Large scale and mobile group communication systems

Roland Parviainen

Division of Media Technology
Department of Computer Science and Electrical Engineering
Luleå University of Technology
S-971 87 Luleå
Sweden

May 2005

Supervisor

PhD Peter Parnes, Luleå University of Technology

Abstract

This doctoral thesis examines different attributes of large scale group communication systems such as scalability, security and mobility by studying two different prototype systems - mIR (multicast Interactive Radio) and MES (Mobile E-meeting Services). mIR is a system for large scale real-time music distribution, designed as an interactive radio system for the Internet. MES is a collection of tools for improving the use of e-meeting and video conferencing tools in a mobile environment.

The mIR prototype has been used to study scalability and security. Scalability in mIR concerns how to support as many users as possible without degrading the experience. This is achieved using IP multicast together with algorithms that limits the bandwidth usage regardless of the number of users. The work on security have focused on copy prevention through digital watermarking. By adding a unique watermark, i.e. a fingerprint, to each media copy a pirated copy can be traced back to a specific user, which can act as a deterrent. The thesis shows how we can combine the different goals of fingerprinting and IP-multicast while still maintaining the scalability features of multicast.

Many issues need to be considered if e-meetings and video conferencing will become widespread and popular. Scalability and security, discussed in the first part of the thesis are two examples, and the second part of the thesis tries to address a third issue: mobility. In particular we are interested in enabling access to an e-meeting in a mobile environment, where we often have difficult conditions such as bad network connections, the user only have access to the Internet through a web browser or the available devices are small and limited. In many cases it is currently impossible to participate in an e-meeting when you're not in the office. The prototype system developed in the second part of the thesis aims to enable participation from any location and device that have some sort of Internet connection. We try to achieve this by allowing a mobile user to access an e-meeting session from a web browser or from a Java enabled mobile phone. Further, the system makes it possible to review missed events in an e-meeting as it is likely that there are many times where no Internet connection at all is available.

The general style of work has been prototype driven with a goal of creating usable prototypes - i.e. the prototypes should be easy to deploy and it should be possible to use and test the prototypes daily. Most the prototypes described in this thesis have indeed been deployed and have seen daily use.

Contents

Publications

This thesis consists of an introduction and seven papers, of which six have been published and one has been submitted for review. I am the main author of all papers and have also done all work described in the thesis, with the exception of paper VI. See section 1.6 in the thesis introduction for more details. The papers are:

Paper I Roland Parviainen.
Multicast Interactive Radio.
Published in the proceedings of PA Java 99, April 1999, London, United Kingdom.

Paper II Roland Parviainan and Peter Parnes.
Large Scale Distributed Watermarking of Multicast Media Through Encryption.
Published in the proceedings of Communications and Multimedia Security, May 2001, Darmstadt, Germany.

Paper III Roland Parviainan and Peter Parnes.
Mobile Instant Messaging.
Published in the proceedings of ICT 2003, February, 2003, Papeete, Tahiti.

Paper IV Roland Parviainan and Peter Parnes.
The MIM Web Gateway to IP Multicast E-Meetings.
Published in the proceedings of SPIE Conference on Multimedia Computing and Networking 2004, January, 2004, San Jose, USA.

Paper V Roland Parviainan and Peter Parnes.
A Web Based History tool for Multicast e-Meeting Sessions.
Published in the proceedings of the IEEE International Conference of Multimedia and Expo (ICME'2004), June 2004, Taipei, Taiwan

Paper VI Mikael Drugge, Marcus Nilsson, Roland Parviainen, and Peter Parnes.
Experiences of using wearable computers for ambient telepresence and remote interaction.
Published in Proceedings of the 2004 ACM SIGMM Workshop on Effective Telepresence, October 2004, New York, USA.

Paper VII Roland Parviainen and Peter Parnes
E-Meeting Services for Mobile Users.
Submitted for review.

Preface

During the end of the final year of my undergraduate studies I heard that the Software Engineering division at the Computer Science and Electrical Engineering department was looking for PhD students. Within a week of my first talk with the head of the division, Dick Schefström, I was taking part in the start-up meeting of the EU project Roxy.

The goal was do my master thesis while working in the Roxy project and continue on to graduate studies when the thesis and my Master of Science degree was done, which happened during the spring of 1999.

It was my current supervisor, Peter Parnes, that suggested that I would start to work on the mIR - multicast Interactive Radio - prototype. This turned out to be a good choice as the continuation of that work resulted my Licentiate thesis, which is typically done midway between the MSc and PhD degrees, in November 2001.

After the Licentiate thesis I worked in project such as RadioSphere and VITAL Community, creating several new prototypes; MIM which led to MIDPro and the Marratech Pro web interface, which in turn led to the history tool. It was during this time our research group switched from being the Software Engineering division to the Media Technology division.

Luleå, March 2005
Roland Parviainen

Acknowledgments

First, I would like to thank my supervisor Dr Peter Parnes. It was Peter who wrote the first basic parts of mIR and who suggested that I should continue working on mIR, which resulted in my Licentiate thesis. I would also like to thank my collegues and friends at Media Technology and CDT, especially the ones who started roughly at the same time me such as PB and James. Finally, I would like to thank my family and friends (especially the d94 guys and hhhh) who made sure that not all my time have been spent i front of computers.

Dick Schefström deserves special thanks. He was one of the initiators behind CDT, started several of the projects I worked in and hired me, among other valuable things. It was also the distributed multimedia course he held while I was an undergraduate student that got me interested in the field of distributed multimedia. He always had new ideas about the role of computers and computer communication and most of the time, in the end, it seems like he was right.

Funding for the research have been provided by the following projects: the EU Esprit project 28410 Roxy, 5th Framework Program of the European Union, the RadioSphere project, which was supported by the Swedish Foundation for Strategic Research and the Objective 1 Norra Norrland - EU structural fund programme for Norra Norrland and the VITAL project. Support was also provided by the Centre for Distance-spanning Technology (CDT).

Luleå, March 2005
Roland Parviainen

Part 1

Thesis Introduction and Summary

1.1 Introduction

This doctoral thesis presents work on three specific problems of designing and using distributed multimedia systems. The type of distributed multimedia systems that I am interested in are systems where users consume media collectively and communicate with each other, using media such as audio and video. In order for these systems to be successful there are many technical problems that have to be solved. In this thesis, I focus on three different problems: *scalability*, so we can reach as many people as possible, *security*, so unauthorized users can not get access to sensitive information, and *mobility*, so users can continue to communicate efficiently while away from their office PC.

Scalability is needed to avoid designing a system that imposes unnecessary limits on its use, while security is needed to protect sensitive information, to ensure that the information is correct and to ensure availability of the system. These two problems are investigated in part I of the thesis using the prototype system mIR - multicast Interactive Radio. The problem can be summarized as *how do we design an scalable interactive real time media distribution system that is also secure?* While parts of this problem only seems applicable to one to many distribution systems, the results can easily be generalized to many to many systems

Part II focus on the single problem of mobility. While we increasingly are using personal computers to communicate, we also have more possibilities to be mobile using new devices and wireless communication networks. This combination creates some interesting problems to solve. The problem in part II can be summarized as: *what can be done to help mobile users to maintain contact with colleagues and other people and maintain a sense of presence and group awareness in a mobile environment where the users are often away from their office PC, traveling and visiting other locations?*

These problems are addressed by designing and testing prototypes and algorithms that shows possible ways to achieve a satisfactory solution.

1.2 Brief summary of the papers and prototypes

In this section we give a brief introduction to the different included papers and the different prototypes as well as a summary of the relationship between the different parts and prototypes in the thesis. Section 1.6 presents the different papers and their contribution in more detail.

The work described in the first part of this thesis started as a small project; to finish an existing application called mIR - multicast Interactive Radio. Contrary to the name, mIR was not interactive at all: it was just a "radio" application using the IP multicast protocol. It also had some other limitations; for example it could only use a certain kind of files, used a non-standard protocol, etc. The work on mIR, adding interactivity, standard compliance, etc. led to *paper I, "multicast Interactive Radio"*.

In that paper several problems were identified. For example, the system suffered from bad error resilience, the chat messages where unreliable, there was no congestion control, etc. The final goal was to create a system that was scalable, interactive, secure, reliable and capable of handling users with different bandwidths and packet loss ratios. In *paper II, Large*

Scale Distributed Watermarking of Multicast Media Through Encryption, we describe a new scalable algorithm for watermarking.

The prototype described in *paper III, Mobile Instant Messaging*, MIM, is a system for improving the usage of different chat, messaging and e-meeting (primarily the textual media) tools in a mobile environment. The target user is one who uses one or more of these tools at the same time, from more than one device or computer.

By using the World Wide Web (WWW), we can get access to e-meetings from almost anywhere were there is some Internet connectivity. *Paper IV, The MIM Web Gateway to IP Multicast E-Meetings* introduces an extended version of Marratech Pro which tries to provide a maximal amount of participation using only basic web standards such as HTTP and HTML.

In a similar fashion, the history tool presented in *paper V, A Web Based History tool for Multicast e-Meeting Sessions* is a modified version of Marratech Pro which captures events in real-time from an e-meeting and stores the information in a database. This information can be viewed and searched using a web interface, in order to see missed events, to quickly get a feeling of what have happened in an e-meeting etc.

Paper VI, Experiences of using wearable computers for ambient telepresence and remote interaction discusses the usage of Marratech Pro and wearable computers for providing telepresence to members of an e-meeting. The experiences of real-life trials are investigated both from the perspective of remote and local participants.

In the final paper, *paper VII, E-Meeting Services for Mobile Users*, the systems from paper IV and V and an extension to an extension to the mobile phone client of paper III are combined to a single, easy to deploy system for mobility called MES, Mobile E-meeting Services.

1.2.1 The relationship between the papers and prototypes

All papers are in one way or another linked to the mStar architecture described later in the thesis: in part I we examine some basic foundations of a distributed multimedia system such as mStar: scalability and security while we in part II try to extend the usability of the architecture to a mobile environment. For example, the results from paper II could have been used in Marratech Pro to enable traitor tracing: the possibility to find who leaked a recording of a sensitive e-meeting.

Figure 1.1 shows a high-level view of the relationship between the prototype systems of part II and different clients and services. The prototypes from paper IV and V are included into MES, described in paper VII, which combines the web interface, the history tool and the mobile phone prototype. The bottom part of the figures show what clients a user can use to communicate with different services, shown at the top. When for instance at the office, the regular Marratech Pro client which connects directly to an e-meeting would be used. If the Marratech Pro client is not available, a web browser or a Java enabled mobile phone connecting through the MES server can be used instead. More limited features of e-meetings together with other text-based services can also be through MIM. Both the regular Marratech Pro client and the MES system is used in paper VI to get access to e-meetings.

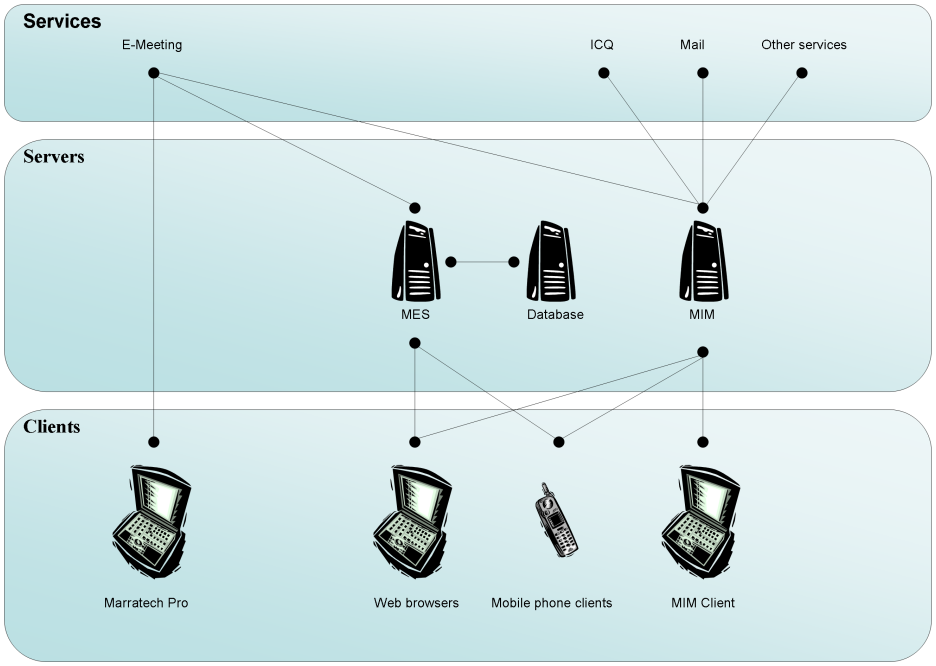

Figure 1.1: Overview of the relationship between the different prototypes described in part II of the thesis.

1.3 Thesis organization

The rest of the thesis is organized as follows. This chapter is an introduction and summary of the thesis. In the following section the background to the work and some technical background are described. The research issues and methodology is presented in section 1.5. In section 1.6 the included papers and their contributions are discussed. Section 1.7 presents work done during the graduate studies that is not included in the thesis. Section 1.8 describes trends and future research issues, both concerning the work in this thesis and in distributed multimedia in general. Finally, sections 1.9 and 1.10 concludes the introduction with a description of how the different prototypes have been used and a summary. The included papers are in chapters 2 to 8, followed by the bibliography.

1.4 Background

The work on the mStar[68] environment started in 1995 when CDT, the Centre for Distance Spanning Technology, started. CDT is a research organization with members both from academia and industry. The mStar environment was a collection of tools for scalable

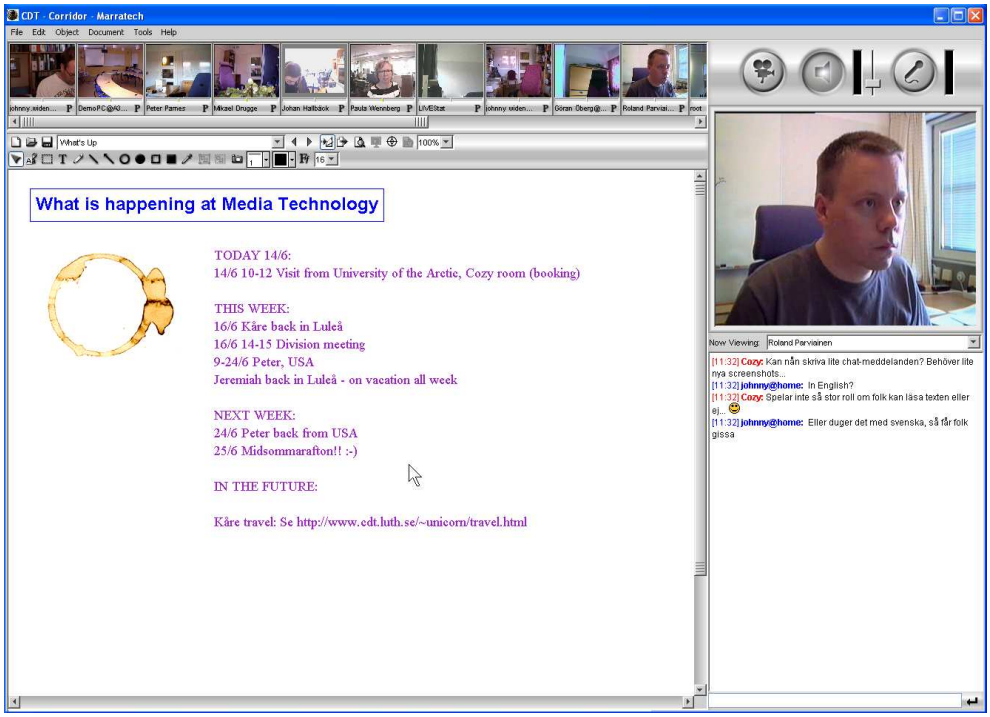

Figure 1.2: Screen-shot of Marratech Pro showing the e-meeting participants, video window, shared whiteboard and public chat. Private chat and the shared browser are not shown.

and effective distribution of real-time media and data between a large number of users and the main uses of mStar were for net based learning and distributed collaborative team-work. The goal has always been to create tools that can be used daily in our own work. mIR, presented in the first part of the thesis, was developed as a part of the mStar environment.

The work on the mStar environment led to formation of the company Marratech AB in 1998 and the Marratech Pro e-meeting product. The Media Technology research group has access to the source code to the Marratech products, which have enabled the prototypes in the second part of the thesis.

At the Media Technology research group collaborative work-spaces are used daily in a wide variety of situations. Example uses include allowing students to view classroom lectures from home, enabling members of discussion groups to interact from a distance and providing members of projects and research groups with increased presence of each other throughout their work day. The last mentioned case is referred to as the “e-corridor”.

In the e-corridor members of the session can be either active or passive and the possible uses of the workspace range from giving a formal presentation to passive monitoring of video. Communicating with colleagues through the e-corridor often replaces other communication

media such as e-mail, phone calls or personal visits. Users with broadband Internet access at home can choose to be part of the e-corridor, participate in meetings or follow presentations.

The software used for the e-corridor is Marratech Pro, which provides the users with the ability to send and receive audio and video to and from other participants. In addition there is a shared whiteboard and a shared web browser. The chat, shared whiteboard and shared web browser all use proprietary protocols while the audio and video tools use standard protocols such as RTP. Marratech Pro can use either IP multicast or unicast. In the latter case traffic is tunneled through a media gateway called the "E-meeting Portal". The video streams from participants in an e-meeting are presented in the "participants" window, which gives a thumbnail overview of all the video streams currently received from the group, while a "focus" window displays the video obtained from a single group member with a higher resolution and frame rate. All audio streams that are not manually muted by the user are mixed and played synchronized to the video streams. The shared whiteboard is mainly used to make simple drawings, application sharing and sharing Microsoft PowerPoint slides during presentations. It is implemented mostly in Java 2 with some parts such as audio and video decoding in native C/C++ code. See figure 1.2 for a screen-shot of Marratech Pro.

