Mobile Computing to Facilitate Interaction in Lectures and Meetings

Isabelle Demeure, Claudie Faure, Eric Lecolinet,
Jean-Claude Moissinac, Stuart Pook
École Nationale Supérieure des Télécommunications
GET – CNRS UMR 5141 – LTCI
46, rue Barrault – 75634 Paris Cedex 13 – France
email: isabelle.demeure@enst.fr

Abstract

The *Campus Mobile* project explored how PDAs and innovative interfaces can improve interaction during lectures and in small meetings. These mobile computers (small PCs or PDAs) are nomad mediators that provide the link between the public space and the user's private space. A lecturer uses an augmented whiteboard to annotate her presentation, while the slides and annotations are broadcast in real time to the students' PDAs. The students can also annotate the presentation. They can then replay the lecture at their leisure. We developed and tested the user interfaces for the interactions on the whiteboard and on the PDAs, the data formats to store the annotations and temporal data, and the network protocol for reliable wireless broadcast transmission. Small meetings have a very different interaction pattern. Instead of a lecturer that transmits knowledge, the participants collaborate in modifying documents. We developed groupware services that encourage these interactions and show how the different interaction patterns imply different network usages.

1. Introduction

The *Campus Mobile* project was created to find ways in which mobile computing and new interaction techniques can be used to improve interaction on a university *campus*. Users have a 'nomad mediator' to interact and share information with other users and with fixed facilities such as interactive stands and enhanced computer screens. These mediators are wireless portable computers or personal digital assistants (PDA). This project analysed several scenarios in which these mediators are useful. The two scenarios that we discuss treat face-to-face lectures and small meetings.

The first scenario, which we call 'CORAO', takes place in a lecture theatre where the lecturer uses an augmented whiteboard to control and annotate the slides of her presentation. The whiteboard transmits the slides and annotations in real time over the wireless network. The students are equipped with mediators which receive the slides and annotations. Using their mediator, students can add their own annotations either during the lecture or during a later review.

The second scenario, called 'REGROUP', is a face-to-face meeting between a small number of participants. We facilitate their sharing of interactive computer resources so that participants can work together on a project. These resources are typically a video projector connected to a machine in the room. The meeting might be a business reunion where several people work together to prepare a document or a tutorial where several students present their work.

In this scenario any participant can control the documents shown on the video projector (without having to change places or computer connections) via his mediator. The control is shared: several participants can display and modify documents on the video projector, or any other machine, at the same time. Each document is also shared using synchronized views that are displayed and modified on the mediators of multiple participants. *Single display groupware* is thus generalised in our scenario as both the output device and the documents are shared.

This scenario may appear to be similar to the previous scenario however the interaction patterns are very different and result in different network usage patterns. In particular, REGROUP does not involve transmission from one master machine to many slave machines but rather bi-directional communication between a small number of machines. The interaction and network patterns are analysed in section 3.

An augmented whiteboard offers new interaction possibilities. In section 4 we present our user interface that facilitates the use of these possibilities without over complicating the interface. At the end of the lecture the students' meditators have a copy of the slides and the lecturer's annotations. This information is not static: each slide and annotation is stored with time information that allows the student to replay the lecturer's presentation of her slides and their annotations synchronized with

Proceedings of the First International Conference on Distributed Frameworks for Multimedia Applications (DFMA'05)
0-7695-2273-4/05 $ 20.00 IEEE

an audio replay of the lecture. The data structures that used to store this temporal information are described in section 5.

In the CORAO scenario, unicast transmissions to a possibly large number of receivers would exceed the limited bandwidth of a wireless network. In section 6 we describe our broadcast solution that avoids this problem.

