This paper appears in the Proceedings of the Seventh Youth Science Conference, 2001, Ministry of Education, Singapore.

A COLLABORATIVE LEARNING ENVIRONMENT USING JAVA SOCKETS

Lin Jiaqi[1] and Tay Seng Chuan[2]

[1]The Chinese High School, 673 Bukit Timah Road, Singapore 269735.
[2] Centre for Remote Imaging, Sensing and Processing, National University of Singapore,
3 Science Drive 2, Singapore 117543.

Abstract

This project makes use of the latest JAVA networking technology to create a collaborative learning environment for students to discuss academic problems. The participants in our environment can be distributed and they are not constrained by geographic space. Several design issues have been considered, including deadlock and livelock problems and the disparities in national culture. Our product can be used by schools for students to interact collaboratively in a virtual environment with their counter-parts in other schools and even in other countries.

Introduction

Much of the world is in the midst of a great transformation fueled by continuing advances in computing and networking capabilities. Today, the Internet and the World Wide Web, with its expanding user base, have changed the shape of modern life and altered the way people interact. Internet is in fact a global and virtual system, a powerful communication system that is open, free and non-propriety. It is also a common space in which we communicate by sharing information. From there we are able to access any sort of information, be it personal, local or global, and draft or highly polished. With this powerful communication channel comes the concept of collaborative learning. This term refers to an instructional method in which people at various performance levels and usually different geographic locations work together towards a common goal. The participants are responsible for one another's learning as well as their own. Proponents of collaborative learning claim that the active exchange of ideas within small groups not only increases interest among the participants but also promotes critical thinking.

Before the Internet was created and HTTP was specified, a collaborative learning environment seemed almost impossible because of geographic restrictions. With the introduction of ARPANET in 1968 and the establishment of TCP/IP in 1973, such restrictions are actually solved elegantly.

Distributed computing and JAVA go together naturally. As the first language designed with networking in mind, JAVA makes it very easy for computers to cooperate. In this project we make use of JAVA socket to develop a collaborative environment used for learning.

Materials and Methods

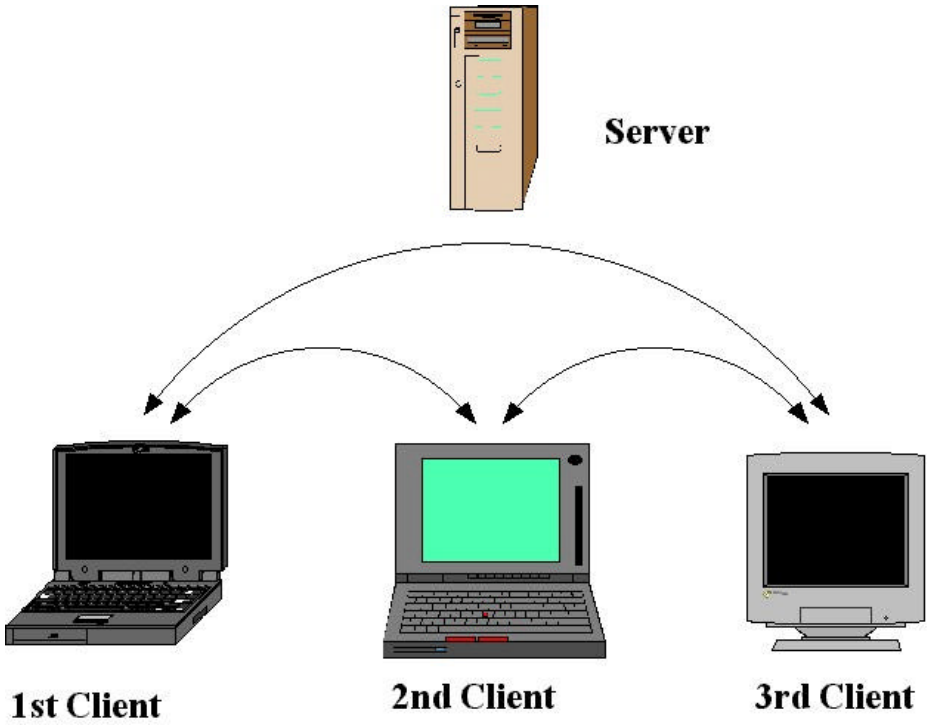
Computers can be connected using several ways including electrical wiring, special cables, and radio frequencies. The common way, however, is to use the existing telephone networking system. Using this approach, telephone network adapter cards are installed on each computer. The adapter cards are connected to the telephone wiring system using standard telephone connections. One advantage of our product is flexibility. The computers used in the collaborative environment can run different Operating Systems. To facilitate communication, a JAVA virtual machine is embedded in all machines.

Materials Used

The configuration of our product is shown in the following table. It is noteworthy that the number of clients participating in the collaborative environment is not constrained in our product.

Hostname	**Functionality**	**Operating System**	**Configuration**	**Java virtual machine**
Celerex	Server	Windows 2000 Professional	Celeron 500 with 640MB RAM	Java HotSpot Virtual Machine 1.3.1
Excellente	1st Client	Windows 98 Second Edition	Pentium MMX 200 with 32 MB RAM	Java HotSpot Virtual Machine 1.3.1
Appliz	2nd Client	Mac OS X	PowerPC 300MHz with 96 MB RAM	Java HotSpot Virtual Machine 1.3.1
Xguin	3rd Client	Red Hat Linux 7.1	Pentium III 733MHz with 128MB RAM	Java HotSpot Virtual Machine 1.3.1

A graphical representation of a collaborative session is given in the following diagram. The participants at the client computers can communicate with each other at real time. In addition, the contents of their communication will be displayed on all computers located at different geographic locations.