1.4.1 Video conferencing and e-meetings

There are mainly two types of video conferencing or e-meetings: room-based and desktop video conferencing. In room-based video conferencing involves a specifically equipped room and desktop video conferencing uses a standard PC, often the user's regular office computer.

We will often use the term "e-meeting" as a synonym for desktop video conferencing. "E-meeting" is often used instead of "video conferencing" as a way to distinguish modern systems from older room-based video conferencing, which many organizations have negative experiences of. While the system discussed in part I of the thesis is not strictly an e-meeting system, it share many of the same properties and the technologies described in that part are necessary or useful in e-meeting systems.

Many different fields of science are needed to create a successful system such as audio and video coding, real-time delivery of media such as audio and video, chat, shared applications etc, computer networking, user interface design and social dynamics and group psychology.

In [45] Kouadio and Pooch examines factors that affect the adoption of video conferencing systems. They identify five attributes that need further research: mobility, memory, application integration, collaboration, informality. In their opinion, the more of these attributes a system have, the more versatile and effective it gets. In this thesis we will investigate two of these attributes: memory and mobility. Other important attributes of video conferencing systems include scalability and security.

1.4.2 Privacy

Many people feel uncomfortable whenever there is a camera in the room, no matter if it is turned on or not. Through the work done in this thesis, we are not only asking people to distribute their picture to their colleagues when they are in the office, but to also make it possible

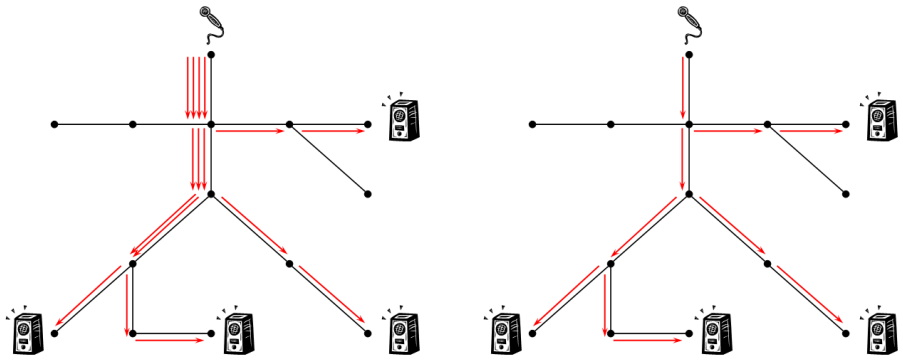

Figure 1.3: The left figure shows media distribution using unicast: most links need to transmit more than one copy of the stream. With IP multicast, shown in the right figure, each link only transmits a single copy of the stream.

to access it through the web from any Internet-connected location and to store it in a searchable database. As privacy can be very subjective, it is important for an e-meeting system to be flexible enough give users control over media. For example, it should be possible to turn off video transmission and to enable users to selectively allow or forbid storage of certain media. Some privacy issues are related to security, such as maintaining the confidentiality of information and proper authentication of people who can access information.

1.4.3 Multicast and IP multicast

Multicast is a data traffic model that delivers information to multiple destinations simultaneously. This can be compared to unicast, where information is delivered to a single destination and broadcast, where information is delivered to "all" destinations simultaneously.

One implementation of the multicast model is IP multicast[21], a network layer solution that works on TCP/IP networks. IP multicast does not only implement a multicast model, it also tries to achieve this using an efficient strategy: a single network link should never deliver the same information twice. Copies are only created when the network paths to information receivers diverge. A single copy is inserted into the network and delivered to anyone who asks for it, with additional copies created by routers only when needed. See figure 1.3 for an illustration of the difference between unicast and multicast.

When the work in this thesis started we believed that IP multicast would soon be available on a global scale; that many users with broadband Internet access would be able to use IP multicast to communicate with other users around the world efficiently. This has not happened yet. There are many reasons for the failure of IP multicast; some technological, others economical or political.

The multicast model is however a very attractive model for application developers, and there are ways to utilize this model without IP multicast. Marratech Pro uses a media gateway,

the m-Portal, to deliver information to all interested parties when IP multicast is not available. The application itself still works exactly as if IP multicast was used although on the network layer only unicast is used. Other solutions create a virtual network layer that implements multicast on the application layer, such as End System Multicast, ESM[17].

Real-time Transport Protocol

The Real-time Transport Protocol, RTP[88], is a standard protocol for transmitting real-time data such as audio and video. It was explicitly designed for multicast in mind, and is typically run on top of UDP[76]. However, RTP may be used with other underlying network or transport protocols. RTP provides no additional reliability and assumes that low levels of loss are acceptable for audio and video applications.

The data transport is augmented by a control protocol, the RTP Control Protocol (RTCP), which consists of session messages sent periodically to the same destination as the data.

SRRTP (the Scalable Reliable Real-time protocol)[64] is an extension to RTP that has been developed for the mStar architecture, which implements the ideas in the SRM [26] framework. It is used for reliable transport in Marratech Pro and later versions of mIR for chat messages, whiteboard pages, etc.

1.4.4 Media coding

In order to use multimedia data such as audio and video in a system such as video conferencing or real time streaming it have to encoded into a suitable form, that can be decoded and played at the receiving side. This coding and encoding is done by a codec, short encoder/decoder. Multimedia codecs are either loss-less or lossy. Loss-less codecs can recreate a bit-identical media object from its compressed form, while lossy codecs are destructive: information is lost during compression, but the recreated media object is still perceptually similar: the audible or visible difference from the original is small enough. What the limit of this difference is depends on the application and often a tradeoff with bandwidth has to be made: higher quality typically requires more bandwidth. Lossy codecs takes human anatomy into account: by knowing how the human visual system works and how humans perceive sounds the codecs can remove information that we do not perceive.

The prototypes of part II utilizes both lossy and loss-less codecs. For example, MES, described in paper VII, uses both PNG and JPEG as image codecs. PNG creates larger files for video snapshots than JPEG, but do not work in all web browsers. It also has faster compression times and is better suited for whiteboard pages than JPEG. PNG is also the only image codec supported on the J2ME platform used in part II of the thesis. Many of these differences are due to the fact that PNG is a loss-less codec while JPEG is a lossy codec. The whiteboard pages contain large continuous surfaces and sharp edges that most lossy codecs have difficulties in representing accurately. MPEG-1 layer III, MP3, is used in both mIR in part I and in the web interface and history tool in part II. Marratech Pro currently uses H.261 as a video codec, but H.264[93] will also be supported in the future. Several audio codecs especially designed for speech are supported such as iLBC, iPCM-wb and GSM.

When the media data is encoded it has to be transmitted and handled efficiently by the application that uses it. The Application Level Framing, ALF[19], principle shows way to achieve this: In ALF, the application should break data into suitable aggregates called Application Data Units, ADUs. It should be possible to process these ADUs independently of each other and arrival time. It is also important that framing done by the application into ADUs is maintained by lower network levels. Although the original ALF paper does not mention multicast at all, Handley [35] has shown that the ALF principle is important for designing scalable applications that uses multicast. As an example, RTP was designed with multicast and ALF in mind. This principle have been followed when designing the protocols used by mIR in part I and in the design of the binary protocol used in MIM and the mobile phone client of MES in part II.

1.4.5 Scalability

Scalability in group communication is the capability of a system to continue to perform well when the conditions or prerequisites changes. Available bandwidth, CPU power, screen size or number of concurrent users are examples of changing conditions.

One facet of scalability concerns media distribution and heterogeneous clients: users with widely different bandwidth capacities. Ideally, we want to support low bandwidth users but at the same time we do not want to compromise on the media quality for high bandwidth users.

A simple way of supporting heterogeneous clients is to use simulcast. With simulcast, more than one version of each media stream are sent, using different qualities and bandwidths. A user chooses a stream that has a bandwidth appropriate for the available network connection. One problem with this solution is that bandwidth is wasted. Media gateways and transcoders in the network can also be used to transcode a media stream to a version with lower quality and bandwidth. Some features of web interface described in paper IV can be seen as a version of both simulcast and a transcoder.

Another, often more effective, way of solving this problem is layered multicast, where the media stream is split up into different layers, each sent separately. Usually there is one base layer and several enhancement layers. These layers can be combined to create a stream of higher quality. The more layers a user receive, both the quality of the media and the necessary bandwidth increases. Users with limited bandwidth will only be able to receive the base layer and therefore a reduced quality. See for instance RLM, Receiver Driven Layered Multicast[57] for an example of a layered multicast system.

The type of scalability we study in paper I concerns how to maximize the number of simultaneous users. This usually depends on either bandwidth (at the source or the receiver, or both) or resources locally at the terminal, such as CPU and memory. Multicast, described in section 1.4.3 is one way of reducing the bandwidth needed as the number of users increase.

1.4.6 Security

Security is an important field with many aspects such as confidentiality, integrity and authorization. Confidentiality concerns how to make the information unavailable to unauthorized

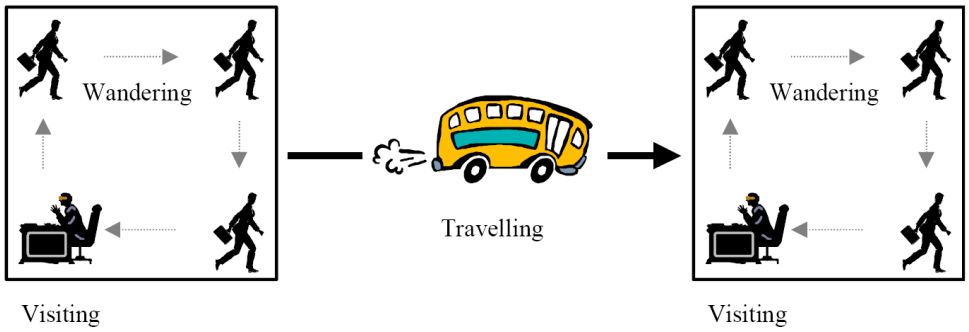

Figure 1.4: Three kinds of mobile modality according to Kristoffersen and Ljungberg.

persons, while integrity concerns how to make sure the information is not altered or falsified. Authorization is the process of deciding if something or someone has access to a service, and relies on proper authentication of the device or user.

Confidentiality and integrity for e-meetings can easily be achieved today. For example, the Secure Real-time Transport Protocol, SRTP[3], a profile of RTP, can be used provide both confidentiality and message authentication, along with replay protection for both RTP and RTCP traffic. There are several ways of achieving user authentication, depending on the level of security that is needed, from basic HTTP authentication to public key certificates, using for instance HTTPS.

Sometimes we also want to protect against unauthorized redistribution of information. Unfortunately, purely software based copy protection and prevention schemes have never worked and requiring that all users have some specific hardware solution is often not workable. Every proposed software solution has been cracked and most security experts believe it is impossible to create a system that can not be attacked. Instead we could try to discourage copying, by being able to find the user who made and spread an illegal copy. Fingerprinting is one way of achieving this. Fingerprinting is an application of digital watermarking, a technology that try to embed information in a media object that cannot be removed without destroying the media object itself. In fingerprinting each copy of a media object has a unique watermark, linked to for example the identity of the license owner of that particular object. If an illegal copy is found, the fingerprint is extracted and we get an indication of who made the copy.

1.4.7 Mobility

Mobility is a reality in today's life, both in our social life and in the work place. The recent developments in mobile devices and communication infrastructure have enabled people and organizations to work and communicate in new ways.

Mobile technologies often tries to achieve "Access anytime, anywhere". What this exactly means depends on the situation, the persons involved, the available technology, etc. In

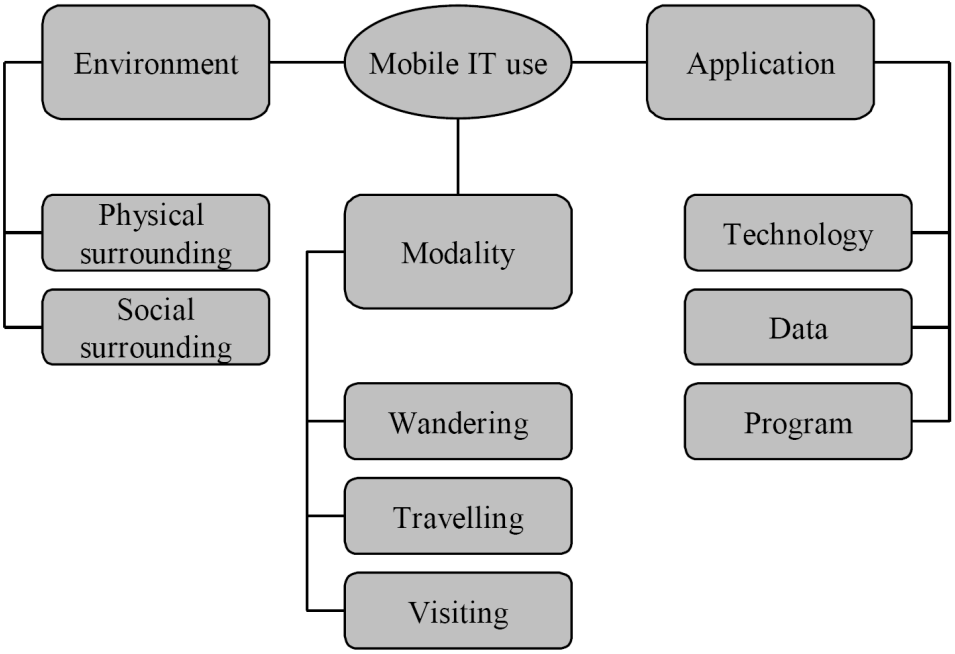

Figure 1.5: Kristoffersen and Ljungberg reference model of Mobile Informatics.

this thesis we want to enable access to the e-corridor or some other similar communication system, anytime and anywhere where there are some Internet access. E-meetings are often seen as a way to cut down on travel, but it can just as well be a way of maintaining contact with colleagues and friends while traveling.

Models of mobility

Kristoffersen and Ljungberg[46] presents of model of mobile IT use with three main components being *environment*, *modality* and *application*, where the environment is the physical and social surroundings, modality is the fundamental patterns of motion and application is the combination of technology, program and data that is used.

The physical environment is the observable surroundings where mobile IT use takes place, while the social environment concern factors such as both formal and informal structures and social conventions. Figure 1.4 shows the three different modalities; visiting, wandering and traveling. According to the model, traveling is an activity that takes place while in a vehicle. Visiting happens in one place for a temporal period of time, while wandering takes place when a user is moving around in for instance an office. Application is the technology part of mobile IT use, consisting of the three components of technology, data and program. Technology is hardware platform for the application (can be either stationary or portable) while a program processes the data part of the application. Figure 1.5 gives an overview of

this reference model. This model will be used in section 1.4.7 to describe the mobility side of part II of the thesis.

In the terms of Kakihara and Sørensen[42], there are three dimensions of mobility: spatiality (where), temporality (when) and contextuality (in what way, in what circumstance, toward which actor(s)). They point out that context and temporal aspects are also important parts of mobility.

Bellotti and Bly[4] describe a study of two design teams at two different locations. Here they define the term "local mobility", which is similar to the wandering modality of the Kristoffersen and Ljungberg model. Their results suggest two design goals for computer software for collaborative work, which the different prototypes try to reach:

1. To replicate for remote colleagues some of the opportunities for building awareness and for informal communication and coordination that local mobility enables

2. To reduce the penalties for distributed colleagues of trying to communicate, collaborate and coordinate with others who are away from their desks.

Perry et al.[74] describes a study of mobile workers that highlights different facets of access to remote people and information and different facets of anytime, anywhere. One of the activities they identified as a common theme for mobile work was the use of technologies for remote monitoring of the activities of colleagues, which is part of what we want to achieve in the second part of the thesis.

Technical problems and possibilities

Mobility comes with certain problems: both the available networks and devices are limited compared to desktop PCs. While the capabilities of mobile devices such as PDAs and mobile phones are steadily improving, they still have small screens, limited CPU power and memory, short battery life and limited input capabilities.

Faster 2.5G and 3G mobile phone networks such as EDGE, GPRS and UMTS in Europe and Cdma2000 in the US are becoming widespread while Wi-Fi technologies (IEEE 802.11a,b,g and soon n) are more and more common in locations such as city centers, airports and hotels. Compared to the fixed networks commonly available in offices and homes, these technologies usually have several limitations such as lower bandwidth, higher packet loss and higher latency. It is often not possible to maintain a continuous connection using these technologies as well, as there might only be coverage in hot-spots, the cost is too high or the connecting device has battery limitations.

In many places it is possible to use a PC, for instance either a personal laptop or a computer at a web cafe or public Internet terminals to connect to the Internet. Wherever there is some Internet access and PCs, there is usually at least WWW access through a web browser as a minimum. In many places, this is all that is available so web access to services has to be considered if maximum availability is important.

The prototypes according the Kristoffersen and Ljungberg mobility model

In this section we describe the different prototypes of part II of the thesis in the view of the Kristoffersen and Ljungberg mobility model. The *social environment* is characterized by a need to provide means for informal communication between a work group in order to for instance improve group awareness, ask questions and in general be part of the e-corridor when not in the office. In paper VI we're using a wearable computer to convey a feeling of "being there" to people who are not there physically. The user of the wearable computer have to interact, sometimes at the same time, both with locally present people and with people who are only present virtually.

