The Essential Elements of Interactive Multimedia Distance Learning Systems


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

{maly, wahab, cmo, wild, hamid, ghanem, farag}@cs.odu.edu

ABSTRACT

In recent years, we have seen the introduction and use of many distance learning systems. Some of these systems are characterized as Interactive which means the dominant mode of instruction is live or synchronous using networked multimedia technology such as audio, video and shared workspaces. In this paper, we present the common features and essential elements that should be implemented in such systems. Throughout this paper, we will use as a model and case study the IRI-h system (for Interactive Remote Instruction-heterogeneous) that we have developed and implemented in Java and have used to support distance learning at Old Dominion University.

1. INTRODUCTION

In recent years, interest in using the Internet and the World Wide Web as tools for distance learning has increased. Many of these systems are based on an asynchronous learning paradigm which makes course contents available on the web; students learn the material on their own pace, they may seek help from the instructor using email and various multimedia teleconferencing tools (e.g., [4,5]). Other systems are based on a synchronous learning paradigm where the instructor and the students meet at the same time, but not necessarily in the same room and use the Internet as the primary means of communications [4, 10, 14]. An example of such synchronous systems is the Interactive Remote Instruction (IRI) project that has been developed and used in the Department of Computer Science at Old Dominion University over the last six years [10,11]. The IRI system offers a synchronous virtual classroom environment, with audio, video, and tool sharing capabilities. Past experience with the original IRI system [10] has enabled us to identify several inherent deficiencies that limit its large-scale deployment. The main identified deficiencies were platform dependence, limited scalability, and the need for a homogeneous controlled network environment. The need for a multi-platform, multi-network environment scalable system prompted us to embark on the design and implementation of a new IRI system which we termed IRI-h [6,11]. The “h” in the acronym stands for heterogeneous; it distinguishes the fact that IRI-h is designed to run on heterogeneous platforms and within heterogeneous network environments. An IRI-h prototype [11] was fully implemented in Java [7] and has been tested on multiple platforms including PCs running various versions of Windows and Unix machines running the Solaris operating system. In addition, the developed prototype has been successfully used to teach a semester-long computer science course across sites 20 miles apart.
In this paper, we leverage our previous and ongoing experience in the distance learning arena to identify a set of essential features for interactive distance learning systems. We present as a case study our own IRI-h system, highlighting how most of the identified features are integrated and implemented. Section 2 discusses a set of features that we believe should be available in any interactive distance learning system. Section 3 presents a general design overview of IRI-h highlighting main design components and functionality, including the session participant, the session manager, application-level gateways to handle network heterogeneity, and recording and playback services to provide asynchronous learning capabilities. Section 4 summarizes how most of the essential features identified in section 2 are supported by IRI-h design and components. Finally, the paper is concluded in Section 5 along with future work.

2. ESSENTIAL ELEMENTS OF AN INTERACTIVE DISTANCE LEARNING SYSTEM

In this section, we identify a set of essential elements that we believe should be present in any interactive distance learning system. Typically, a semester long class meets as a series of synchronous 
sessions lasting an hour or so. Some of the session participants can be gathered or co-located in specially equipped sites, e.g., a university lab, or a conferencing room in a corporate network setup. Other class members participate from home or work. In terms of network connectivity, some participants can utilize high-speed scalable group communication, e.g., by being located on a high-speed multicast-capable Intranet [2] while other may be less capable in terms of the unavailability of multicast communications, limited connectivity bandwidth, or long incurred delay. It is desirable not to reduce quality of service to some participants even when others have more limited connectivity.

Based on our experience with IRI, we strongly recommend that most, if not all, of the following features should be available in any interactive distance learning system:

1. **Audio and Video Conferencing**: For the instructor and students to talk and see each other and facilitate interaction. Any participant should be able to transmit her audio and/or video to join a discussion, although desktop landscape and the available bandwidth might limit the number of simultaneous video and/or audio transmitters and may require a management scheme for controlling the audio and/or video transmission.

2. **Application Sharing**: To present and enhance class material by sharing tools. Examples include presenting class material through a Web browser or PowerPoint. Tool-sharing should allow any participant to control and manipulate the tool. For example, an engineering design tool may allow participants, one at a time, to alter the design parameters, and subsequently view the effects of their input on the design.

3. **Scalable Group Communication**: For efficient and scalable delivery of data streams to session participants. Participants should be shielded from network heterogeneity specifics through an abstracted view of session membership, regardless of the participant’s network connectivity. Some participants may rely on scalable one-to-many group communication such as multicast technology, whereas other participants may need unicast services from an application-level gateway.