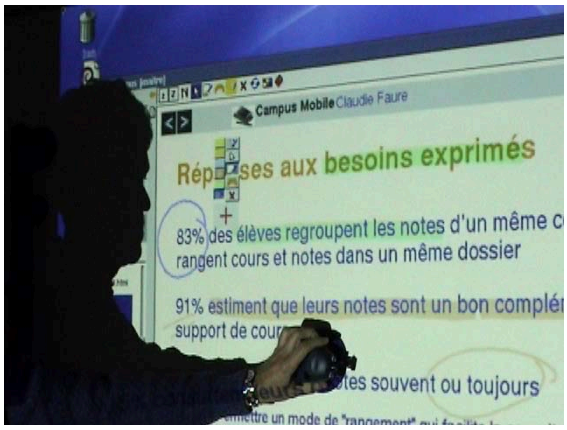


Figure 1: interactive whiteboard

2. Related Work

Improving teaching and learning with new technologies is now a priority in education. CORAO and REGROUP were defined in response to inquiries and potential user's observations. This led us to design a system which integrates several technologies to address classroom and meeting activities as a whole while other systems focus on some of them.

In response to students' requests, a system can replay the lecture. This request was explicit when we asked students to imagine scenarios for a future ideal university. However, presence in a classroom seems to be the best way to acquire knowledge. Teachers can take advantage of mutual interaction and students' questions to adjust the lecture. The capture of classroom events and interaction between participants during lectures are important topics when attempting to improve teaching and learning.

Students say that after several days they have problems in fully understanding their notes and even the lecture slides (if available). Most of the lecture content is conveyed by short lived events, such as oral explanations and handwritten notes. *Classroom2000* [1] was one of the first projects that captured and integrated the different media (slides, notes, audio, video, web sites) involved in a lecture. Video offers a strong feeling of presence, but it requires a large amount of data storage, and the attention of the reader is divided between the video and the slides during playback. For these reasons, we chose to eliminate video capture. Recording audio and annotations ensures a permanent access to these short-lived events.

Using an interactive whiteboard requires specific interaction techniques. In the *Interactive Presentation Manager* [14], interaction with a large vertical surface is very much like user interaction with applications on a desktop computer. A strong analogy between the use of the traditional chalk and board and the use of an interactive (or computerized) whiteboard is attractive for users who do not wish to change their habits [1, 9]. We feel however that the computerized whiteboard opens up new interaction possibilities. Contextual menus and pen-based commands are well suited for interacting with a board [23]. Handwritten words and formula drawn on the interactive whiteboard are recognized in [9] during mathematics and physics lectures. In large lecture theatres the top of the whiteboard is often too high to be easily reached. In this case, a pen-based device may be used as a remote sensor to send input to the board [21]. This kind of device also allows the lecturer to keep eye contact with the students and to move freely in the classroom [2, 22].

A teacher standing alone in front of a dense group of students is not the best setting for interpersonal interaction. A close view of the lecture slides and the teacher's annotations, instead of a distant view of the public board, may help each student to feel that the lecture is for him, and not for a group to which he belongs as one member among many. Student personal devices are used for displaying the content of the public board [7, 19, 22, 32]. Adding (or attaching) personal notes to the slides helps students customize the lecture document [7, 19, 32]. These notes remain related to the slides even after media integration [17]. Students may exchange notes during the lecture [12, 21]. Personal devices can also be used to increase interaction in classroom with questions and answers [12, 21, 22, 25].

3. Interaction Patterns and Communication Support

As illustrated by the above scenarios we focussed on situations involving several simultaneous participants: in the CORAO scenario the lecturer sends slides as well as her annotations to the students' PDAs; in REGROUP, views of the same document can be displayed and modified on the mediators of multiple participants. REGROUP is therefore a *Single Display Groupware* (SDG) and thus multiple users can share a common display or a common set of displays located in the same room.

The remainder of this section is divided into two parts: we first discuss issues related to the support of the 'one-to-many' interaction patterns seen in services in-

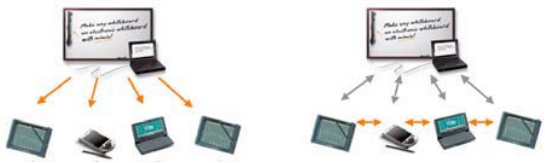
volving several simultaneous participants; we then present services for single display groupware.

3.1 One-To-Many Interaction Pattern

The *one-to-many* interaction pattern contrasts with the *one-to-one* interaction pattern found in the widespread client/server model. In this model the service user (or client) proceeds by sending a request to a server that subsequently replies thereby providing the requested service.

