DOCUMENT RESUME

ED 427 725 IR 019 276

AUTHOR Ndinga, S. S.; Clayton, P.
TITLE Distance Education Infrastructure for Rural Areas Using Java as a Development Tool.
PUB DATE 1998-11-00
NOTE 6p.; In: WebNet 98 World Conference of the WWW, Internet and Intranet Proceedings (3rd, Orlando, FL, November 7-12, 1998); see IR 019 231. Figures may not copy clearly.
PUB TYPE Reports - Descriptive (141) -- Speeches/Meeting Papers (150)
EDRS PRICE MF01/PC01 Plus Postage.
DESCRIPTORS *Computer Assisted Instruction; *Computer Mediated Communication; Computer Software Development; *Computer System Design; *Distance Education; Foreign Countries; Higher Education; Information Technology; Navigation (Information Systems); Online Systems; *World Wide Web
IDENTIFIERS Client Server Computing Systems; Interactive Systems; Java Programming Language; Rhodes University (South Africa)

ABSTRACT

New information technology is rapidly becoming part of the localized education process, while offering the tools and the infrastructure for the establishment of a distance education process. At Rhodes University (South Africa), an Interactive Remote Tutorial System (IRTS) was built to support distance education. IRTS will be used as an instructional medium to meet the needs of the learner in a manner that is instructionally effective and economically prudent. This paper describes the design and implementation of IRTS as a World Wide Web-based distance education delivery system that allows an instructor sited remotely and connected via normal phone lines to provide two-way voice communication within a class and remotely navigate Web-based lesson material. Topics discussed include: (1) background on distance education technologies; (2) the IRTS architecture, including the MiniNavibar, Whiteboard, Chat-box, and Event Manager; (3) the collaboration client server design; (4) the relay servers; (5) the Event Manager, including flattening/sending events and reconstructing/consuming events; (6) lesson management; (7) hardware and software requirements; and (8) future work. A figure presents the overall system architecture. (Author/AEF)

Distance Education Infrastructure for Rural Areas Using Java as a Development Tool

S.S. Ndinga and P. Clayton
Department of Computer Science
Rhodes University, South Africa
Grahamstown, 6139
{cssn, cspc}@cs.ru.ac.za

Abstract

New information technology is fast becoming part of the localized education process, whilst offering the tools and the infrastructure for the establishment of a distance education process. At Rhodes University, we have built IRTS (Interactive Remote Tutorial System); a computer based system to support distance education. IRTS will be used as an instructional medium that would meet the needs of the learner in a manner that is instructionally effective and economically prudent. In this paper we describe the design and implementation of IRTS as an web-based distance education delivery system, which allows an instructor sited remotely, and connected via normal phone lines, to provide two-way voice communication within a class and remotely navigate web-based lesson material.

Key words: distance education, shared application, remote control, and web-based instruction.

1. Background

Within the context of rapid technological change and shifting market conditions, many educational institutions are challenged with providing increased educational opportunities. Most of these institutions are answering this challenge by enhancing their education process and adding new technologies within their distance education programs. At its most basic level, distance education takes place when distance and technology separate a teacher and students. These span the technologies of:

Voice: Instructional ***audio*** tools, which include interactive technologies of telephone and audio conferencing.

Video: Instructional video tools, which include still images such as slide presentations and video conferencing.

Data: Computers send and receive data electronically.

Print: This is the fundamental element of distance education information systems and the basis from which all other technologies have evolved.

The use of these technologies within distance education is driven by opportunities to:

- reach a wider student audience
- meet the needs of students who are unable to attend campus classes and
- involve outside speakers who would otherwise be unavailable

With the above technologies in mind, the Leather Industries Research Institute (LIRI) of Rhodes University, which has long been involved in distance education in the form of **Print**, decided to use the Internet as one of their delivery mediums. The main drawback with using the Internet was bandwidth, since most of their students are spread across the countries of Southern Africa and are connected to the Internet with low bandwidth, if at all. To address this shortcoming, we built a point-to-point Interactive Remote Tutorial System (IRTS), which allows remote navigation of HTML-based lessons. The Web-based lesson material is pre-loaded onto the local and remote computers. IRTS is a client-server Java application, which can be used from Java-enabled Internet browsers such as Netscape Navigator and Microsoft Internet Explorer. IRTS consists of: a MiniNavibar which is a graphical interface that allows local and remote navigation of HTML slides; a chat environment where participants can type and exchange textual conversations; and a shared whiteboard that allows teacher and student to sketch diagrams or drawings simultaneously.