4. **Application-level Gateways**: To handle network heterogeneity and accommodate less capable participants. Gateways may perform format transcoding, and rate adaptation of media streams. In addition, packet relaying may be necessary to bridge the gap between multicast-enabled sites and multicast-disabled participants.
5. **Recording and Playback**: To record and playback different media and data streams. Students may review recorded material at any time on their own pace. Playback can also take place during a live session to review material recorded in an earlier session. In addition, recording provides a backup capability in case a student is not able to attend the live lecture or due to technical difficulties was not able to join the session.

6. **A Common Shared View**: To maintain a common view of any shared activities across all participants within a session (e.g., windows content and positions, pointers, annotations). Such a feature provides the same view for all participants as dictated by the person who controls the shared view. This feature requires a layout management tool that disseminates shared view changes to all session participants.

7. **Presentation Aids**: To assist in class material presentation capabilities, such as annotation and pointer tools. Such presentation aids need to be accessible for all participants, albeit in a token-controlled manner. Annotation and pointer tools are used to annotate and point at the common shared view, hence resulting annotations, or pointer movements must be propagated to all participants.

8. **Note Taking and post-session Notes Availability**: To allow participants to privately take notes during the session, and the availability of these notes after the session termination, either locally on their private disks, or through a web browser.

9. **Monitoring and Feedback**: To monitor the status of the session and report any exceptional events. Such tools can make available to a teacher or a system administrator a snapshot of the current status of the system, e.g., the list of current participants or the quality of a participant’s video or audio transmission. Furthermore, special purpose tools such as site videos can be deployed to allow monitoring of students in remote sites. Other intelligent feedback tools can be used to automatically adapt the behavior of the system, e.g., change the transmission rate of a video stream due to network congestion.

10. **Simple Interface**: Easy to use and learn by both the instructor and the students, providing one-click access to commonly used tasks. A participant should be able to quickly learn how to manipulate such interface. This is a very subjective feature that is hard to quantify, however we listed it here to emphasize the importance of this issue to the success and wide adoption of distance learning systems.

11. **Tutoring and subgroup collaboration**: To allow students to study together in small groups, or to enable an instructor to partition a class of participants into discussion groups to work on an assignment or a term project. Most, if not all, the feature listed here should be made available to the participants of such groups.

12. **Testing and Evaluation tools**: For taking tests, submitting assignments and grading. Examples of such tools include assessment capabilities within Learning Space [9], and the Blackboard learning system [16].

13. **Administration tools**: To efficiently manage and administer the distance learning system and environment. Administration tools can be classified as offline and as in-session management tools. Offline tools are used to setup and startup a session (based on administrative registration lists). In-session tools can be used for management of an ongoing session, for example selecting which student joins an ongoing discussion, or answers a question.
3. IRI-h DESIGN AND ARCHITECTURE

Providing the features identified in section 2 in an integrated fashion requires a robust infrastructure for establishing and coordinating sessions. In this section, we describe the design of IRI-h’s software architecture as an example of a successful distance learning system capable of providing these essential features. Figure 1 illustrates a typical setting for an IRI-h session with 2 sites involved. A session consists of a single session manager (SM), and several session participants (SP). The SM is a central server that runs on a designated machine, represents a rendezvous point for SPs, and provides control information for all participants. The SP is a client running on each desktop participating in the session. A special purpose SP is deployed in site 2 to act as a Recording Replay Server (RRS). By default, IRI-h sessions are recorded for backup and to provide asynchronous learning capabilities through a playback functionality. Please refer to section 3.4 for more details on the recording and replay functionality within IRI-h. The SM operates and manages several virtual rooms. Virtual rooms can be used to subdivide SPs, for example to form special discussion groups. A SP is a member of one virtual room at one instant of time. A session can be pre-configured with several virtual rooms, and new virtual rooms can be created as needed. In addition, SPs can move from one virtual room to another. In each room, a set of services is available for SPs including audio, video, tool sharing, annotation, and pointer services. Services use shared resources that are allocated and managed through the SM, e.g. group communication channels. IRI-h services require one or more group communication channels. We classify a group communication channel as unreliable, reliable, or semi-reliable. Unreliable group communication can be provided through IP multicast [2]. Reliable group communication can be provided through a reliable multicast protocol, e.g. RMP [15]. Semi-reliable group communication can be achieved by means of a tailored retransmission protocol that provides some degree of reliability. The current IRI-h prototype implementation offers a mix of unreliable and semi-reliable group communication. Unreliable group communication is provided through IP multicast and is used for audio and video services. Semi-reliable group communication is offered through basic IP multicast, enhanced using a controlled retransmission policy, and is used for tool sharing, pointer, and annotation services. In adopting a semi-reliable group communication model for data services, we rely on an eventual consistency paradigm that improves the architecture support for heterogeneous network environments and avoids the extra overhead if a reliable multicast protocol is used otherwise.