The *one-to-many* interaction pattern is most naturally supported by broadcast communication technologies. Therefore we made it a requirement that the *Campus Mobile* communication middleware supports broadcasting. Other requirements include the support of 802.11 wireless local area networks and the support of terminals with limited capabilities. The characteristics taken into account in the support of 802.11 local area networks are the possibly frequent disconnections and the limited bandwidth. In support of terminals with limited capabilities we took into account the limited CPU power and memory. Although battery autonomy is an important issue we have not yet addressed it.

Traditional middleware solutions often implement tightly coupled communication mechanisms such as remote object method invocation (as in CORBA and Java RMI). In these mechanisms, the message destination must be known when sending, which is difficult when destinations change or the number of recipients varies. In addition, this type of communication is synchronous. By contrast, MOM (Message-Oriented Middleware) encourages loose coupling between message senders and receivers with a high degree of anonymity. MOM also supports asynchronous communications that are more adapted to wireless and mobile environments. Other MOM characteristics such as guaranteed delivery and store-and-forward messaging are useful in mobile applications.



one-to-many pattern **SDG pattern**

Figure 2: communication patterns

Sun's Java Message Service (JMS) has become a reference interface for MOM. There are several projects, such as Pronto [35], that have tackled the issue of developing JMS systems adapted to wireless and mobile environments. We are particularly interested by Pronto because it uses IP multicast as its transport mechanism. Pronto provides efficient transmission of messages from one publisher to many subscribers, and the number of subscribers can increase without increasing network traffic.

Wireless JMS [16] is another JMS compliant system whose development was partly inspired by the CORAO scenario. In parallel with the *Wireless JMS* project we developed a robust broadcast communication system named *campcast* specifically designed to support the CORAO scenario and similar ones. This ad hoc system was chosen over existing systems because we wanted to have full control over the broadcast protocol and a system with a very small memory footprint. The implementation of *campcast* is described in section 6.

3.2 Single Display Groupware

Single Display Groupware (SDG) has been proposed as a new approach to facilitate face-to-face collaboration in which multiple computers can be used to control one or several large screens.

Projectors are now commonly used during meetings and presentations to display a PC's output on a large screen. Usually only one person (the one who is interacting with the PC) can control the screen. Other people cannot display documents simultaneously on the screen nor take control of the mouse or keyboard. There are many cases when such interaction patterns would be useful: for instance, a teacher could ask a student to show how she solved an exercise; several people could interact simultaneously on the same document for solving a problem. SDG makes such interactions possible.

Unfortunately, SDG systems are notoriously hard to build. Common graphical toolkits and windowing systems offer little support for developing software that manages multiple input (multiple mice and keyboards) and output (remote and replicated views) in an appropriate way for SDG applications. For instance, current windowing systems assume that only one person is interacting with a computer at a given time. As a consequence they do not make it possible to distinguish events coming from different devices (controlled by different users). Several research systems have been proposed recently in the literature [4, 15, 22, 29] but none of them seems to address all the requirements of single display groupware. The rest of this section describes the kind of services that are relevant for SDG and the architecture of the system that we have developed for this purpose.

SDG multiplexes event input sources or graphical output on a single computer and on a computer network. Several kinds of **input multiplexing** can be envisaged:

– *Single remote control:* one user using a given computer (for instance a handheld device) can take control of the input resources (mouse/keyboard) of a remote computer (typically the PC whose output is

displayed on the large screen). Hence several users can *successively* take control of the shared screen.

- *Multiple local control:* several users can interact *simultaneously* on the same computer (a separate pseudo-pointer being associated with the input device of each user and continuously displayed on the screen). Two sub-cases must be considered depending on whether these users are interacting with the same application or on different applications located on the same display.
- *Multiple local/remote control* that combines the two previous cases: several participants can interact *simultaneously* on a given display, either locally or remotely.
- Single or multiple remote control *on multiple screens,* two cases that are immediate extensions of previous ones.

Single remote control is a basic service that is offered by all SDG systems. Multiple synchronous control is a much more complex case because current graphical systems cannot display several pointers on the screen nor to distinguish input sources. Hence, this type of service is generally unavailable or solved in a specific way at the application level. A SDG application must thus directly manage alternate input devices and remote connections. It must also display pseudo-pointers on the top of its windows. As a consequence, other applications will not receive events from alternate devices or remote connections and pseudo-pointers will not be shown on them.