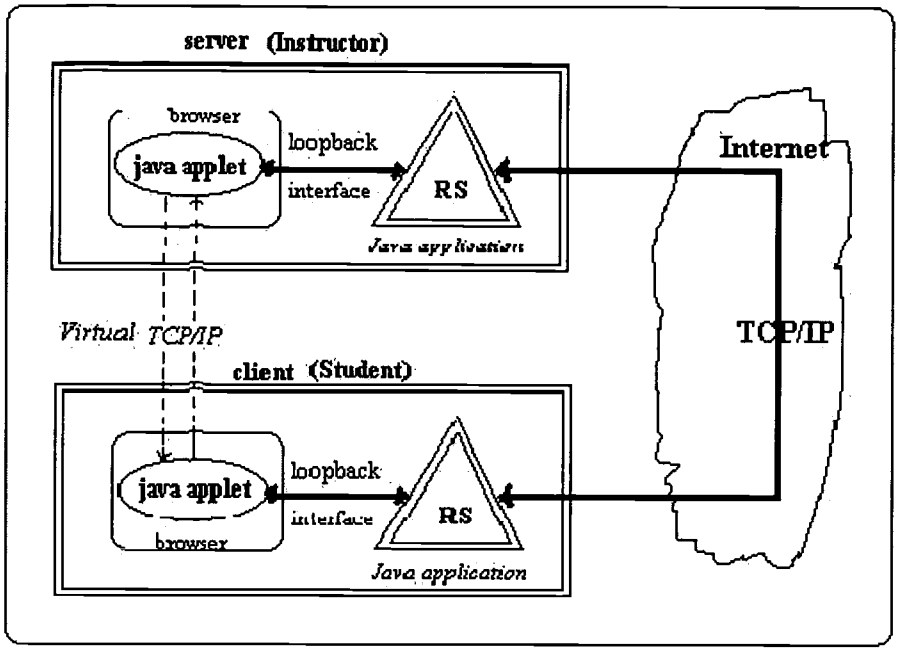

Figure 1. Overall System Architecture

2. IRTS Architecture

Figure 1 depicts the overall system architecture of IRTS, the relationship and communication paths among the two participating parties in a tutorial session. Our model takes advantage of the client – server architecture of the Internet [2], in which a client represents the student's corner and the server being the instructor's corner. As shown in Figure 1, IRTS consists of two components: a Java applet and a Relay Server (RS) for both client (Student) and server (Instructor).

The Relay Servers (RS), shown on Figure 1, are stand-alone Java applications, which have one distinctive function of forwarding all messages from the client applet (students) to the server applet (server) and vice versa. The dotted lines between the two applets represent a two-way TCP/IP communication between the instructor and the students, but due to the security restrictions imposed by Java (i.e. an applet cannot establish a network connection to any server other than its host machine), we had to implement two Relay Servers in order to establish a TCP/IP link between the client and the server. Another factor contributing to the design of the Relay Servers was the fact that the lesson material has to be pre-loaded locally onto the client and server applet's machines, before each tutorial session. Thus, during a tutorial session both applets will access the lesson material from the local disks. The applet acts as Tutorial Session Manager and it provides the user with a graphical interface with options to start or quit a tutorial session. It is composed of four components that run as separate threads of execution. These include a MiniNavibar, a whiteboard, a chat-box, and an Event Manager.

- **MiniNavibar**: This is for controlling local and remote simultaneous display of slides between the instructor and the students. Its navigation panel consist of a button bar with *Home*, *Next*, and *Previous* buttons. When IRTS is started, both the client and the server applets instantiate a copy of MiniNavibar and is displayed throughout the tutorial session.
- **The Whiteboard**: This is simply a "computer" whiteboard with tools for sketching diagrams or drawings. The whiteboard can only be started and destroyed by the instructor and it is opened as a separate window, which can be minimised and maximised. When the instructor opens the whiteboard, a copy of it is automatically opened on the student's computer.
- **The chat-box**: It is implemented as a client – server process, where the client thread runs on the student's machine while the server process runs on the instructor's machine. It also is automatically instantiated when an ITRS session is established and it runs as a low priority thread
- **Event Manager**: This forms the core of the collaboration mechanism between the instructor and the students. It is started automatically every time a tutorial session is established. The Event Manager

consists of two processes: an event *Sender* and a *Consumer* .The *Sender* is responsible for sending the mouse events and other IRTS related messages among the participating parties. The *Consumer* is responsible for receiving these events and other messages from the sending application.

