IRI-h, A Java-Based Distance Education System: Architecture and Performance

K. MALY, H. ABDEL-WAHAB, C. WILD, C. M. OVERSTREET, A. GUPTA, A. ABDEL-HAMID, S. GHANEM, A. GONZALEZ, and X. ZHU
Old Dominion University

We used our original Interactive Remote Instruction (IRI) system to teach scores of university classes over the past five years at sites up to 300 km apart. While this system is a prototype, its use in real classes allows us to deal with crucial issues in distributed education instruction systems. We describe our motivation and vision for a reimplementation of IRI that supports synchronous and asynchronous distance education. This new version, called IRI-h (h for heterogeneous), is coded in Java and executes on several different platforms. IRI-h extends IRI both to multiple platforms and heterogeneous network infrastructures, including delivery to home users. In this article we describe IRI-h's architectural experiences with the developing prototype, including preliminary performance evaluation, and also unresolved issues still to be addressed.

Categories and Subject Descriptors: C.2.4 [**Computer-Communication Networks**]: Distributed Systems—*Distributed applications*; K.3.1 [**Computers and Education**]: Computer Uses in Education—*Collaborative learning*; *Distance learning*

General Terms: Design, Management, Performance

Additional Key Words and Phrases: Heterogeneity, Java, platform independence

1. INTRODUCTION

We have used the current version of our Interactive Remote Instruction (IRI) system to teach dozens of for-credit university classes over the past five years. These classes use university-provided equipment in sites up to 300 km apart connected by an Intranet with dedicated bandwidth. The current IRI supports multiple simultaneous video/audio, tool-sharing, and many other services in which any class member, student, or instructor can become a presenter, controlling a collection of shared tools at will to provide

The IRI-h project is partially supported by NSF under grant NSF-RED-9554261.
Authors' address: Department of Computer Science, Old Dominion University, Norfolk, VA 23529–0162; email: {maly; wahab; wild; cmo; ajay; hamid; ghanem; gonza_a; zhu}@cs.odu.edu.

a synchronous virtual classroom environment (see Maly et al. [1997] for more complete descriptions of IRI's current capabilities and design).

Even with all of this experience, IRI is still a work-in-progress. These live-classroom experiences provide the bases for a better understanding of both technical implementation issues, features important to teachers and students, and interface issues. We used our improved understanding to design a new IRI called IRI-h [Maly et al. 2000] that we are now completing. The "h" in the acronym stands for heterogeneous to indicate that, in contrast to the old system, it functions on multiple platforms and network environments.

The rest of the article is organized as follows. Section 2 summaries limitations of the current IRI system, and hence the motivation and general overview for IRI-h. Section 3 presents a view of the IRI-h use of heterogeneous networks; Section 4 depicts IRI-h software architecture. Section 5 explores the IRI-h prototype implementation and capabilities. Section 6 presents preliminary performance results for IRI-h major services, including audio, video, and tool-sharing. Section 7 highlights some unresolved issues, along with preliminary suggested solutions. Finally, the article concludes in Section 8.

2. IRI-h: MOTIVATION AND OVERVIEW

Though we are now using the current IRI system each semester to teach distributed synchronous classes, we recognize several deficiencies in this version:

(1) *Platform-independence problem.* The current IRI architecture is heavily dependent on UNIX system calls and the X windowing environment.

(2) *Heterogeneous network environment problems.* The current IRI is designed to be run over a private Intranet where QoS (Quality of Service) is achieved by careful engineering of this controlled environment (both in terms of network and platform resources).

(3) *Late join problem.* The design of the tool-sharing engine makes it difficult for participants to join a session late or to rejoin after network or platform failures. Many times, the only successful strategy for dealing with system failures is to restart the entire session.

(4) *Single shared view of all participants.* It is often beneficial to divide classes into groups for parts of a class session such that groups can work independently. The current IRI only supports a single shared view of all class activities.

(5) *Scalability.* IRI uses a reliable multicast protocol as the communication architecture underlying the tool-sharing engine; it proved to be not truly scalable both in terms of number of users and amount of traffic generated by such heavy uses as downloading large image files in a browser.

The limitations of the current IRI system can be summarized as *homogeneous platform, homogeneous network, homogeneous start time, and homogeneous room view*. The move to a heterogeneous environment requires solutions to the following problems:

—*Collaboration engine with multiple-platform-tool-source, platform-independent delivery*. These allow sharing of computer tools that exist in different hardware platforms by all participants in the session. In particular, the new version of IRI will have to make available the rich set of applications running on Windows 9X/NT/2000 environments.

