An Architecture and Initial Performance Data for a JAVA-based Distance Education System

K. Maly, H. Abdel-Wahab, C. Wild, C. M. Overstreet, A. Gupta
A. Abdel Hamid, S. Ghanem, A. González, X. Zhu

Computer Science Department
Old Dominion University
Norfolk, VA 23529-0162, USA

Abstract

We describe our motivation and vision for a reimplementation of our Interactive Remote Instruction (IRI) system that supports synchronous and asynchronous distance education. The new version is coded in Java and executes on several different platforms. We used the original system to teach scores of university classes over the past five years at sites up to 300 km apart. This system was built as a prototype, its use in real classes has forced us to deal with crucial issues in distributed education instruction systems. The new implementation of IRI, called IRI-h (h for heterogeneous), extends the IRI system both to multiple platforms and to heterogeneous network infrastructure, including delivery to the home user. This paper describes the architecture of the new implementation, experiences with the developing prototype including preliminary performance evaluation, and unresolved issues to be addressed.

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 a synchronous virtual classroom environment. See [8] 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 and of features important to teachers and students. We have used our improved understanding to design a new IRI called IRI-h that we are now completing. The 'h' in the acronym stands for heterogeneous to indicate that, in contrast to the old system, it will handle multiple platforms as well as multiple network environments.

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

2. IRI-h: Motivation and Overview

Though we are using the current IRI system to teach distributed synchronous classes each term, 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 in which 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. *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, and homogeneous start time.* The move to a heterogeneous solution will require that the following problems be solved:

- *Collaboration Engine with Multiple-Platform-Tool-Source, Platform-Independent Delivery*: This allows the sharing of computer tools that exist in different hardware platforms by all participants in the session. In particular the new version of IR will have to make available the rich set of applications running on windows 9X/NT/2000 environments.
- *Scalable Semi-Reliable Communications*: IRI currently uses RMP [13] 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 semi-reliable multicast is the key to achieving scalability in a heterogeneous environment.
- *Shared Multi-program/multi-window Graphical User Interface*: In a learning environment it is important that the teacher can focus the student's attention on the current topic of discussion. 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 be already delivering information from the active participants. In addition a student can join an on-going session at any time and fully participate in that class. Upon termination of a session, individual student notebooks and the session-recorded information is being formatted and made available on the IRI-h web site for access outside of 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 latest 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 on-going 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:

- *Non-multicast 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 kms – 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 and include the use of Java [5] APIs for distributed systems development (Java Shared Data Toolkit–JSDT [7]), multimedia libraries (Java Media Framework-JMF [6]) and GUI interface design (Java Foundation Classes–JFC). Overall, initial results are encouraging.

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

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/semi-reliable/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 [10] 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 semi-reliable transport protocol. Because of the high resolution on many computer applications, the tool sharing engine allows for control of delivered bit rate ranging from 10Kb/s to 4Mb/s. Tools are captured at this time only at a PC although the PC is running an X server and thus can remotely run an X application on a

UNIX machine. The IRI presentation tool [8] is eliminated and we instead use a class web site to store presentations accessed through the browser in class. Students and teachers can also use their own web site for presenting their slides using the tool of their choice. major reliability problem in the current IRI was the lack of the late join/leave feature. In IRI-h we support both; again we have 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. Software Architecture

Figure 1 depicts the software architecture for IRI-h main components, namely, SP (Session Participant – one per desktop), and SM (Session Manager one per class session). Both are multi-threaded processes. SP connects to SM through a permanent TCP/IP connection [9], maintained during 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 for each service its allocated *resources*. In addition, SM maintains the participants' 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 pertaining to 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.