The SP and SM exchange control information through a client-server approach in which the SP connects to SM by means of a permanent TCP connection. This connection remains active as long as the session is ongoing, and is used to convey control information between SP and SM. We have made a tradeoff between reliability and scalability by eliminating any distributed servers that maintain considerable state information about the session. A session is managed by only one SM and is responsible for distributing any control or state information to all participants. Although a central server approach for session control reduces the robustness and fault tolerance of such system, it greatly simplifies the design, and achieves the desired level of scalability of supporting a maximum of hundreds of control connections to the session manager within a session.

Some of the SPs might be located in high bandwidth multicast-enabled Intranets. Other SPs can have limited capabilities, say a student participating in the class from home or work. The latter SPs may require the services of a gateway (GW), which offers data packet relaying from multicast groups, transcoding, and rate adaptation services (see section 3.3). The SM is connected to each of the GWs involved in the session with a permanent TCP connection to ensure that such GWs remain functional during the life of the session and to pass control information from the SM to the GW. Within each site data for various services are sent through multicast channels.
IRI-h supporting servers are deployed independently of IRI-h sessions, including a Directory Server (DS) and a Notes Server. The Directory server provides lookup functionality for ongoing sessions. Upon startup, a SM is required to attempt to register with a well-known DS by establishing a permanent TCP connection. An SP later queries the DS through a transient TCP connection, selecting which ongoing session to join. The Notes Server provides a back-end server for collecting participants’ notes when a participant leaves a session and stores these notes in a class directory structure to allow later retrieval through a web interface. During the course of the session, a participant can take notes using a note-taking tool and is asked when exiting if she wishes to upload her notes to be accessible through an IRI-h web server. If the upload option is selected, a transient TCP connection is established between the SP and the notes server to transfer the notes, and create the appropriate notes directory structure.

Section 3 is organized as follows. Section 3.1 presents the session participant along with available services and session views. In Section 3.2 the session manager's software architecture is introduced, explaining how resource management, group communication, and late join are realized. Section 3.3 explains how the architecture supports network heterogeneity by offering gateway services to less-capable session participants. Section 3.4 presents the design of recording and replay services to allow for asynchronous learning capabilities.

3.1 Session Participant
Figure 2 illustrates the SP software architecture in which the SP operates several services controlled through service managers. Supported services are audio, video, site video, tool-sharing, annotation, pointer, layout management, and replay. The Audio and Video service allow sending and receiving audio and video streams. The Java Media Framework (JMF) [8] is used for both capturing and playing
the audio and video of each participant. JMF uses RTP/RTCP [13] to transmit the captured media across the network. The Site Video service allows sending and receiving a site video at a low frame rate from a machine configured to act as the video server for a site. Hence, the service’s receiver component is available at all participants, whereas, the sender component is only available at a designated video server per site. The site video sender operates at a low frame rate, e.g. one video frame per second. The Tool-Sharing service allows any participant to share applications with other participants. At the sender side, images of the windows in the application being shared are captured, compared to previous images to see if the image has changed (for removing temporal redundancy), compressed, and transmitted over a group communication channel. At the receiver side, the images are received, decompressed, and displayed. See [3] for design details and performance results for the tool-sharing engine (IPV for Interactive Program Video). The Annotation, Pointer, and Layout Management services are token-controlled and allow the current token holder to annotate on top of the shared view, move a shared pointer to any position in the shared view, and arrange all shared view windows (videos, and shared tools, etc.), respectively. The Replay functionality is managed by a Main Replay Manager which controls several individual replay services which in turn operate replay receivers that are invoked during playback of a previously recorded session (see section 3.4 for more details on the recording and playback services). Note that each recorded service requires a corresponding replay service. Currently, the recorded services are audio, video, tool-sharing, annotation, pointer, and layout management.

Figure 2: Session Participant (SP) software architecture.