The *physical environment* is anywhere where there is some Internet connectivity, except in paper VI where the environment is limited to a specific location such as a fair or exhibition. This results wide range of needed *technology* such fixed PCs, laptops, PDAs and mobile phones and any available network infrastructure such as the public Internet, both fixed and wireless, or mobile phone networks. The *data* is the information in an e-meeting: both current and past events and data such as chat messages, video data and activity measures. The programs are the prototypes them self together with web browsers.

The MIM prototype of paper III is intended to be used in all three *modalities*. The mobile phone client is useful while traveling and wandering and the regular Java application is designed to be used while visiting, while the web interface and history tool are intended while visiting or wandering. Paper VI concerns several participants: one is wandering while the other participants are visiting.

1.4.8 Web applications

Web applications are applications using web browsers for presentation, delivered from a server over the World Wide Web (WWW) or an intranet. The applications are created using HyperText Markup Language (HTML), a textual format, and presented in a web browser. HTML is typically transported over HTTP, the HyperText Transport Protocol, created with the purpose to provide a way to present HTML pages to users. A web browser makes a request over HTTP to a web server which parses the request and formats a response, typically in HTML. This request/response can result in further requests to retrieve for instance embedded images or style sheets. HTML is often combined with CSS, cascading style sheets, to separate document structure, defined in HTML, from presentation aspects such as fonts, colors and layout. While HTML is static, it can be combined with Javascript, a client-side scripting language, to create interactive web application. This combination is called Dynamic HTML, DHTML. Macromedia Flash is often used as an alternative to DHTML. Multimedia objects such as video and audio can be added using plug-ins, third party programs running inside the web browser.

Web applications can be particularly suitable for mobile systems as it available almost anywhere, without any need for software installation thanks to the ubiquity of the Internet and web browsers. While different web browsers still differ in standard support and plug-in availability it is relatively easy to restrict applications to universally supported specifications.

The drawbacks of a web application include the need for a network connection, no guarantees of what features the users' web browsers support and that the user experience can be quite different than a native application with higher response time, lack of common features such as drag and drop, different widgets, a more static interface, etc. This creates a trade off that application developers have to be aware of.

1.4.9 J2ME CLDC MIDP

In the cases where web applications can not be used, another solution is needed. These cases mostly concern small devices such as mobile phones that either lacks a web browser or where the displays are so small that the web applications can be hard to use. One obvious solution is to make sure that the web applications are usable, which can be hard and require specialized versions. Another solution is to create a specialized application for these devices. A typical example of this is the use of the J2ME CLDC MIDP 1.0[1][95] environment for developing applications for mobile phones in papers III and VII. A recent report by the Zelos Group[56] states that the annual sales of Java-capable phones will grow from 39 million to 63 million between 2004 and 2009 in the US alone. The list of J2ME devices from Sun Microsystems[2] lists more than 135 J2ME devices, from 19 different manufacturers.

One drawback of MIDP 1.0 is its limited features, especially when it comes to interaction with built-in phone capabilities such as cameras, the microphone and the speaker. Other development environments often enables more advanced features, but are often specific to certain phone brands or even certain models, which creates portability problems. However, if these advanced features are needed this can not be avoided.

1.5 Research issues and methodology

The work presented in this thesis is a clear example of applied research: we are trying to solve and answer specific and practical questions, through both experimental and theoretical approaches[33]. In order to find answers and solutions to these questions we have designed and implemented prototype systems which, if possible, have been deployed in a real world environment. This has been a very important part of the research as the problems we are trying to solve are of a practical and concrete nature; to test the prototypes in the same environment where research problems and questions have been observed are quite natural. The development of the prototypes has been an iterative process: initial prototypes have been tested and evaluated and then further enhanced based on feedback. The state of the prototypes as described in the included papers and this thesis represents a certain point in this process; in most cases the development has continued afterwards.

In order to evaluate the results we can ask the question: have the problem or question been solved or answered? We can also evaluate the results by comparing the properties of the technical solutions against the stated goals as well as other solutions in the literature.

[1] Java 2 Micro Edition, Connected Limited Device Configuration, Mobile Information Device Profile 1.0
[2] http://wireless.java.sun.com/device/

1.5.1 Research questions

The specific research questions the work has been aimed at can be divided into three fields: scalability, security and mobility:

Scalability How do we create a scalable real-time music distribution system? The concept of scalability is further discussed in 1.4.5, but briefly we are mostly interested in supporting as many simultaneous users as possible.

Security How do we protect the content in a large scale real-time media distribution system? While there are many important security aspects involved, we have focused on one specific problem: prevention of unauthorized copies. The main research issue here is to combine existing solutions for two problems that have opposite goals: scalable distribution and fingerprinting, linking the issue of security to scalability.

Mobility How to create a sense of group awareness in an environment where people are highly mobile, travel often and use a multitude of devices with high variations in access technologies and available resources such as CPU, memory and screen-size? What kind of systems can we create to help these mobile users communicate with colleagues, friends and family?

Sections 1.4.5, 1.4.6 and 1.4.7 discusses the issues of scalability, security, and mobility in more detail while section 1.6 describes the different included papers and what specific issues they deal with and outlines how we try to address them.

1.5.2 Delimitations

In this section we describe problems and methods that have not been considered or used. In general, we have focused on the technical problems and their solution, not on the usability on the system.

- In paper II, the existence of a robust watermarking algorithm is assumed and key distribution is not considered in detail.
- In part I, security aspects other than traitor tracing are assumed to be solved.
- In papers VI and V different algorithms for detecting activity in video streams have not been investigated.
- To create support for all features of Marratech Pro in a mobile environment is not attempted.
- No qualitative investigations have been performed, apart from informal interviews with users.
- No quantitative investigations have been done in order to evaluate the usability of various aspects of the prototypes.

Figure 1.6: The mIR receiver

1.6 Summary of included publications

This section describes the papers in the order they were published, and this also represents the chronological order in which the work was done. The papers have been reformatted, but not otherwise changed since their publication.

1.6.1 Paper I - multicast Interactive Radio

mIR - multicast Interactive Radio - is the oldest work included in the thesis and was started as part of my master thesis. The goal was to create an interactive and scalable real-time music distribution system, using a radio model. That is, one source sends the same audio stream to a large number of receivers. Scalability in this case means that we want to support a large number of receivers at the same time without degrading the service or running out of bandwidth. Interactivity can mean many things; in mIR we mean interaction between different listeners, for instance human to human. These interactions are done through a chat tool, the possibility give comments and see information about songs and voting for the next songs to be played.

If possible, we want to keep the bandwidth usage constant - in other words, not dependent on the number of receivers. The main technology used to achieve this in mIR was IP multicast, which enabled an efficient many-to-many distribution model. Still, as each receiver also send traffic such as chat messages and votes their bandwidth need to be controlled as well. To restrict the total bandwidth usage a receiver is allocated a bandwidth that is inversely proportional to the total number of receivers.

MPEG 1[60] was chosen as the audio codec (coder/decoder) to be used in mIR. At that time, MPEG 1, and specially MPEG 1 layer III audio (known as MP3), was the best choice due to several factors: it was an open standard, had a good compression ratio and codecs were easily and freely available. One drawback was that the standardized method at that time [36] had bad error resilience properties: a small amount of packet loss resulted in bad audio quality at the receiver side. Some solutions to this problem are discussed in section 1.7.

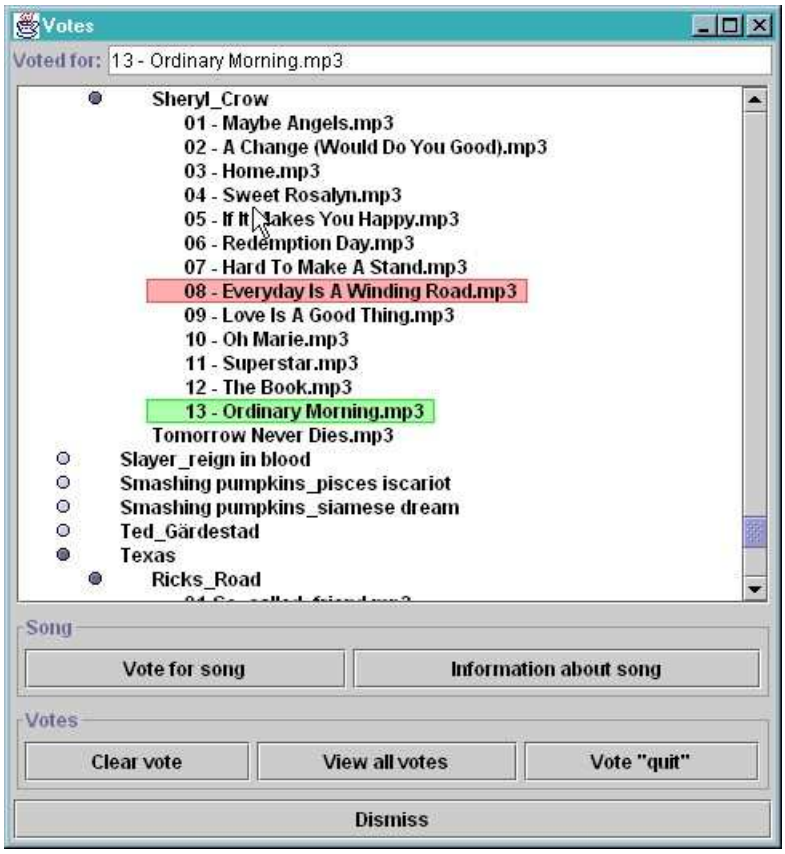

Figure 1.7: The mIR voting tool

1.6.2 Paper II - Large Scale Distributed Watermarking of Multicast Media Through Encryption

The second paper continues the work on the mIR prototype by focusing on a specific security problem: we try to create a scalable fingerprinting system for real-time multicast media delivery systems.

The goal of fingerprinting is contrary to the goal of multicast: in fingerprinting every user should receive a different, unique copy of for example a radio broadcast while in multicast we send the exact same data to each listener. In paper II ("Large Scale Distributed Watermarking of Multicast Media through Encryption") a way of combining these two goals is presented. By using encryption we can, for a cost of doubling the used bandwidth, ensure that each user receives a unique fingerprinted stream.

These solutions do not discourage or prevent larger groups of pirates or professional pirate copy facilities, but that is not the goal either. The issues of encryption and authentication, which also are very important security requirements are not discussed in this thesis. A lot other work in this area has been done; see for example [8][11][12][92][90].

The media stream that will be protected is divided into packets, and each packet is watermarked twice (with different watermarks), resulting in a pair of marked packets for each packet in the original, unmarked stream. Each marked packet is encrypted with a unique key before transmitted to all receivers. While receivers have access to both packets, only the decryption key to one of them is available to the user. The specific set of decryption keys each user has access to should be unique among all receivers. If this is the case, each user will receive a uniquely watermarked stream.

A prototype implementation has been incorporated into mIR, using Blowfish as the encryption algorithm and a simple, non-robust, watermarking method. Compared to the original mIR version, the watermarking enabled version requires slightly more than 2x the bandwidth and an increase in CPU usage. The extra bandwidth above 2x was due to padding added by the encryption algorithm. Key distribution was not included in the evaluation.

1.6.3 Paper III: Mobile Instant Messaging

Different Instant Messaging systems such as MSN Messenger, ICQ and AIM have become extremely popular during the last years. Other similar systems also exist, for example the chat media of Marratech Pro or the IRC chat networks. Unfortunately most of these systems have not been designed with mobility in mind. Users are assumed to connect from only one computer or device at a time, preferences and data such as history and contacts are often stored locally making it unavailable from other locations, etc.

Mobile Instant Messaging, MIM, is an attempt to create a system design with mobility in mind. It is not meant to replace the existing systems, but instead tries to adapt them to a mobile environment instead.

The MIM system consists of plug-ins, clients and a server. Plug-ins manage different existing IM systems and translates user identities and messages to and from the native IM format to the MIM format, acting as a proxy between MIM and the original IM system.

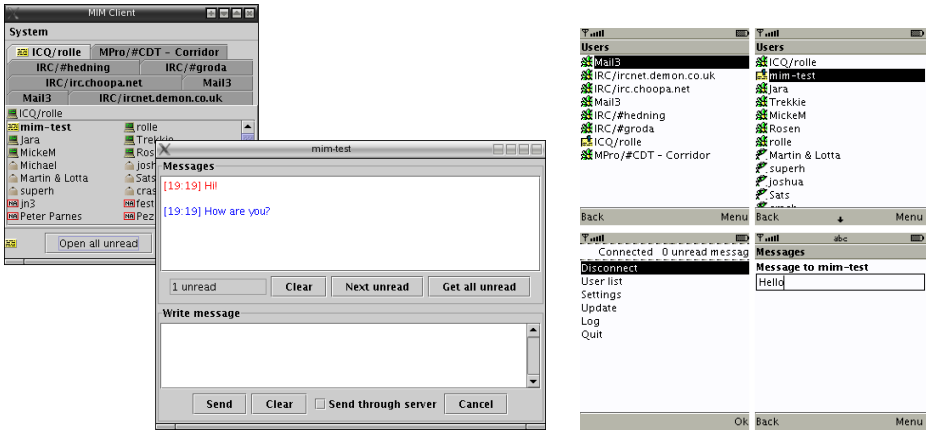

Figure 1.8: MIM running a PC and on a phone (The screen shots are from the J2ME Wireless Toolkit emulator).

Clients connect to the server and only know about the MIM format of users and messages. A user can at any time use any number of clients simultaneously. The server handles synchronization of messages and make sure all connected clients maintain the same state regarding users and messages. If a message is read using one client, it will also be marked as such in all other clients. No important state is stored at clients and they can disconnect and reconnect at any time.

The system was implemented in Java, and includes clients for PCs, PDAs and Java enabled mobile phones (using the J2ME CLDC MIDP environment) and plug-ins for ICQ, IRC, Marratech Pro and e-mail.

After the paper was published work was done to support more advanced features of Marratech Pro such as video snapshots and activity indicators. Some of this work is reused in paper VII for the MIDP Marratech Pro client. These more advanced features eventually led to the web interface, which is discussed in the next section.

1.6.4 Paper IV: The MIM Web Gateway to IP Multicast E-Meetings

When working with retrieving information from an e-meeting for the MIM system a dymanically updated web page containing video snapshots from the e-corridor that was created. This early prototype was well received and a more advanced prototype was requested.

The idea was to use the WWW as a way to enable participation in e-meetings from places where it earlier was impossible. This includes locations such as web cafes or Internet terminals at airports where it is often hard or impossible to use the regular e-meeting clients. To be able to get as wide support as possible only simple web standards such as HTTP 1.0 and HTML was used. The main goal so far is to provide group awareness and not full participation with complete audio and video support.

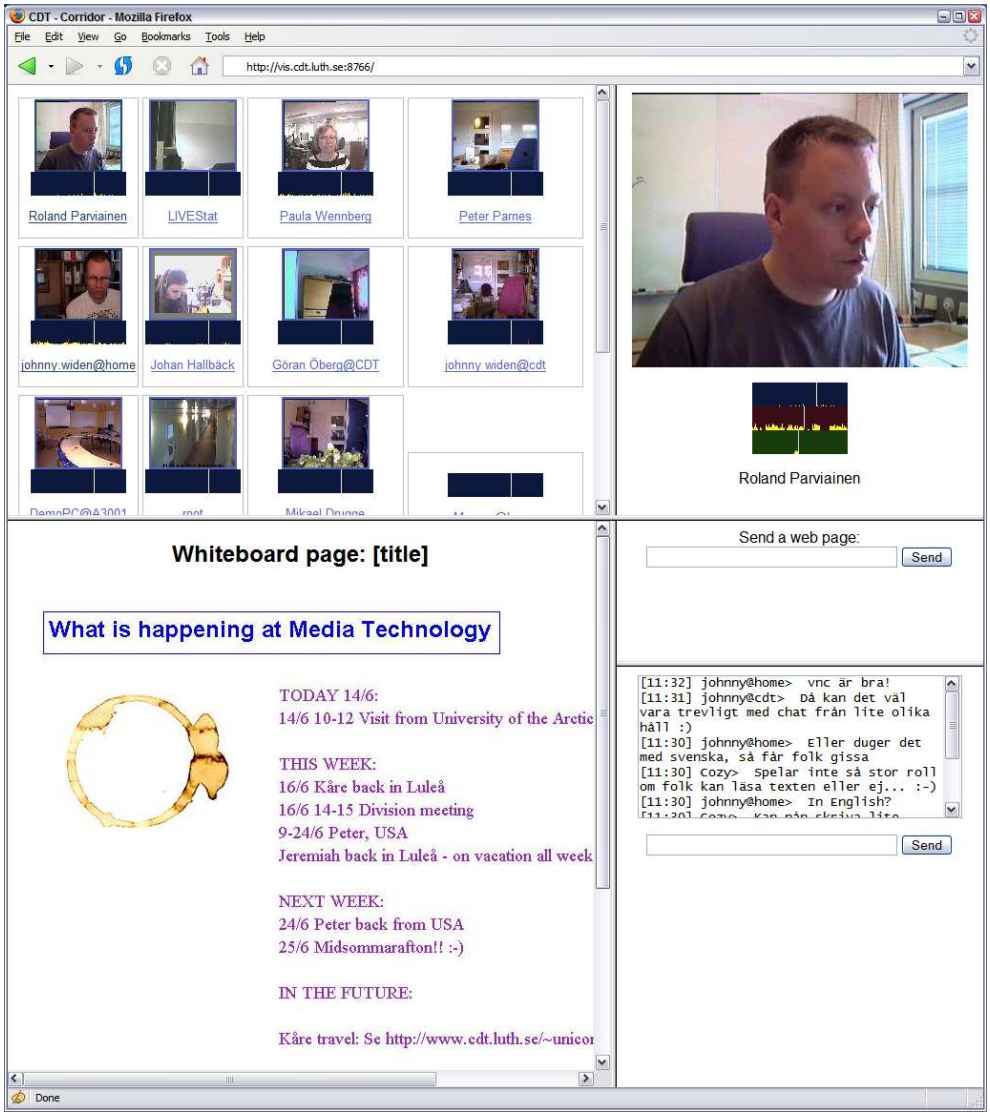

Figure 1.9: The web interface, showing the corridor, video and activity indicator for one user, a whiteboard page and a public chat window.