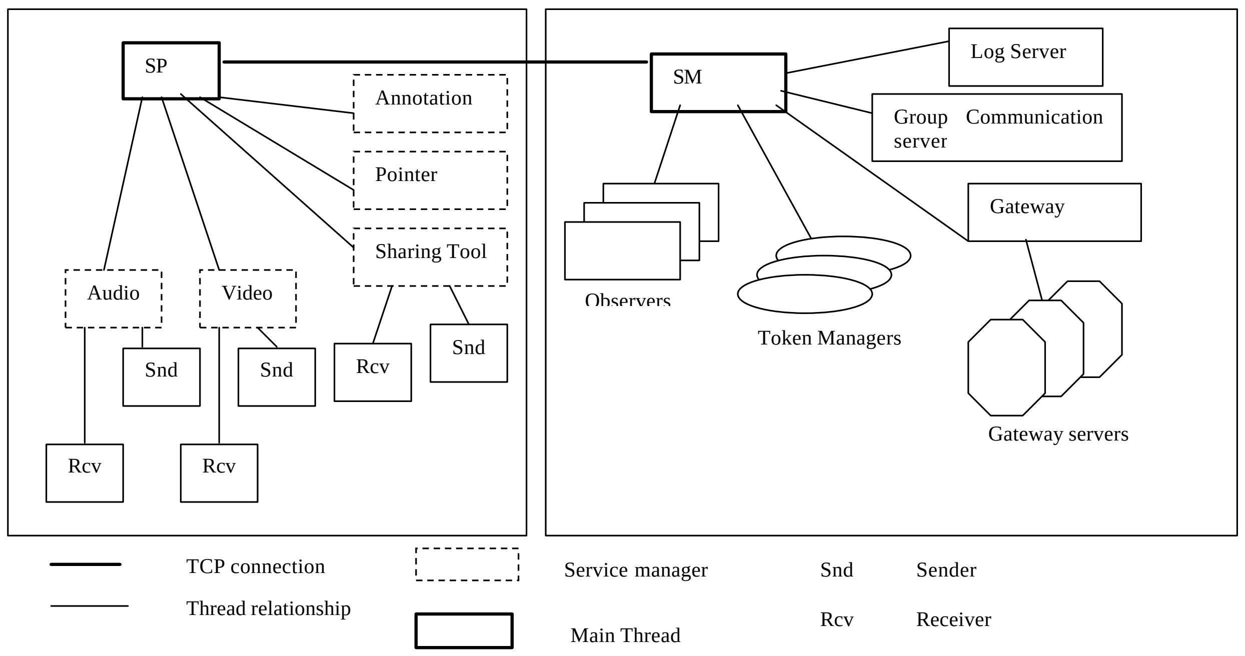


Figure 1: IRI-h Software Architecture

3.1. 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 4). The TCP/IP connection is used to transport messages to the log server. The messages are saved in an archival file, or 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 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 used and then all other students use the late-join feature to join an on-going class on their particular machine.

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 [3]. 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 IP address, session manager server port, and the room number. Ports are allocated by subdividing a preset range of ports between a maximum number of allowed session managers per machine. Within each session, ports are divided between 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 4 as a thread within the session manager. For bandwidth, and physical network connectivity issues, the gateway might 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 subject 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 later playback. Recording and playback of generated data streams is an ongoing effort, and is not provided as part of the current prototype implementation.

4. 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. 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 sharing tool engine. The following sections present various session startup scenarios, the SP desktop GUI, and startup capabilities for SM, and SP.

4.1. Session Startup Scenarios

An IRI-h session can be started through an individual join mechanism, or through an automated startup procedure. Figure 2 illustrates the different components that are involved in the startup procedure. For example, the SM running on machine H, and the SP running on machine G are started by an individual using command line startup. The SP running on machine B is started with the help of a *"Directory Server"* that is running on machine D. While, SM running on machine F and the SP running on machine E are started by an automatic startup procedure.

The *"Directory Server"* is always running on a well-known machine and port, and all SMs register themselves providing their machine names, and ports. The *"SP Startup"* module contacts the *"Directory Server"* querying the current registered sessions, and retrieves the machine name and port of the SM for the class he wants to join, 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 him 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. Wait 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, contact 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 invoking components remotely on machines. The HA is running on dedicated classroom machines, or the participants can choose to run this server (service) on their machine, so they are notified when classes start allowing SP to run automatically. The SM when started, registers with the *"Directory Server"* his machine name and port, to help users join current sessions after his startup.