3. The Collaboration Client-Server Design

In our system both the client applet and the server applet share the same MiniNavibar and whiteboard graphical interfaces. We implemented a technique to intercept all user's MiniNavibar and whiteboard events, and send them to the other participant during a tutorial session. This technique is discussed in section 5.1 and 5.2. All the GUI components defined in the MiniNavibar and the whiteboard from both applets implement this technique. When starting a tutorial session, the first task of IRTS is to establish a network connection between the two applets. Since the applets reside locally on the remote machines and are not allowed to directly make network connections, each applet (i.e. instructor's applet or student's applet) establishes a connection with its RS using the loop-back interface. This loop-back interface prevents unnecessary traffic and limit bandwidth utilisation on the LAN.

Once these inter-connections have been established by both participating parties, the two Relay Servers, i.e. the client RS and the server RS, establish a TCP/IP link with each other. The solid line in Figure 1 represents this link. When the clientRS-to-serverRS connection has been established, events and any other messages can be sent and received between the two applets. During a tutorial session both the instructor and the students can remotely control the display of the slides using MiniNavibar. But, the instructor has higher authority over the slide presentation, hence the instructor can disable

When a user interacts with a GUI component of either the MiniNavibar or the whiteboard in IRTS, the Java component Event-handlers, *handleEvent()* and *action()*, are called. We then override these two methods such that a *Sender* method, which sends the event generated to the corresponding participant, is called every time the two event-handlers are called. The structure of the event is flattened to a string representation before it can be sent across the network, to be received in the corresponding applet by a thread in a busy-waiting loop, receiving events and other

the student's MiniNavibar and thus gaining total control over slide presentation.

4. The Relay Servers

The major function of the Relay Servers implemented in IRTS is to act as distributors of IRTS messages among the two session participants i.e. student applet and the instructor applet. The IRTS Relay Servers make use of two communication channels:

1) Client/Server Applet–to-RS channel
   This channel is responsible for transporting messages between the client/server applets and the Relay Servers. It uses the loop-back interface.

2) Client RS –to- Server RS channel
   This channel uses TCP/IP for communication between the client RS and the server RS.

Each RS has two threads that loop continuously: one for forwarding messages between the client/server applet and the RS, and another one for forwarding messages between the two Relay Servers. The packet format to exchange information implemented by IRTS is:

length	*packet type*	*data*

The packet type field represents the following event classes:

START	Start a new tutorial session
QUIT	End the current session
CYBER	A miniNavibar message
CHAT	A chat-box message
DRAW	A whiteboard message

5. Event Handling

IRTS related messages. The received event is then reconstructed from the string representation and the target component is located and is finally posted by the receiving applet. When the event is posted in the receiving applet, the Java event-handlers are called as if it was a normal local user-generated event. The MiniNavibar and the whiteboard events that come from the network are tagged, such that they cannot be re-sent to the sending applet. This technique prevents an infinite loop of sending the same event

between the instructor and the student machines.

5.1 Flattening and Sending Events

Before an event is sent by the *Sender* process of the Event Manager, it has to be converted to a representation suitable for transportation over the network. This representation has to contain sufficient information for the event to be easily reconstructed and posted to the correct target component by the *Consumer* process of the receiving application. The attributes of a Java event object are shown on Table 1. With bandwidth restrictions in mind we minimised the information on our event representation as much as possible without sacrificing too much information. We did not incorporate the *clickCount, when,* and the *evt* fields in the representation. In our string representation of the Java component event, the remaining fields get their designated positions separated by a colon (:). Each event is flattened by concatenating the string representation of each field and sending the resulting string in the *data* field of IRTS packet format. The resulting string representation is shown below.