Output multiplexing allows two possibilities: displaying an application on a remote computer and replicating the application windows on several computers.

Under the first possibility, one or several participants can display documents stored on their own computer on the shared screen (controlled by a remote PC) without having to change places or computer connections. Hence, several documents coming from different computers can be displayed simultaneously on the same remote screen. But, they can only be controlled from this screen (the participant who launched the application loses control on them). This problem can be solved through input multiplexing, as seen before.

The second possibility is even more interesting because the application will be shown on the originating computer and on the shared screen (and possibly on the screens of the computers of other users). The application can thus be controlled from different screens. Certain systems will even allow several users to interact simultaneously on the same application as described before in the case of *multiple local control*.

Common graphical systems, such as X Window or RDP (for Microsoft Windows) allow applications to display on remote computers. However, such systems have obviously not been primarily designed for SDG and some kind of encapsulation is necessary to make remote display easily usable. Graphical replication raises more complex problems and is thus rarely available. VNC [27] provides graphical replication but with several limitations: multiple users cannot really interact simultaneously and graphical replications cannot be adapted to the output device. In addition, some existing groupware toolkits [28] provide tools that do not rely on graphical replication for displaying distributed applications. Such systems have primarily been designed for the more complex case when users are working at different locations (a situation that raises group awareness problems that are out of the scope of this paper).

3.3 The Campus Mobile SDG Environment

The SDG environment that was developed for the *Campus Mobile* project supports all the types of input and output multiplexing presented in the previous section.

Input multiplexing is managed by a set of *interaction servers* (a server on each computer). An *i-server* offers the services required by *multiple local/remote control.* When connected together, interaction servers remotely control all the corresponding screens and define a screen topology.

Each *i-server* can manage several pointers simultaneously on a given display. More precisely, an *i-server* is able to manage *several* input sources without merging them and to display the corresponding pointers anywhere on the screen. Each input source can be attached to one or several physical devices (e.g. second mouse, trackball or MIMIO) or to a flow of remote events coming from another computer (via a simple protocol over a TCP/IP connection). The events generated by all input sources are then 'sent' to the appropriate application according to their location or to the input focus.

This client/server architecture has several advantages as it manages all interaction resources at the 'display' (or computer) level. First there is no need to handle alternate devices (and device drivers) or remote communications at the application level. All applications receive standard and alternate events in a normalized form. Similarly, pseudo-pointers are created automatically by the *i-server* and appear above all applications. Hence, there is no need to modify applications in order to interact with them in a simplified *multiple local/remote control* mode. For instance, two people can interact locally or remotely on two applications shown on the same display. They will be able to control each application independently with their own device or computer.

The case where several users interact simultaneously on the *same* application is more challenging. Such applications (and the graphical toolkits they rely on) must separately process the events coming from several input

sources so that users can interact simultaneously with different widgets. This is generally impossible because typical graphical systems combine the events coming from all input devices both at the windowing system and GUI toolkit levels. As said before, *i-servers* can manage several separate input sources. The events they generate have a logical source ID. SDG applications can thus distinguish various input sources and separately process the corresponding events. Unfortunately, classical GUI toolkits assume that a single user interacts with an application and do not provide such capabilities. We developed a new GUI toolkit, called *Ubit* [18, 33] that removes this restriction. *Ubit* can handle an arbitrary number of event flows, each of them controlled by a separate event loop. A great advantage of this architecture is that any *Ubit* application automatically supports multiple user interaction.

Output multiplexing. An application written using the *Ubit* toolkit can replicate any part of its output on displays controlled by remote computers. This mechanism is dynamic so that any user can temporarily show a replicated view of his application (or of any subpart of it) on the shared screen or on the screens of the other participants. This application can then be controlled from all computers where it appears. Synchronous interaction from all these devices is possible as the toolkit supports multiple user interaction.

The main advantage of graphical replication is that it makes it very simple to create collaborative applications. Distributed applications are generally hard to design because they involve complex synchronization mechanisms. Graphical replication provides a trivial solution to this problem: there is no duplicated data to synchronize because there is only one application running. This application exports and controls GUIs on the other machines. The current version of the *Ubit* toolkit uses the *X Window* communication protocol as base for graphical replication.