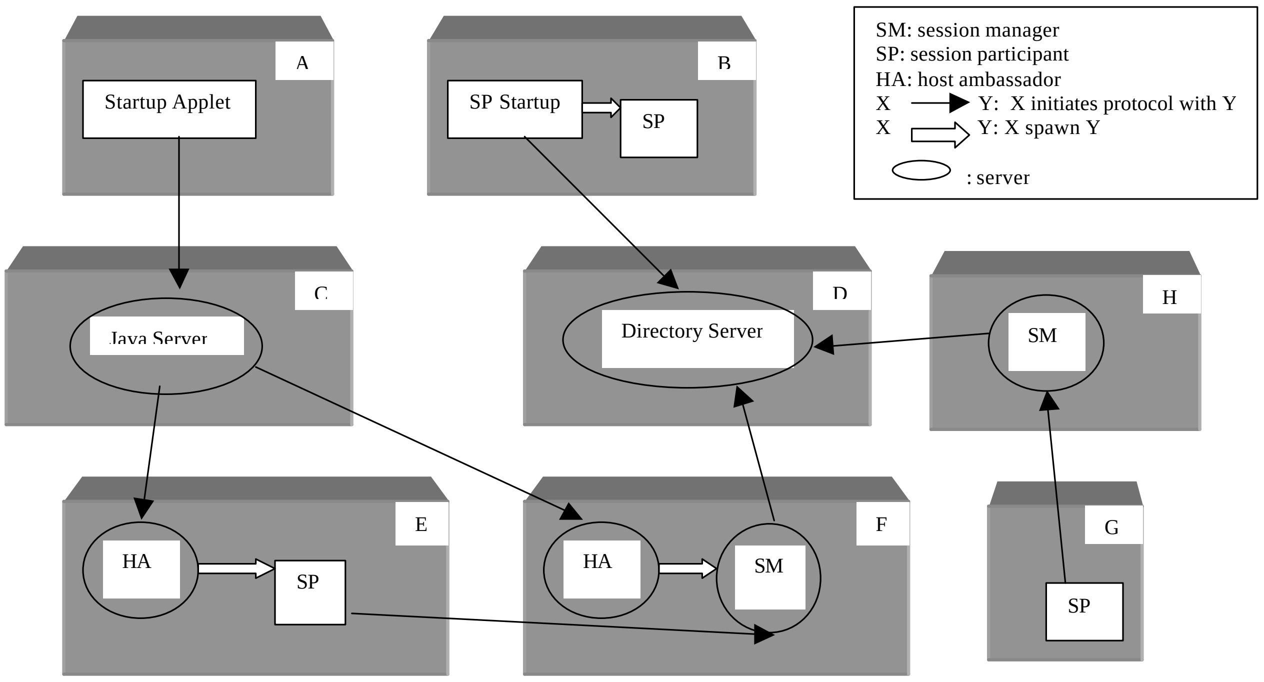


Figure 2: Various IRI-h session startup scenarios

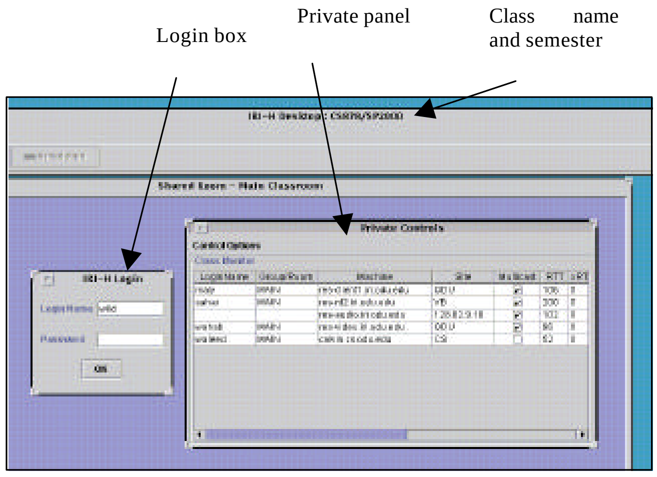
4.2. IRI-h Desktop

An SP has two views available through the IRI-h desktop, a private view, and the 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 login name and password before receiving the shared view and actually participating in the session. Before login he is given access to a private panel that provides a display of a feedback window, as shown in figure 3. The feedback window illustrates members who are logged-in and some information about each SP. An SP desktop is shown in figure 4. 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 4 depicts that the SP is receiving two video streams identified by the sender 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 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 3, the shared view is annotated by drawing a circle and an arrow. The annotation tool utilities (draw, erase, write) appear for its token holder on the bottom bar as shown in figure 4.

Due to bandwidth limitations, and shared view real-estate constraints, the shared view can hold a maximum of 3 video windows. In case the maximum limit is reached, a student wishing to transmit his video, preempts one of the existing student video sources. The preempted video source is instructed by SM to stop transmitting his video stream. The presenter cannot be preempted if he is transmitting his 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 machine. Figure 4 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 PowerPoint presentation.



