

Reliability, Scalability and Robustness Issues in IRI*

Ehab Al-Shaer

Alaa Youssef

Hussein Abdel-Wahab

Kurt Maly

C. Michael Overstreet

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

email: (ehab, youssef, wahab, maly, cmo)@cs.odu.edu

This paper appeared in WETICE'97: the IEEE 6th Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises, held in Cambridge, MA, USA, June 1997.

Abstract

Over the past two years, we have used the IRI (Interactive Remote Instruction) system to teach several live interactive classes with students in different cities. While this system is a prototype - we are using it to better understand both system performance requirements and what tools can be effective for remote instruction and how to use them - we have used it repeatedly to teach regularly scheduled for-credit university classes. This repeated use has resulted in significant improvements in IRI's functionality, but its evaluative use in real classrooms situations has required that we address significant scalability, reliability, and robustness issues. We discuss features of IRI's software architecture and basic functionality motivated by these scalability and reliability issues.

1. Introduction

IRI (Interactive Remote Instruction) is a system under development at Old Dominion University which we are using to support distance learning - education without a central classroom. In this paper we describe the actions we have taken to enhance the scalability, reliability, and robustness of the IRI system. We begin by explaining what we mean by remote education; we then briefly describe IRI's current functionality. We emphasize that IRI is an experimental system; as we continue to use the system in real classroom situations,

it continues to evolve in functionality, performance, and reliability.

The goal of IRI is to provide a distributed "virtual classroom" so that students and teachers have effective tools for preparing, presenting and assimilating class material and can interact much as they might if all were in the same room even when they are geographically dispersed. Figure 1 shows one of the virtual classrooms which we currently use. Note that all class participants (students and instructor) have individual networked multimedia workstations (each has its own camera and microphone). IRI provides a rich set of communication tools so that teachers and students can present material (text, image or audio/video based), take notes, replay material for review or missed classes, ask questions (using either audio only or combined audio/video), take quizzes, be tutored in the use of computer-based tools (since any class member can operate any computer-based tool selected by a presenter). A more complete description is available from our Web site <http://www.cs.odu.edu/~tele/iri> and [5].

Since IRI use is focused on educational use, it provides a scope of support for both teachers and students not available in video conferencing or collaborative software systems. To support activities before class, IRI provides planning tools to facilitate collection, distribution, and orchestration of materials to be presented in individual lectures. Class members can take notes and record images during class with an electronic notebook. To support after class, students can use an IRI-provided WebBook to access any of their personal notes made in class and any public material presented in a lecture. During class presenters can "call on" any class member so that the person's video image appears on all workstations. Students can easily ask questions (using audio only or both audio and video). Class comprehension can be quickly assessed using an instantly tabulated survey tool.

Figure 2 shows a workstation screen during a class ses-

*This project is partially supported by the National Science Foundation, Sun Microsystems and COX Fibernet.



Figure 1. View of One Site of Our Virtual Classroom

sion. While a large NTSC quality image with a high frame rate of a presenter is displayed if no presentation tool is in use, in this figure the instructor video is displayed in a small window in the upper right corner and the large window is occupied instead by the presentation tool. Rather than using the IRI presentation tool, a presenter can choose to switch to any NTSC video so that VCRs, TVs, and laptops can be used. In this figure, the instructor is currently presenting a set of PowerPoint slides using the IRI Presentation Tool. He has annotated this current slide during his presentation using a stylus and also by typing some new text. Partially hidden by this tool is a Netscape window used earlier in the class. Figure 2 also illustrates IRI's support of group discussions; this image has two students participating in a discussion. The last image, in the lower right corner, flips among the various participating classrooms with a view into each room.

The experiences gained through using IRI in many classes and demonstrations has caused us to modify and enhance both external functionality (to improve usability and the richness of options available to users) and internal functionality (to improve performance, reliability, robustness, and scalability). Functionality in both areas continues to evolve as we gain more experience in its use, but most relevant for this conference are those made to improve IRI's scalability and robustness; this is the primary focus of this paper. Section 2 presents a brief description of IRI software architecture, Section 3 describes the design aspects and functionalities that support reliability, scalability and robustness of IRI system, and Section 4 presents the concluding remarks and the future work.

2. Overview of IRI Architecture

IRI's software architecture is detailed in [5]. In this section, we briefly describe the major components of IRI's software architecture, and explain how this architecture provides the functionality described in the previous section.