A SP has two views of the current session: a shared view, and a private view. The virtual room that the SP belongs to has a shared view that is consistent across all participants in that room. Changes can be made to the shared view by the presenter, the token holder of the layout management service, which are propagated to other participants. The shared view consists of any received video windows,
annotations, and shared tools' windows. In addition to the shared view, a participant has a configurable private view of private tools such as a collection of session monitors, and a note-taking tool. Session monitors include a "Class Monitor", a "Participant State Monitor", a "Log Viewer", and a "Site Camera View". The “Class Monitor” displays information about all members currently logged in or expected to participate in the session. Class participants are known beforehand through a class enrollment database. For example, a currently logged-in participant entry would provide the participant login, used machine name, and perceived role for this participant as being a student, teacher, or administrator. In addition, the "Class Monitor" is capable of providing a view of the currently connected machines to the session manager, regardless of whether a member is logged in using this machine or not. The "Participant State Monitor" shows each service state at each logged-in machine. For example the state of a video sender could be one of three states (can’t send video, can send video and is not sending, or is sending his video). The "Log Viewer" displays messages reported by all participants. Session monitors are provided with up-to-date information affecting the state of the session through the session manager, which has a global view of all participants. The site camera view provides a "site view" through a camera at each site (site video service in figure 2).

Figure 3 is a collection of snapshots of the IRI-h interface in different session scenarios. Figure 3(a) is a snapshot of a discussion scenario where two participants’ video windows (small size) and a presenter’s video window (large size) are visible. Figure 3(b) is a snapshot of a presentation scenario where the presenter is presenting class material using a shared Web browser, a shared Unix tool (an xterm), some annotations, and a pointer. Figure 3(c) depicts a snapshot of a collaboration scenario where a simulation tool is being shared, and participants are taking turn in controlling the simulation parameters affecting the outcome of the simulation.

Figure 3(a): A shared view snapshot illustrating a discussion scenario.
Figure 3(b): A shared view snapshot illustrating a presentation scenario.

Figure 3(c): A shared view snapshot illustrating a collaboration scenario.
3.2 Session Manager

Figure 4 illustrates the SM software architecture. The SM maintains the session's state including session rooms and services running within each room. Furthermore, the SM allocates the resources needed by each service. Resources are observers, group communication channels, tokens, queue managers, or gateway servers. For example, the resources required by the video service are a group communication channel, 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. Observers are used to maintain the state of stateful services such as annotation, and are used to provide a newly logged-in participant with the relevant services’ state which helps solve the late join problem. Group communication channels are used to transport each service's data streams and are allocated by a Group Communication Server (GC server). Tokens are used within token-controlled services, and are controlled by Token Managers running within the SM machine. Queue managers are used within video and audio services to manage the current audio/video senders. Gateway servers provide data packets relaying and rate adaptation services to participants with no multicast capabilities, or limited connectivity bandwidth (section 3.3).

A “Log Server” is responsible for collecting all reported messages from all participants and writing them into a log file. Upon startup, Participants connect to it through an independent TCP connection maintained for the lifetime of the session. Each reported message has a message header that includes a timestamp, the reporting participant IP, the reporting object name/method and a message level to allow filtering of messages. Messages can be classified as error, status reporting, or program debugging messages.

Late Join

While a session is in progress, participants joining the session need to know the current shared view state such as the presence of video windows, their positions and if any annotations exist on the shared view. Note that any previous session views could have been recorded as part of a recording functionality but only the latest updated view is provided to a latecomer. Observers are threads running within the SM for pointer, annotation and layout-management services. The observer joins the service's data communication channel (unreliable channel) and receives every packet sent through the channel. Upon a late join (participant login), SM notifies all observers to send their latest state of each service. In order to avoid a startup explosion problem (many requests in a short period of time), a timer is set by

Figure 4: Session Manager (SM) software architecture.
the observer to guarantee a minimum period of time before honoring any new requests to resend the maintained state.

The pointer service requires a stateless observer that keeps only the last snapshot of the pointer state that is the last position of the pointer (x and y). New packets with newer timestamps continuously update such state. On the other hand, the annotation service requires a stateful observer that maintains the state of a whole annotation frame including any annotation objects and their respective positions within that frame. Whenever a new frame is detected, the saved frame’s state is cleared. The layout management service requires a window observer that maintains the latest boundaries (x, y, height, width) of every uniquely identified window within the shared view. New participants detect existing video and audio streams through RTP mechanisms [13] after joining the audio/video channels. Hence, no observer is required for such services.

**Group Communication**

A group communication API is designed to provide group communication capabilities to IRI-h services in a uniform and transparent manner. Transparency in this context implies that an API layer hides from upper code layers the implementation details of the communication channel. For example, the same annotation service code can be used unchanged for a multicast-enabled SP that utilizes a multicast-based group communication channel or a multicast-disabled SP that utilizes a unicast communication channel. Figure 5 illustrates the API layers used to provide group communication capabilities to IRI-h services.

