BlueCT - The Interactive Learning Tool Using Bluetooth.

Gang Peng
Department of Computer Science
University of British Columbia
Vancouver, British Columbia, Canada-V6T 1Z4
Email: shuaige@cs.ubc.ca

Anirban Sinha
Department of Computer Science
University of British Columbia
Vancouver, British Columbia, Canada-V6T 1Z4
Email: anirbans@cs.ubc.ca

Abstract— **BlueCT is an excellent tool providing an interactive platform between students and instructors changing the traditional one-direction teaching model. With the help of BlueCT, students can answer questions and submit feedback information in real time. Instructors can be made aware of students' understanding of questions and their feelings about the class spontaneously. Unfortunately, prior to this work, there existed some shortcomings in BlueCT, such as having limited functionalities and non existence of security mechanisms. This work is aimed at fixing its shortcomings and extending its functionalities to make it a perfect interactive platform for classroom learning.**

I. Introduction

BlueCT is an excellent interactive platform that shortens the distance between students and instructors and changes the traditional one-direction teaching model. With the help of BlueCT, students can answer questions and submit feedback information online, and the instructor can post questions and check students answers and feedback messages on the fly. Thus, BlueCT can help students stay concentrated in the class and provides the students an interesting way to communicate with the instructor.

BlueCT has a Client/Server design in which the instructor starts a server on his laptop and the students use the client side software on the handset to communicate with the server. Figure 1 shows the structure of BlueCT. When a student wants to communicate with the server, he should first log in. Thereafter, he can send messages to the server to answer questions posted by the instructor or send feedback information.

Figure 2 shows the software architecture of the BlueCT system on both the server side and client side. In the server side, "MS Bluetooth Stack" is a socket-style C API which allows any application written in C to operate using a generic USB Bluetooth dongle. JSR-82 API is the Java version of the "MS Bluetooth Stack" API. The integration between the JSR-82 and the MS Bluetooth API is done via JNI, a technology used by Java applications for using C libraries. The client side software is also built on top of JSR-82. The prerequirement is that the OS on the handset must support JSR-82. We are using Nokia S6620 handset that uses Symbian OS platform and is the first handset supporting JSR-82.

Although BlueCT is an excellent teaching platform, prior to this work, it had some shortcomings that made it less perfect. Firstly, BlueCT only supported very limited functions.

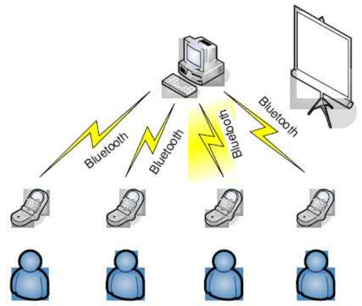

Fig. 1. The BlueCT Structure

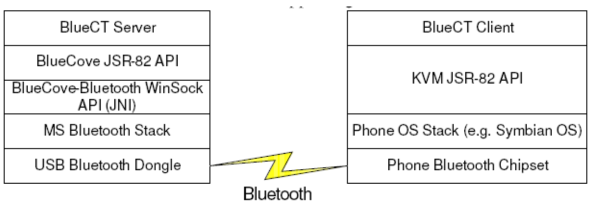

Fig. 2. The BlueCT Software Layers

For example, BlueCT only supported single-choice questions. Secondly, there were no security mechanisms in BlueCT. Students could use any login account and no password was ever required. This meant that there was no way for the instructor to map a login account to a student. Thirdly, clients could not communicate with each other, meaning that students could not discuss questions and ideas through BlueCT. In a word, BlueCT had some major disadvantages and our work is aimed at fixing these problems and in making BlueCT perfect. The rest of the paper is organized as follows. Section II deals with all the improvements we made in BlueCT and describes them in details. Section III discusses some of the experiences and challenges we faced while working on BlueCT and then it finally concludes the paper.

II. Improvements

Due to the existing shortcomings of BlueCT we mentioned above, we decided to improve BlueCT to make it a perfect interactive platform between instructors and students. Our improvements include supporting multiple choice questions,

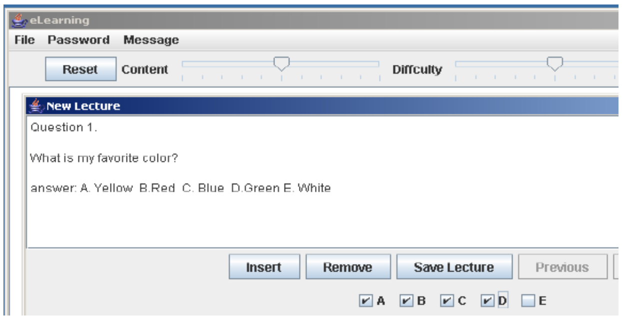

Fig. 3. The Server side GUI for inserting multiple choice questions

supporting text input, viewing active students from instructors side and students side, a "username and password" security mechanism and the communication mechanism between students. In the following sections, we describe each of them in greater details.