Methods

Our product uses JAVA sockets to make communication possible between different machines. A server runs on a specific computer and has a socket that is bound to a specific PORT number normally above 1024. The server will wait forever, listening to the socket for a client to make a connection request. For example, the server can be waiting on a computer with a hostname called localhost and waiting on PORT 3999.

The client will be required to know the hostname of the machine on which the server is running and the PORT number to which the server is connected. To make a connection request, the client tries to connect to the server using the server machine's hostname and PORT. In this case, the hostname is localhost[1] and the PORT is 3999.

Upon request by the client, the server will accept the connection. After which, the server gets a new socket bound to a different PORT. It will also require a new socket so that it can continue to listen to the original socket for connection requests from other clients while at the same time attending to the needs of the connected client.

If the server accepts a connection from a client, a socket will be successfully created and the client can use the socket to communicate with the server. Note that the socket on the client side is not bound to the PORT number used to communicate with the server. Rather, the client is assigned a PORT number local to the machine on which the client is running.

Once the handshaking protocols are established, the client and server can communicate by writing to or reading from their sockets[2]. The server can accept multiple clients by storing all the socket information into a hash table and using enumeration to multicast the information to all the clients.

Results

The following snapshots illustrate the use of our product. It is noteworthy that the collaborative session is distributed at different geographic locations.

(i) A teacher will start the server program and enter a PORT number.

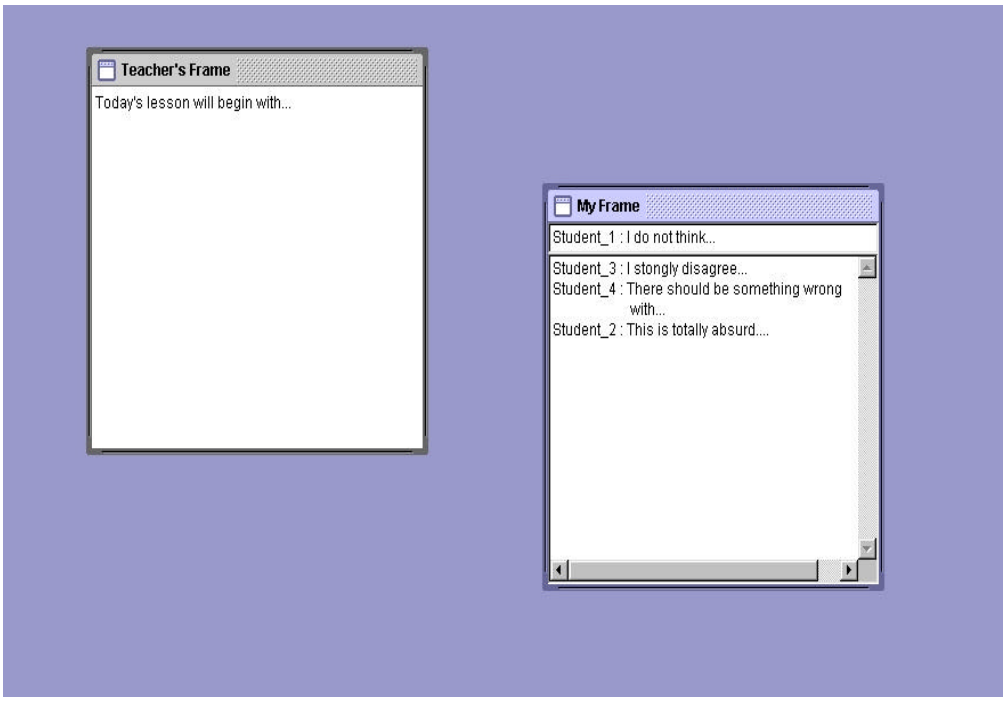
[1] localhost is usually used for debugging purpose. In the operational environment the hostname refers to the destination of the participating machine.

[2] The paradigm for data communication in JAVA is similar to file I/O operation.

(ii) User name has to be entered.

(iii) The server program can be started.

(iv) Participants at different geographic locations can start the client program and collaborative session begins.



Discussion

This project has resulted in the completion of a software used for online collaborative learning. By using JAVA socket, a cross-platform product is successfully developed. As a start, our product can be used in schools for students to interact with their teachers from all over the globe, without worrying about geographic limitations.

As this is a multi-tasking system, we have to ensure that deadlock does not occur in the collaborative environment. Deadlock is the permanent blocking of a set of processes that either compete for system resources or communicate with each other. We can show deadlock graphically by building the *waits-for* graph. This is done by drawing each process as a little circle, and drawing an arrow from *P* to *Q* if *P* is waiting for *Q*. In this project we have checked that no cycle occurs in the waits-for graph, thus preventing the possibility of deadlock. We are also aware that a deadlock-free solution can introduce the livelock problem, a situation in which the system is prevented from making progress while the processes in the system are active. Our product has gone through a rigorous integration test and is now functional. The product can be upgraded to include the transmission of video and audio files.

Acknowledgment

This project would not have been so successful if not for Dr Tay Seng Chuan at the Centre for Remote Imaging, Sensing and Processing in NUS. Dr Tay is a strict and helpful mentor who has consistently put in his effort and time to teach me advanced JAVA features and has assisted me in the product development. Even at the last phase of report writing, Dr Tay has repeatedly revised and improved the contents of this paper. I would also like to thank my school teacher, Mrs Ke Ee Kian, for coordinating with Dr Tay for this project work.

References

1. Advanced JAVA Networking, 2nd Edition, Dick Steflik, Prashant Sridharan, ISBN 0-13-084466-7, PH PTR.
2. JAVA 2 Networking, Justin Couch, ISBN 0-07-134813-1, McGraw Hill.
3. Server-based Java, Ted Neward, ISBN 1-884777-71-6, Manning.