ISBN: 972-8924-15-1 © 2006 IADIS

BLUETOOTH INTERACTION SUPPORT IN LECTURES

Henning Bär, Guido Rößling, Erik Tews, Elmar Lecher
Department of Computer Science
Darmstadt University of Technology, Darmstadt, Germany
{hcbaer, guido, e_tews, lecher}@rbg.informatik.tu-darmstadt.de

ABSTRACT

This paper stresses the need for an electronic interaction supporting system, which allows students to use their mobile phones to interact with the educator during lectures. The focus of this representation is a Bluetooth-based approach which offers the interaction services to the students free of charge. This idea came up because only very few students were willing to pay the GSM fee for the interaction service being offered.

KEYWORDS

Mobile Phone Support, Learner Interaction, Bluetooth.

1. INTRODUCTION

Undergraduate courses in German universities can easily reach 500 or even 1000 students in a single lecture. Alas, an increase in lecture attendance beyond a certain level often leads to a decrease in lecture intensity, quality, and active student participation. For example, many students may find it more difficult to ask a potentially "obvious" or "stupid" question in large lectures. Even if they do, they will often find it difficult to reach the educator with their question. In especially large lectures, students sitting toward the back may require a microphone to be heard by the lecturer.

Several studies (Ruhl et al. 1995, Waite et al. 2003) have shown that a media break including asking questions during a lecture can help in keeping students' attention. Asking the audience questions serves two functionalities: the context switch for a better attention and an opportunity for students to reflect the learned content. Alas, similar difficulties arise as for students who want to ask questions in large scale lectures. Students who need a bit longer than the fastest may be reluctant to continue thinking about the question after another student has given the answer and the educator continues. Furthermore, students in a large lecture hall may not be able to hear the answer unless the educator repeats it using a microphone.

Using mobile devices – notebooks, PDAs, or mobile phones – allows all students to answer a question. The correct answer can be shown afterwards. It is also possible to show how many students knew the answer and how many were wrong. Such technology-based solutions often require students who wish to participate using a mobile phone to pay a communication fee, unless a free transfer approach is used. The following chapter will give a short overview over such mobile phone-based interaction support systems.

2. RELATED INTERACTION SYSTEMS

Several university projects and companies have created systems for interaction support during live lectures. Apart from systems for browsers and larger devices, a few also support mobile phones.

Students using IVES (Brehm et al. 2003) can visit a special interaction website which is generated dynamically. They can submit evaluations of the lecture from *too slow* to *too fast*, or they can submit text messages. The educator can integrate these comments or questions into his talk, when he thinks they best fit to the content. The same research group also experimented with TCP over Bluetooth using mobile phones.

Using OCLI (Trompler et al. 2002), educators can display multiple choice questions on a beamer. Students can answer them by using WAP browsers on mobile phones. They can also submit text messages or an evaluation of the lecture to the educator.

A system developed at Kingston University, England (Stone 2003) allows deploying complex tasks using text messages (SMS). The students can answer them and also submit their solution using SMS. The system has an interface to the Learning Management System *Blackboard* (Blackboard Inc. 2005).

The Pervasive Learning Environment (PALe) (Chow et al. 2005) allows students to submit a text message or an answer to a multiple choice quiz by SMS or WAP. A card reader allows students to register, so the educator gains an overview over class attendance.

A system developed at Lappeenranta University in Finland (Hämäläinen et al. 2003) allows students to submit text messages, evaluations of the lecture, or answers to multiple choice quizzes with a mobile device which offers a browser. They are planning a Bluetooth or SMS solution. However, the system currently only supports wireless LAN.

Using Java-based browsers, students can use many of the discussed systems with mobile phones. However, they will be charged for the connection. Some of the developers think about a solution using Bluetooth, but we did not find a system which was ready to use. We therefore developed our own solution.

3. THE TVREMOTE INTERACTION SUPPORT SYSTEM

The TVremote system (Bär et al. 2005) shown in Figure 1 consists of a server, a client for the educator, and a set of clients for the students. It is designed to be highly flexible and to be equipped with many degrees of freedom for configuring the system to match personal expectations.