—*Scalable semireliable communications*. IRI currently uses RMP [Whetten et al. 1994] for reliable multicast communications. While important in the current success of IRI, RMP is limited by the slowest client and requires careful tuning for different network configurations. In IRI-h we use unreliable multicast at the core with our own mechanisms for enforcing reliability. We believe that semireliable multicast is the key to achieving scalability in a heterogeneous environment.

—*Shared multiprogram/multiwindow graphical user interface*. In a learning environment it is important that the teacher focus the student's attention on the current topic being discussed. This means, among other things, that the position and focus of the windows displayed on the student's workstation be coordinated with the instructor's machine. In the original IRI, students could rearrange their view of the shared screen, but could resynchronize to the teacher's view or be resynchronized by the teacher. This mechanism was costly to implement and subject to anomalous situations.

—*Platform/environment management including late join*. IRI-h can start and stop a session with no student intervention. Students can arrive in class late and the system will already be delivering information from the active participants. In addition, a student can join an ongoing session at any time and fully participate in that class. Upon termination of a session, individual student notebooks and the session-recorded information is formatted and made available on the IRI-h web site for access outside the IRI-h environment.

—*Delivery to the home user*. IRI-h will include home users who access the session over a regular Internet connection using the current generation of high-speed at-home Internet connections.

—*Virtual rooms*. The class can be divided into groups by assigning each group a virtual meeting room. Students can move from room to room and join in different ongoing discussions.

—*Situational awareness*. Students, teachers, and technical engineers are made aware of the current operating environment and are notified about noteworthy changes or unusual situations.

Regarding network heterogeneity, we consider the following cases:

—*Nonmulticast enabled nodes.* In general, we can expect that nodes at the typical home will connect over a route that does not support multicast routing.

—*Low-bandwidth nodes.* Although most nodes participating in an IRI-h session will be at sites with high-quality networks, some will be reached over networks with low bandwidth.

—*High-delay nodes.* When nodes are at significant distances, say greater than 140 km, then the impact on interactivity due to transmission delays can be significant. While delivery of the session can tolerate rather significant delays for passive participants, interactivity, in the form of question asking or session presentation, will be impacted and will require different management of the network resources.

Providing all of these services in a platform-independent version is not easy. However, we have been experimenting with a platform-independent implementation of selected services. Most of these experiments use the Java programming environment [Sun Microsystems Java home page]; and include the use of Java APIs for distributed systems development (Java Shared Data Toolkit–JSDT[Sun Microsystems JSDT home page]); multimedia libraries (Java Media Framework–JMF [Sun Microsystems JMF home page]); and GUI interface design (Java Foundation Classes–JFC). Overall, initial results are encouraging.

As we complete the new implementation, we must ensure that we retain the features, abilities, and performance characteristics that proved to be most successful while solving the problems described in this article. Desirable features from the old IRI system include ease-of-use for students at all levels of the technology curve and ease-of-interaction for students in different locations.

The IRI-h prototype is implemented fully in Java, and has been tested on PCs running the Windows operating systems (NT, 98, 2000) and on Unix machines running the Solaris operating system. It implements tool-sharing and has a simple interface adaptable to the various roles of an IRI-h session participant: student, teacher, presenter, and monitor. The communication infrastructure utilizes a combination of reliable/semireliable/unreliable uni/multicasting. Video management is performed by a single click and audio is basically hands-free. JMF is used for both capturing and playing the audio and video of each participant. JMF uses RTP/RTCP [Schulzrinne et al. 1996] to transmit the captured media. The following key decisions were made in light of the expressed goals for IRI-h. Tool-sharing is implemented using a lossless video encoding in conjunction with a semireliable transport protocol. Due to the high resolution of many computer applications, the tool-sharing engine allows control of delivered bit rate ranging from 10Kb/s to 4Mb/s. At this time, tools are captured only at a PC, although the PC runs an X server and thus can remotely run an X application on a UNIX machine. The IRI presentation tool [Maly et al. 1997] is eliminated, and we use instead a class web site to store presentations