The web interface is implemented by extending Marratech Pro with a web server and component to facilitate communication between the users through the web server to an e-meeting. The web interface can either be incorporated into the same application that the user uses normally, running on for example a desktop computer in the office or run at a central location. In the former case each user accesses their own application to view the web interface directly and in the latter case all users access the same server.

The different media of Marratech Pro posed different challenges. Chat is a textual media and very easy to both present and input using web pages. However, private chat to/from the web interface to a single user in an e-meeting presented some problems: a single instance of Marratech Pro need to be able to represent several web users; otherwise there would be no way for the web interface of knowing who should be able to see an incoming private messages from a "normal" user. This was solved by adding virtual users to Marratech Pro and user authentication to the web interface. Virtual users are visible in the e-meeting and represent one user of the web interface, but all virtual users are managed by a single Marratech Pro instance.

Video relatively easy to show on a web page - especially if only group awareness is the main goal. Research has shown that for many uses, there is no need to support the full frame rate of live video streams: in [13] the required frame rate for video conferencing is reported as being no more than at least 5 frames per second. If we are only interested in providing awareness, [24] states that a frame rate as low as one snapshot every five minutes can provide group awareness in a work environment.

Mixed and decoded audio from the e-meeting is encoded as a 16kbit/s MP3 stream, which is served over HTTP. This is the only exception to the rule about using only simple web standards as an MP3-plug-in or an external MP3-player is needed. Audio input is not currently available.

Shared web pages are listed as a list of linked URLs. URLs can also be distributed to an e-meeting; as a URL is received from a web user Marratech Pro downloads the web page and distributes it to the other participants using reliable multicast as usual. Shared whiteboard pages are converted to JPEG or PNG images when requested by a user. Input from a web user to an e-meeting session is currently limited.

Activity indicators are added to make it easier to get an overview of the recent activity of the e-corridor or a single user since we cannot rely on the video stream to provide the information when video updates are far apart. The algorithm for detecting activity in video is similar to the one used in NYNEX Portholes[48]. Other activities such as sending audio or chat messages are also added to the indicator. Different activity is displayed in different colors and painted in a specific order: video first, then audio, whiteboard, web and finally chat. The reason is that for instance video activity, which is happens frequently, should not hide the more important chat activity.

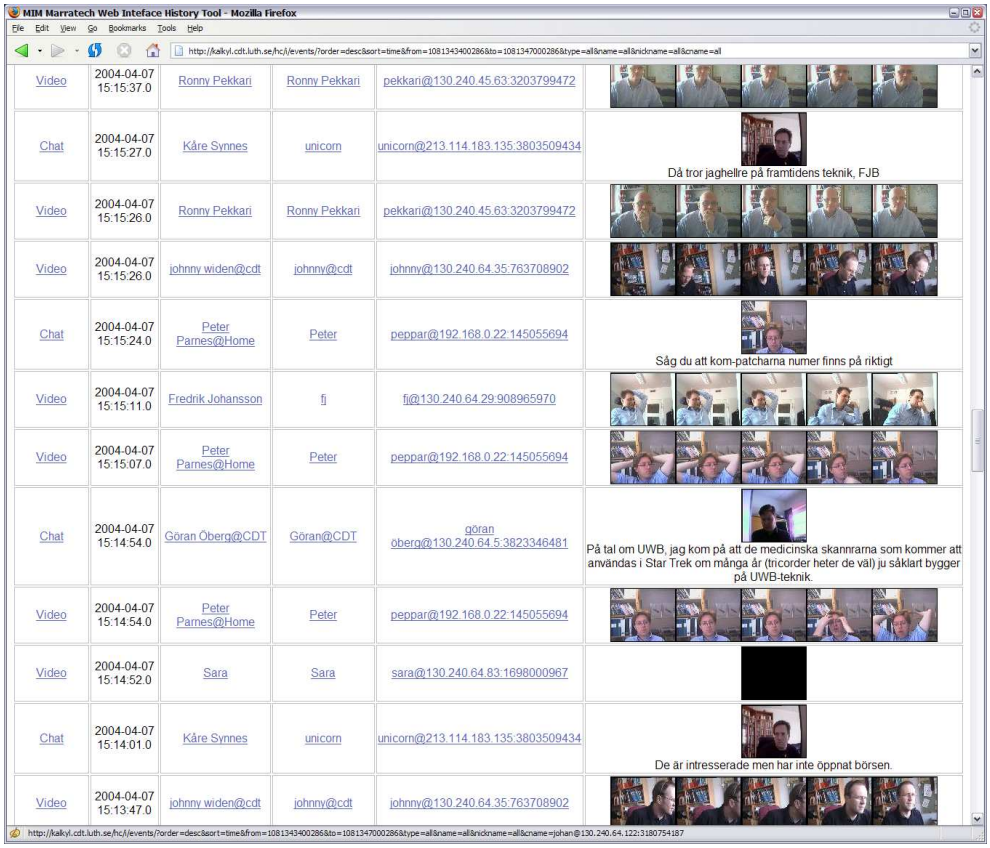

Figure 1.10: The event view of the history tool.

1.6.5 Paper V: A Web Based History tool for Multicast E-Meeting Sessions

During the initial evaluation of the web interface described in paper IV people expressed that it would be interesting to make past information available as well. All data were easily available as JPG or PNG images, text or MP3 frames in the web interface, so why not store it so it can be retrieved later?

As a mobile user travels or moves between devices and networks, there will be times, sometimes quite long, were it will be impossible or not practical to access an e-meeting even with the previously described tools. To be able to see what was missed during these times, the history tool was developed. Tools for reviewing the history of an e-meeting have several uses in a mobile environment such as enhancing group awareness and reviewing past events. A tool like this can also be used for surveillance and monitoring of staff and can have severe

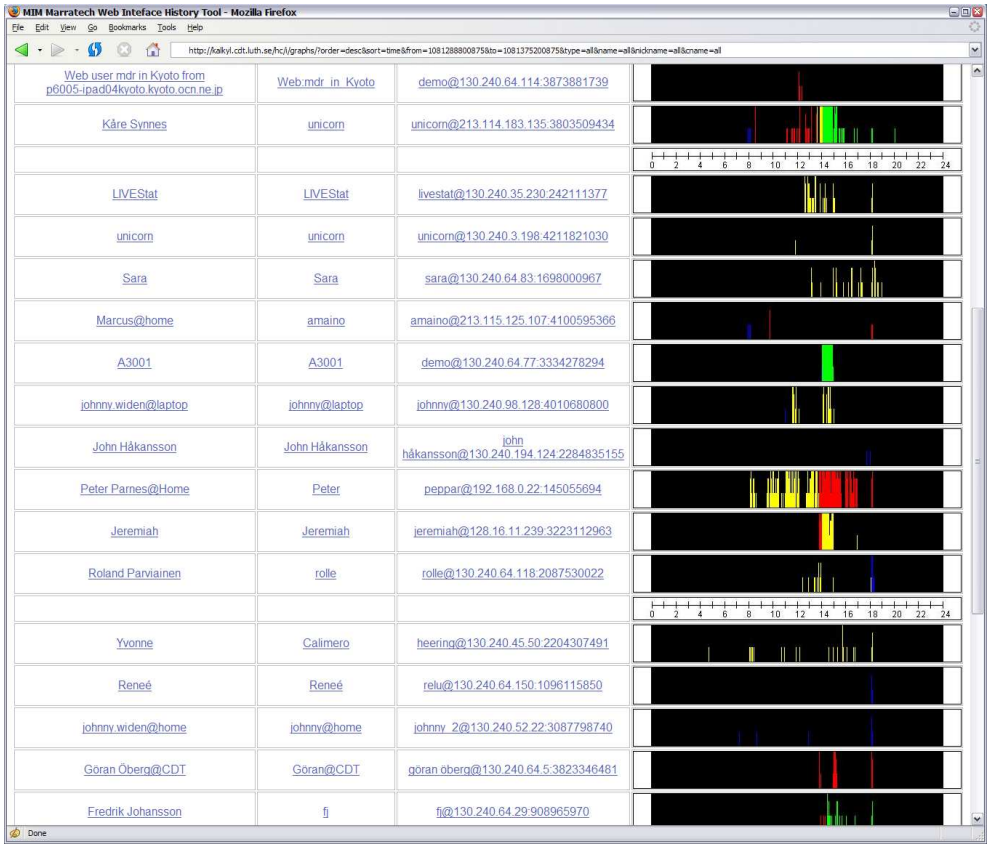

Figure 1.11: The activity indicator view of the history tool. The time span is one day.

privacy issues. The implementation of a tool like this in a work place will need to seriously consider privacy issues.

The history tool is implemented in three parts: first, Marratech Pro is modified to collect data and store it an SQL database, which is the second part. Finally, a Java Servlet is used as a front-end to present the information to users through a web interface.

A main goal has been to store as little as possible to reduce the storage needs, which also can expand the history to longer time frames. While the capacity of hard drives increases rapidly and the prices are reduced there are always limitations and recording an e-meeting session directly can use up to several Mbit/s. During playback it can be hard and cumbersome and take long time to find a specific point in time or get a quick overview of the events.

Video data is the media that requires the most bandwidth, so the focus has mostly been on reducing the storage needs for video. While video is a continuous media, only changes are important, as there for example is no need to store 25 frames/s during night when nothing happens. In the history tool video frames are only stored when there has been a large enough

change in the picture. The history tool compares consecutive video frames against each other to calculate a video activity value. If this value differs enough from a dynamically calculated threshold, a video event is generated. As a single video frame might not be enough to capture the meaning of an event (is someone sitting down or getting up from their chair?) five frames are stored for each event, localized in time both before and after the event was detected.

1.6.6 Paper VI: Experiences of using wearable computers for ambient telepresence and remote interaction

This is the only paper in the thesis were I am not the main author. Mikael Drugge's and Marcus Nilsson's, the main authors and contributors, research are focused on different aspects of wearable computers. This paper presents experiences of using wearable computers to provide telepresence to members of an e-meeting.

A wearable computer user (the "remote user") joins an e-meeting, usually the Media Technology e-corridor, when visiting fairs and exhibitions. Figure 1.12 shows the wearable computer being worn by Mikael Drugge. The idea is to provide a feeling of "being there" to those who are locate elsewhere. An unmodified version of Marratech Pro was used for communicating with the e-meeting which have some drawbacks, as the user interface is not adapted to the specific concerns of a wearable computer. The traditional WIMP-paradigm does not work as well on a wearable computer as on a desktop computer. The research issues concerns how wearable computers can provide telepresence in real-life scenarios and what is needed to enhance the experience.

The conclusions of the experiments shows that today's wearable computing technology can provide a very encompassing form of telepresence, but the time to prepare, setup and use the system will influence how much it will be used. The aesthetical appearance of the wearable computing equipment is important as this will influence people at the remote location. Remote users need to be free to interact with their environment without social or technical obstacles and the equipment should not interfere with the environment and therefore have to be unobtrusive and less noticeable than today.

1.6.7 Paper VII: E-Meeting Services for Mobile Users

This paper combines paper IV and V and introduces a new mobile phone client, the MIDLet client. The MIDLet client focuses exclusively at providing access to e-meeting sessions through Marratech Pro to a Java enabled mobile phone, as apart from the MIM MIDP client introduced in section 1.6.3 and paper III. The three tools are presented together as a single system called Mobile E-meeting Services (MES). All three tools have similar goals and purpose and need similar services from Marratech Pro, which have been collected into a single component, called the *mobility manager*.

While Internet-connected computers that can be used to access the web interface and the history tool are common, they are not everywhere. If it would be possible to access the e-meeting from a device such a mobile phone, the range of locations from where a mobile user can be part of a session increases greatly. To exploit this possibility a Java 2 MIDP client, the

Figure 1.12: The wearable computer prototype being worn by Mikael Drugge, one of the authors of paper VI.

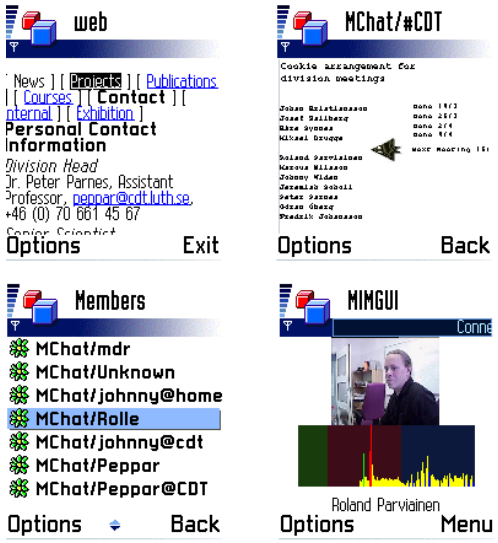

Figure 1.13: Examples of the shared web browser, whiteboard, user list and combined activity and video display running on a Nokia 7650 phone.

MIDlet client, was developed using the MIM MIDP client as base. The new client connect directly to the same HTTP server the other tools of MES utilizes. It supports chat, viewing of video snapshots, activity indicators, whiteboard pages and distributed web pages. The current Java implementations are not advanced enough to support any form of audio communication.

By combining the three different tools into one system, we get a set of simple and easy to deploy services which can enable users to access an e-meeting from locations and devices where it was previously impossible.

1.6.8 Contributions

I am the main author of and contributor to all papers except paper VI, *Experiences of using wearable computers for ambient telepresence and remote interaction*, where Mikael Drugge and Marcus Nilsson did most of the work. My main contribution to that paper is the usage of the web interface and the history tool. Below is a list the most important contributions of this thesis:

Paper I *Multicast Interactive Radio.*

The main contribution of paper I is the design and implementation of a scalable and interactive real-time music distribution system. The use of multicast and algorithms that limit bandwidth usage enables the system to scale as the number of users increase.

Paper II *Large Scale Distributed Watermarking of Multicast Media Through Encryption.*

The main contribution of paper III is a novel algorithm for uniquely fingerprinting media streams in a multicast based large scale distribution system, in which the necessary bandwidth is independent from the number of receivers.

Paper III *Mobile Instant Messaging.*

The main contribution of paper III is that it presents a system for efficient and convenient textual communication in a mobile environment, where users simultaneously utilizes several Internet connected devices such as mobile phones and computers to access chat services online.

Paper IV *The MIM Web Gateway to IP Multicast E-Meetings.*

The main contribution of paper IV is the design and implementation of a web gateway system for e-meetings which enables participation in through web browsers. The inclusion of activity indicators gives users a possibility to quickly get an overview of recent activity. A template system enables easy customization of the look and feel of the web interface.

Paper V *A Web Based History tool for Multicast E-Meeting Sessions.*

The main contribution of paper V is a history tool for e-meetings with a web interface for easy access enabling users to easily see past events and activity in an e-meeting. Other contributions include the use of reducing storage needs by only storing video snapshots when necessary, the use of a SQL database which enables easy access statistics and information through other query tools and the use of activity graphs to quickly get an overview of the activity in a session.

Paper VI *Experiences of using wearable computers for ambient telepresence and remote interaction.*

The main contribution of paper VI is the presentation of the key issues that prohibit the remote interaction from being entirely seamless and suggestions on how those problems can be resolved or alleviated.

Paper VII *E-Meeting Services for Mobile Users.*

The main contribution of paper VII is the combination of the systems described in paper IV and V together with a mobile phone client into a coherent, easily deployable system for mobile access to e-meetings.

1.7 Work not included in the thesis

.

This section presents work that has been done during the graduate studies but is not included in the thesis as it has never been published.

1.7.1 Advanced mIR features

Some work has been done on improving the error resilience of mIR, which is not otherwise discussed in this thesis. Some of it is included in the licentiate thesis, but not otherwise published. An implementation of forward error correction (FEC[89]) was added to the RTP stack, which can be used to protect any kind of RTP streams against packet loss. A more advanced RTP packetization scheme that further improves the error resilience of MP3 was also implemented. Finally, a layered multicast scheme was implemented utilizing the Ogg Vorbis audio codec. In this scheme, layers could have different levels of error protection through FEC.

1.7.2 Dynamic groups

Work is now being done on a new model for e-meetings, by creating yet another prototype based on Marratech Pro. E-meeting software such as Marratech Pro often use a room metaphor which makes it hard for a user to participate with more than a few groups of people at a time. It is also often hard to quickly create a new group when needed, which can limit the level of informality the tools can support.

Many instant messaging tools and social network/community systems such as Friendster and Orkut have support for grouping people together and to dynamically create groups as needed. However, these systems do not handle audio and video communication well or at all for groups.

The idea is to extend Marratech Pro with support for dynamic groups, where a user can easily select whom to interact with based on a number of parameters, some supported today and some not. Dynamic groups can be selected or created by applying parameters such as geographic position (“within 1km of my current location”), user state (“available for chat”) or work relation (“part of my research group”).