This centralized approach has two drawbacks. The first is that the replicated views are usually identical, the views are thus not adapted to their screens. The *Ubit* toolkit removes this limitation because any subpart of the GUI can be displayed remotely or duplicated. Hence, different views can appear on different screens. In addition, it is also possible to adapt the replicated views to the characteristics of each display (such as the size).

The second drawback of graphical replication is that it requires a relatively large bandwidth. The number of remote displays is thus necessarily limited and the network must have appropriate characteristics. These restrictions do not cause a problem in the case of single display groupware: the participants are obviously using a local network and very few of them will interact simultaneously on the same replicated views. In conclusion, graphical replication is appropriate for SDG because it provides a simple and flexible solution and does not cause performance problems.

4. Interfaces: Whiteboard Interaction

The *Campus Mobile* project allows a lecturer to control and annotate her presentation directly from the whiteboard. A MIMIO [20] device is used to capture the movement of the stylus pen on the screen. A UNIX MIMIO driver has been developed and integrated into the interaction server (described in the previous section). With this driver we can use and configure the MIMIO pens in various ways and make them compatible with existing applications. The rest of this section describes how we modified the behaviour of this input device to facilitate the interaction with the whiteboard.

An augmented whiteboard obviously has a richer interaction style than a classical whiteboard. A user can draw and write on the board and also highlight text in the presentation, change pages or documents, scroll or zoom the data in the presentation, and control other applications. Two types of techniques are traditionally used. The first provides several interaction tools. For instance the MIMIO provides four stylus pens of different colours and a combined small and large eraser. Our initial approach was to configure these tools so that performed different actions. For instance, one pen was used for writing, another for highlighting, and a third for controlling applications. However, experimentation showed that using such a large number of tools was rather cumbersome. Moreover, it was still necessary to use other means for changing the writing colour and other actions.

The other way of interacting with a digital whiteboard is to use it as a computer workspace (i.e. with buttons, menus and other widgets). However, this kind of interaction style is not very well suited because of the size and the position of the whiteboard (the lecturer will have to move quite often to reach the widgets and some of them may even be unreachably high).

We propose a solution that combines the two previous approaches. One stylus pen is used as a general pointing/writing device whose effect can be changed by a movable palette. This palette is moved by pointing a second tool on the whiteboard. Hence, the lecturer will be able to perform the most frequent actions just by using two different tools (one in each hand) and she will not have to move to perform these actions. This can be seen as a simplified type of two-handed interaction [3] where the non dominant hand is used to bring the palette to the appropriate location and the dominant hand is used to write, draw or control the interaction.

We have also experimented with the use of *Control menus* [24] instead of a movable palette. Control menus, which are inspired from *Pie menus* [11] select and control an operation in a single gesture. Other studies have

already shown that similar kinds of menus were very appropriate when interacting with large screens [10]. We are now comparing the usability of these two types of interaction.

5. Data Formats

There are two types of data used in CORAO: slides and annotations. For each of these two data types we chose a format based on standards and suitable for both use during a lecture and during later review. Students follow the lectures on a PDA or small portable computer while later review is normally performed on a standard PC. We defined formats for slides and annotations.

Slide Format: Slides are prepared by the lecturer in an external format and before the lecture starts these slides are transformed into a representation usable by the whiteboards interface. At the end of the lecture this document has been enriched with information from the lecture.

The lecturer's slides can be handwritten or digital, most commonly a list of HTML pages, a PowerPoint presentation or a PDF document. Our current prototype can handle HTML or PowerPoint slides.

Our whiteboard uses a transcoding of the original slides into a valid subpart of the HTML specification. This transcoding integrates text and images and is the representation that is sent to the students' mediators during the lecture. The student can use this representation to review the lecture.

The recorded lecture is a multimedia document composed of text (titles, notes, and other comments), images of the slides, graphics (as part of a slide or annotations) and audio. This document has a linear temporal progression structure imposed by its audio component. A number of projects have studied the representation and exploitation of audio (or video) recordings of lectures [1, 5]. In defining our format we privileged flexibility and facilitated modifications by choosing XML grammars. The IMS consortium publishes XML schemas for various educational resources, including lectures. Our top level storage uses the appropriate IMS schema and this format provides high level access to the lectures, as in a lecture repository with search functionalities. However, for the recording and the replay of the lectures, we need a multimedia format. Some projects use custom formats [1] while others use SMIL [30, 34] or MPEG-4 [8]. As described in the next section, we chose SVG as it is the only XML format combining images, annotations and sound.

