E-CLASSROOM THAT LETS INSTRUCTORS MAKE VIRTUAL CLASSROOMS AND DISPERSE ASSESSMENTS

Aldo Stalin JL¹, Thangaraj K², Vasanthi S³, Ilanchezhian P⁴, P.Shanmugaraja⁵
¹,²Assistant Professor/IT, Sona College of Technology, Salem, Tamilnadu, India
³,⁴,⁵Associate Professor/IT, Sona College of Technology, Salem, Tamilnadu, India
aldo@sonatech.ac.in, thangarajkesavan@sonatech.ac.in

ABSTRACT

This new initiative is all about developing a mode of communication between lecturer and the audience. By using our software, you would be able to ask your doubts directly to the lecturer, if you feel awkward to ask it out directly amongst your fellow audience. The lecturer can also send messages to the audiences. While in a lecture of a large number of audiences, we may be sitting somewhere in the crowd. In that case, we will not be able to speak to the lecturer. Or, we may also feel a bit awkward to ask the doubt while you are with your friends. In that case, you can ask them directly to the person on the dais and he gets the message from the audiences. The person who is handling the class can also ask MCQs, send messages, send links and files for references to the audience and also clear their doubts. Also, the lecturer can post programming related questions to the students and provide test-cases to solve the challenges. This establishes a two-way communication between the lecturer and the audience. This paper helps both the lecturer and the audience. It helps the lecturer by making him know how well his audience have understood the concepts and also audience can clear their queries and confusions. Hence, overall, it helps in making the lecture complete and bringing the students and the lecturer closer.

Keywords—lecture, client server, communication system

I. INTRODUCTION

Computers have become a part of human’s life in today’s fast-growing world. Today, we cannot imagine living without computers. Computers are used in all the departments today. For Example: In Healthcare and Medicals, Science and Research, Education, Industries, and even in home.[2]. Today, almost all the Education institutions, Schools and Colleges have computers with them. So, the classes are taken with computers today. When the lecturer is taking class, the students may sometimes feel awkward to pause the lecturer and ask his/her doubts, in front of their fellow mates.[3][4] This lack of communication can lead to misunderstanding or confusion of the concepts.[5]

On the other hand, side, the lecturer should also know how much their students are clear with the concepts. So, he/she must question their students. In a case where there are 300-400 students, it would be very difficult to ask. So, this may also lead to mishandling of students and not taking the lecture properly.[6][7]. Our software application aims in building the bridge between the lecturer and the students, to enhance the classroom environment for the learning purpose.[8][9][10] This paper explains about the application software in detail.

Server-Client Communication

A distributed application system that divides tasks or workloads between a resource or service provider, called servers, and service requesters, called clients, is a client-server model. Clients and servers frequently communicate on different hardware over a computer network, but in the same device, both client and server can reside. One or more server programs run by a server host share their services with clients. A client does not share any of its resources, but asks a server for content or service. Therefore, clients start chat sessions with servers, anticipating incoming requests. Email, network printing, and the World Wide Web are examples of computer programs that use the client-server model. The client-server characteristic defines in an application the partnership of cooperating programs. The server portion provides one or more clients with a feature or service.
that initiates requests for certain services. Servers are categorized by the services they offer. For instance, web pages are served by a web server and computer files are served by a file server. Any of the software and electronic components of the server computer can be a common resource, from programs and data to processors and storage devices. Sharing a server's resources constitutes a facility.[1]

The essence of the program that requires the service functions determines whether a device is a client, a server, or both. For example, at the same time, a single computer will run a web server and file server software to serve various data to customers making different types of requests. It is also possible for client software to connect with server software on the same device. Server-to-server communication, such as data replication, is also called inter-server or server-to-server communication.

In general, a service is an abstraction of computer resources, and when fulfilling the request and providing the response, a client does not have to be concerned with how the server works. Based on the well-known application protocol, i.e. the content and the formatting of the data for the requested service, the client only has to understand the response. In a request–response messaging pattern, clients and servers exchange messages. The client sends a request, and a response is returned by the server. An example of inter-process communication is this message exchange. The machines must have a shared language in order to communicate, and they must obey rules so that both the client and the server know what to expect. A communications protocol specifies the vocabulary and rules of communication. In the application layer, all client-server protocols function.

The fundamental patterns of the dialogue are described by the application layer protocol. The server can implement an application programming interface (API) to formalize the data exchange even further. For accessing a service, the API is an abstraction layer. It enables parsing by limiting communication to a particular content format. It enables cross-platform data sharing by abstracting entry.

In a short period of time, a server can accept requests from many distinct clients. At any moment, a computer can only perform a small number of tasks and relies on a scheduling mechanism to prioritize customers' incoming requests to satisfy them. The server software can restrict customer availability to prevent abuse and optimize usability. Service denial attacks are designed to take advantage of the duty of a server to process requests by overloading them with unreasonable request rates. If confidential information is to be shared between the client and the server, encryption should apply.

The client initiates a submission to the bank's web server when a bank client accesses online banking services with a web browser (the client). The login credentials of the customer may be stored in a database and the web server, as a client, accesses the database server. An application server interprets the returned data by applying the business logic of the bank, and provides the web server with the output. Finally, the result is returned by the web server to the client web browser for display.