As shown in Figure 3, the major components of IRI are: Class Information Server, Session Control, Tool Sharing



Figure 2. User Interface Screen with Presentation Tool and Netscape

Engine, Single-user Applications, Multiuser Applications, Audio Component, Video Component, Reliable Multicast Protocol Server, and IRI GUI Interface.

The *Class Information Server (CIS)* maintains information about all IRI classes such as course information, instructors, registered students and active classes. CIS assigns a unique identifier (ID) for each active class and for each attending student. These IDs are the basis for creating multicast group addresses [5].

The *Session Control (Sess)* process is created whenever a student joins an on-going class. Sess gets the student's unique ID from CIS via TCP connection and then creates all other processes which communicate with Sess using UNIX Socket connections [7]. Finally, it creates the IRI GUI interface through which the users interact with the system. All Sess processes cooperate with each other to control and manage IRI resources (e.g., audio, video channels) in a distributed fashion.

The *Tool Sharing Engine (TSE)* of IRI, based on the XTV system [1], allows sharing of all X windows' applications collaboratively. At any given moment, only one participant (instructor or student) can interact and provide input to the application. Output is distributed to all participants and displayed locally though the X server on that workstation.

Single-user Applications are available at each host and run independently from each other. One example is the *Notebook* described above. The notebook can also be run as a stand-alone tool for editing of notes outside class. Another example of a single user application is the *Session Status* which displays information about the class to the instructor, such as current attendees and their locations.

Multi-user Applications are collaborative-aware, i.e., they know the identities of all participating users and can distinguish and accept input from all of them. Examples are the *Presentation* and the *Survey* tools. The *Presentation*

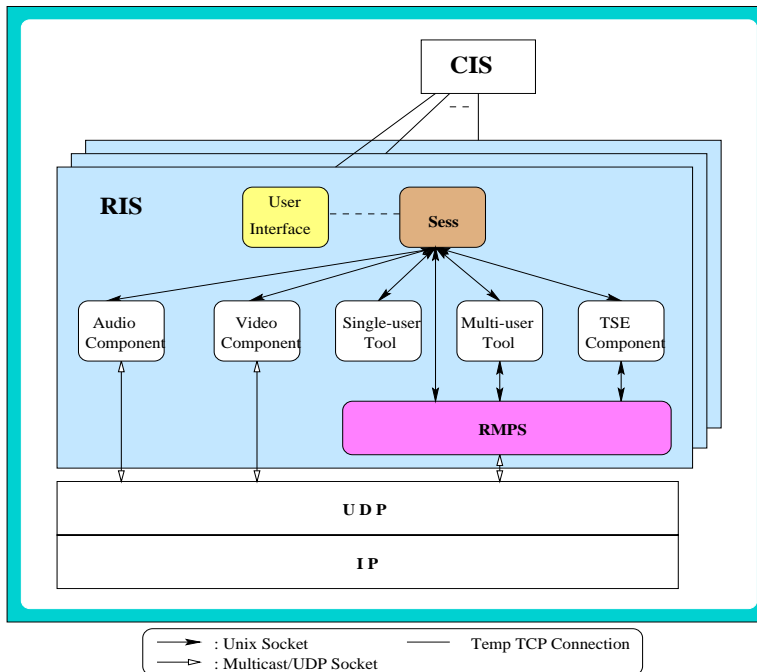


Figure 3. IRI Software Architecture

Tool allows a presenter to display slides on all workstations. The Survey Tool helps the instructor to get feedback from the students.

The *Reliable Multicast Protocol Server* (RMPS) is based on the RMP protocol developed at the UC-Berkeley/West Virginia University [8]. The Reliable Multicast server creates a light-weight process (thread) to handle the communication needs for each multicast group. The Session Control, Tool Sharing Engine and the Multi-user Applications use RMPS.

The *Audio Component* handles voice data. In IRI, one audio channel is reserved for the instructor to use at any time and a group of audio channels is shared by all students. We use audio silence detection for allocating/deallocating the tokens for the audio channels. An audio mixing algorithm is used for playing the audio data received from all channels. At each site one machine (called audio server) sends the mixed audio to the site speaker. To avoid feedback, audio is not played in the originating room.

The *Video Component* captures and displays presenter's, students' and the classroom site images. In IRI, one video channel is reserved for the instructor, a group of video channels are shared by students, and a second group are shared by all classroom site videos. When a student wants to show her picture, the token of any available video channel is allocated and the captured image is sent to that channel. Figure 2 shows the video of the presenter/teacher, two students and the classroom site.