A. Multiple choice questions

One obvious shortcoming of BlueCT is that it only supports single choice questions. If instructors want to set up multiple choice questions, they would have to set up choices like "A, B and C" etc. We modify BlueCT to make it support multiple choice questions. This modification includes two parts. On one hand, we had to change the answer form in the client side to make it support multiple choice questions. On the other hand, we had to modify the lecture creation and lecture modification functions in the server to make the server support creation and modification of multiple choice questions. There is a little story about this modification. Because we do not want to change the original data structure that uses an integer to record the answer, we had to convert the answers in multiple choice questions to integers. We did this by treating the answers as binary codes. For example, if a student's answer is "A, B, C, E", then this would be interpreted as:

$$1 \times 2^4 + 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = 29$$

Using this method, we do not have to change the old data structure and related operations, which greatly simplifies our work. The GUI on the server side looks like Figure 3.

B. Inputting text

Another problem with BlueCT is that students can only send multiple choice answers. There is no way for students to post a question or submit some suggestions. We add the capability for inputting text into BlueCT. We add a new input form into the client and use a panel to output students input on the server side. We also change the existing "Save Student Grade" function into "Save Student Grade and Input" function, so as to save students' grades along with their input text. The GUI looks like the one shown in Figure 4.

C. Use of student authentication through username and password

Lack of student authentication is another serious problem in BlueCT. Students can use any login name and do not need to input password. There is no way for instructors to map a login account to actual student. To fix this problem, we added a new username and password mechanism into the BlueCT. The instructor can use the "Initialize Username and Password" mechanism to initialize login account and corresponding password. This function reads in a text file and treats the first word of very line as a new account username and sets the corresponding password to be same as the username. For example, if one line in the input text file looks like "Gang Peng", this function will add an account "Gang" and set the corresponding password "Gang". Both the instructor and the student who uses this account can change the password conveniently. The usernames and passwords are stored in a file "user.data", which is located in the "res" directory. This mechanism binds students with their account tightly and the instructor can easily find out the corresponding student from his account. The GUI dialog in the server side is shown in Figure 5. We also keep the capability to generate user accounts on the fly from the server side. For this, we keep a separate input dialog on the server side that takes user names and generates random ten character passwords for them. These are added into the system so that new students can be easily accommodated in the system. The GUI for this facility is shown in Figure 6

Because all these mechanisms includes several options, such as "Initialize username and password" and "Change user password", we create a new menu, "Password" in the main menu.

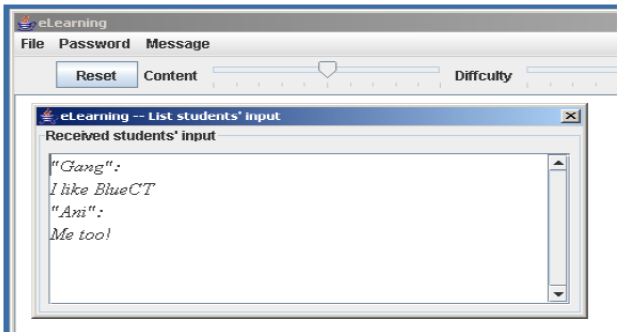

Fig. 4. The server side GUI showing students input

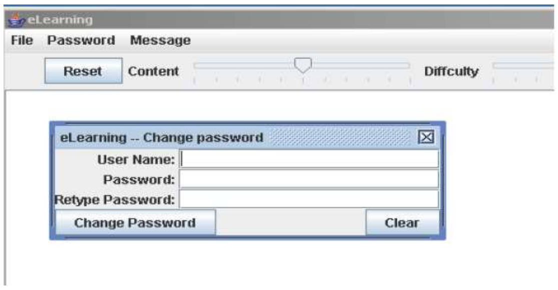

Fig. 5. The server side GUI for changing user password

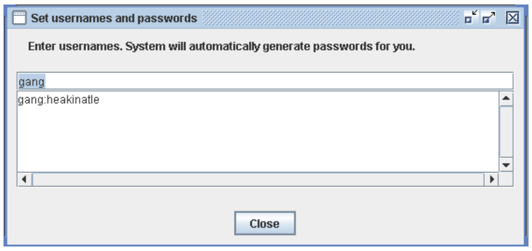

Fig. 6. The server side GUI for on the fly user creation

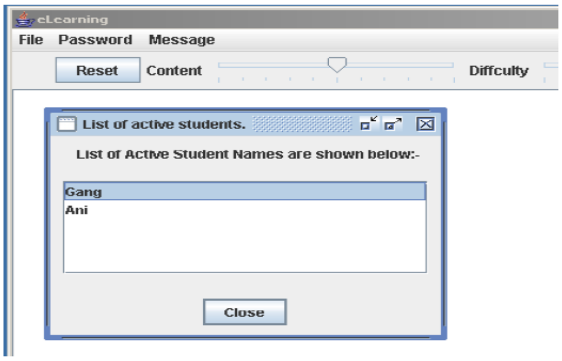

Fig. 7. The server side GUI for listing currently active students.