1.8 Future directions

In this section we discuss technological and commercial developments since the papers were published as well as future research directions. Related work is discussed in more detail in the different papers them self.

Since the publication of papers I and II, Internet radio have grown substantially even though we have not seen a wide scale IP multicast deployment. For example Winamp Radio[3]

[3]http://www.shoutcast.com/

claims to have 3 million listeners per month and thousands of stations and Live365[4] similarly claim to have more than 2.6 million listeners a month. The most popular stations can have more than 5-10000 listeners at a time, requiring an aggregate bandwidth of several hundreds Mbit/s. Many commercial radio stations that also broadcast online, often using proprietary solutions such as Real Audio or Windows Media. The bandwidth requirements of these solutions does not enable regular users to broadcast audio to a large audience in the same way IP multicast would have, even though broadband Internet connections are common. If IP multicast is not deployed a replacement is needed, and many different research groups are working on creating overlay networks at the application layer as a replacement distribution technology. Different systems using P2P technology (which in the case of real time distribution can be seen as an application layer multicast protocol) already exist such as Mercora P2PRadio[5] which have more than 2000 private radio stations and PeerCast[6].

Most of these distribution systems do not employ any specific security technology to protect against unauthorized copying. Non real time music distribution systems such as online music stores on the other hand does often use software based digital right management (DRM) systems to protect against copying. Most have been broken in one way or another.

While watermarking has not been widely used in audio streaming, it is now starting to being used in other media distribution systems. For example, the Swedish company Inprodicon plans to sell music as traceable MP3 files, utilizing watermarking technology by Musictrace[7], a spin-off company of Fraunhofer. Similar watermarking ideas as the algorithm in paper II, where an identical stream of data can generate a unique fingerprinted stream at a receiver or player, will be used in the Advanced Access Content System, AACS, designed for use in the new HD DVD standard.

There have been some new technological developments for multicasting audio, such as a new RTP packetization format[27] that enables better error resilience, which together with forward error correction[89] can maintain reasonable audio quality even in high packet loss environments. Although other audio codecs such as AAC, Windows Media and Ogg Vorbis have managed to make a fairly large impact and have better audio quality at a given bit rate, MP3 is still the most popular format.

Advanced web applications are becoming more popular and the features developers manage to create are becoming more sophisticated; mail.google.com and maps.google.com are just two examples of what can be done with only HTML, CSS and Javascript with modern browsers. Re-creating the web interface and history tool using these tools or other advanced technologies such as Macromedia Flash or Java Applets could make the tools more interactive and flexible. Similarly, other development environments than J2ME for mobile phones could be used to enable more advanced features for the mobile phone client.

The research done in part I is an example of the research into the basic foundations, or "nuts and bolts" of larger multimedia systems that were typical for multimedia research at that time. Current research trends and suggestions for future research directions are described in for instance the report from the ACM SIGMM strategic retreat[84], which presents sugges-

[4] http://www.live365
[5] http://www.mercora.com/
[6] http://www.peercast.org/
[7] http://www.musictrace.de/

tions for future research in multimedia. The report suggests the research community focus on solving three grand challenges:

1. Make authoring complex multimedia titles as easy as using a word processor or drawing program.
2. Make interactions with remote people and environments nearly the same as interactions with local people and environments.
3. Make capturing, storing, finding, and using digital media an everyday occurrence in our computing environment.

Items 2) and 3) closely relates to the work done in part II of the thesis; for example, MIM in paper III and the web interface in paper IV intends to improve the interactions between local and remote participants, and in particular help remote people get a better sense of presence and group awareness. The history tool in paper V was created with the goal of capturing and making the media of an e-meeting easily available, while the work done in paper VI on telepresence through wearable computers and e-meetings in particular fits item 2): the goal is practically identical. The video conferencing research issues (mobility, memory, application integration, collaboration, informality) identified by Kouadio and Pooch, described in section 1.4.1 also fits into these challenges.

One particular issue that have been avoided in this thesis is the usability of the prototypes, which of course is an important part when the systems are intended to be used daily, and should be addressed in the future. Other researchers in the media technology research group are working on different HCI and usability issues in e-meetings in general.

1.9 Real world prototype use

The mIR system designed in paper I was running on the university campus until 2003, when the server computer was shut down for unrelated reasons. The watermarking algorithm designed in paper II has never been used outside tests, as there has not been any need for fingerprinting in our uses of mIR.

MIM is used daily to connect to Marratech Pro and to chat services such as ICQ and IRC. The typical use is to use several PC clients running simultaneously and the Java phone client now and then when in locations where PCs are not available.

Apart from MIM, the web interface is the prototype that has seen the most frequent use. It is used when Marratech Pro is not available and there is not possible or no time to install or use it. The history tool has been used as intended to see what happened, what was said, who was online etc. It has been used both in every day, mostly non-mobile, use and when traveling or visiting. It has also been used for creating long term statistics of e-meeting use for usability research. The mobile phone client for e-meetings has not seen wide use, but there have been instances where no other means of communication have been available.

The telepresence system described in paper VI has seen further use in other fairs and exhibitions that the media technology research group has visited.

1.10 Summary

The first part of the thesis discussed multicast Interactive Radio - mIR, a scalable real-time music distribution system. The work on mIR focused on making it scalable, that is, to support as many listeners as possible. A novel distributed watermarking approach for scalable watermarking of media stream was also presented.

In the second part of the thesis, a prototype system was introduced which includes different tools for supporting mobile users of e-meetings and allows a user to access an e-meeting session from a web browser or a Java-enabled mobile phone and thus extending the e-meeting to devices and locations where it previously was hard to participate. A history tool enables users to easily view past and missed events, making it possible to catch up with events in a session after disruptions in network connectivity. We have also described experiments in using Marratech Pro together with wearable computers for telepresence.

The main contributions of the thesis are the prototypes them self. All of them have been used in the real world, and all of the prototypes in part II of the thesis are still in daily use. Other important contributions are the distributed watermarking algorithm from paper II and the investigation of issues of providing telepresence using wearable computers in paper VI.

Part 2

Multicast Interactive Radio

Multicast Interactive Radio

Roland Parviainen
Centre for Distance-spanning Technology
Department of Computer Science,
Luleå University of Technology,
971 87 Luleå, Sweden

March, 1999

Abstract

This document describes multicast Interactive Radio, a system for distributing high quality audio over the Internet with interactivity. The system is implemented in the platform independent language Java, and uses IP multicast for distributing the audio.

2.1 Introduction

mIR – multicast Interactive Radio – is a suite of applications for transmitting MPEG [59] audio files using IP multicast. mIR consists of three applications: the receiver, the voting tool and the transmitter. The receiver uses an external MPEG player for actually playing out the audio. The voting tool is the application that makes mIR interactive; users can vote for songs, get more information about all the songs that are available or engage in a discussion with the other users.

All applications are written in the platform independent language Java [32] and have been tested with Sun Solaris, Linux, Windows 95 and Windows NT.

Two different "channels" are used for sending data, and each channel uses a different multicast address. On one channel (the audio channel) the actual audio data are sent, on the other channel (the information channel) information about which songs are available at the transmitter and messages to the discussion is sent. The voting tool sends information on this channel too, while the receiver only listens to the audio channel and does not send any information at all to any channel.

The focus when developing mIR has been on interactivity and scalability. The mIR applications can be downloaded from the author's home-page[1].

[1] <URL:http://www.cdt.luth.se/~rolle/mIR/>

2.1.1 Technical background

IP multicast

IP multicast [21] provides efficient many-to-many data distribution in an Internet environment. Senders send datagrams to a "host group", a set of zero or more hosts identified by a single IP destination address. The datagrams are delivered to all members of the host group by the network infrastructure in an optimized way. Neither receivers nor senders need to know who or where the other receivers and/or senders are. The membership of a host group is dynamic; hosts may join and leave groups at any time.

Real-time Transport Protocol

The Real-time Transport Protocol, RTP [87], is a standard protocol for transmitting real-time data such as audio and video. It was explicitly designed for multicast in mind, and is typically run on top of UDP [76]. However, RTP may be used with other underlying network or transport protocols. RTP provides no additional reliability and assumes that low levels of loss are acceptable for audio and video applications.

The data transport is augmented by a control protocol, the RTP Control Protocol (RTCP), which consists of session messages sent periodically to the same destination as the data.

RTP Packet The RTP data packet header contains the following among other things:

- A sequence number, that can be used to discover packet loss and out of order packets.
- A synchronization source (SSRC) which identifies the source.
- A timestamp.
- A payload type field identifying the format of the RTP payload.
- A marker bit. The interpretation of the marker bit is defined by a profile. For audio transmission, the marker bit is usually used for signaling the start of a talk-spurt.

RTCP RTCP is based on periodic transmission of control packets to all participants in the session, using the same distribution mechanism as the data packets. RTCP packets can contain, among other things, any of the following:

- Source description items (SDES) to identify the participant (name, email, etc) and associate the information with the SSRC in the RTP packets.
- Sender reports for transmission and reception statistics from participants that are active senders.
- Receiver reports for reception statistics from participants that are not active senders.

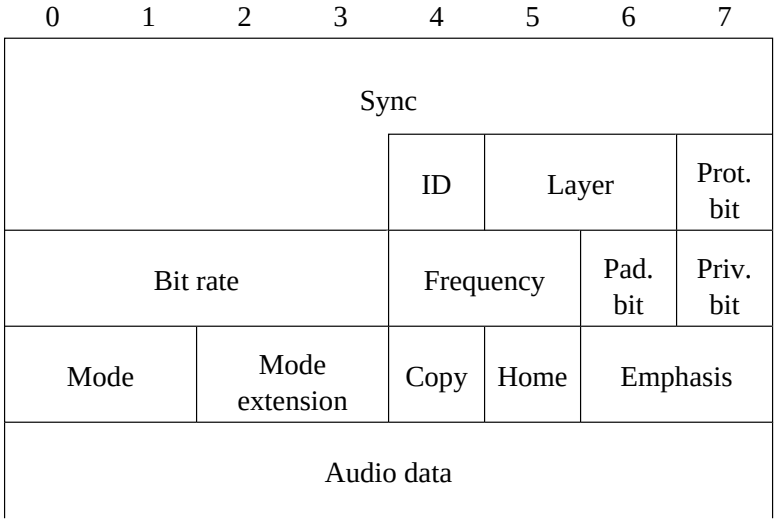

Figure 2.1: The MPEG Audio header

MPEG-1 Audio

MPEG is a working group in a subcommittee of ISO/IEC that generates generic standards for digital video and audio compression.

MPEG works in phases. These phases are denoted by MPEG-1, MPEG-2, MPEG-4 and MPEG-7. Both in MPEG-1 and in MPEG-2, three different audio levels are defined. The layers are denoted by roman figures, i.e. Layer I, Layer II and Layer III. Basically, the complexity of the encoder and decoder, the encoder/decoder delay, and the coding efficiency increase when going from Layer I via Layer II to Layer III.

An MPEG audio stream consists of frames. A frame consist of a header and a data block. The data block contains the actual audio data, while the header contains information about the audio data. See figure 2.1 for an overview of the MPEG audio frame structure. The fields in the header that are relevant in this thesis are sync, layer, bit rate and frequency. The sync fields are 12 set bits in a row, and mark the beginning of a frame. The layer, bit rate and frequency fields describe which MPEG audio layer is used, the bit rate of the stream and the frequency of the audio data.

2.1.2 Scalability

The definition of scalable is seemingly simple: to be able to scale. In other words: How well a solution to some problem will work when the size of the problem increases. For the kind of applications that we consider in this report, the problem size is the number of clients.

Depending on the application, a client can either be an actual user, or just a program that is running.

There are two main aspects of scalability that have to be considered for distributed interactive applications:

1. **The scalability of the application itself.**
   The common mode of operation for interactive applications is to react on events sent from other clients. The response for the client time should be constant with regards to the number of clients, i.e. $O(1)$. Although memory is cheap these days, the memory usage needs to be limited too. The memory requirements of the applications should also be $O(1)$, if possible.

2. **The bandwidth requirements.**
   Although basic data distribution using IP multicast scales well to large groups, there are still some issues that must be considered. One issue here is reliable data distribution. Section 2.2.1 discusses this in more detail. Another issue is the total bandwidth used, which should grow as little as possible when the number of clients is increased. Preferably the bandwidth should be constant ($O(1)$), but this is rarely possible.

2.1.3 Application Level Framing

The design principle of Application Level Framing (ALF) [19] was applied during the design of these applications. The ALF principle states that an application should break the data into suitable aggregates, and the lower levels should preserve these frame boundaries. These aggregates are called Application Data Units, or ADUs. The fundamental characteristics of the definition of the ALF design principle is that it should be possible to process each ADU as it arrives, even if the ADUs arrive out of order.

Although the original ALF paper does not mention multicast at all, Handley [35] has shown that the ALF principle is important for designing scalable applications that uses multicast. As an example, RTP was designed with multicast and ALF in mind.

In this report, packets and messages are used as ADUs.

2.2 mIR – multicast Interactive Radio

2.2.1 The transmitter

The transmitter consists of two parts: one that receives votes from information channel, and one part that transmit the MPEG files to the audio channel. The received votes are stored in a hashtable, and these votes are used to select which song will be transmitted next. All MPEG-1 [60] audio files (layer I,II and III) can be transmitted. The audio data is multicasted using the RTP [87] protocol, using the payload type 14 for MPEG audio [86], [36]. See figure 2.2 for an overview of the functionality of the transmitter.

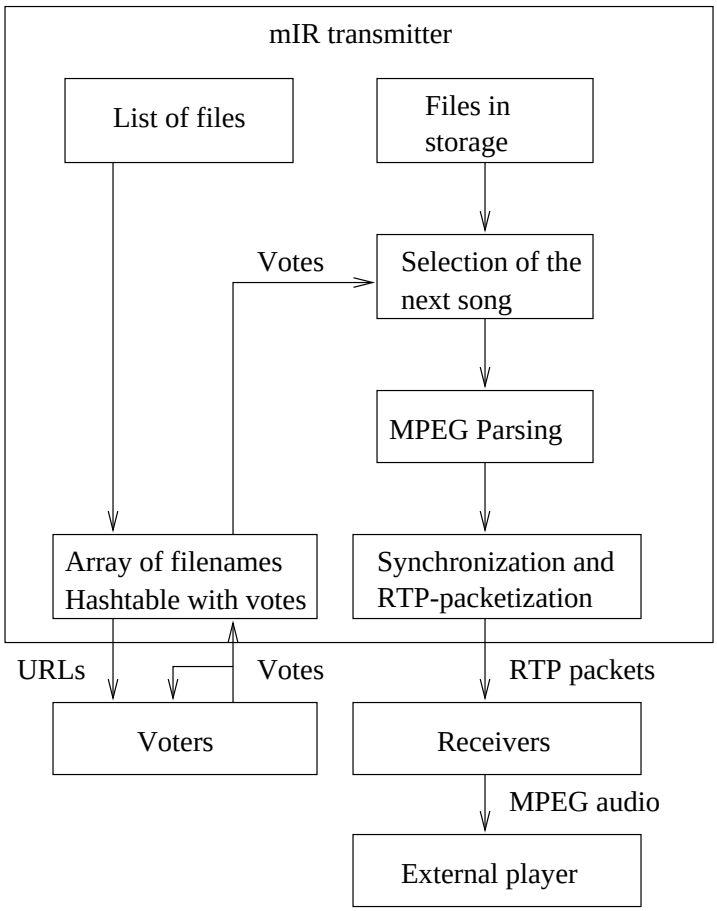

Figure 2.2: mIR transmitter

Command	Argument	Notes
“m”	The message	A message to the discussion
“c”		Clear all received votes (sent by the transmitter at start of a new song)
“v”	The song	Vote for a song
“cv”		Clear the vote from the user sending this command
“q”		Vote for “Quit transmitting this song”
“db”	An URL to the database	This URL is used in the voting tool to access the database of songs
“fu”	An URL to the filelist	The URL to the list of available songs.

Table 2.1: Available commands

There is currently no graphical user interface for the transmitter; it might be available in future versions.

The voting protocol

A simple text based protocol is used for voting and the discussion (see section 2.2.3). One or more commands can be sent in one RTP packet, separated by the newline character. All commands have the form “command:argument”. See table 2.1 for available commands.

Filenames

The URL to a file that contains a list of the songs/files that should be available for transmission is given at the startup of the transmitter. The transmitter downloads the list and check all files if they are available for reading, and removes duplicate names. A thread in the transmitter continuously sends the URL to the list of files to the information channel using the “fu” command.

Transmission of songs

When the transmitter has selected which song will be transmitted next, the filename is sent to a sender object that will read the file and send it to the multicast group. The marker bit in the RTP header is set on the first packet of each song. The marker bit is usually used for signaling talk-spurts when transmitting audio.

The transmission is done in a loop that consists of five steps, which will be performed until the end of the file is encountered, enough “quit” votes have been registered or an error occurs. The steps are:

1. Find next MPEG header.
   An MPEG audio file/stream consists of MPEG frames. Each frame starts with a header

of four bytes, which begins with something called sync. This sync is 12 set bits in a row. The transmitter searches for the sync by reading one bit at a time from the file. If no sync is found in 1024 bits the transmission of this file is aborted.

2. Parse the MPEG header.
   When the sync is found we read the remaining bits of the header and extracts the available information (for example MPEG layer, bit rate, mono or stereo). If this was the first header of a file, information about the song is sent to the information channel using the "m" command.