IP multicast is used by audio and video data. Each continuous medium channel is mapped to a unique IP multicast group. In choosing IP multicast group addresses and port numbers we use the scheme from [6] where the IP address of the host machine and the class IDs are combined [5].

IRI GUI Interface is written in Motif/Xlib. The main interface and the video windows are created and managed by the Session Control process. All IRI's resources and applications are accessed through the main interface while each application, e.g., Notebook, creates and manages its own interface.

Processes inside the same host (RIS) communicate using UNIX sockets [7] and different hosts use RMPS or IP multicast to communicate with each other.

3. Reliability and Robustness in IRI System

A major design objective is supporting reliability and robustness in IRI. We believe that reliability and robustness are key issues that may determine the success of the computer technology in distance learning applications. In traditional classrooms, lack of clarity of the teacher's voice, illegibility of the teacher's writing or getting distracting noises is undesirable (by the students and the teacher) in the education environment. Similarly, in the computer-based distance learning environment, problems such as communication failures, losing edited information or tool crashes cause a considerable inconvenience and degrade the educa-

tion process. With the remarkable advantages of interactive computer-based distance learning applications, these applications may also cause some problems that do not exist in regular classroom such as the previous examples. Moreover, the scalability of such systems represented by the large number of concurrent and distributed interactions (i.e., students interactions) and the wide geographical distribution may also increase the number of failures. Therefore, it is essential that distance learning applications provide reliability and robustness in the design and implementation to efficiently cope with problems during operation. In this section, we will discuss the three major approaches for supporting reliability and robustness in IRI.

3.1. Reliability Issues in IRI Architecture

Protecting IRI from complete (e.g., IRI crash) or partial malfunctions (e.g., a single component crash) is a major design goal. The system may fail due to, for example, software bugs, wrong hardware or software configurations or improper use of IRI (e.g. sending signal). Any one involved in a large software design knows that providing a specific level of reliability can be very difficult. In this section, we discuss the IRI design features that we have employed to improve reliability and robustness of IRI.

Multiprocess Architecture: IRI is a multiprocess architecture in which each major component (e.g., IRI tool) is implemented as an independent process. In fact, most IRI components, such as Presentation Tool (PT), can be used stand-alone. When the IRI main program starts, the Session component does `FORK` and `EXEC` for each components using the proper command argument for each. IRI components communicate with each other via UNIX socket message passing [7]. Each component (specifically Sess and RMPS) has a well-defined interface used for this purpose. The multiprocess architecture of IRI has improved IRI reliability by:

- (1) preventing failures in one component from jeopardizing other components and consequently entire IRI session.
- (2) increasing the maintainability and the testability of IRI by eliminating any shared global state and decoupling the implementation and the modifications among IRI components.

Multithreaded Architecture: The RMPS component is a service layer built over the Reliable Multicast Protocol (RMP) [8] to provide a reliable multipoint communication service to IRI components. The use of RMPS shields IRI components from the details of controlling RMP directly. RMPS is designed as a multithreaded application where each LWP (Light Weight Process) or thread is completely independent from the others. This means no shared access

exists and consequently no locking scheme is necessary. IRI components that use reliable multicast connect to RMPS through two UNIX socket connection called channels: *control channel* and *data channel*. The control channel is used to send out-of-band control information such as requesting the membership status. The data channel is used to send and receive data to and from the multicast groups respectively.

As soon as the UNIX connection is established, RMPS spawns a thread called `RMPThread` to serve the corresponding requester and goes back to accept any new connections. The `RMPThread` is created upon request and lasts for the duration of the requesting component. From this point, the multipoint communication activities between `RMPThread` and the IRI component goes through the established channels. The multithreaded design of RMPS has many reliability advantages: (1) it alleviates the overhead of managing multiple processes such as frequent context switching that may degrade the performance, (2) it delivers control information as out-of-band messages via the control channel which improves the robustness and the reliability of IRI since detection and reaction to run-time problems can be performed immediately and independent of other transmission activity, and (3) from our experience with the current version of RMP (RMP 1.3b), we found that RMP has some run-time problems that could cease an IRI session. The multithreaded design principle in RMPS assists in isolating RMP failures that could occur in an `RMPThread` from affecting the other `RMPThread(s)` and consequently protecting their components.