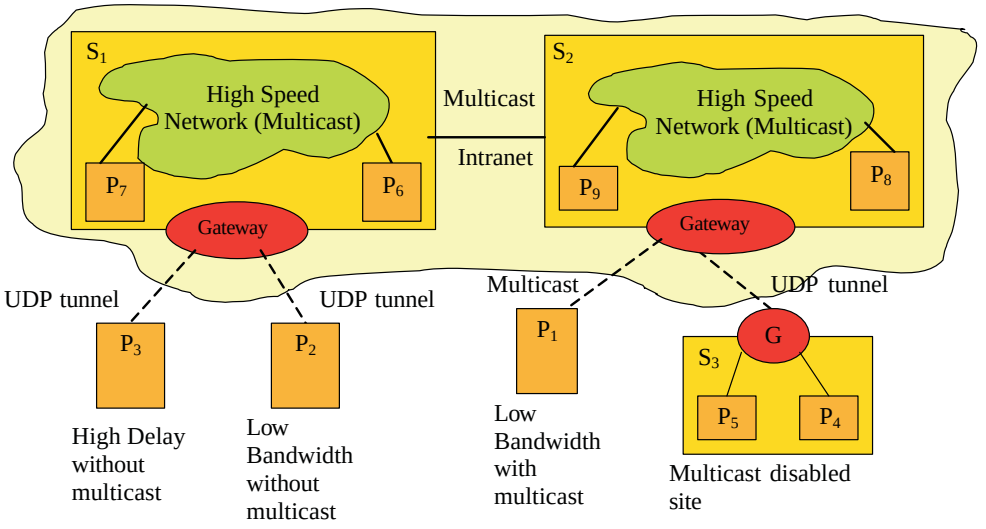

Fig. 1. A typical IRI-h network structure.

accessed in class through a browser. Students and teachers can also use their own web sites for presenting their slides using the tool of their choice. A major reliability problem in the current IRI is the lack of the late join/leave feature. In IRI-h we support both; we have again made a tradeoff between reliability and scalability by eliminating distributed servers that maintain considerable state information about the classes and their sessions. Class information is now only at one server (with a backup), and session information is maintained only at one session manager server. IRI-h has now been explicitly designed to work well for up to one hundred nodes.

3. NETWORK STRUCTURE

Figure 1 shows a mixed network layout as might occur in typical use of IRI-h. As with the previous version of IRI, we expect the system to be used in classrooms equipped with a high-speed network connections to each student's workstation.

Figure 1 illustrates sites 1 and 2 that represent classrooms supported by high-speed network connections and multicast to each class member's workstation. To support other class members, whose connections may have limited capabilities, these sites include a gateway capable of providing appropriately modified services. These can include unicasting to multicast disabled sites such as site 3, where a local gateway can distribute packets to participants; translation of video formats, and reduction of bandwidths to low-bandwidth connections.

4. SOFTWARE ARCHITECTURE

Figure 2 depicts the software architecture for IRI-h's main components, namely, SP (session participant, one per desktop), and SM (session manager, one per class session). Both are multithreaded processes. SP connects

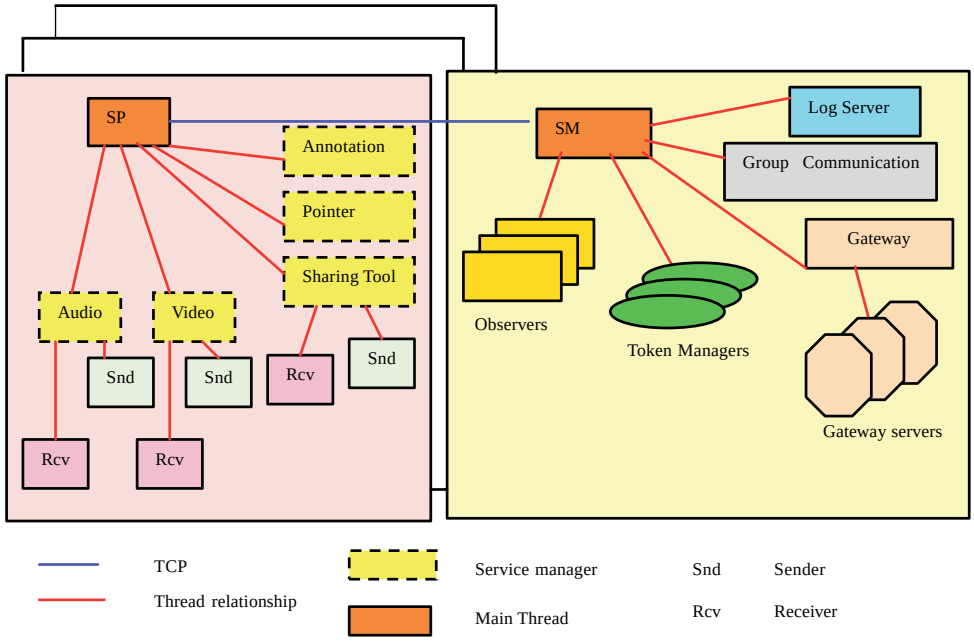

Fig. 2. IRI-h software architecture.