![Figure 5: API layers to provide group communication to IRI-h services.](image)

As noted previously, The Group Communication Server (GC server) allocates group communication channels requested for services. A group communication channel identifier is a (textual-name, implementation-type) pair such as ("Room1-Video Group", "Unreliable Multicast"). The SM automatically generates the channel textual name, in order to ensure its uniqueness. The GC server maintains a database of mappings from a group communication channel identifier to networking entities such as a (multicast-IP-address, port) pair. The SM requests from the GC server the creation of a new group communication channel within a specific virtual room. The GC server generates the associated networking entities and saves this mapping in its database. Later on, a SP connects to the GC server using a transient TCP connection in order to request the associated networking entities of a specific group communication channel. A transient TCP connection implies that the connection is
closed after the SP receives the reply. Each service requiring a group communication channel is
guaranteed a unique (multicast-IP-address, port) pair, across all running IRI-h sessions (if any), through
the GC server allocation policy. 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 SM machine's IP address, SM server port, and the room
number.

**Token Management**

A token manager is required for each token-controlled service (pointer, annotation, and layout-
management). The SM allocates the token managers, and provides a SP with their corresponding IP
addresses and port. A SP connects to the token manager through a TCP channel. The token manager
manages one token and guarantees that only one participant holds that token at one instant of time. The
token holder is the only participant that is allowed to send data through the token-controlled service
channel(s). A *stateless token* resource is allocated for pointer and layout-management services. Alternatively, a *stateful token* resource is a token that maintains a state and sends it reliably from the old
token holder to the new token holder. It is used by the annotation service to send a frame number and a
sequence number. This information is crucial in ensuring the correct sequence of annotation packets,
since the annotation packets are sent unreliably.

**Queue Management for Audio and Video Services**

Queue-managers are data structures used by audio and video services to manage participants currently
transmitting their audio and video. This management is required because of the need to impose a
physical constraint on the number of simultaneous video streams and to limit the number of participants
joining a discussion. The physical constraint on the number of simultaneous video streams arises from
shared view landscape limitations and the need to minimize the generated video streams' bandwidth
flowing in the network. When a participant is transmitting his audio/video, his ID is kept in the queue.
When there is no room in the queue, the first participant who started transmitting is forced to stop
transmitting his audio/video and is removed from the queue. The current presenter, the layout-
management token holder, is shielded from this forced remove until he is no longer a presenter.

3.3 Application-level Gateway

In this section, we introduce the design and operation of application-level gateways that are seamlessly
integrated within the overall architecture to handle class participants with no multicast connectivity or
limited bandwidth such as home users. Each IRI-h site has one or more designated gateways. A session
manager is provided the list of currently designated gateways and establishes TCP connections to all
the identified gateways. Such TCP connections are used to pass requests from the SM to the GW and to
monitor the availability of the GW during the life of the session. The list of gateways to which a SM is
able to connect constitutes a list of currently active gateways that is supplied to a newly connected SP
as part of a participant startup protocol. When allocating a new multicast-based group communication
channel, the SM requests the creation of a corresponding gateway server (GS) from each active
gateway. Gateway servers are gateway components that can provide data packet relaying and rate
adaptation services for multicast-disabled participants or participants with limited connectivity
bandwidth.

A GW is responsible for classifying whether a SP is multicast-enabled or disabled by means of a
multicast capability test. If this SP is multicast-disabled, a *Round Trip Time* (RTT) measurement is
conducted between the SP and the set of designated GWs for this session. The purpose of the RTT
measurement is to identify which GW will service this SP, i.e. provide data packets relaying functionality to this SP. The GW with the minimum RTT is selected as the candidate GW to service this SP. Other factors affecting the candidate GW selection are GW load, and network path bandwidth between the SP and each GW. Incorporating such factors in the candidate GW selection process is left for future investigation. The SP reports back to the SM the results of its multicast test, and if multicast-disabled, the candidate GW. A SP needs to be informed if its serving gateway becomes unavailable during the life of the session. Hence, the SM uses its GW permanent TCP connection to sense if a GW becomes unavailable and informs all session participants serviced by that GW. In this case, each affected SP attempts to recover by choosing a new candidate GW.