3. Read the rest of the MPEG frame.
   If the header parsing was successful the rest of the MPEG frame is read.

4. Send a RTP packet to the multicast group.
   The complete MPEG frame is copied to a buffer, which is then put on the outgoing queue.

5. Synchronize.
   To ensure that the audio data is sent with the correct bit rate, the transmission is synchronized before we read the next MPEG frame. After every tenth[2] packet the transmission is synchronized against the time the transmission of this file started. The time the thread should sleep is calculated as

$$time = n \cdot t_{frame} - (current\,time - start\,time)$$

   where n is the total number of packets sent during the transmission of this file and t_{frame} is the duration of one frame. The other packets are sent with a slightly shorter interval. The reason for this is that the granularity of the system clock in Java is often as high as 10 milliseconds, and the value t_{frame} is often at around 25 milliseconds, so to avoid time drifting we need to synchronize against the start time once in a while.

After the last MPEG frame has been sent the hashtable of received votes (see section 2.2.1) is cleared, and a "c" command is sent to the information channel (the clients should now clear all data structures containing received votes). If an error has occurred during one of the steps the transmission of the file is aborted. If the transmitter has not been able to read at least 200 MPEG frames (200 frames usually represents about 5 seconds of audio) before the error occurred the file is marked as corrupt. Corrupt files will not be selected again.

Votes

The only commands the transmitter listens for are the "v", "cv" and "q" commands.

When a vote is received (the "v" command) an identifier for the user voting is retrieved. The RTCP [87] SDES CNAME is used for this purpose. The vote is inserted into a hashtable of received votes, using the user identifier as the key and the song as the data. This has a number of consequences:

[2]The default value is every tenth packet, but this can be changed at the startup of the transmitter.

1. Each user will only have one vote.
2. The running time of the operation is $O(1)$ (i.e. the time needed is constant with regards to the number of users and the number of available songs).
3. If two users vote for the same song, there will be two entries in the hashtable for this song.

When a "cv" command is received, the identifier for this user is retrieved and if this identifier is used as a key in the hashtable this entry is removed. Again, the running time is $O(1)$.

The "Quit transmitting this song now" (the "q" command) votes are also stored in a hashtable, using the user identifier as the key. If the size of this hashtable becomes larger than half the number of listeners, the transmitter interrupts the current transmission and selects a new song directly. The hashtable is cleared whenever the transmission of a song starts.

The song that will be played next is selected among the songs the users have voted for by randomly selecting one entry in the hashtable. The probability for $song_i$ to be selected is thus

$$P_{song_i} = \frac{votes\ for\ song_i}{\sum_{j \in all\ songs} votes\ for\ song_j}$$

. This means that one has complete control over the selection of songs if there is only one user (the probability is then 1 for the song the user has voted for), and if there are many users and many votes there is still a chance that a song with only one vote will be selected.

After a song is selected all the votes are reset to zero. If there are no votes at all a song is selected at random.

Reliability of votes

To increase scalability, the received votes are the only state that the transmitter keeps. The clients are responsible for acquiring this state from the traffic on the information channel, and the transmitter does not send any information about this state to the channel.

Since neither IP multicast nor RTP provides reliable transmission, reliability must be ensured in some other way. There exist a number of solutions to this problem:

1. Accept the packet loss.
2. Transmit redundant data, for example use Forward Error Correction (FEC) or retransmit data.
3. Use Quality of Service (QoS) techniques and allocate/reserve enough bandwidth
4. Use a reliable multicast protocol.

Two of these solutions have been implemented in mIR: The use of a reliable multicast protocol and the use of redundancy.

The SRRTP solution SRRTP (the Scalable Reliable Real-time protocol)[64] is an extension to RTP that has been developed at CDT, which implements the ideas in the SRM [26] framework. The use of SRRTP does not just solve the problem of packet loss, but also creates new ones:

1. Out of order packets.
   The number of packets that arrive out of order increases when packet loss occurs. Unless this is handled in some way, this can result in an inconsistent state at the voters and the transmitter, for example if a user changes a vote quickly and the two votes for different songs arrive out of order, the first received vote is the correct one and the second should be ignored.

2. Duplicate packets.
   When a packet is lost and subsequently retransmitted, more than one copy of the packet can arrive. For votes, this would not be any problem, but for the discussion messages this would be rather annoying.

3. Late-comers.
   When voters first start the voting tool, they need to get the current state, i.e. all the votes.

4. Scalability.
   There are still questions regarding the scalability of the different reliable multicast protocols. If not even the underlying network protocols are scalable, is there any reason for the application to be scalable?

Following the ALF design principles (See section 2.1.3), the receiver can handle each packet (i.e. ADU) as soon as it arrives, so the first problem was solved by simply ignoring packets that arrive to late. Removing duplicate packets provides a solution for the second problem. Since the global state is cleared after each song, the problem with late-comers can be ignored. The only consequence of this choice is that it may take longer for new clients to acquire the complete global state.

The redundancy solution The problem of packet loss can also be solved by periodic retransmission, i.e. redundancy.

In the redundancy version of mIR, the votes are periodically retransmitted from the voters. The time between the retransmissions is calculated as

$$t_{sleep} = \frac{n}{k}$$

where n is the total number of members in the information channel and k is a constant (currently $k = 2$).

This means that the votes become more unreliable as the number of users increases. This might be acceptable, since these votes really are “unreliable” anyhow (a vote only guarantees that there is a chance that the song will be selected). The real consequence is that the

probability that a song a user has voted for will be selected is lower than the probability calculated in Section 2.2.1, due to the fact that the probability that the vote gets lost also has to be accounted for.

To avoid synchronization of votes (i.e. if two voters both vote at time t, they shouldn't both retransmit their votes at $t + t_{sleep}$ since this would create a higher load on the network and on the transmitter), a short random time is added to t_{sleep}. The discussion messages are not currently retransmitted, so the discussion must be considered as unreliable.

Reliability of the audio transmission

A harder problem to solve with packet loss is the problem of audio quality. MPEG-1 audio was not designed for network links with data loss, and the audio quality decreases rapidly when packets are lost. A packet loss of 1 % makes, for example, music encoded at 128 kbit/s almost impossible to listen to. The consequence is that a solution for the problem with packet loss is *more* important to solve for the audio transmission, since the whole system becomes impossible to use even for low ratios of packet loss.

None of these solutions described above, except the first, is trivial. Initial experiments with the use of the SRRTP protocol for the audio distribution have started.

2.2.2 The receiver

The receiver can receive any MPEG audio stream that the transmitter can send, but not all RTP MPEG audio streams. The receiver fails if:

- There is more than one MPEG frame in each RTP packet.
- The MPEG frames are fragmented.

MPEG audio frames usually are smaller than 500 bytes, which means that there seldom is any need for fragmentation of frames since this is smaller than the usual MTU[3] on the Internet today. Support for multiple MPEG frames in each RTP packet is planned for future versions of mIR; this would reduce the bandwidth of the audio transmission since there would be less overhead due to packet headers for each MPEG frame.

The data is sent to the player either by writing to standard input of the player process, or over an HTTP stream. Because most MPEG audio players can not handle changes in bit rate or MPEG layer in a stream, the player process is restarted when the transmission of a new song starts, i.e. when a packet where the marker bit is set is received. For users running mIR with an external player that uses a GUI, this might be an annoyance.

Receiving packets

When a RTP packet is received, the sequence number is checked to detect packet loss. If packet loss has occurred the count of the number of packets received is adjusted so the syn-

[3]MTU: Maximum Transmission Unit, the size of the largest packet that can be sent without fragmentation.

Figure 2.3: The receiver

chronization step will be carried out correctly. The MPEG frame is then retrieved from the packet, and the packet is placed in the play-out buffer. Information about the audio stream is retrieved from the MPEG frame, and the current bit rate and the MPEG layer is displayed in the user interface (see figure 2.3).

A player thread in the receiver writes the MPEG frames from the packets in the play-out buffer to an output stream. This stream is either the standard input of an external player process, or the output stream of a TCP socket. The receiver listens for HTTP connections on this socket.

After each frame has been written to the output stream the player thread is synchronized in a similar fashion as the synchronization of the transmission in the transmitter.

Performance

The receiver has been tested on a PC with a Pentium 90 MHz processor[4] with adequate results, although this is highly dependent on the external player that is used. The total bandwidth required for the receiver is about 10% more than the bandwidth of the transmitted song. For MPEG 1 layer III songs encoded at 128 kbit/s the bandwidth used[5] is 142 kbit/s and for songs encoded at 112 kbit/s the bandwidth used is 125 kbit/s. A 128 kbit/s ISDN connection is thus enough for receiving and playing songs encoded at 112 kbit/s with full quality[6].

2.2.3 The voting tool

The voting tool is the application that makes mIR interactive. A user can see all available songs, vote for songs, communicate with other listeners and get more information about the songs. There are two different kinds of votes: votes for a specific song and “quit” votes. When the transmitter has received enough quit votes, the transmission of the currents song is interrupted and a new song is selected.

The voting tool listens for commands on the information channel and sends votes and messages from the user to the discussion to the channel (see figure 2.4).

[4]Tested with both Windows 95 and Linux.
[5]Measured using IP firewall rules/accounting on a Linux workstation.
[6]This has been tested good results. All other applications that uses bandwidth must of course be closed.

Figure 2.4: The voting tool

Available songs

When an URL is received via the ”fu” command for the first time, the filelist is downloaded, and the filenames are inserted into a hashtable (using the filename as the key). A tree structure of the filenames is then created. The user can vote for songs by selecting them in this tree structure (see figure 2.5). All votes received from other users are also available for viewing.

Information about songs

Information about all the available songs is kept in a SQL[38] database[7]. Currently the database contains information such as bit rate and song length. It is possible to add comments about the songs and to search the database. The function “Information about a song” in the voting tool opens a web browser with a web interface to the database, and automatically views the correct entry.

Discussion

A simple discussion tool is included in the voting tool. Information about the current song is sent to the discussion by the transmitter. See figure 2.6 for an example view of the discussion. The messages to the discussion are not retransmitted, so some messages might not reach all other users due to packet loss.

2.2.4 Scalability

Both the transmitter and the voters perform operations based on commands sent to the information channel. The commands are sent by both the transmitter and the voters. These operations have to be executed quickly as incoming packets with commands may be lost otherwise. Preferably these operations should also scale well, i.e. the running time of the operations should be $O(1)$ with regards to the number of users (the time to perform an operation is constant and independent of the number of users). The memory requirements of the application should also be bounded.

[7] MySQL, <URL:http://www.mysql.com>

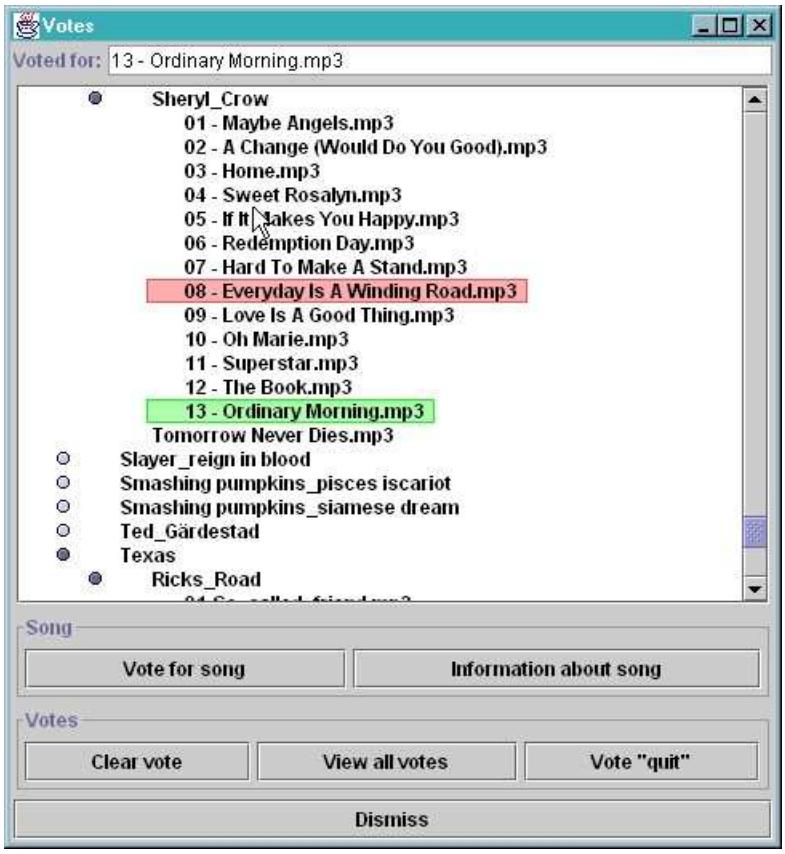

Figure 2.5: Voting

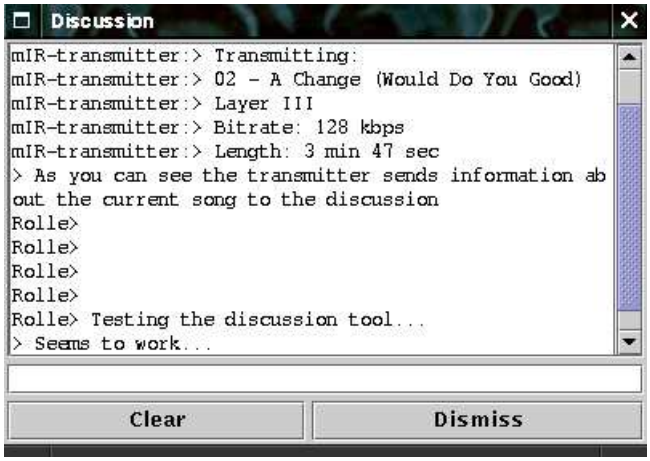

Figure 2.6: The discussion tool

The transmitter

As shown above (see section 2.2.1), the running time of all operations performed by the transmitter when commands are received from the voters is $O(1)$. The memory requirement is not constant, but grows linearly ($O(n)$) with the number of active voters (i.e. voters that actually vote). For each user voting about 100-200 additional bytes will to be stored in the hashtable containing votes, so even if 10000 users are voting this amounts to 1-2 Mbytes, which is acceptable when one considers that the transmitter process usually uses 5-6 Mbytes of memory anyway. The hashtable is cleared after each song has been sent.

Although all operations can be performed in constant time, there might still be a scalability problem if all voters happen to send votes at exactly the same time. The effect at the transmitter of this is a higher rate of packet loss, but if the voters are not completely synchronized (i.e. the next retransmission of votes does not also happen at exactly the same time) the retransmission of votes will assure that all votes are received.

The voting tool and the receiver

The transmitter uses the same algorithms for storing votes as the transmitter, and is therefore as scalable as the receiver is. The receiver just receives a stream of audio data with constant bit rate and sends it to an external player, so it is as scalable as RTP/RTCP.

Bandwidth

The SRRTP version The total bandwidth used by mIR can be calculated as:

$$B = B_{audio} + B_{discussion} + B_{votes}$$

where B_{audio}, $B_{discussion}$ and B_{votes} is the bandwidth for the transmission of the audio data, discussion and votes respectively. The value of B_{audio} is constant for a fixed bit rate. The use of SRRTP makes $B_{discussion}$ and B_{votes} hard to calculate, but $B_{discussion}$ is usually low. B_{votes} can be estimated as

$$B_{votes} = \frac{v \cdot S}{t_{song}} + B_{SRRTP}$$

where v is the total number of votes transmitted, S is the average size of one vote, and B_{SRRTP} is the overhead for SRRTP, which depends on the packet loss and the number of receivers/senders.

The redundancy version Again, the total bandwidth used by mIR can be calculated as:

$$B = B_{audio} + B_{discussion} + B_{votes}$$

where B_{audio}, $B_{discussion}$ and $B_{votes}$is the bandwidth for the transmission of the audio data, discussion and votes respectively. The value of B_{audio} is constant for a fixed. $B_{discussion}$ is not constant, but usually quite low.

To calculate an estimate of B_{votes}, the maximum number of votes sent during the transmission of one song is calculated, and B_{votes} is then calculated as

$$B_{votes} = \frac{v \cdot S}{t_{song}}$$

where v is the number of votes sent, S is the average size of one vote and t_{song} is the length of the transmitted song. Unfortunately, v is only limited by how often users change their votes:

$$v \leq m \cdot (\frac{t_{song}}{\min(t_{sleep}, t_{song})} + C)$$

where C is the number of times one user changes the vote on average, m is the number of users voting, t_{song} is the length of the current song and t_{sleep} is as above (see Section 2.2.1).

If we only consider the case were users votes once per song ($C = 0$), then

$$v \leq m \cdot \frac{t_{song}}{\min(t_{sleep}, t_{song})}$$

If $t_{sleep} \leq t_{song}$, then

$$v \leq m \cdot \frac{t_{song}}{t_{sleep}} = m \cdot \frac{t_{song}}{(n/k)} = \frac{m}{n} \cdot k \cdot t_{song} \leq k \cdot t_{song}$$

Users	Unreliable	SRRTP: 0%	SRRTP: 5%	SRRTP: 10%
1	1.8	1.8	1.8	1.9
10	6.1	6.2	6.5	7.2
25	13	25	32	44
50	25	105	126	136

Table 2.2: Bandwidth measurements. Values for SRRTP are packet loss in percent, for bandwidth the values are in kbit/s.

since $m \leq n$. The condition $t_{sleep} \leq t_{song}$ can also be written as $n \leq k \cdot t_{song}$. When these conditions hold, i.e. users don't change their votes and the total number of users is less than $k \cdot t_{song}$, then

$$B_{votes} \leq \frac{k \cdot t_{song} \cdot S}{t_{song}} = k \cdot S$$

i.e. the bandwidth is always smaller than a constant value. In the general case,

$$B_{votes} \leq m \cdot (\frac{t_{song}}{\min(t_{sleep}, t_{song})} + C) \cdot \frac{S}{t_{song}}$$

If the value of C can be considered constant, then B_{votes} grows linearly when the number of users voting increases.