to SM through a permanent TCP/IP connection [Postel et al. 1981], maintained for the lifetime of the session. Session participants run a set of *services* that use shared *resources* that are allocated and managed through the SM. The SM maintains a set of *rooms*, each room contains a set of *services*, and each service has allocated *resources*. In addition, SM maintains the participant's attributes, e.g., his current room. Currently, the supported services are audio, video, presenter, annotation, pointer, and sharing tool. For example, the resources required by the video service are a group communication channel, a video observer, and a gateway server. The resources required by the annotation service are a group communication channel, a token, an annotation observer, and a gateway server. Resource management by the session manager is performed through a two-phase process. In the first phase, an *allocating SP* requests the startup of a service in his current room and the allocation of its shared resources. In the second phase, all participants within this room receive the information on how to contact the service resource managers. For example, when the session manager allocates a token, a token manager is created, and the (IP address, port) pair for this token manager is returned to the participants. SM sends any service information to the latecomers after they connect to it and login.

4.1 SM Components

The *log server* logs messages from each IRI-h process participating in the IRI-h session. Each process connects to the *log server* through a permanent

TCP/IP connection (not shown in Figure 2). The TCP/IP connection is used to transport messages to the log server. The messages are saved in an archival file, or are output to a monitoring display. We also have an automated start-up process that starts IRI-h sessions on a configurable or predefined set of machines and users. Since the start-up process runs before a session is running, messages are logged separately and are viewable through the browser involved in starting a session. The concept is to start IRI-h services on those machines that are known to be part of a class, and then other students use the late-join feature to join an ongoing class from their particular machines.

The group communication server allocates group communication channels requested for services. Group communication can be reliable or unreliable. The current prototype implementation provides unreliable group communication through IP multicast [Deering 1989]. Each service requiring a group communication channel is guaranteed a unique (multicast IP address, port) pair across all running IRI sessions (if any) through the group communication server allocation policy. The server provides a mapping from textual group communication names such as *"Room1-Video Group"* to communication entities such as a (multicast IP address, port) pair. To support virtual rooms, each room is assigned a unique multicast address. This multicast address is used by all services requiring group communication channels within this room. It is formed as a function of the session-manager machine's IP address, session manager's server port, and the room number. Ports are allocated by subdividing a preset range of ports among a maximum number of allowed session managers per machine. Within each session, ports are divided among a maximum number of allowed virtual rooms.

The *gateway* solves the heterogeneity problem manifested by varying connectivity bandwidths available to participants. The gateway detects the multicast ability of a session participant by performing a simple multicast test with this participant. If the participant is multicast-disabled, tunneling services are provided through the gateway to deliver any multicast data streams. In addition, the gateway provides for adaptive content delivery for participants to meet their connectivity bandwidth constraints. An example is a gateway server for a video service adapting the video stream frame rate from 15 frames/second to 5 frames/second. The gateway plays a crucial role in making IRI-h available to home users with limited connectivity bandwidth. Note that, for simplicity, the gateway component is illustrated in Figure 2 as a thread within the session manager. For bandwidth and physical network connectivity issues, the gateway may be running on an independent machine, other than the session manager machine. Tunneling services and adaptive content delivery are not implemented in the current prototype and are subjects for ongoing research.

Service observers provide recording capabilities to the IRI-h session. Several services that need to be recorded include audio, video, whiteboard, and tool-sharing, among others. Service observers act as passive receivers to any data stream generated by a service, and save these data streams for

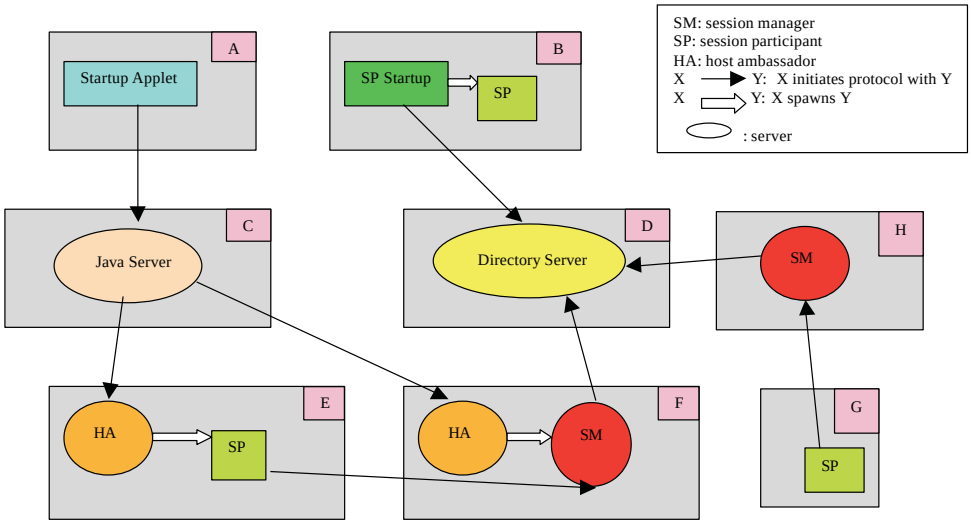

Fig. 3. Various IRI-h session start-up scenarios.

later playback. Recording and playback of generated data streams is an ongoing effort, and is not part of the current prototype implementation.