Annotation Format: Our prototype stores annotations from the lecturer and the students. Annotations are graphic plots accompanied by temporal information. We recorded the start of each plot. A plot is all the graphics elements created from the moment the pen touches the surface until it is lifted from the surface.

Several projects use a custom format to store annotations [2, 12, 19]; these projects require a specific player to replay annotations. We chose to follow a standard. We compared the use in our context of InkML [13], an XML grammar proposed by the W3C specifically to store annotations, and the SVG standard [31], designed by W3C to represent vector graphics with XML.

We chose SVG as it provides the required geometrical representation of the annotations. InkML would have allowed a more elaborated representation, including the semantic grouping of fragments of plots, but is limited to annotations. Another advantage of SVG is that players are widely available and that these players can replay the whole presentation: images, text, and sound (this is an Adobe extension in SVG 1.1 but is included in the upcoming SVG 1.2).

The SVG document generated during the lecture allows students to replay the lecture with the correct temporal association between the slides, the audio and the annotations. As SVG is an XML grammar and because of its vectorial nature, our recorded lectures can be transformed and viewed using many different representations.

6. Implementation

This article analysed, in two scenarios, how mobile mediators can improve inter-participant interaction during lectures and in meetings. This section presents implementations of the resulting proposals.

6.1 CORAO: Lectures

The lecturer's slides are each stored in one or more files. The lecturer can annotate her presentation during the lecture.

The students receive a copy of each slide of the presentation no later than when it is projected. The students receive the lecturer's annotations in real time and can add their own annotations. Students who arrive late receive the current slide immediately and the annotations that they have missed. There can be many students (more than 50) present during a lecture.

The lecturer uses a MIMIO and a video projector connected to a computer installed in the lecture theatre. This computer is connected to an IEEE 802.11b wireless (also known as WiFi) network. The lecturer uses the MIMIO to control the presentation and to annotate the slides**.**

Students have PDAs (or portable computers) equipped with WiFi cards. These machines are *Hewlett-Packard iPAQs* with 96 millimetre screens, 400 MHz processors and 128 MB RAM. They run *Linux Familiar*. The student machines receive their network configuration from a DHCP server or, until a DHCP server replies, by imple-

menting the *ZeroConf* protocol (part of the protocol proposed by Apple under the name *Rendezvous* [26]).

The computer in the theatre and the students' PDAs all use the same program (called *campus*). This program is based on the *Ubit* toolkit and adapts automatically to the different styles of interaction required by a MIMIO or by a stylo on a touch screen and to the different presentations appropriate on a video projector or the small screen of a PDA.

Given the limited network bandwidth available per student we decided to broadcast the slides and the annotations to the students. Several systems exist that provide reliable transmission using UDP broadcast. As our PDAs have limited resources we decided to design a transmission protocol that minimises the use of these resources**.**

This protocol is implemented in two programs. The first, *campcasts*, runs on the computer and receives commands from *campus* and sends broadcast messages on a well known port. The second, *campcastc*, runs on the PDAs, stores all received slides and annotations, requests retransmissions when necessary, accepts requests for slides from the program *campus* on the PDA, and informs *campus* when requested slides arrive. Figure 3 shows how the different programs that run on the computer and a PDA are interconnected as well indicating where we store the information concerning the lecture.

At least once a second, *campcasts* broadcasts a status packet containing the name of the current slide, the number of the last annotation sent and a status that indicates if the computer is ready to accept requests for retransmissions. These status packets and the transmissions requested by *campcastc* make the transmission protocol robust in the face of the frequent packet losses that occur when using UDP broadcast over a WiFi network.

In our protocol, the slides are divided into packets (so that only lost packets need to be retransmitted) that fit into an UDP packet (because if one fragment of a fragmented UDP packet is lost the whole packet must be retransmitted). Before changing to the next slide, the computer sends the slide and any associated images. This avoids having every PDA request the slide. Annotations are sent as the lecturer makes them so that *campus* can draw them immediately.

