (13 senders)	1011.14	117.32	1.59
Individuals (12 senders)	670.43	113.82	86.56

Table 2: Video Applications' Network Traffic

System	Ctxsw/s	NIC Intr/s
chai (AVMS)	116.84	481.31
sport (rcv merge)	91.67	312.59
sport (rcv indivs)	159.91	312.13
hafez (rcv merge)	965.24	360.82
hafez (rcv indivs)	1263.87	355.49

Table 3: Video Processing Performance

The receiver for the performance measurements was also a modified vic that would display the merged or unmerged YUVCR. Receiver statistics were taken with the receiver windows selected, not in thumbnail form (the individual flow windows are bigger than thumbnails). The measurement runs were a minimum of 300 seconds long.

The results of the measurements are shown in Tables 2 and 3:

The results are clear that the use of merged flows for the DA is likely to improve the end-user's system performance in at least one of the metrics described above, the context switch processing. (There is also a surprising, but consistent difference in the magnitude of the measured context switches per second between FreeBSD and Linux, which we have not yet explained). The fact that measurement senders failed during our individual runs makes it hard to say that the merge service has no impact on interrupts per second for the receiver; the individual runs all unfortunately had one less sender than the merged runs. More measurements are planned, of course.

CPU load average is another metric that is obviously of interest for assessing the DA design, but the measurements of CPU shown by the tools for these runs were very low for both merged and unmerged experiments (ranging from 0.02 to 0.15, with considerable variation). CPU load experiments with much larger numbers of senders will be a future study for the DA project.

An exception with respect to load average was when our initial naive rendering of the YUV planar data into RGB caused the 200 Mhz measured system to thrash. Those measurement results have guided a revised receiver design for both the test tool (the modified vic) and the DA user application (Section 8).

8 Conclusions and Future Work

As expected, we found some processor performance benefits in having a merged video stream instead of individual streams, despite similar packet per second rates of the two flows. In future work, we will understand the difference between Linux and FreeBSD results better. We will investigate the impact of the larger frames on quality of video under loss. We expect to be able to show more benefit when at higher CPU loads, but this result will require larger tests.

As we work towards characterizing the space of the benefits from a merged video stream for very large conferences, we also continue to develop the DA application software. Our future work will include making a general release of the DA source and seeking a hundred volunteers to participate in testing it. We expect that the video flows for a full conference could exceed 10 Mbps, so we expect that good sites for users will be on US and international educational testbeds.

The measurements are important to efficient implementation of the application. In early runs, the large merged video stream caused a 200 Mhz PC to reach full CPU utilization, with large numbers of page faults registering in the xvmstat output.

We speculate that this was the result of conversion between image formats when the merged video was displayed. The YUVCR codec used in the DA encodes 16x16 blocks of pixels in YUV 4:2:0 planar with conditional replenishment.

In order to be displayed, the YUVCR encoded video must be decoded into a sequence of frames. The frames are typically created in YUV planar format and converted to RGB for display. Unfortunately, as the size of the video frame increased, as in merging a large number of video streams, converting from YUV planar to RGB apparently caused significant page swapping.

In order to eliminate this thrashing due to converting the large merged frame from YUV planar to RGB, we plan to generate an RGB format image directly from the 16x16 blocks of the YUVCR stream. Although the input blocks are still in planar format, their small size will lead to far fewer page faults.

References

[1] E. Amir, S. McCanne, and R. Katz. An Active Service Framework and its Application to Real-time Multimedia Transcoding. In *Proceedings of ACM SIGCOMM*, Vancouver, 1998.

[2] M. Handley and V. Jacobson. SDP: Session Description Protocol. RFC 2327, IETF, April 1998.

[3] M. Handley, C. Perkins, and E. Wheland. SAP: Session Announcement Protocol. Work in Progress NA, IETF, March 2000.

[4] Mark Handley. YUVCR Codec. personal correspondence, AT&T Center for Internet Research at ICSI.

[5] H. Holbrook and B. Cain. Source-Specific Multicast for IP. Work in Progress NA, IETF, March 2000.

[6] H. Holbrook and D. Cheriton. EXPRESS Multicast. In *Proceedings of ACM SIGCOMM*, Cambridge, MA, 1999.

[7] Anil K. Jain. *Video Compression.* Prentice Hall, 1989.

[8] A.S. Patrick. The Human Factors of MBONE Videoconferences: Recommendations for Improving Sessions and Software. Technical report, Communications Research Center, Ottawa, 1998.

[9] H. Sandick and B. Cain. PIM-SM Rules for Source-Specific Multicast. Work in Progress NA, IETF, March 2000.

[10] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. RTP: A Transport Protocol for Real-Time Applications. RFC 1889, IETF, Jan 1996.

[11] J.C. Tang and E.A. Isaacs. Why Do Users Like Video? *Computer Supported Cooperative Work (CSCW)*, 1:163–193, 1992.

[12] http://www-mice.cs.ucl.ac.uk/multimedia. Freely available vic, sdr, and other tools, University College London, 2000.