5. IRI-h PROTOTYPE IMPLEMENTATION

As mentioned previously, an IRI-h session is composed of a session manager, SM, and a set of session participants, SPs. The SM is a server that has to be running before any SP can join a session, and acts as a coordinator between all SPs. SPs running on distributed machines initiate the connection with SM. An SP GUI gives a participant access to a set of shared tools (services) that allow collaborative work between all SPs. The available tools are audio, video, annotation, pointer, note-taking, and tool-sharing engine. The following sections present various session start-up scenarios, the SP desktop GUI, and start-up capabilities for SM and SP.

5.1 Session Start-up Scenarios

An IRI-h session can be started through an individual join mechanism, or through an automated start-up procedure. Figure 3 illustrates the different components involved in the start-up procedure. For example, the SM running on machine H and the SP running on machine G are started by an individual using command-line start-up. The SP running on machine B is started with the help of a *"directory server"* running on machine D, while SM running on machine F and the SP running on machine E are started by an automatic start-up procedure.

The *"directory server"* always runs on a well-known machine and port, and all SMs register themselves, providing their machine names and ports. The *"SP start-up"* module contacts the *"directory server"* querying the

current registered sessions and retrieves the machine name and port of the SM for the class the user wants to join and then spawns an SP.

An administrator can configure and run a session through a Java applet interface, *"startup applet"* on machine A. The applet allows the administrator to choose a class configuration file, a machine to run the session manager SM, and a set of machines that will participate in the session SPs. The applet contacts a *"Java server"* (running on machine C), and passes all session parameters. The *"Java server"* triggers an automatic start-up procedure that performs the following:

(1) Contacts HA (host ambassador) on the session manager machine, requesting to run SM and passing all session configuration files.

(2) Waits for an acknowledgment that SM is ready. The SM sends the acknowledgment message after reading all the configuration files, and just before waiting for participants to connect to it. The acknowledgment message contains the SM server port.

(3) In parallel, contacts all HAs on all participating machines, requesting to run an SP, passing it the SM machine name and server port.

The *"host ambassador,"* HA, is an IRI-h agent that facilitates remote invocation of components. The HA is run on dedicated classroom machines, or the participants can choose to run this server (service) on their machines, so they are notified when classes begin to allow SP to run automatically. The SM, when started, registers his/her machine name and port with the *"directory server"* to help users join current sessions after start-up.

5.2 IRI-h Desktop

An SP has two views available through the IRI-h desktop, a private view and a shared view. The shared view is a consistent view across all participants. All changes to the shared view by one participant are propagated to other participants. A participant needs to provide his/her login name and password before receiving the shared view and actually participating in the session. Before login he/she is given access to a private panel that provides a display of a feedback window, as shown in Figure 4. The feedback window illustrates members who are logged-in and some information about each SP. An SP desktop is shown in Figure 5. The top bar contains buttons for token-controlled tools such as presenter, pointer, and annotation. The bottom bar contains a fixed set of buttons for controlling audio and video and a dynamic set of buttons for controlling annotation, pointer, and a sharing tool engine. The shared view consists of any received video windows, annotations, and shared tools. Figure 5 depicts the SP receiving two video streams, identified by the sender's machine name.

The current presenter controls the layout of the shared view. The presenter is a token controlled tool, with the name of the current token holder appearing on the button. Any participant is allowed to grab the presenter token by pressing that button, and can arrange the shared view

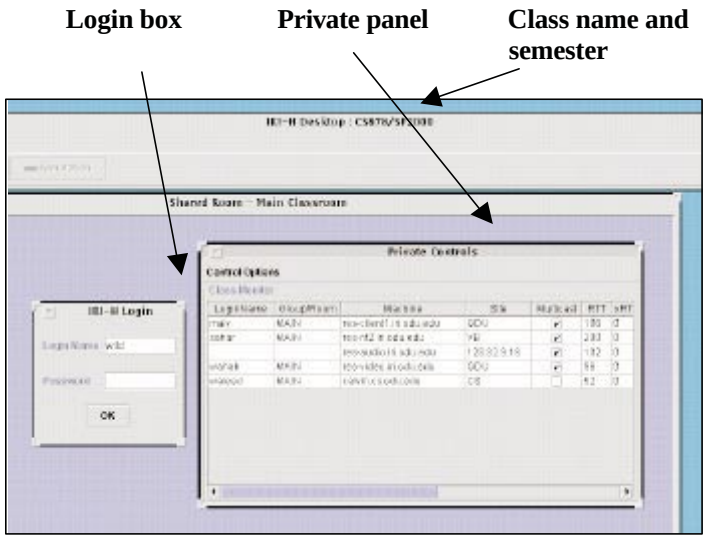

Fig. 4. A session participant GUI before login showing the class monitor in the private panel.