Attribute	Description
id	Type of the event
x	x coordinate of the event
y	y coordinate of the event
key	Key pressed in a keyboard event
clickCount	Number of mouse clicks
modifiers	State of the modifier keys
when	Time stamp
evt	Next event
arg	An argument of the target component
target	The target component of the event

Table 1. The attributes of a Java event

```
id:x:y:key:modifiers:arg
```

5.2 Reconstructing and Consuming Events

When the Event Manager's *Consumer* process of the receiving application receives a message that originated from the chat-box, it extracts the data from the *data* field of the packet. It then forwards this data to the chat-box process of the receiving application, which then displays the data on the chat-box's Text Canvas. On the other hand, if the message originated from either MiniNavibar or the whiteboard, the *Consumer* extracts the event and forwards it to either MiniNavibar or the whiteboard. The event is then reconstructed and posted to the target component of the receiving application. For the event to be posted, the matching target component must be located. In Java, a GUI component has no explicit identity. In our case, we use the *arg* field of the generated event to locate the target component. This is possible because both the client and the server applets (for students and the instructor) execute the same copies of MiniNavibar and the whiteboard. For example, when the instructor presses the MiniNavubar's *Next* button on the server applet, an action event is generated. This event is then intercepted and sent to the student's computer (client applet) by the *Sender* process. This event is then received by the *Consumer* process of the client applet and the *arg* field, which is the label of the *Next* button, is used to uniquely identify the target component, which in this case will be the MiniNavibar's *Next* button of the client applet.

5. Lesson Management

When IRTS is loaded onto a computer, two named frames are created. The client/server applet is loaded into one frame and the second frame is used for displaying the lesson material. The lesson material has to be developed into a sequence of HTML slides. Since the lesson material resides on the local disks of the participating parties, the client/server applet loads the slides into an array during start-up. MiniNavibar is used for the presentation of the slides. It keeps track of the current page, and its navigation buttons simply navigate the internal tree in an obvious sequence and the relevant URL (slide) is loaded into the second frame. For example, when either an instructor or a student presses the *Next* button, the array index is incremented and causes the appropriate slide to be displayed on the second frame.

6. Hardware and Software Requirements

The hardware required for IRTS at each site is:

1.
   a) Any Java 1.0 compatible computer

2. A monitor capable of SVGA graphics, 800x600 resolution; suggested sizes are:
   a) 14 – 15" inch monitor for 2 participants
   b) 14– 17" inch monitor for 2 – 4 participants
   c) 17 – 21" inch monitor for 4 – 5 participants
   d) more than 5 participants require an extra monitor or a projection system

The IRTS software requirements include any Java capable Internet browser and Sun Microsystems JDK 1.0 SDK. The type of audio equipment needed varies depending upon the number of participants during a tutorial session, the size of the classroom venue, acoustics, and other factors. However, each site must have a full-duplex speakerphone, and a direct analogue phone line.

7. Conclusions and Future Work

The Interactive Remote Tutorial System (IRTS) presented in this paper provides a point – to-point link between two applications running on any Java-capable Internet browsers, and it allows remote navigation of HTML slides by an instructor or students sited different geographical locations. It is able to be used on very low bandwidth connections, as only the event information is parsed between the instructor and student machines. IRTS also provides extra features, which include a computer whiteboard and a chat environment. The system is currently useful for delivering distance education to remote regions, using two dial-up telephone lines (one for computer communication and the other for interactive voice communication), or a basic Internet connection and a telephone call.

Our next goal is to extend IRTS into a point–to-multipoint system. Other issues to be addressed in our future work are scalability through use of reliable multicast protocols; the integration of audio and video; and use of centralised server for keeping the lesson material.

References:

[1] P. Wentworth, "Navibar: A tool for Web Tutorial Authoring", published on the Internet at Http://cs.ru.ac.za/coe/research/navibar.htm

[2] S. S. Ndinga et al, "An Investigation into Tools and Protocols for Commercial Audio Web site Creation", to appear on the proceedings of SATNAC, September 1998

[3] B. Kvande et al, "An Internal Collaborative Environment for Sharing Java Applications", Old Dominion University, 1997

[4] H. Abdel-Wahab et al, "Using Java for Multimedia Collaborative Applications", Proceedings of the 3rd International Workshop on Protocols for Multimedia Systems (PROMS'96), October 1996, Madrid

[5] M. Campione and K. Walrath, "The Java Tutorial", at Http://www.javasoft.com

U.S. Department of Education
Office of Educational Research and Improvement (OERI)
National Library of Education (NLE)
Educational Resources Information Center (ERIC)

NOTICE

REPRODUCTION BASIS

☒ This document is covered by a signed "Reproduction Release (Blanket) form (on file within the ERIC system), encompassing all or classes of documents from its source organization and, therefore, does not require a "Specific Document" Release form.

☐ This document is Federally-funded, or carries its own permission to reproduce, or is otherwise in the public domain and, therefore, may be reproduced by ERIC without a signed Reproduction Release form (either "Specific Document" or "Blanket").