The server hosts the TVremote services. The communication layer offers interfaces for the different clients. Another layer reads and stores the interaction contents using a database. An email system can be connected.

The educator's client can be connected to the server to fetch text messages of the students, multiple choice questions to ask the audience, or the students' ranking of the lecture. When connected, it is also possible to transmit an image of the current lecture slide to the server. Information which is difficult for the students to copy on a paper and pen basis, such as long numbers or URLs, can also be posted. The students can then fetch this information using their client. We used UNO, the programming interface of OpenOffice, to retrieve the current slide from the presentation software OpenOffice Impress.

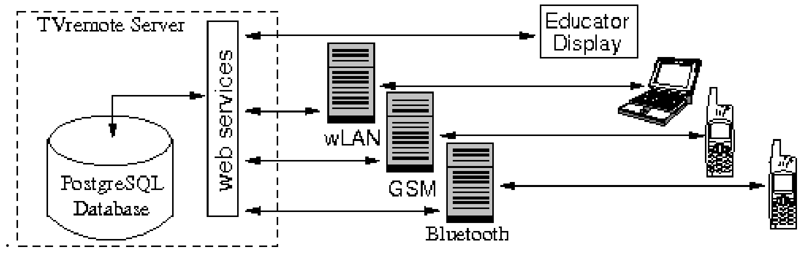

Figure 1. Architecture of the TVremote system

The students' clients consist of a user interface which is built for our network API written in Java. Currently, we offer user interfaces for notebooks, PDAs, and mobile phones. The commonly used Java Standard Edition (J2SE) cannot be used for mobile phones. Instead, there is a Java Micro Edition (J2ME) which supports less GUI functions than J2SE. This leads to a client for J2ME PDAs and mobile phones offering a menu to select a desired interaction option, as shown in Figure 2. However, there is no difference in functionality: both clients support submitting free texts and evaluations about the lecture to the educator. They also allow submitting answers to multiple choice quizzes and the retrieval of textual information.

Using web services, we successfully connected an experimental client written in the Microsoft .NET programming language. It was easy to create a PHP Client, which can offer the same functionalities to users of devices which support an HTML Browser. However, there is one device that most students already own and use daily: the mobile phone.

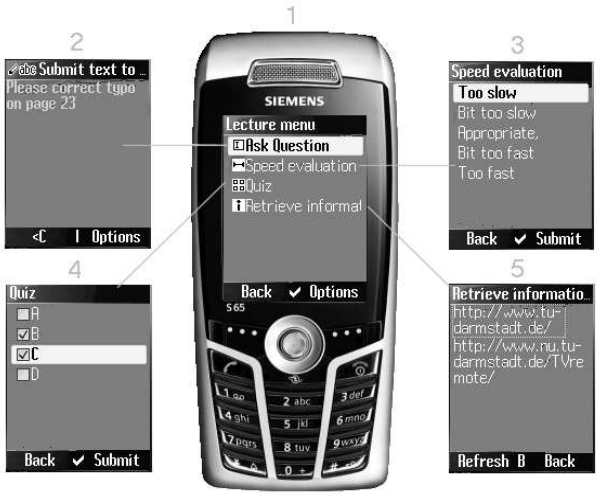

Figure 2. Client for mobile phones

4. COMMUNICATION LAYER

The communication layer is designed to reduce communication to a minimum number of transmissions. Otherwise, the protocol used for the communication would slow down the communication and increase costs in case of GSM connections because of protocol overhead, especially header and footer. So we preferred transmitting complex objects instead of using several requests. An example is the user interface, which can be configured differently for each lecture. When the student client is started, it fetches this configuration. Using web services technology, this leads to a transmission of about 7 kB.

```
POST /axis/services/TVremote2 HTTP/1.1
User-Agent: kSOAP/2.0
SOAPAction: ""
Content-Type: text/xml
Connection: close
Content-Length: 751
Cache-Control: no-cache
Pragma: no-cache
Host: localhost:8080
Accept: text/html, image/gif, image/jpeg, *; q=.2, */*; q=.2

<v:Envelope xmlns:i="http://www.w3.org/2001/XMLSchema-instance"  […]>
  <v:Header />
  <v:Body>
    <n0:submitText id="o0" c:root="1" xmlns:n0="tns:TVremote2">
      <token i:type="d:string">f35c56a817ad6f3addb852d7be59b7827229b6</token>
      <submit i:type="n1:submit" xmlns:n1="tns:TVremote2Beans">
        <dateCreated i:type="d:dateTime">2006-01-19T20:20:01.040Z</dateCreated>
        <id i:type="d:int">1</id>
        <button i:type="d:int">0</button>
        <uid i:type="d:string">-1204137030</uid>
        <message i:type="d:string">Mobile Learning</message>
        <dateNow i:type="d:dateTime">2006-01-19T20:20:01.042Z</dateNow>
      </submit>
    </n0:submitText>
  </v:Body>
                              </v:Envelope>
```