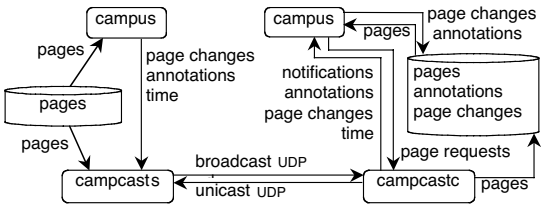


Figure 3: information flows in CORAO

When *campcastc* receives a status packet indicating that the current slide has changed, it informs *campus*. *Campus* will then display this slide. However if it has not been received, *campus* will ask *campcastc* to request a retransmission of the missing parts. As *campcastc* receives annotations it forwards them to *campus*. If a status packet informs *campcastc* that it has missed annotations, it will send a unicast UDP packet requesting a retransmission. *Campus* can thus receive annotations out of order. All traffic from the computer is broadcast so that all the PDAs see all retransmissions thus reducing the number of retransmissions.

6.2 REGROUP: Meetings

Meetings involve a small number of participants that wish to simultaneously control one or several shared applications. Meetings do not require a large amount of bandwidth since there is only a small number of active participants at any one moment. During meetings we thus connect the participants' computers using TCP/IP. The display of documents on multiple screens and the remote control of screens is directly handled by the toolkit *Ubit* described in section 3. Since we use point-to-point connections, we need to know the addresses of the other participants. We find these addresses using the *Rendezvous* resource discovery service [26].

7. Conclusion

We gave an overview of the *Campus Mobile* project and its current state of development. This project aims at using wireless technologies to facilitate and to improve interaction in education and collaborative work. Throughout the project we observed the interactions that occur in meetings and lecture theatres and we noticed that there are many different activities and needs in these situations: annotation, document exchange, document sharing, collaborative editing, shared and remote control of applications, creation and playback of multimedia documents, etc. Satisfying this wide range of activities required knowledge and skills from a number of domains including human computer interaction, single display groupware, wireless networks, middleware, document engineering, and user centred design. The originality of our approach was to find a global solution that takes into account all these various aspects. This approach contrasts with the more focussed proposals of other projects.

We developed the integrated *Campus Mobile* environment to facilitate the observed interactions. This environment has been successfully tested with a small number of participants. We will now undertake experiments with a larger number of participants.

8. References

[1] Abowd G.D. *Classroom 2000: An experiment with the instrumentation of a living educational environment*. IBM Systems Journal, Vol. 38, No. 4, 1999.

[2] Anderson R., Anderson R., Simon B., Wolfman S. A., VanDeGrift T., Yasuhara K. *Experiences with a Tablet PC Based Lecture Presentation System in Computer Science Courses*. Proc. SIGCSE'04, ACM, Norfolk, USA, 2004.

[3] Bier E.A., Stone M, Pier K., Buxton W., DeRose T.D., *Toolglass and Magic Lenses: The See-Through Interface,* Proc. ACM- SIGGRAPH, pp. 73-80, 1993.

[4] Booth K.S. et al., *The "Mighty Mouse" Multi-Screen Collaboration Tool,* Proc UIST 2002, ACM, 209-212.

[5] Brotherton J.A., *Enriching Everyday Experiences through the Automated Capture and Access of Live Experiences: eClass*. Georgia Tech, Ph.D. Thesis, Dec. 2001.

[6] *CampusMobile Project*: www.infres.enst.fr/~elc/campmob/

[7] Denoue L., Singh G., Das A., *Taking Notes on PDAs with Shared Text Input*. EDMEDIA, Lugano, Switzerland, 2004.

[8] Envivio, *the 4Forum product*: www.envivio.com

[9] Friedland G., Knipping L., Rojas R., *E-Chalk - Technical description*. Freie Universität Berlin, Institut für Informatik, Technical Report B-02-11, 2002, www.echalk.de

[10] Guimbretière F., Stone M., Winograd T., *Fluid Interaction with High-resolution Wall-size Displays*. ACM UIST 2001.

[11] Hopkins D., *The design and implementation of pie menus*. Dr. Dobb's J. of Software Tools, 16(12), pp. 16–26, 1991.