Gateway Design

Gateway servers handle a specific multicast-based group communication channel and hence are supplied with the multicast group and port used for this channel in the high-speed multicast-enabled Intranet. When a GS receives a new datagram packet from one of its clients, it extracts the payload of the packet and encapsulates this payload into datagram packets destined for each individual serviced client except the originating client. Such datagram packet processing is the same whether the data packet was received from a multicast group or from one of the SPs currently serviced by this GS. In addition, this approach does not preclude the GS from serving other SPs that are multicast-enabled but only require rate limiting or media transcoding.

Tool-sharing, layout management, pointer, and annotation services require generic gateway servers. However, video, and audio services are RTP-based and hence require an RTP gateway server. In our current prototype, RTP gateway servers are implemented using the Java Media Framework (JMF) [8]. Adopting a relay approach creates a problem for video window identification for purposes of layout management within the SP’s interface. A video window identifier is based on the originating sender machine IP address. Obviously a video stream received at the multicast-disabled SP will be identified as originating from the GW and not from the original video sender. To solve this problem, an identified video source at the SP is inspected to determine if it is a gateway machine, and if so it is contacted to retrieve the original video sender machine name corresponding to the relayed RTP stream. To provide such lookup functionality, each GS stores the SSRC (Synchronization Source) [13] for each generated RTP stream.

Another functionality of gateway servers is to perform rate limiting or media transcoding, if required, to cope with the expected limited connectivity bandwidth for home users. For example, a video gateway server might drop video frames, or transcode the incoming video stream to a less bandwidth-consuming video format. Tool-sharing and video streams are the most bandwidth consuming streams within IRI-h services [11]. IRI-h video streams use a JPEG capture format, hence can be rate limited by dropping entire video frames. The tool-sharing data stream requires a buffering and flow-control approach in order to effectively implement rate limiting and smooth peak bit rates (bursts).

3.4 Recording And Playback Service

In this section, we present the design and functionality of the recording and playback service. Six main streams need to be recorded for proper playback: video, audio, tool-sharing, annotations, pointer, and layout management. Each stream is recorded along with the timing information needed for future synchronized playback. Figure 6 depicts the architecture of the recording service. A recording agent runs on a separate machine (the recording and/or replay server) that can be regarded as a special purpose participant. Such architecture has the advantage of enabling playback of recorded sessions
during live sessions. As illustrated in Figure 6, the recording agent spawns a different thread upon the reception of a new stream from any of the multicast channels. Each IRI-h service requires a service-specific recording agent. For each IRI-h stream, two types of files are recorded. The first contains the payload information for a specific stream while the second type is either timing files for audio and video streams or token information for token-controlled services (tool-sharing, pointer, annotation, and layout). Because storage requirements are a concern, several experiments were performed to measure typical values of storage spaces. In a typical IRI-h session, assuming the continuous presence of two videos, three audio streams, a shared application, and moderate use of pointer, annotations, and presenter services, an estimate for one hour’s storage is about 840 MB. This figure is affordable due to the reduction in storage media cost offered by current secondary storage technologies.

**Video Recording Agent (VRA):** Up to three participant videos can be active at the same time. Hence, the recording agent can receive a maximum of three simultaneous video streams. Each of the received streams is processed and converted to a format suitable for storage (a QuickTime format in our case). Other timing information is stored in index files to facilitate synchronized playback of these streams.

**Audio Recording Agent (ARA):** Unlike video streams, many audio streams can be active at the same time. For each received stream a separate audio processor is created whose function is to process the stream by converting it into a basic audio (.au) format then saving it on secondary storage. The timing files mark the beginning and end time stamps (TSs) of all received streams. All TSs are measured at the receiver side for both audio and video index files. This is mainly due to the variable times taken by each processor to be created and consequently we are recording all TSs at the receiver side.

**Tool-sharing Recording Agent (IPVA):** Tool sharing is provided by IPV [3]. The IPV recording agent listens to the specific multicast channels for IPV streams and directs any received payload to disk files.
along with the required time indexes. More than one IPV application can be shared at a time provided that each sender is on a different machine. A separate thread is created for each IPV stream to handle any simultaneously active streams.

**Pointer Recording Agent (PRA):** The coordinates of the pointer are sent periodically as packets over the multicast channel. These packets are captured by the recording agent then directed to a disk file following a specific protocol for organizing the recorded data. The pointer is a token-controlled service that only one participant can control at a time. Thus, in addition to normal pointer packets token events are recorded in order to duplicate what happened in the live session during future playback.

**Annotations Recording Agent (ANA):** Performs a function similar to the pointer agent with the difference that annotation objects (circles, polygons, text, etc.) are the recorded payloads instead of pointer packets.