3.2. Automatic Fault Recovery Mechanisms in IRI

Despite the low failure rate of IRI components, failure detection and recovery is an essential mechanism in distance learning applications. Integrating fault recovery procedures into IRI components is important to protect IRI from run-time problems and improve the robustness of IRI system. Different IRI components may require different fault recovery mechanisms. In this section, we will describe the fault recovery mechanisms designed for IRI components.

3.2.1 Fault Recovery in Sess Component

Since the Sess component is the core of IRI's architecture, protecting Sess processes is fundamental to IRI's fault recovery mechanism. In this section, we discuss failures that can jeopardize Sess operation. As discussed previously, RMP version 1.3b has some problems which can cause all multicast groups (`RMPThread`) attached to a specific component to halt. These faults are usually identified in Sess by "Failure Detection" messages sent by `RMPThread` to Sess

(or the attached IRI component). One source of such failures is the abnormal exit of a member while another member is still sending. In the following, we describe the automatic fault recovery algorithm for Sess component:

1. Each Sess process establishes an additional Data Channel (DC) with RMPS called “Emergency Control Channel” (ECC). Each ECC is attached with a separate RMPThread and therefore will not be affected by faults which occur in other RMPThreads including those attached with the Primary Data Channel (PDC) in Sess.
2. When a “Failure Detection” message is received or the transmission timer expires in a Sess process, then:
 - (a) The Sess process disconnects the current PDC by dropping the attached RMPThread and requests a new PDC and RMPThread.
 - (b) Also, the Sess process uses the ECC to multicast a control message declaring a global communication error. Other Sess processes will automatically drop the RMPThread of PDC (i.e., disconnect from the group) and re-establish a new PDC for the same group name with a fresh RMPThread.
 - (c) When the Sess of the teacher receives the failure declaration via ECC, it immediately informs the teacher that a Sess automatic recovery procedure has started.
 - (d) If any other Sess members discover a communication fault after receiving a failure notification by a short time¹, this Sess suppresses this failure notification because it must correspond to the same failure.
3. The Sess process receives a notification when each Sess process joins the group; this information is provided by the RMP itself. When the Sess process receives the first join notification, it sets another time-out timer called the group timer.
4. The Sess process waits until all members join the group or the group timer expires:
 - (a) If the group timer expires, then the Sess process sends a *selective multicast* message via ECC to the non-responding Sess processes. The selective multicast messages are only processed by members whose names are listed in the message and ignored by the others. A new time-out timer (called response timer) is set for retransmission this message.

- (b) If the response timer expires and some member has not yet joined the group, the previous step is repeated for several times (we have found 3 times to be acceptable).
- (c) If some members have still not joined but the Sess of the teacher has already joined, then the newly reformed group consists of the current members.
- (d) If the teacher’s Sess did not join the new reformed group, the student is notified and IRI is terminated.

5. The reformed group now resumes sending/receiving messages (using the new PDCs) and the teacher receives an instantaneous status report of the recovery process.

The worst execution time for this recovery algorithm is $3 * GroupTimerValue + MaxErrPropagationTime = 69$ seconds. However, when all machines are alive and the network is not partitioned, Sess is completely recovered in few seconds.

3.2.2 Automatic Local Fault Recovery

In this section, we discuss the failures of IRI components that can be sufficiently recovered by including only the local machine that has encountered a problem. We call this kind of recovery a *local fault recovery*. Examples of such failures includes: crashing of local processes such as audio send/receive, video send/receive and the presentation tool. For such failures, the local Sess process will immediately detect the crash of such components because it will receive a UNIX `SigPipe` on the corresponding socket. In case of audio or video processes crash, the local Sess will automatically `fork` and `exec` a new audio and video processes which immediately joins the multicast group and resume working successfully. Similarly, if the presentation tool crashes the local Sess will restart a new presentation tool automatically with a *recovery mode* set on and release all the resources held by this presentation tool such as the token. When the presentation tool is restarted, it will contact the presentation tool of the token-holder (since this information is available in Sess) to get the slide information including existing annotations. If no presentation tool has the token, the restarted presentation tool will send a selective multicast requesting the slide information from any member in the group since, in this case, any presentation tool might have the recent updates. To avoid the reply implosion, only the group member with the lowest member identification number replies to this request by sending the slide number and its update. The member identification numbers are assigned by the CIS and the Sess keeps track of the lowest identification number at any given time. After

¹About 60 seconds which is the maximum error propagation time in the group.

the presentation tool receives the reply, it resumes working in a normal mode (e.g., getting the token or sending annotations). This requires that presentation tool has the capability of saving and restoring the state of the current page, similar to Wb in [4]. However, the presentation tool has to restore the slide identification (i.e., name) instead of the contents of the slide as in Wb. It is important to know that PT was designed as a stateless tool as described in [3].