Figure 3. A SOAP message encapsulated in HTTP

We currently use the web services standard protocol SOAP (Brown et al. 2002), which is relatively easy to read and understand. Furthermore, SOAP supports derivation and subobjects, which allows simple mapping of Java objects to SOAP objects. Figure 3 shows an example for a transmission of a small text object. The SOAP message is encapsulated into HTTP. It defines two objects, a text called *token* and a bean called *submit*. The *submit* bean itself consists of a few further types. On the server, we installed the Axis libraries (Brown et al. 2002) to handle the SOAP communication. However, they are too space intensive to use them on mobile phones. So we decided to work there with kSOAP (Brown et al. 2002) instead.

For the server and the educator's client, there was no need to use anything else than TCP over Ethernet or rarely wireless LAN. The students, however, did not have any Ethernet outlets, so we first relied on wireless LAN. Alas, typical mobile phones are not equipped with wireless LAN. For submitting data over GSM, kSOAP establishes an HTTP connection to submit the message. However, students were reluctant to pay the fee for the GSM connection. So we extended the kSOAP libraries to offer Bluetooth submissions of SOAP objects. In GSM networks, the network provider offers a gateway where the data leaves the ether and enters the Internet. A similar solution for our Bluetooth approach is called Blueproxy (Madhavapeddy 2005). Using this solution, the SOAP objects are not encapsulated into HTTP as the Axis server expected them. So Blueproxy had to forward the data to a program which wraps incoming SOAP messages into HTTP.

5. BLUETOOTH USAGE

When using a GSM connection like GPRS or HSCSD, the phone sends requests using SOAP over HTTP, so any common J2EE web service can handle it. This is directly supported by kSOAP without modification.

We will now have a closer look at our Bluetooth networking protocol. We developed a special Bluetooth server running currently on Linux using a modified version of Blueproxy and a Bluetooth-to-HTTP protocol gateway written in Java. The server can run on small off-the-shelf hardware or even a Linux based wLAN access point, if it has a USB connector for adding a Bluetooth dongle.

The student client uses the jsr-82 API (Sun Microsystems 2006) which offers all the needed Bluetooth functionality to Java. Most current mobile phones supporting Java and Bluetooth support the jsr-82 API. Support for transporting SOAP messages over Bluetooth could be added very easily to kSOAP.

Whenever a mobile device starts the student client and needs a Bluetooth network connection, it first scans its environment for other discoverable Bluetooth devices. This usually takes 10 seconds and could be speeded up a little bit, but could then miss some devices. The student client cannot connect directly to a Blueproxy server without scanning first, because it does not know the addresses of all currently installed Blueproxy servers. Even if we bundled a list of all Blueproxy servers with the application, it would not know the lecture hall where the student is sitting in and would have to try each server until a server responds.

After a list with all devices in the environment is built, we start asking each device if there is a TVremote Blueproxy service running on it. We try all devices in our list until the list is completed or a TVremote Blueproxy service was found. This can take some time, if there are a lot of discoverable Bluetooth devices in the room. However, we found out that most students turn discovery off for safety reason which leads to 10-15 active devices in a big lecture with about 400 students. If we found a service, we store the address of this service for later use. All further network requests will try this service first, and only start a new discovery in the case of a network failure.