**Layout Recording Agent (LRA):** The semantics of the shared view in IRI-h is achieved if all participants can see exactly the same view. To preserve that assumption during playback, windows’ location information is recorded during a live session. In addition, the layout management token events (change of presenter) are recorded since this service is token-controlled.

**Playback Service**

The replay functionality can be invoked by any session participant upon startup or later during the course of the session. Replay services differ from normal IRI-h services in that they only require the receiver component of the service and do not need a sender component (see figure 2 in section 3.1).

A participant can navigate through the recording archive and select the class, semester, and date of the session to be played back. Every class member can initiate replay with the restriction that the initiator of the replay must hold the presenter token. After browsing the recording archive and selecting a session to be replayed, the initiator participant needs to communicate the selected session to the replay machine through the session manager. At the replay server, a separate reader reads each recorded stream file, i.e., payload data and timing information, and sends the payload data over the multicast channel based on the timing information. Each reader tries to emulate the functionality of one IRI-h service and sends payload information using the same data packets’ format as live session packets. Moreover, any IRI-h visual effect that occurred in a live session is replicated in the replayed session, for instance the change of the color of the indicator light signaling whether a participant currently has the presenter token.

In order to extend late join functionality (see section 3.2) to accommodate replay services, an observer is associated with each of pointer, annotations, and layout replay services. The purpose of such an observer is to maintain the current state of the service. For example, all annotation objects sent within the current annotation frame are saved and later sent to the multicast channel when a late joiner logs in. This insures that all late joiners get the current status of all replay services. The current prototype implementation does not allow recursive playback, i.e., playback of a recorded session that contains a previously recorded session. We limit the level of playback recursion to one due to the performance capability of current machines.

Recorded sessions can be replayed during a live class, either as part of the lesson plan or in response to questions from students regarding previously covered material. Because the availability of events that are recorded along with the information streams (e.g. who has control of the token, when does a student speak, etc), it is possible to more quickly access relevant material than is possible using VCR replay technology. We are working on adding additional semantic information (such as titles of slides or URLs of browsed sites) that would allow the selection of semantically related material to be presented over all class session. In addition, one issue that we leave for future research is the
availability of class material for online access through the web using any web browser without the need to actually start an IRI-h session.

4. INTERACTIVE DISTANCE LEARNING SYSTEMS FEATURES REVISITED

In this section, we summarize how the interactive distance leaning systems’ features, identified in section 2, are manifested in IRI-h design and functionality. We attempt to clarify, where appropriate, how the availability of a feature affected the design and components of IRI-h.

1. Audio and Video Conferencing: Audio and video conferencing is fully supported in a platform independent manner. Desktop landscape and bandwidth limitations necessitated a management scheme for video transmitters (see section 3.2). Up to three video windows can be active at any point of time, with one of the video senders acting as a presenter. Similarly, an audio transmitters management scheme is imposed to limit the number of participants able to join a discussion; this is currently set to ten participants.

2. Application Sharing: IPV (Interactive Program Video) [3] offers tool-sharing capabilities. Any desktop application with multiple windows can be shared. Windows applications are shared directly. Unix applications can be shared by running on a PC through an X server. A tool-sharing engine relying on capturing shared windows necessitated a separate dedicated machine to be deployed as an “application server”. Hence, a subsequent problem presented itself in how to remotely control such dedicated machines. We devised a session-independent solution based on remote controlling the application server desktop from the teacher machine by using the remote desktop sharing capabilities in Microsoft Windows NetMeeting [12]. We are currently developing a session-integrated customized remote desktop control solution that will allow selecting tools to be shared and control sizes and positions of such tools’ windows.

3. Scalable Group Communication: A group communication API provides a middleware layer for IRI-h services (section 3.2). Where applicable, multicast communications is adopted as a scalable group communication implementation. Audio and video streams use unreliable multicast communication. However, data streams rely on a semi-reliable multicast scheme (section 3).

4. Application-level Gateways: Session-independent gateways are deployed within high-speed multicast-enabled Intranets. Availability of gateways enables the system to handle network heterogeneity, however raises several issues which we attempt to summarize along with our current solutions (section 3.3).
   - How does a Participant know about the availability of a gateway: through a session manager participant startup protocol.
   - How does a participant know if she requires services of a gateway: through a multicast capability test with the set of available gateways.
   - Which gateway serves the participant: the gateway with the least Round Trip Time (RTT) measurement to the participant.