layout. There are two more token-controlled tools, namely annotation and pointer that allow their token holders to annotate or point to any location in the shared view. Referring to Figure 5, the shared view is annotated by drawing a circle and an arrow. The annotation tool utilities (draw, erase, and write) appear for its token holder on the bottom bar, as shown in Figure 5.

Due to bandwidth limitations and shared view real-estate constraints, the shared view can hold a maximum of 3 video windows. When the maximum limit is reached, a student wishing to transmit his/her video preempts one of the existing student video sources. The preempted video source is instructed by SM to stop transmitting the student's video stream. The presenter cannot be preempted if transmitting his/her own video.

Members capable of sharing tools will have a button for the sharing tool engine, allowing them to start sharing any tool running on their machines. Figure 5 shows a participant sharing a Netscape. Every shared application is token-controlled, and any participant can grab that token to control its actions, such as browsing in Netscape or flipping the slides in a PowerPoint presentation.

Each dedicated classroom machine belongs to an IRI-h site. Normally during an IRI-h session, a site contains a machine that acts as an audio server, playing audio streams received from other session sites. If the site is not in the site-machine list, the default site of a participant is his/her IP address. An SP can optionally be configured as a site audio server or as no audio receiver at all. The default audio receiver behavior is to run a local audio receiver, in which case all received audio streams are played (except his/her own audio stream).

The whole class can meet in one virtual room or it can be divided further into smaller groups. Each group meets in an individual virtual room, with

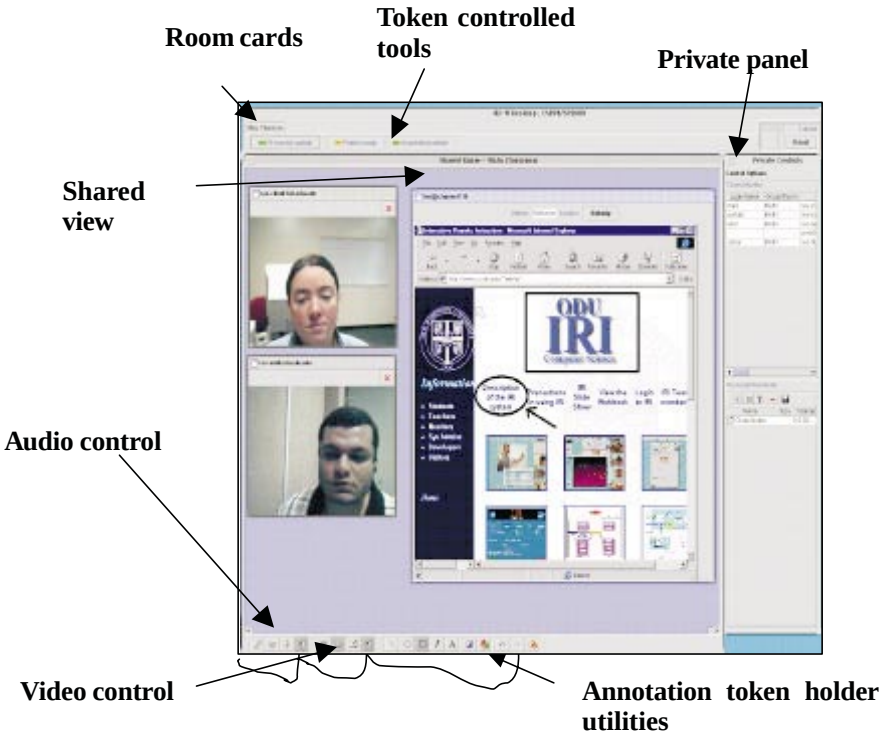

Fig. 5. A session participant GUI showing the shared view.

the ability to move between rooms. The virtual room paradigm is adopted to allow for IRI-h to be used by simultaneous discussion or study groups.

The only required parameter in order to run an SP is the SM (machine name, port) pair, which is provided by the automatic start-up procedure, the directory server, or, assuming the member knows it through some other means, email notification.

6. PRELIMINARY PERFORMANCE RESULTS

IRI-h was designed for both heterogeneity and scalability. In this section we report on the performance of the current prototype. These results are reported for each of the major services (audio, video, and tool-sharing) separately in order to demonstrate the demands each makes on the network and platform resources.