Now, an RFCOMM connection is established to this service and the SOAP request is sent. It is not encapsulated into an HTTP-request. Instead, we send the length of the request first in a four-byte encoded integer, then the request itself. The response of the other party is sent in the same way. After the response was received, the RFCOMM connection is closed. No connection to the Blueproxy is being held active. So a single Blueproxy can support a large number of clients, as long as only a few of them send data in parallel.

We can even use multiple Blueproxy servers in one room. The students' client automatically picks a Blueproxy, which is in most cases the one with the best connection quality. If one of the Blueproxy servers is shut down, or the student changes the room, a new one will be found without the need of user interaction or restarting the application.

We noticed Bluetooth to be a quite appropriate technique for supporting interaction in classes. Using class 1 Bluetooth devices, we were able to transmit data through an 800 seat lecture hall even with students in it. Even though there cannot be more than eight coupled devices simultaneously, this is no real restriction. The submission of objects does not require handling any sessions, so the time slice in which a device is coupled to the server is limited by the size of the data block being submitted. If there is really need to support more than seven transmissions in parallel, further USB Bluetooth dongles can be plugged into the computer.

6. CONCLUSION

We introduced a Bluetooth solution for supporting interaction in classes. By using such a solution, students do not need to pay any connection fee to participate in the interaction supporting system. This solution is restricted by connections initiated by the client devices. However, our current web service solution does not allow connections initiated by the server anyway.

Currently, we are working to port our Bluetooth server to an embedded off-the-shelf system which costs about $50. With this no additional server installation would be necessary anymore.

ACKNOWLEDGEMENTS

We thank CellIQ for lending us different Nokia mobile phones and Siemens for two Siemens S65 mobile phones. Jessica Heymann worked on the mobile phone application. Collin Mulder gave us some hints about Bluetooth.

REFERENCES

Bär, H., Rößling, G., Köbler, S. and Deneke, M. 2005. Evaluation of Digital Interaction Support in a Large Scale Lecture. *Proceedings of the IADIS International Conference Applied Computing 2005*. pp. 63-67.

Blackboard Inc. 2006. Blackboard Homepage. WWW: http://www.blackboard.com

Brehm, J., Brancovici, G., Müller-Schloer, C., Smaoui, T. and S.Voigt. 2003. The Interactive Lecture – An Integrated Multimedia Supported Teaching Experiment. *Proceedings of Interactive Computer Aided Learning 2003*. Villach, Austria, pp. 1-12.

Brown, C., Sarang, P., Ayala, D. and Chopra, V. 2002. *Professional Open Source Web Services*. Wrox Press Ltd. Birmingham, UK.

Chow, K., Fan, K., Chan, A. Ip, H. and Kwok, L. 2005. Enhancing Teacher-Student Interactions with Multiple Handheld Devices. *Proceedings of Mobile Learning 2005,* Qawra, Malta. pp. 79-86.

Hämäläinen, H., Ikonen, J. and Porras, J. 2003. Applying Wireless Technology to Teaching Environment. Workshop on Applications of Wireless Communications. Lappeenranta, Finland. pp. 29-36.

Madhavapeddy, A. 2005. Blueproxy project. WWW: http://anil.recoil.org/projects/blueproxy.html

Ruhl, K. and Suritsky, S., 1995. The Pause Procedure and/or Outline: Effect on Immediate Free Recall and Lecture Notes taken by College Students with Learning Diablilities. *Learning Disability Quarterly*, Volume 18. pp. 2-11.

Stone, A. 2003. Towards implementing m-learning support for first year students at Kingston University. *Learning Technology newsletter*, Los Alamitos, USA, published as an online journal

Sun Microsystems. 2006. JSRs: Java Specification Requests – detail JSR# 82 WWW: www.jcp.org/en/jsr/detail?id=82

Trompler, C., Mühlhäuser, M. and Wegner, W. 2002. OPEN CLIENT LECTURE INTERACTION: An Approach to Wireless Learners-in-the-Loop. *Proceedings of the 4th International Conference on New Educational Environments*. Lugano, Switzerland, pp. 43-46.

Waite, W., Jackson, M. and Diwan, A. 2003. The Conversational Classroom. *Proceedings of the 34th SIGCSE Technical Symposium on Computer Science Education*. Reno, Nevada, USA, pp. 127-131.