5. Recording and Playback: IRI-h sessions are by default recorded. Recording functionality is provided through a deployed specialized session participant, the Recording and/or Replay server. In-session playback of previously recorded sessions is provided. The ability to access recorded material independent of the session, for example through web access, is left for future work.
6. **Common Shared View:** We enforce a common shared view policy in which shared view updates, such as windows movements and resizing, are propagated to other participants which requires a layout management service. Changes to the shared view are token-controlled and only allowed by the current presenter.

7. **Presentation Aids:** Annotation and pointer services are provided as part of IRI-h services. Access to both services is token-controlled. The token holder of either the annotation or pointer service might be different than the current presenter. This is useful in situations where for example a student would like to point a location in the common shared view to ask a question, while an instructor has the presenter token.

8. **Note taking and post-session notes availability:** A note-taking tool is provided with the ability to upload notes into an IRI-h repository for post-session access through a web browser. The note-uploading capability requires the availability of a back-end notes server that receives and stores the notes (section 3).

9. **Monitoring and Feedback:** Several monitoring and feedback tools are available within IRI-h. A site video tool provides a capability to monitor students in remote sites. A class monitor provides the current list of participants along with a summary of their network connectivity capabilities. A services’ state monitor report on the state of various IRI-h services at each participant, for example a transmitted video stream bandwidth, and the total video bandwidth received by a participant. A log viewer permits inspection of current status and/or error messages generated by all session participants and the session manager. Integration of intelligent feedback tools to report and adapt to network delays and congestion is planned for future work, e.g., feedback between a participant and the serving gateway.

10. **Simple Interface:** The IRI-h interface is role-based, which implies it can be customized for students or teachers and session administrators with more control capabilities for teachers and administrators, and a concise set of capabilities for students. A human-machine interface study may be required to evaluate how each type of user perceives the current interface design.

11. **Tutoring and subgroup collaboration:** Virtual rooms offer opportunities for subgroup work. Session participants can be subdivided into groups for smaller group discussions or collaborative work. Supporting virtual rooms affects graphical user interface design, group communication channels allocation, and gateway servers operation. The current IRI-h interface operates using only one virtual room. Nevertheless, the underlying group communication channel allocation process and the operation of gateway servers fully support virtual rooms.

12. **Testing and Evaluation tools:** A session-integrated testing or evaluation tool is not currently part of the IRI-h prototype. Nevertheless, the previous IRI system [10] offered an in-session exam tool by directing a student web browser to a Learning Space [9] server that hosted assignment and exams for IRI courses. A similar approach is feasible within IRI-h.

13. **Administration tools:** IRI-h administration tools can be classified as offline, startup, and in-session management tools. Offline and startup tools are web-based for “anywhere” accessibility. Offline tools allow management of various configuration files, for example, identification of available Intranet machines and servers within sites, creation of new classes along with list of participants, and access to participants’ session notes [6]. Startup tools automate the startup of a session by selecting required server machines and actually performing a startup process, where a session is initiated on participating Intranet machines [11]. In-session management tools include a call-student tool that offers the capability to remotely call upon students to join an ongoing discussion, or to present their class material, hence activating their video and audio transmission.
5. CONCLUSION AND FUTURE WORK

In this paper, we presented the basic building blocks for interactive multimedia distance learning systems as typified by the IRI project at Old Dominion University over the last 6 years. The latest version of IRI, called IRI-h is completely written in Java and provides a platform-independent virtual classroom with audio, video, tool-sharing capabilities. Furthermore, network heterogeneity is overcome by deploying application-level gateways to serve less-capable participants, in terms of multicast capability and connectivity bandwidth. In addition, asynchronous learning capabilities are provided through recording and playback services. Experiences with an IRI-h prototype demonstrated design feasibility and student acceptance, but required an expected administration overhead to setup and maintain IRI-h sessions.

Future IRI-h session management work includes implementing the virtual rooms functionality including: initial setup of a session with several virtual rooms, extending the session participant interface to allow for creation and removal of virtual rooms, navigation through a set of virtual rooms, and effect of multiple virtual rooms on existing components’ functionality such as recording, playback and gateway services. In addition, we are currently developing a customized solution for remote desktop control of the application server machine.

Future application-level gateways work includes implementing rate control and bandwidth management policies, e.g. [1]. Moreover, we will explore the use of semantic information to guide the intelligent dropping of data streams to reduce bandwidth when necessary and the addition of priorities to relayed data streams, for example prioritize more crucial audio streams over other session streams. In addition, we intend to research the effect of delays introduced by buffering and flow control on the interactivity feature. Finally, we are currently in the process of establishing a test bed to evaluate IRI-h’s performance using currently available high-speed home connections such as xDSL, and cable modem technologies.

REFERENCES