Measurements

Instead of trying to analyze the behavior of periodic retransmission and SRRTP with different degrees of packet loss, practical experiments have been conducted. For measuring the used bandwidth a PC was configured as a firewall[8], which reported the actual value of bits/s transferred to and from a multicast address.

Table 2.2 shows the measured bandwidth. The values are the average value over 1 minute. The users were simulated with a "ghost client" application. The ghost client operate in a simple loop:

1. Send one vote and one message to the discussion.
2. Sleep for a random time. The distribution is a uniform distribution of $[0..10]$ seconds.
3. Go back to 1.

We can clearly see that the SRRTP solution is not scalable: the bandwidth increases rapidly even without packet loss when the number of users increases. The redundancy solution on the other hand behaves as predicted: it shows a small and linear growth when the number of users increases.

[8]Running the Linux operating system, <URL:http://www.Linux.org>

2.3 Related work

There exist a few other programs for distributing high quality audio over the Internet. A regular WWW server can also be used for this, both for "asynchronous" listening (first the song has to be downloaded, then it the song is played) and for streaming. No publically available systems that incorporates interactive parts are known by the author.

2.3.1 Real

The Real Server and the Real Player[9] are often used for transmission of audio, but the quality is far from the quality one can get from MPEG audio. Both unicast and multicast can be used.

2.3.2 Shoutcast and Icecast

Although unicast solutions have existed for a long time, it is the recently released Shoutcast[10] that is most well known.

Shoutcast encodes the audio currently being played through the sound card of the computer as MPEG audio and transmit this data to a server program that handles the distribution of the music. The server program can be running on the same workstation or on a separate server, and works much like a simple WWW server.

Since Shoutcast uses unicast it is not very scalable. The bandwidth usage can be calculated as

$$B \geq n \cdot bitrate$$

where n is the number of listeners and $bitrate$ is the bit rate of the song currently being transmitted.

A similar system is Icecast[11], which was created as an free GPL alternative to Shoutcast that could run on Linux and other Unix platforms.

2.3.3 liveCaster

Ross Finlayson's liveCaster[12] program is similar to mIR, but does not contain any interactive parts. Since liveCaster transmits MPEG Audio files from storage using RTP over IP Multicast, mIR and liveCaster is compatible. liveCaster can also do transcoding of MPEG streams to lower bit rates.

[9] <URL:http://www.real.com/>
[10] Released in December 1998: <URL:http://www.shoutcast.com/>
[11] <URL:http://icecast.linuxpower.org/>
[12] <URL:http://www.live.com/liveCaster/>

2.4 Conclusion

mIR provides a scalable system for distributing high quality audio over the internet with interactivity. Although the applications still must be considered to be prototypes, they are very robust. The transmitter for instance, has been running for over three months without any failures. Implementing the applications in Java has had a number of advantages, for example the applications are robust and platform independent and the development has been fast.

By following the ALF design principles, each incoming packet to the different application can be handled directly. Combined with efficient constant-time algorithms make the applications scalable.

Although the bandwidth is not limited, the growth is small for each additional user. The main uncertainty is the use of SRRTP, which has not yet been proven to be scalable. Another solution, the use of redundancy has though been shown to be scalable.

2.5 Future work

The main area of future work is reliability: how to provide reliability for scalable applications that uses multicast. The SRRTP protocol is not really scalable, and solutions such as redundancy are not fool-proof: we can not guarantee that all packets arrive. For applications that demand truly reliable traffic, another solution is needed.

One other interesting idea for future work is to transmit layered MPEG audio. The idea is to divide the audio data into two or more layers at the transmitter. The layers can be incrementally combined to produce higher quality audio at the receiver. Each layer would be transmitted on a different multicast group, and a receiver joins an appropriate subset of the multicast groups to receive the audio. A user could then listen to the audio at a lower quality even if it can't receive 125-145 kbit/s, which is needed now.

Most of the following ideas are planned for future versions of mIR.

- The transmitter
  - Reduce the bandwidth for audio sent by sending more than one MPEG frame in each RTP packet.
    This could reduce the bandwidth used by about 5-6 kbit/s.
  - A graphical user interface.
  - Statistics.
    The statistics could for example be inserted into the database.
  - Some sort of long term memory of votes.
    When there are no votes, a song is selected at random. Songs that have received many votes in the past could perhaps have a higher probability of being selected again.
  - Congestion control.
    Since mIR uses a considerable amount of bandwidth, some sort of congestion

control would be appropriate (in the transmitter, the receiver or in both the transmitter and receiver).

- The receiver
  - Run-time configuration of the external player.
    This would make it easier to use other players than the default ones.
  - An Applet version.
- The voting tool
  - Repeatable votes
    This could be an option such as “vote for this song until it is selected”.
  - An Applet version.

Part 3

Large scale distributed watermarking of multicast media through encryption

Large scale distributed watermarking of multicast media through encryption

Roland Parviainen, Peter Parnes Department of Computer Science/Centre for Distance-spanning Technology
Luleå University of Technology, 971 87 Luleå, Sweden
Roland.Parviainen@cdt.luth.se, Peter.Parnes@cdt.luth.se

February, 2001

Abstract

In this paper we describe a scheme in which each receiver of a multicast session receives a stream with a different, unique watermark, while still retaining the scalability of multicast. The watermarked streams can be used to trace those users who make unauthorized copies of a stream. The watermarking is enabled by encryption of two slightly different copies of the original stream with a large set of different keys.

3.1 Introduction

IP Multicast [21] provides efficient many-to-many data distribution in an Internet environment. Senders send datagrams to a 'host group', a set of zero or more hosts identified by a single IP destination address. The datagrams are delivered to all members of the host group by the network infrastructure in an optimized way.

Multicast is very well suited to use for large scale media distribution because of the scalability: each network link in the network only has to transport one copy of each packet regardless of the number of receivers. The drawback is that receivers do not have to be authenticated and can easily eavesdrop on the traffic without being detected. A unicast solution, where we send one copy of the stream to each user, is easier to protect but is infeasible for large scale transmissions to 100,000 to 1000,000 users and above.

Authentication and confidentiality can be solved with the use of encryption, but there is still a problem with malicious users retransmitting the media data unencrypted to other users. One way to detect whom the illegal copy originated from is *fingerprinting*, embedding unique information, a watermark, into each copy of the media that identifies the user receiving the copy. This information should be robust against any possible user manipulation, and in the remainder of this paper we will assume a robust watermarking scheme that can perform this fingerprinting exist. A fingerprinted stream might discourage illegal copying of the media,

since the origin or the buyer of stream can be identified. This might be the only option for pure software solutions where tamper-resistant hardware is impossible.

The objectives of multicast and fingerprinting seem contradictory: multicast sends the same stream to everyone while to achieve the goals of fingerprinting every receiver should receive a different stream. In [10] Perkins, Brown and Crowcroft solve this problem by using active network elements that make sure all receivers get slightly different streams. In this paper we present a solution that does not suffer from the requirement of trusted active network elements. We propose a scheme were encryption is used to ensure different users receive fingerprinted streams. No trusted or active network elements are needed; all security is handled by the applications.

The remainder of this paper is structured as follows. In the next section we describe related work on multicast security. Then in section 3.3 we describe our approach in detail. In section 3.4 we describe possible attacks against the system. In section 3.5 an experimental implementation of this system is described. Sections 3.6 and 3.7 conclude this paper with limitations and conclusions.

3.2 Background

3.2.1 Multicast Security

Since IP multicast provides no authentication or confidentiality it is very easy to eavesdrop on, record and copy or retransmit a media stream completely anonymously. Basic encryption of multicast streams is not sufficient to protect important media streams since it is possible to retransmit either the content stream or the keys to untrusted users.

In [16] a ‘virtual key’ is marked instead of the plain-text. A certain minimum number of users need to collaborate to construct a key that works but identifies none of them. This does not prevent retransmission of media content and the bandwidth needed for the control messages may be too high.

Scalable content control schemes such as Nark [9] provides scalable authentication and encryption but need tamper-resistant smart-cards. Watermarking is not possible in todays limited smart-cards and have to be done off-card, but can be done using a system such as Chameleon [2].

Chameleon by Anderson and Manifacas is a similar scheme to ours, where a stream cipher is adapted to give slightly different output depending on a large unique key for each user. Our scheme can handle much more flexible watermarking algorithms; the two different watermarked packets do not have to have any common bits at all.

In the watercasting scheme [10] the source sends multiple subtly different copies of each packets and routers at the nodes in the multicast distribution tree discard packets, such that the stream delivered to each receiver is unique. This approach has several problems: support for this protocol in routers is needed which is probably hard to achieve, the routers have to be trusted, and the source must send d copies of each media packet, where d is the depth of the multicast tree.

3.2.2 Watermarking

A simple watermarking method is to change the least significant bits in for example an audio clip or an image. For an audio clip, we could put our embedded message into to least significant bit of some or all samples. These methods are easily broken. More advanced methods often use spread spectrum techniques or transforms such as Fourier and DCT to make the watermarks more robust.

A watermarking method that fulfills some requirements for the difficulty in removing it is called *robust*. Some examples of robustness requirements for audio recordings from IFPI, the International Federation for the Phonographic Industry [37] are:

- The sonic quality of the sound recording should not change
- The marking information should be recoverable after a wide range of filtering and processing operations, including two successive D/A and A/D conversions, MPEG compression, etc.
- There should be no other way to remove or alter the embedded information without sufficient degradation of the sound quality as to render it unusable

Similar requirements can be made for still images, video and other media types. A good introduction to the subject is [44].

3.3 Method description

The source sends two different copies of each media packet, each with a different watermark. Both copies are encrypted with two different, random encryption keys. The encrypted packets are then sent to all receivers using IP multicast. Any given receiver has access to the key of only one of the two encrypted packets of one media packet.

3.3.1 Transmission of packets

The source has access to k media packets: $P[1], P[2], \cdots, P[k]$ and an encryption algorithm E, such that $P = D(E(P,K),K)$. That is, $E(P,K)$ encrypts P with key k and $D(P,K)$ decrypts P. A watermarking algorithm W: $P_w = W(P,w)$, $w = U(P_w)$ and two watermarks, w_0 and w_1 are also needed. W embeds the watermark w in the cover object P, and U extracts the watermark from the marked object. A detection algorithm that detects if the watermark is still present can be used instead: $U(P_w,w) = B, B \in \{true, false\}$. The source needs $2k$ random encryption keys, $SK[1], SK[2], \cdots, SK[2k]$, to be able to transmit the media packets. A receiver r has access to k of these keys: either $RK_r[i] = SK[2i-1]$ or $RK_r[i] = SK[2i]$, $i = \{1, 2, \cdots, k\}$.

To transmit media packet i the source perform the following operations:

1. Read the media packet, $P[i]$

2. Create two watermarked packets, $V_0[i] = W(P[i], w_0)$ and $V_1[i] = W(P[i], w_1)$
3. Get two encryption keys: $SK[2i-1]$ and $SK[2i]$
4. Encrypt $V_0[i]$ and $V_1[i]$: $C_0[i] = E(V_0[i], SK[2i-1])$ and $C_1[i] = E(V_1[i], SK[2i])$
5. Transmit $C_0[i]$ and $C_1[i]$ together with i

At the receiver side, the client receives both packets and tries to decrypt them:

1. Receive two packets: $C_0[i]$ and $C_1[i]$
2. Get the decryption key for packet i: $RK_r[i]$
3. Try to decrypt both packets with key $RK_r[i]$
4. Only one packet will decrypt into a proper media packet: $V_{j_i}[i] = D(C_{j_i}[i], RK_r[i])$, $j_i \in \{0, 1\}$
5. Decode and render $V_{j_i}[i]$

For each media packet the receiver will be able to decode exactly one of the watermarked packets. Which of the two packets is decided by the keys the source has assigned to the receiver.

3.3.2 Identity strings

If the keys a receiver have access to is unique among all receivers and known by the source, a unique identity string for that user can be defined: $id_r = B_r[1], B_r[2], \ldots, B_r[k]$, $B_r[i] \in \{0, 1\}$.

This identity string can be derived by the source from both the keys given to the receiver and the stream the receiver decrypted. From the keys the source sent to the receiver:

$$B_r[i] = \begin{cases} 0, & \text{if } RK_r[i] = SK[2i-1] \\ 1, & \text{if } RK_r[i] = SK[2i] \end{cases}$$

From the decrypted stream for the user:

$$B_r[i] = \begin{cases} 0, & \text{if } U(V_{j_i}[i]) = w_0 \\ 1, & \text{if } U(V_{j_i}[i]) = w_1 \\ undefined, & \text{if neither } C_0[i] \text{ nor } C_1[i] \text{ was received or decrypted} \end{cases}$$

If the receiver do not receive all packets, due to for example packet loss or that the receiver tuned in late, the identity strings will not match completely. If n is large enough, the partial identity string will still be long enough to be unique among all receivers although some bits are undefined.

3.3.3 Bandwidth usage

Since 2 copies have to be sent for each media packet, the bandwidth usage is doubled both for the source and the receivers. This can be decreased by some optimizations.

At any given time, only one of the two watermarked packets is actually useful for a single receiver since the other packet can not be decrypted. If we send the two copies on different multicast groups the receivers can hop between the groups by joining and leaving them as the group the correct packet is transmitted on changes. In this approach we not only have to send the keys to each receiver but also which stream to receive; one extra bit for each key is needed. Unfortunately the join/leave latency for IP multicast is currently too large for this approach. Also, if more than one receiver is on the same network segment most of saving is lost.

As pointed out in [10] another optimization would be to only watermark 1 in every x packet, thus reducing the bandwidth to $(1 + 1/x)$ times the bandwidth of the original stream. Unfortunately a malicious user could remove these watermarked packets and retransmit the resulting degraded stream if x is large. We must therefore make sure that the degradation is large enough to discourage removal of the watermarked packets. One example of this is to only add watermarks to the I frames of an MPEG video stream or only watermark the last 10 minutes of a movie.

3.3.4 Key Distribution

The receiver keys can be treated as a long term key distributed by out-of-band means when the users registers, either as a downloadable file (protected by e.g. SSL/TLS [22]) or delivered to the user on a floppy or cdrom. All these solutions have problems when revocation of access is considered. The keys can also be continuously streamed to the users. The bandwidth per user needed for this is small, but it is still a serious scalability problem.

The amount of keys that each receiver needs depends on the required security (see section 3.4.1). The total size of the keys for one receiver is then $keys \cdot keysize$. Using 10000 keys and a key size of 128 bits thus requires 160 kbyte per user. The source only has to store a bitmask of length $keys$ for each user and $2 \cdot keys \cdot keysize$ bytes of key material. A cryptographically secure random number generator can also generate the bitmasks instead to further reduce storage needs at the source.

3.3.5 Key size

It can be argued that attackers with sufficient funds to break, for example, 56 bit keys also have enough funds to get keys in other easier ways, for example using false identities while registering, and thus we do not need any stronger keys. On the other hand, getting access to computing power does not necessarily require monetary funds; a distributed attack is also possible.

It is not enough to break one of the keys since it only gives the attacker the option to change one of the watermarked packets in the stream. An attacker has to break a sufficient amount of keys to get enough packets to create an unidentifiable watermarked stream.

3.4 Attacks

We assume that it is not possible to either remove the watermark or break the encryption in reasonable time. We also assume that the attacker can not steal the non-watermarked stream from the source by breaking into the server.

If the encryption algorithm is broken an attacker can choose the final watermarked stream and make traitor tracing impossible, but if the encryption algorithm is chosen with care and with large enough key size and the keys are generated properly this can be avoided.

It is possible to attack another receiver's computer and steal the stream or keys from there, thus indicating someone else as the pirate. These kinds of attacks are out of scope for this paper.

Removal of the watermark might be possible if a robust watermarking algorithm is not chosen. The existence of such an algorithm is not considered in this paper.

Several users can collaborate and combine their streams to try to prevent watermark detection, either by choosing packets from different streams or by merging packets with for example bit voting. The two watermarks w_0 and w_1 and the watermarking algorithm should be chosen so that at least one of the watermarks is still present and recoverable after the bit voting.

Getting access to several streams and selecting packets from different streams is an easy and powerful attack. In the next section we analyze this attack further.

3.4.1 Traitor tracing

If p users collaborate, at least k/p of the original bits from one of the streams will always remain. We must make sure that this fraction is large enough to ensure a high probability of detection of the collaborators.