In case of failure, a status report is sent to the teacher stating the problem, the machine name and the recovery results. This information is important for the teacher to recap and compensate for the disconnected time, if it is necessary. In practice, the process time of this type of fault recovery does not normally exceed a few seconds.

3.2.3 Automatic Global Fault Recovery

Global fault recovery in IRI is required because some faults may affect most or all virtual classrooms. An example of this type of error is the communication failures described in the Section 3.1 (e.g., "Failure Detection" RMP message). When an IRI component such as TSE receives a communication failure from RMPS, it informs its local Sess which then multicasts this information to all other Sess(s). As soon as Sess processes receive the global error notification, each Sess terminates and restarts its local TSE tools. The termination of the TSE is important to avoid state inconsistencies within the system. The restarted TSE allocates a new RMPThread joining the same multicast group. The same global recovery is used if one or more TSE components crash because the late join is not yet supported in TSE as opposed to the presentation tool. The teacher is also notified when the failure is detected and after the recovery is completed. The average process time for this recovery as measured experimentally is 3 to 6 seconds.

3.3. IRI Management Tools to Support Reliability

IRI management tools have been developed to inspect the system components and report their status at run-time. Examples of IRI management tools include class management, class feedback meter, automatic saving, automatic restart/shutdown. In [3], we describe the IRI management tools and explain their functionalities.

4. Conclusion and Future Work

In this paper we focus on a crucial issue in the design of interactive distance learning systems: the support of reliability and robustness. Reliability and robustness are key issues that will affect the success of the use of computer technology in distance learning applications. We discuss

features of IRI's software architecture and basic functionality motivated by these robustness and reliability goals. In IRI, three major schemes are jointly employed to achieve these goals: (1) a multiprocess and multithreaded architecture that isolates individual components of the system from each other and prevents failures in one component from propagating to another; (2) group of management tools that automate the environment setup and startup of IRI sessions, and provide for on-line inspection of the status of different components; and (3) fault recovery mechanisms that allow for automatic detection and recovery from local or global failures. The application of these schemes in IRI resulted in significant improvements in IRI's reliability and robustness.

Our future work in this context includes implementing and using the on-line monitoring architecture proposed in [2] to observe the behavior of IRI at run-time and to provide status information necessary to improve the reliability and robustness of IRI. The main distinctive feature of this new approach over the presented ones is the flexibility and the dynamism provided by the monitoring architecture.

References

- [1] H. Abdel-Wahab and M. Feit. Xtv: A framework for sharing x window clients in remote synchronous collaboration. *Proceedings, IEEE TriComm '91: Communications for Distributed Applications & Systems*, pages 159–167, April 1991.
- [2] E. Al-Shaer, H. Abdel-Wahab, and K. Maly. High-performanced monitoring architecture for large-scale distributed systems using event filtering. *CS&I'97: Third International Conference on Computer Science & Informatics*, 3:42–46, March 1997.
- [3] E. Al-Shaer, A. Youssef, H. Abdel-Wahab, K. Maly, and C. Overstreet. Reliability, scalability and robustness issues in iri. Technical report, TR-97-27, Computer Science Department, Old Dominion University, March 1997.
- [4] S. Floyd, V. Jacobson, S. McCanne, C.-G. Liu, and L. Zhang. A reliable multicast framework for light-weight sessions and applicati on level framing. In *Proceedings of ACM SIGCOMM'95*, pages 342–356, October 1995.
- [5] K. Maly, H. Abdel-Wahab, C. M. Overstreet, C. Wild, A. Gupta, A. Youssef, E. Stoica, and E. S. Al-Shaer. Interactive distance learning over intranets. *Journal of IEEE Internet Computing*, pages 60–71, Feb. 1997.
- [6] S. Pejhan, A. Eleftheriadis, and D. Anastassiou. Distributed multicast address management in the global internet. *IEEE Journal on selected areas in communications*, pages 1445–1455, Oct. 1995.
- [7] W. R. Stevens. *TCP/IP Illustrated, Volume 3: TCP for Transactions, HTTP, NNTP and the UNIX Domain Protocols*. Addison-Wesley, 1996.
- [8] B. Whetten, T. Montgomery, and S. Kaplan. A high performance tot ally ordered multicast protocol. *Theory and Practice in Distributed Systems*, 1994.