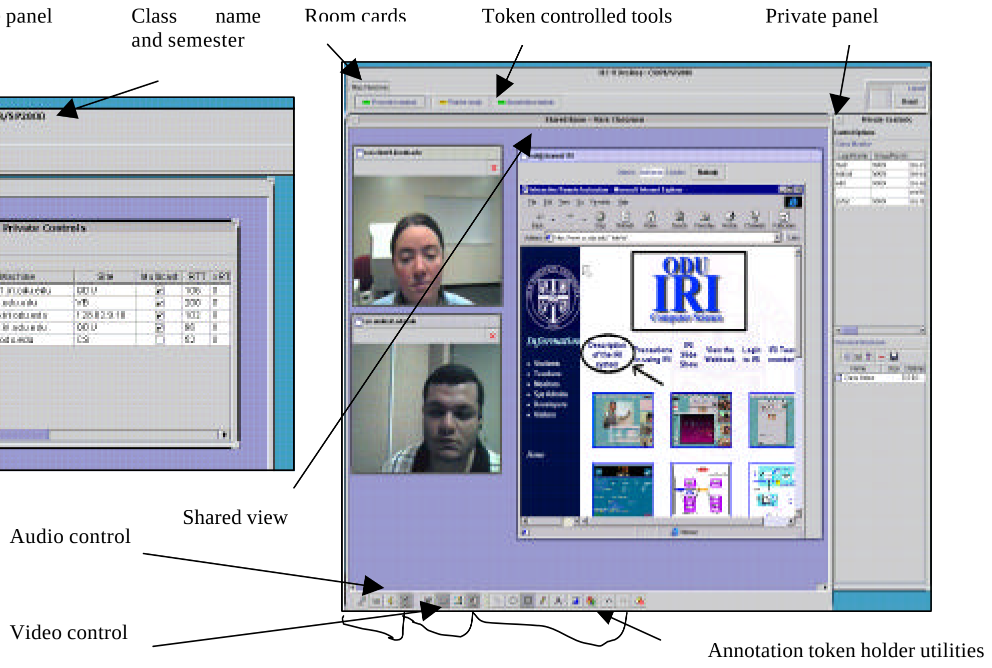


Figure 3: An SP GUI before login showing the class monitor in the private panel

Figure 4: An SP GUI showing the shared view

Each dedicated classroom machine belongs to an IRI-h site. Normally during an IRI-h session a site contains a machine, which 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 IP address. An SP can optionally be configured as a site audio server, or 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 own audio stream).

The whole class can meet in one virtual room, or it can be further divided into smaller groups. Each group meets in an individual virtual room, with the ability to move between rooms. The virtual room paradigm is adopted to allow for IRI-h to be used by simultaneous discussion groups, or study groups.

The only required parameter to run an SP is the SM (machine name, port) pair, which is either provided by the automatic startup procedure, the directory server, or assuming the member knows it through any other means, e.g., email notification.

5. Preliminary Performance results

IRI-h was designed both for heterogeneity and for scalability. In this section we report 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.

NETWORK RESOURCES: We tested network bandwidth on the configuration that consists of three Local Area Networks (running 100Mbps Ethernet) interconnecting by 10 Mbps links. To calibrate the network resources, we used a multicast ping program (mping) from which we calculate the raw bandwidth available on the various networks. The performance between two machines on the local area network RESIRI is on the order of 70 MBPS, and between networks is approximately 17 MBPS.

Audio: The 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 playing audio. Using JMF's audio services should therefore scale well, which is verified in our preliminary testing.

Video: Video bandwidth requirements are around 340kbps for 5 frames/sec given the image size of 320x240. At 15 frames/sec the bandwidth requirements approximately triple to 850 kbps. Thus given the "normal" IRI-h video load of two low- and one high-rate video streams, the video bandwidth requirements would be on the order of 1.5 Mbps.

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 capability of video cards. Because of its image quality, we encode video using JPEG compression. Our Unix machines are equipped with video cards which 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 the video cards and are thus required to compress/decompress in software. This places a significant load on the CPU. While we have available H263 encoders 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 client machine. Extensive studies were done in the development of this protocol and have been reported elsewhere [4] 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 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 300-500 msec. Compression time is a function of the compression algorithm and ranges from approx 1000 to 3000 ms since this is performed in software. Transmission time depends on image type and ranges from 20msec for text images to 350 msec for picture images (using PNG). On the receiver's side, performance is dominated by the time to decompress which is around 500 msec. Because of the large size of images, IPV has been provided with a rate control that limits the bandwidth requirements requested by the originating machine.