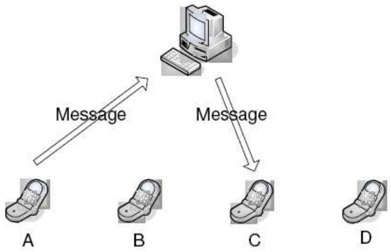

Fig. 8. The C/S Structure of Communication.

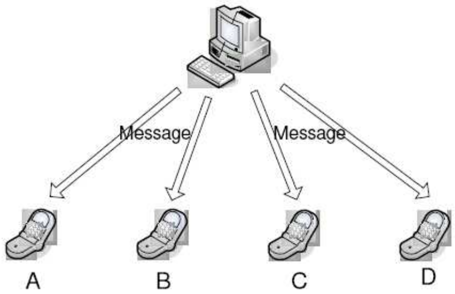

Fig. 9. The broadcasting of messages from server.

D. Listing active students

To be able to list all active students is a useful functionality for both the instructor and the student. Instructors can use this functionality to check how many students have logged in and who they are. Students can use this functionality to list other students and find out the one he/she wants to talk to. The server side GUI for this looks like Figure 7.

E. Communication between students

Supporting communication between students can make BlueCT much more powerful. Students can use this functionality to discuss questions and share their knowledge and feelings about the class. When we tried to implement this functionality, we had different ideas about it. One author insisted that we should use traditional Client/Server model, in which when one student wants to send messages to any other student, he would send the message to server first and the server would then forward these messages to their destinations. Another author had a different idea about this. He thought that we should use a peer-to-peer (P2P) structure similar to Napster, in which every client functions both as a server as well as a client. In C/S structure, all the messages would be sent to the server and then would be forwarded by the server to the corresponding destination after a while. This generates a heavy workload on server. In P2P structure, clients can communicate with each other directly, without bothering the server. However, in this case, the server has no way to control the messages sent between clients. At the end, we decided to use the C/S structure. This was because on one hand, we found that it was very important for instructors to control the messages sent between students, on the other hand, C/S structure was much easier to implement than the P2P structure. In designing this C/S model, we use polling mechanism on the client side. This means that the client polls the server at regular intervals (15 seconds) for any new messages from the server. The server, in response, either sends "NO OP" meaning that the client does not need any further processing or sends specific commands to the clients for doing some specific jobs. Figure 8 shows the C/S structure we choose to use.

F. Broadcasting messages to students

This is another interesting extension to BlueCT. With the help of this functionality, instructors can send questions, answers and other information to some specific student or to all the students conveniently. We implement this functionality by inserting the broadcast messages into the message queue of every active student and then invoking the communication method mentioned above for sending them to every student. The GUI looks like the one shown in Figure 9.

III. Conclusion

Our work was interesting. We learnt a lot from this project, especially in the areas of J2ME and Bluetooth. J2ME is a powerful and interesting environment that entitles us to run our own programs on mobile devices, such as PDA and handsets. We developed using a J2ME development environment by installing Nokia J2ME plug-in into Eclipse. In this environment one can code, debug and generate executable file. One can also try the executable file on an inbuilt simulator. The Nokia J2ME plug-in has several kinds of simulators that support most modern Nokia handsets. One interesting feature about J2ME is that its Java language packages are slightly different from the packages used in in J2SE. This bothered us for a

moment. For example, in J2ME's java.util package, there are no StringTokenizer class, hence we had to implement our own StringTokenizer.
We found that the old "send" and "sendNoReply" functions in BlueTooth API were not so stable. When exceptions are generated, they would fail and would print out exception information. We modified these two functions to make them reliable. We made them so that if any exceptions are generated during data transmitting, these two functions would retransmit automatically. This provides upper level functions a sense of reliable connection.

ACKNOWLEDGMENT

The authors would like to thank Dr. Son Vuong for giving the time, connecting them to the right people in this area and helping them with answers to the different questions. The authors also likes to thank Kamran Malik, the original designer of BlueCT for giving them the source code to work with. We also thank Mike Tsai for providing us with interesting ideas to work with in the project.

REFERENCES

[1] "The Bluetooth Special Interest Group", http://www.bluetooth.com
[2] Jaap Haartsen, "Bluetooth - the universal radio interface for ad hoc, wireless connectivity", Ericsson Review, no.3, pp. 110-117, 1998.
[3] Jennifer Bray and Charles F Sturman, Bluetooth: Connect Without Cables, Prentice Hall, 2001.
[4] Pravin Bhagwat, "Bluetooth: Technology for Short-Range Wireless Apps" , IEEE Internet Computing, June 2001.
[5] Nokia 6600 Internet over bluetooth - How to. http://www.arjunprabhu.com/blog/archives/2005/03/30/nokia-6600-internet-over-bluetooth-how-to/
[6] Playing with Series 60 Phones and Linux.http://www.gagravarr.org/series-60/
[7] The official website for the Bluetooth short range wireless connectivity standard. http://www.bluetooth.com/bluetooth/