Audio. Audio bandwidth requirements are modest, on the order of 8 kbps per audio channel. In addition, because sound cards support the transmitted audio encoding, little CPU power is consumed in capturing or audio playback. Using JMF's audio services should therefore scale well, which is verified in our preliminary testing.

Video. Video bandwidth requirements are around 250kbps for 5 frames/sec given a JPEG image size of 320x240. Thus, given the "normal" IRI-h

video load of three video streams, the video bandwidth requirements would be on the order of 750 Kbps.

Platform loads for handling video are tied to the availability of video-compression hardware on the video cards. Here we have found difference in the platforms due to the capabilities of video cards. Owing to its image quality, we encode video using JPEG compression. Our Unix machines are equipped with video cards that compress/decompress JPEG encoded video streams, and thus do not require a lot of CPU power. However, on the PC machines, we do not have JPEG encoders on video cards, and are thus required to compress/decompress in software. This places a significant load on the CPU. While we have H263 encoders available for the PC, we felt the quality of the transmitted video precluded their general use.

Tool-sharing. The performance of the sharing tool – IPV (interactive program video) depends on the following activities: (1) capture images of the windows in the application being shared; (2) compare these images with previous images to see if the image has changed (for removing temporal redundancy); (3) compress the image; (4) transfer; (5) decompress; (6) display images on the client machine. Extensive studies were performed during the development of this protocol, and have been reported elsewhere [Gonzalez 2000]. In these studies two compression algorithms (JPEG and PNG—a public domain GIF-like algorithm) and two image styles (photo of two boys in the woods 388x566 pixels), and a Microsoft Word document containing plain text (680x580 pixels) were used. Capture time is a function of the image size only (measured around 220 msec for a 700x700 image on a Unix machine). Comparison time is between 300 and 500 msec. Compression time is a function of the compression algorithm, and ranges from approximately 1000 to 3000 msec, since this is performed in software. Transmission time depends on image type and ranges from 20msec for the text images to 350 msec for picture images (using PNG). On the receiver's side, performance is dominated by the decompression time that is around 500 msec. Due to the the large size of the images, IPV has been provided with a rate control feature that limits the bandwidth requirements requested by the originating machine.

7. UNRESOLVED ISSUES

Figure 2 depicts the software architecture of the IRI-h system's building blocks. However, several issues still need to be addressed, e.g., interstream synchronization and gateway functionality. The following sections present our vision and preliminary solutions for these problems.

7.1 Interstream Synchronization

A presenter in the IRI-h session might be transmitting his/her video and speaking while sharing a slide in a browser. Meanwhile, he/she is also pointing at a certain location in the browser window using the pointer service or annotating the slide using the annotation service. This scenario entails the presenter sending a video stream, audio stream, sharing a tool

stream and a pointer motion stream, or sending an annotation tool stream to the other participants. At the receiver participant, these streams may be received out of synch, due to differences in stream characteristics, size of data packets, and network delays. At the receiver side, an interstream synchronization protocol [Stoica et al. 1997] is required to maintain the timing relationships across the streams received. Participants connected to the IRI-h session through a gateway encounter the same problem. The gateway might transcode a video stream into another video format more suitable for a limited bandwidth connection, introducing an extra delay and aggravating the out-of-synch problem.

7.2 Gateway Components and Functionality

In terms of network bandwidth consumption, IRI-h services can be classified into *lightweight* (pointer and annotation) and *heavyweight* (video, tool-sharing, and audio) services. Lightweight services may only require a *tunneling gateway server* bridging the multicast gap between multicast-able IRI-h sites and multicast-disabled IRI-h participants. A received multicast packet is encapsulated in a unicast packet and forwarded to each gateway participant. No knowledge or processing of data stream content is required. Hence, such services require a generic tunneling gateway server. Heavyweight services might require a gateway server that performs tunneling, and/or data rate limiting and/or transcoding. Transcoding can be used for video streams, as an option to lower the bandwidth requirements of the video stream, e.g., transcoding from JPEG to H.261 [Amir et al. 1995]. Heavyweight services require specialized gateway servers geared towards the characteristics of the generated data streams. These gateway servers use their knowledge of the received data stream content to accomplish the gateway function. A number of issues need to be investigated to efficiently perform such gateway functionality.

—Can limits on the data rate be performed through a buffering approach at the gateway and playback at a lower data rate to gateway-serviced participants? Will the source of the data stream need to be notified to "slow down" in case the gateway is running out of buffer space?