6. Unresolved Issues

Figure 1 depicted the software architecture of the building blocks of the IRI-h system. However, a number of issues need to be addressed such as inter-stream synchronization, and the gateway functionality. The following sections present our vision and preliminary suggested solutions for these issues.

6.1. Inter-stream Synchronization

A presenter in the IRI-h session might be transmitting his video, speaking while sharing a slide in a browser. Meanwhile, he is 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 tool stream, and a pointer motion stream, or 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, data packet sizes, and encountered network delays. A receiver side, inter-stream synchronization protocol [11], is required to maintain the timing relationship across the received streams. The same problem is encountered for participants connected to the IRI-h session through a gateway. 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.

6.2. Gateway Components and Functionality

IRI-h services can be classified in terms of network bandwidth consumption into *light-weight* (pointer, and annotation), and *heavy-weight* (video, tool sharing, and audio) services. *Light- weight* services might 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-serviced participant. No knowledge or processing of the data stream content is required. Hence, such services require a generic tunneling gateway server. *Heavy-weight* 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 [2]. Heavy-weight 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 data rate limiting 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? Recently, [1] introduced a receiver driven bandwidth adaptation strategy for such gateways. Meanwhile, [12] suggested a quality of session control layer for interactive multimedia sessions.
- Is the transcoding process required, or can the data rate limiting operation by itself, perform the required task of adapting the incoming data stream to lower bandwidth requirements?

7. Conclusions

Although the current version of IRI is by all measures a success, we recognize that extending it's reach to a wider audience will require fundamental changes in the way the services of the IRI system are delivered and managed. We believe a key feature of the new version of IRI will be 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 have described a prototype implementation in JAVA which 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 we listed at the beginning of the paper. We have shown and tested the system on a variety of platforms simultaneously and we have done initial scaling experiments with close to fifty nodes with no discernible performance degradation while using heavy tools to share among the users. Late join and leave have been completely solved and reliability of the system has improved dramatically. Situational awareness has been addressed only with regard to participants knowing the performance aspects of the system, we have not yet addressed the issue of providing learning context to all beyond the services 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 heavy services and that of 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 then is currently possible in the current IRI system.

References

[1] E. Amir, S. McCanne, and R. Katz, "Receiver-driven Bandwidth Adaptation for Light-weight Sessions", in Proceedings of ACM Multimedia 97, Seattle, WA, Nov. 1997.
[2] E. Amir, S. McCanne, and H. Zhang, "An Application Level Video Gateway", in Proc. ACM Multimedia '95, San Francisco, CA, Nov. 1995.
[3] S. Deering, "Host Extensions for IP Multicasting", RFC 1112, Aug. 1989.
[4] A. J. González, H. Abdel-Wahab, and C. Wild, “Lightweight Scalable Tool Sharing for the Internet”, March 2000, submitted to the 7th International Workshop on Interactive Distributed Multimedia Systems and Telecommunication Services, IDMS 2000.
[5] Sun's Java Home Page, available at http://java.sun.com.
[6] Sun’s JMF's Home Page, available at http://java.sun.com/products/java-media/jmf/index.html.
[7] Sun’s JSDT's Home Page, available at http://java.sun.com/products/java-media/jsdt/index.html
[8] K. Maly, H. Abdel-Wahab, C.M. Overstreet, C. Wild, A. Gupta, A. Youssef, E. Stoica and E. Al-Shaer, "Distance Learning and Training over Intranets", IEEE Internet Computing, Vol 1, No. 1, pp. 60-71, 1997.
[9] J. B. Postel (ed.), "Transmission Control Protocol," RFC 793, Sept. 1981.
[10] H. Schulzrinne, et al., " RTP: A Transport Protocol for Real-Time Applications", RFC 1889, Jan. 1996.
[11] E. Stoica, H. Abdel-Wahab, and K. Maly, “Synchronization Algorithms for the Playback of Multiple Distributed Streams”, in Proceedings of the 4th International Conference on Multimedia Modeling, pp. 143-158, Singapore, Nov. 1997.
[12] A. Youssef, H. Abdel-Wahab, and K. Maly, "The Software Architecture of a distributed Quality of Session Control Layer", in proceedings of the Seventh IEEE Symposium on High Performance Distributed Computing (HPDC-7), Chicago, IL, July 1998.
[13] B. Whetten, T. Montgomery, and S. Kaplan, “A High-Performance Totally Ordered Multicast Protocol”, Theory and Practice in Distributed Systems, Springer Verlag LCNS 938, Sept. 1994, pp. 33-57.