A machine processes a request in each step of this series of client-server message exchanges and returns data. This is the messaging pattern for request-response. The sequence is complete when all the demands are met and the web browser presents the customer with the details.

A design trend specific to the client-server model is demonstrated in this example: separation of concerns.

**Working of the Software**

With the Client-Server UDP Multicast Architecture as shown in the figure 1, we can send message from a number of computers to a single computer. This kind of the network communication is possible only with UDP Server-Client Model.
So, all the students can send their doubts to the lecturer taking the class as a message, and the staff can receive it and see them with the help of this architecture. Also, the lecturer can post objective questions to their students. This can also be achieved by UDP Server-Client Model.

Moreover, the lecturer can also post programming related questions to his/her students with test cases, and the students would be able to submit their programs to the lecturer. All of these can also be achieved with the help of the tools in the internet. But now, it can also be achieved within a closed network without connected to the outside world. Figure 2 and Figure 3 describes the steps and flow for the client side configuration and Figure 4 and Figure 5 describes the steps and flow for the host system.
Step 1) Start the program.
Step 2) Create Multicast socket for receiving messages.
Step 3) Get the clients name.
Step 4) Get the Sending Socket's IP address and Port Number.
Step 5) Create Sending Multicast Socket.
Step 6) Receive Network password from client.
Step 7) Ask Client to enter password.
Step 8) If both passwords match, then:
    Go to Step 8.
Else:
    Go to Step 6.
Step 9) Ask client to enter username.
Step 10) Send the username to host for verification.
Step 11) Receive verification signal from host.
Step 12) If the Username is available, Then
    Go to Step 12.
Else:
    Go to Step 8.
Step 13) Receive Messages.
Step 14) If message type is "Chat", then:
    Display the chat message on screen.
Step 15) If message type is "Participants list", then:
    Update the participants list with the received one.
Step 16) If the message type is "Question", then:
    Display the question.
    Ask the user to enter answer.
    If the answer is correct, then:
        Display "Answer is correct".
    Else:
        Display "Answer is incorrect".
Step 17) If message type is "Disconnect", then:
    If the username matches with my username:
        Send Disconnect signal to host.
    Close all the Sockets.
    Stop the program.
Step 18) Stop the program.

Fig.2 Client Algorithm
Fig. 3 Client Block Diagram
Fig. 4 Host Algorithm

Step 1) Start the program.
Step 2) Create Multicast socket for receiving messages.
Step 3) Get the clients name.
Step 4) Get the Sending Socket's IP address and Port Number.
Step 5) Create Sending Multicast Socket.
Step 6) Receive Network password from client.
Step 7) Ask Client to enter password.
Step 8) If both passwords match, then:
   Go to Step 8.
   Else:
   Go to Step 6.
Step 9) Ask client to enter username.
Step 10) Send the username to host for verification.
Step 11) Receive verification signal from host.
Step 12) If the Username is available, Then
   Go to Step 12.
   Else:
   Go to Step 8.
Step 13) Receive Messages.
Step 14) If message type is "Chat", then:
   Display the chat message on screen.
Step 15) If message type is "Participants list", then:
   Update the participants list with the received one.
Step 16) If the message type is "Question", then:
   Display the question.
   Ask the user to enter answer.
   If the answer is correct, then:
      Display "Answer is correct".
   Else:
      Display "Answer is incorrect".
Step 17) If message type is "Disconnect", then:
   If the username matches with my username:
      Send Disconnect signal to host.
      Close all the Sockets.
      Stop the program.
Step 18) Stop the program.
The figure 6 shows the lecturer login screen through which the lecturer can login to the system by giving a valid network password. Then the lecturer allowed to enter in to the question posting system. The figure 7 shows the format of Lecturer Question Posting Window.
The Figure 8 shows the lecturer monitoring window through which the lecturer can monitor the students’ queries and activities. The simultaneous questions from different students also addressed without disturbing other students as in the face to face class. The figure 9 shows the students’ login screen where using their first name and last name they can enter themselves into the system. Figure 10 shows the students’ dashboard through which they can take up their eClass activities and interact with the lecturer. Figure 11 shows the mcq test window and figure 12 shows the mcq test response window. Through the system the lecturer can address to all the students and clarify their queries and also able to test whether they understood the particular session or not.
Fig. 9 Student Login Window

Fig. 10 Student Dashboard

Fig. 11 Student MCQ Window
II. CONCLUSION

In this paper, we analysed the possibilities of achieving a software model done with the help of internet, within a closed LAN network, with only a limited number of computers restricted access to the internet and the other networks. Thus our software helps both the lecturer and the audience. It helps the lecturer by making him know how well his audience have understood the concepts and also audience can clear their queries and confusions. Hence, overall, it helps in making the lecture complete and bringing the students and the lecturer closer.

Acknowledgment

The authors would like to thank the faculty and staff of Sona College of Technology for providing them motivation, knowledge and enough facilities to do experimentation and implementation of the paper inside their campus.

REFERENCES

1. en.wikipedia.org/wiki/Client-server_model