—What is the bandwidth-allocation strategy for data streams originating from the gateway towards the gateway-serviced participants? Amir et al. [1997] introduced a receiver-driven bandwidth adaptation strategy for such gateways. Meanwhile, Youssef et al. [1998] suggested a quality-of-session control layer for interactive multimedia sessions.

—Is the transcoding process required, or can limiting the data rate by itself perform the required task of adapting the incoming data stream to lower bandwidth requirements?

8. CONCLUSIONS

Although the current version of IRI is by all measures a success, we recognize that extending its reach to a wider audience will require fundamental changes in the way IRI services are offered and managed. We

believe a key feature of the new version of IRI is its heterogeneity: heterogeneous delivery platforms, heterogeneous network environments, and heterogeneous arrival and departure times. We have presented a software architecture for achieving these goals and described a prototype implementation in Java that supports audio, video, application-sharing, annotation in a shared view and note taking, and situational awareness in a private view. At the time of this writing we have a complete base version of IRI-h that solves the first four problems listed at the beginning of this article. We have shown and tested the system on a variety of platforms simultaneously, and done initial scaling experiments with close to 50 nodes with no discernible performance degradation while sharing heavy tools among all users. Late join and leave were solved completely, and the reliability of the system has improved dramatically. Situational awareness has been addressed only with regard to participants who know the performance aspects of the system; we have not yet addressed the issue of providing learning context to all beyond the services already provided. The issue of virtual rooms will mostly be a performance-tuning, technical problem, although it should greatly increase the learning paradigm possibilities. The one really significant problem we need to address is that of the gateway for heavyweight services and synchronous presentation of services. Not all of the intended features have been implemented; but from our experience to date, we believe IRI-h will be more scalable and more robust, with better quality of service to a wider range of participants, than is possible in the current IRI system.

REFERENCES

AMIR, E., MCCANNE, S., AND ZHANG, H. 1995. An application level video gateway. In *Proceedings of the 3rd International Conference on Multimedia* (Multimedia '95, San Francisco, CA, Nov. 5–9), P. Zellweger, Chair. ACM Press, New York, NY, 255–265.

AMIR, E., MCCANNE, S., AND KATZ, R. 1997. Receiver-driven bandwidth adaptation for light-weight sessions. In *Proceedings of the Conference on Multimedia* (Seattle, WA, Nov. 9–13), E. P. Glinert, M. S. Johnson, J. Foley, and J. Hollan, Chairs. ACM Press, New York, NY, 415–426.

DEERING, S. 1989. Host extensions for IP multicasting. RFC 1112. Internet Engineering Task Force.

GONZALEZ, A. J. 2000. A Semantics-based middleware for collaborative multimedia applications. Ph.D. Dissertation. Old Dominion University, Norfolk, VA.

MALY, K. ET AL. 2000. IRI-h home page. http://www.cs.odu.edu/~iri-h.

MALY, K., ABDEL-WAHAB, H., OVERSTREET, C. M., WILD, C., GUPTA, A., YOUSSEF, A., STOICA, E., AND AL-SHAER, E. 1997. Distance learning and training over ntranets. *IEEE Internet Comput. 1*, 1 (May/June), 60–71.

POSTEL, J. 1981. Transmission control protocol. RFC 793. Internet Engineering Task Force.

SCHULZRINNE, H., CASNER, S., FREDERICK, R., AND JACOBSON, V. 1996. RTP: A transport protocol for real-time applications. RFC 1889. Internet Engineering Task Force.

STOICA, E., ABDEL-WAHAB, H., AND MALY, K. 1997. Synchronization algorithms for the playback of multiple distributed streams. In *Proceedings of the Fourth International Conference on Multimedia Modeling*. 143–158.

SUN MICROSYSTEMS. 2000. Sun's Java home page. http://java.sun.com.

SUN MICROSYSTEMS. 2000. Sun's Java media framework (JMF) home page. http://java.sun.com/products/java-media/jmf/index.html.

SUN MICROSYSTEMS. 2000. Sun's Java shared data toolkit (JSDT) home page. http://java.sun.com/products/java-media/jsdt/index.html.

YOUSSEF, A., ABDEL-WAHAB, H., AND MALY, K. 1998. The software architecture of a distributed quality of session control layer. In *Proceedings of the Seventh IEEE Symposium on High Performance Distributed Computing* (Chicago, IL). IEEE Computer Society Press, Los Alamitos, CA.

WHETTEN, B., MONTGOMERY, T., AND KAPLAN, S. 1995. A high performance totally ordered multicast protocol. In *Theory and Practice in Distributed Systems*, K. Birman, F. Mattern, and A. Schiper, Eds. Springer-Verlag, Berlin, Germany, 33–57.

Received: August 2000; revised: October 2000; accepted: November 2000