[12] Iles A., Glaser D., Kam M., Canny J., *Learning via Distributed Dialogue: Livenotes and Handels Wireless*. Proc. of CSCL, Boulder CO, Lawrence Erlbaum, pp. 408-417, 2002.

[13] *Ink Markup Language*: www.w3.org/2002/mmi/ink

[14] IPM, *Interactive Presentation Manager.* Tutorial For Windows, Numonics Corporation, 2002 www.touchboards.com/numonics/ipm500TM.html

[15] Johanson B., Hutchins G., Winograd T., Stone M., *PointRight: Experience with Flexible Input Redirection in Interactive Workspaces,* Proc. UIST, ACM, 227-234, 2002.

[16] Kaddour M., Pautet L., *A Middleware for Supporting Disconnections and Multi-Network Access in Mobile Environments*. Proc. of Middleware Support for Pervasive Computing 2nd Conference on Pervasive Comp. (Perware'04), USA, 2004.

[17] Landay J.A., Davis R.C., *Making sharing pervasive: Ubiquitous computing for shared note taking*. IBM Systems Journal, Vol. 38, N. 4, pp. 531-550, 1999.

[18] Lecolinet E., *A molecular architecture for creating advanced interfaces,* ACM UIST, pp. 135-144. 2003.

[19] Lienhard J., Lauer T., *Multi-Layer recording as a New Concept of Combining Lecture Recording and Students' Handwritten Notes*. Proc. Multimedia'02, ACM, 2002.

[20] MIMIO: www.mimio.com

[21] Mühlhäuser M., Trompler C., *Digital Lecture Halls Keep Teachers in the Mood and Learners in the Loop*. Proc. of E-Learn, AACE ed., Montreal, Canada. pp. 714-721, 2002.

[22] Myers, B., *Using Handhelds and PCs Together*. Communications of the ACM, Vol. 44, No. 11, pp. 34-41, 2001.

[23] Mynatt E. D., Igarashi T., Edwards W. K., Lamarca A. *Designing an Augmented Writing Surface*. IEEE Computer Graphics and Applications, pp. 55-61, 2000.

[24] Pook S., Lecolinet E., Vaysseix G., Barillot E., *Control Menus: Execution and Control in a Single Interactor*. Proc. CHI, pp. 263-264, ACM Press, April 2000.

[25] Ratto, M., Shapiro, B. R., Griswold, W. G., Truong, T. M., Griswold W.G., *The ActiveClass Project:Experiments in Encouraging Classroom Participation*. Computer Support for Collaborative Learning, Kluwer, pp. 477-486, 2003.

[26] Rendezvous: developer.apple.com/macosx/rendezvous/

[27] Richardson T, Stafford-Fraser Q., Wood K.R., Hopper A, *Virtual Network Computing*. IEEE Trans. on Internet Computing, 2(1), pp. 33-38, 1998.

[28] Roseman M., Greenberg S., *Building Real Time Groupware with GroupKit, a Groupware Toolkit*. ACM ToCHI, 3(1), pp. 66-106, 1996.

[29] Stewart J., Bederson B.B, Druin A., *Single display groupware: A model for co-present collaboration.* Proc. CHI, ACM. pp. 286-293, 1999.

[30] *Synchronized Multimedia :* www.w3.org/AudioVideo/

[31] *Scalable Vector Graphics, XML Graphics for the Web*, W3C recommendation: www.w3.org/Graphics/SVG/

[32] Truong K.N., Abowd G.D., Brotherton J.A., *Personalizing the Capture of Public Experiences*. Proc. UIST, Asheville, NC, CHI Letters 1(1), ACM, pp. 121-130, 1999.

[33] Ubit GUI toolkit: www.enst.fr/elc/ubit/

[34] Yang C.-C., Yang Y.-Z., Lin K.-Y., *A SMIL-based Lesson Recording System for Distance Learning*. Proc. of Conf. on Distributed Multimedia Systems, pp. 486-489, 2001.

[35] Yoneki E., *Pronto: Mobile Gateway With Publish-Subscribe Paradigm Over Wireless Network*. DS Online: (dsonline.computer.org), Expert Authored Articles, May 2003.