We assume we have one identy string for the illegal stream id_I and n identy strings for the legal watermarked streams: $id_{L[1]}, id_{L[2]}, \ldots, id_{L[n]}$. We want to identify the identy string of at least one of the streams that was used to create the illegal stream. We can calculate a measure on how good a watermarked stream match the illegal stream by adding the XOR value of bits from the identity strings:

$$M(L[i], I) = \sum_{j=1}^{k} B_{L[i]}[j] \oplus B_I[j]$$

We define the set S_{nc} as the values $M(L[i], I)$ where i is not one of the collaborators, that is $L[i]$ was not used to create the illegal stream. S_c is the values $M(L[i], I)$ where i one

of the collaborators. The values in S_{nc} have a binomial distribution $X_{nc} = Bin\left(k, \frac{1}{2}\right)$, and the distribution of the values in S_c is $X_c = Bin\left(k\left(1 - \frac{1}{p}\right), \frac{1}{2}\right) + \frac{k}{p}$. These distributions can be approximated with a normal distribution,

$$X_{nc} = N\left(\frac{k}{2}, \sqrt{\frac{k}{4}}\right)$$

and

$$X_c = N\left(\frac{k}{2}\left(1 + \frac{1}{p}\right), \sqrt{\frac{k}{4}\left(1 - \frac{1}{p}\right)}\right).$$

As the number of collaborators, p, increases X_c converges towards X_{nc} as expected. If p is relatively small, the probability that all values in S_c are less than $E[X_c]$ is very high. For example, if $n = 100000, p = 10, k = 10000$ the distributions are $X_{nc} = N(5000, 50)$ and $X_c = N(5500, 47.43)$. The probability that at least one of the values in S_c is greater than $E[X_c]$ is $A = 1 - P(X_c < E[X_c])^{100000} = 1 - P(X_c < 5500)^{100000} = 0.76 \cdot 10^{-18}$ The probability that at least one of the values in S_{nc} is greater that $E[X_c]$ is $B = 1 - P(X_{nc} < E[X_c])^{10} = 1 - \left(\frac{1}{2}\right)^{10} = 0.9990$. That is, there is a high probability that the identy string with the highest value of $M(L[i], I)$ is one of the collaborators. For $p = 50$ on the other hand, we have virtually no chance to detect any of the collaborators.

3.5 Implementation

To test the feasibility of our scheme it was implemented in an existing Java application system[69] for audio transmission over multicast which uses the MPEG-1[60] audio compression standard. The system consists of a server application and a client application. The server read MPEG-1 audio data from disk and send it to a multicast address using RTP[87], the client receives this data and decodes it. The MPEG decoding is not done in Java but in native code.

Blowfish was chosen as the encryption algorithm, and was provided by the reference implementation of the Java^{TM} Cryptography Extension (JCE) 1.2.1 from Sun. Since the implementation is not supposed to be used in a production environment a very weak watermarking algorithm was chosen for simplicity: a few otherwise unused bits in the MPEG audio frame header are used for the watermark.

Prior to the transmission, the server generate $2n$ random keys and keys for p users, where the values n and p are given. These keys are stored in files until they are used. If the stream is longer than the value of n, the keys 'wrap around': the keys for packet i is $SK[1 + (2i - 1 \mod n)]$ and $SK[1 + (2i \mod n)]$.

When we enable the watermarking scheme in the server the CPU usage increases from 1.0% to 12%[†]. The large increase is because the server now has to encrypt every packet

[†]Measured with perfmon.exe on a Pentium III 550 MHz computer running Windows NT4. The bit-rate of the media stream was 128 kbit/s and each key was 40 bits.

twice. For the client application the increase is from 6.7% to 18%. The bandwidth usage is the same for both a client and a server, independent of the number of users. As expected, the bandwidth[‡] usage increases by a factor of roughly two, from 133.3 kbyte/s to 270.4 kbyte/s. That the factor is 2.03 and not 2 is due to a small extra overhead for padding when doing encryption. The cost of key distribution is not included.

3.6 Limitations

Our scheme relies on the existence of a robust watermarking scheme. No perfectly robust watermarking algorithm has been proposed so far, and it is not clear such a algorithm is possible. Without any optimizations, we need at least twice the bandwidth of the original media stream which might be too much for some applications. We have not considered the problem of revoking access for a receiver; with the current scheme this would require that new keys have to be distributed to all receivers again.

Although only the use of two watermarks are described in this paper we are not limited to this. Instead of sending two different packets for each media packet it is possible to send any number of copies, although this increases the bandwidth substantially. The watermarks w_0 and w_1 do not have be constant and can change at any time as long as $w_0 \neq w_1$ and the source keeps track of them.

3.7 Conclusion

We have described a scheme for scalable fingerprinting of multicasted media. Contrary to other proposed schemes no active network components or tamper-resistant smart-cards are needed. By increasing the bandwidth with a constant factor we can ensure that every receiver get a unique fingerprinted stream. The watermarks that make up the fingerprints are not fixed to a certain number of bits or format but can be of any format the watermarking algorithm requires for robustness.

Like other traitor tracing schemes based on watermarks, it has only limited collusion resistance. The most promising attack is to get hold of several streams and mix them together so the source can no longer be identified. The main countermeasure is to increase the number of watermarks in one stream.

[‡]This includes RTP headers but not IP and UDP headers

Part 4

Mobile Instant Messaging

Mobile Instant Messaging

Roland Parviainen, Peter Parnes Department of Computer Science/Centre for Distance-spanning Technology
Luleå University of Technology, 971 87 Luleå, Sweden
Roland.Parviainen@cdt.luth.se, Peter.Parnes@cdt.luth.se

February, 2003

Abstract

In this paper we describe a Mobile Instant Messaging system, MIM, designed for mobile environments. During design of mobile applications several new problems and possibilities have to be considered that do not exist with applications targeted at desktop PCs. One example of an application not designed for a mobile environment is current, very popular, instant messaging systems such as ICQ, AIM and MSN messenger.

We describe why current systems are not suitable in a mobile environment and present our architecture for a new system, MIM, and show various implementations for different mobile devices such as PDAs, wearable computers and mobile phones.

4.1 Introduction

Today wireless technologies such as GRPS, new 3G networks and different WLAN technologies are rapidly being developed and deployed providing wireless access to the Internet in a much larger scale than before. At the same time, new mobile platforms like PDAs, advanced mobile phones and wearable computers are becoming more and more common.

Together it will mean that the Internet will be accessible from anywhere, on a wide variety of different platforms and connection technologies. A user should be able to expect to access the same applications and services that are available on a fixed, connected PC from a mobile connected unit.

Software applications for these mobile platforms face a number of different problems that applications designed for PCs do not experience, especially for distributed applications that need network connectivity. In addition new possibilities are opened up for mobile applications such as location independence, location and context awareness. Some of the unique conditions mobile distributed applications face are:

Wide variety of user interface possibilities This can range from a normal PC environment to a mobile phone or a wearable computer with only audio user interaction capabilities.

Resource limitations Resources available on most mobile devices are usually very restricted. Different limitations such as weight and size mean less CPU power and storage capabilities.

Heterogeneous networks The connectivity can vary from a low latency always connected broadband connection to a low bit rate wireless connection with high packet loss that only supports HTTP connections through proxies and firewalls. Often the network connectivity can be temporary as well, for example when we move in and out of areas without coverage in a wireless network or for devices such as phones that are not always connected.

Mobility and state We should not be restricted by the limited user interface of mobile devices when we are for example in the office and have access the work PC. It should be possible to easily switch the device we work with while always having access to the same data and applications. Long term and short term state, such as program state and configuration details have to be synchronized between several application instances running on different devices at the same time. A new instance should have access the same state as other connected instances.

These are of course not all the problems and unique conditions that exist, but are the problems that we focused on in the system described in this paper.

4.1.1 Instant Messaging

Instant Messaging (IM) systems such as ICQ, AIM and MSN Messenger have become extremely popular during the last few years. All these systems are designed for desktop PCs with the assumption that a user only uses one computer, or at least only one computer at a time. New protocols and standards for instant messaging are being developed by e.g. the IETF and the Jabber Software Foundation, but it is unlikely that these protocols are adapted by any of the major systems any time soon; although Jabber has had some success on internal corporate networks and a client is included in certain mobile phones.

If we want to use these systems in a mobile environment we either have to try to adapt to the assumptions about usage these systems have made or try to adapt the IM systems to better suit the needs of a mobile environment.

In this paper we will present a system that can be used to adapt current instant messaging and similar systems to a mobile environment. In the next section we describe in more detail the problems with the current systems, and in section 4.2 we introduce our new platform. Then, in sections 4.3 and 4.4 we present the architecture and implementation. Sections 4.5 and 4.6 discusses related and future work and finally we end with a discussion and conclusions in sections 4.7 and 4.8.

Most current IM systems are designed in a way that makes it impossible or very hard to use more than one client program per user at the same time; if a user wants to connect both from their office PC and a mobile client at the same time they're in trouble since the system only allows one connected client at a time. In addition, even if more than one client can be connected at the same time, messages are only forwarded to one of the clients.

Configuration state such as the user's contact list, history, settings etc. is often stored by the client locally, making synchronization a big problem. This is mostly due to scalability issues since most systems have millions of users. Some systems such as the latest versions of ICQ supports hosting the user's contact list at the server which makes it easier to run clients from more than one computer at the same time but does not completely solve the problem. For example, the message history is still stored locally at each client. To get a complete view of the history, access to the history from all clients the user has used is needed.

Also, if the network connectivity is lost or the user switches connection method the user is marked as being "Off-line", although there might be a high probability that the connectivity is regained shortly. In a wireless network this will be problematic since short losses of connectivity might occur often when a user for instance move from one cell to another or briefly looses coverage.

Since current systems assume that the user's position is fixed to one PC there is no location or position information. This is not the case for a mobile device; here a user's position dynamically change and can provide important and valuable information to other contacts.

Implementations of different Instant Messaging systems already exist for platforms such as PocketPC and J2ME MIDP for PDAs and mobile phones, but they still have similar problems.[†] They do not attempt to solve the problem of mobility; it is just implementations of the same applications for different platforms.

4.2 Mobile Instant Messaging

Designing a new IM system and expecting all users to adopt it is unrealistic; most users simply use the same system all their friends and contacts use. The extremely large user base of the systems makes it hard to see any new better system gaining any large market share. Therefore, a system that can use the current IM systems is highly desirable. It is also desirable to be able to connect to several systems at the same time and aggregate the different user communities and present a single list of contacts to the user. Clients that can connect to different systems already exist, for example EveryBuddy[‡], Trillian[§] and GAIM[¶].

A user should be able to move from his home PC, to a mobile unit such as a phone while on the road and finally to his work PC without having to remember to manually logout from each unit before switching to a new one. The user status should not switch from "Online" to "Off-line" and back again each time the user switch unit or when the network connectivity is temporarily lost since the user will probably regain connectivity shortly. The contact list should be automatically synchronized among our clients; if a new contact is added while using a mobile phone this new user should also show up in the contact lists in all other clients. If a contact sends a message, we should be able to receive and read it on any of the connected applications at any time. None of the current systems can do this today.

[†] See e.g. QSM (http://www.softex-india.com/qsm/) or MiMessenger (http://www.mordax.com/manual/manual_mim_en.html)

[‡] http://www.everybuddy.com/

[§] http://www.trillian.cc/

[¶] http://gaim.sf.net/

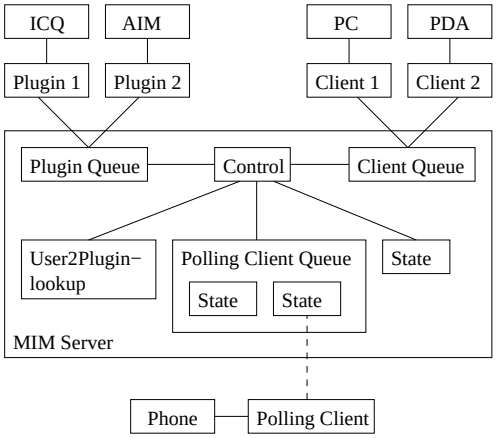

Figure 4.1: Architecture

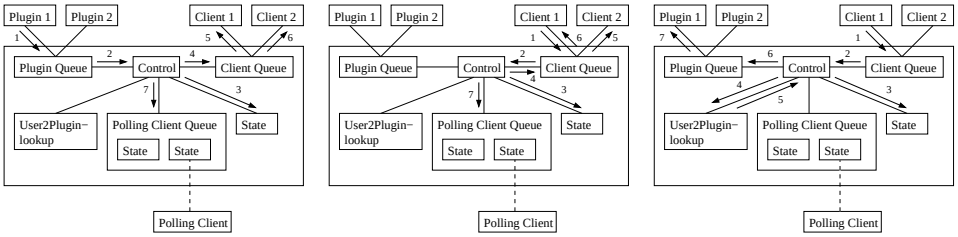

Figure 4.2: Example of a message exchange

4.2.1 Limitations of current solutions

4.3 Architecture

The goals from the previous section can be achieved by using a hierarchical system with a Mobile Instant Messaging (MIM) server running on the **Home PC**, which is for example the user's actual home PC, office PC or a central network server. Different IM systems are connected by MIM plugins, providing the MIM server with contact lists, users' status, functions for setting our own status, receiving and sending messages etc.

Clients connect to the MIM server, which handles all synchronization and stores state such as received and unread messages, history and status. When a client first connects, the full state is retrieved from the server. Further updates to the state, such as an incoming message, are sent to all connected clients. The MIM server is always connected to the different IM systems so a user will be able to receive messages at any time. If no clients are connected over a duration of time, the status and auto response message can be set to pre-configured values.

If clients can not keep a connection to the MIM server open continuously, it can also run in a polling mode, where the client regularly connects to the server for synchronization: messages that have been written are sent to the server and on to their recipients, new messages are downloaded, etc. This mode is primarily for clients running on phones, for example the MIDP profile as used by many "Java" phones only guarantees network connectivity through HTTP.

See figure 4.1 for an example of this architecture with two different MIM plugins, one for the ICQ network and one for the AIM network. Three clients are connected to the MIM server: One on a PC, one on a PDA and one on a mobile phone. The clients on the PC and the PDA use a full graphical user interface, while the Phone client use a limited user interface adapted to the possibilities of the unit.

4.3.1 Communication Protocol

We use a simple light weight, low bandwidth message passing system that can work over any protocol that supports binary data transfer, although it could be implemented on top of for example a tuplespace system such as JavaSpaces[97] or T Spaces[49], but work would be needed to make these systems work efficiently over the limited network services that are currently provided by e.g. Java MIDP phone. This limitation is one of the reasons that rules out protocols based on IP multicast that would have been more efficient in a system like ours.

4.3.2 Example of a message exchange

In figure 4.2 we see an example of a message exchange in MIM. Three clients and two plugins are connected to the server.

In a) a message is received from a IM system by a plugin, then in b) the messages is read by a user using one client and finally in c) a message is sent as a reply back to the plugin which forwards the message to the IM system.

An incoming message is sent through the plugin queue to the server control which sends it on to both the client queue and the polling client queue. It also forwards the message to the state component for storage. The client queue sends the message to all clients that are connected, while the polling client queue stores it in a state component for each connected polling client.

When a user reads the message, the client sends a notification back to the server through the client queue. This notification is broadcasted to all connected clients as described above. Clients can use this notification to change the status of received messages from "Unread" to "read". The notification is also sent to the central state which updates the status of the message in storage.

A message from a client to an IM system is sent through the client queue to the server, which first sends it to the state component. It then uses the "User to plugin" component to look up which plugin that is connected to the target IM system. Then the message is sent to the correct plugin through the plugin queue.

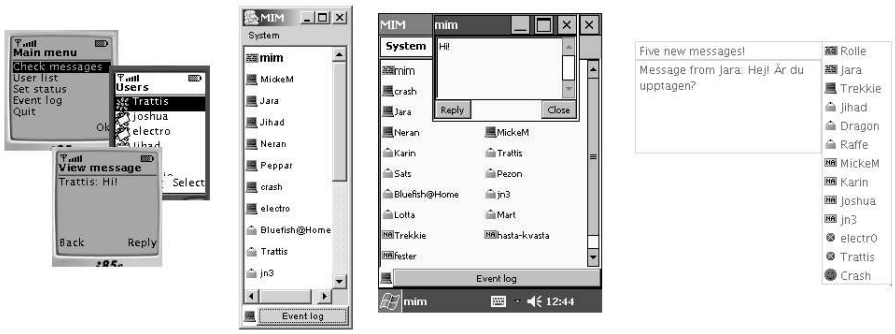

Figure 4.3: MIM running different platforms; on phones (The screen shots are from the J2ME Wireless Toolkit emulator), on a PC, on an iPAQ Pocket PC PDA and on a wearable computer (the screen shot shows what is displayed in the top right corner of the head up display).

4.4 Implementation

The current implementation consists of a mix of Java and C++ applications. All communication is based on TCP or HTTP. The communication protocol is a simple message based protocol, where each message such as a status update or user status information usually is between 20 and 30 bytes.

4.4.1 Clients

All clients are implemented in different versions of the Java platform. They can be delivered to the client platform by regular software installation mechanisms, via applets, Java Webstart or Over the Air (OTA) provisioning for WAP and GPRS. See figures 4.3 and 4.4 for some screen shots of different clients.

The interface for PDAs and PCs is created by the same application, while it had to be completely rewritten due to the different nature of the user interface API in J2ME MIDP. A user interface description language such as UIML[75] could be used instead to provide different user interfaces for different platforms automatically from one specification.

Although Java is platform independent this is not enough to create one user interface for different clients for two reasons; the same user interface APIs are not available on all platforms that runs Java, and the limitations and possibilities of the available platforms varies widely. It does not help that the user interface is written in Java if it is based on a graphical user interface and our platform only have audio and text user interface capabilities. A user interface that works well on a PC does most often not work well on a PDA due to the limited screen size.

PocketPC 2002 The implementation for Pocket PC PDAs uses the Jeode Personal Java runtime which is almost a full JDK1.1.8 implementation.

Figure 4.4: MIM running on a Nokia 7650 phone

PC The PC version uses the same implementation as for PocketPC, but the different platforms are detected at runtime and the user interface is adapted to better suit the particular platform.

Phone The implementation for mobile phones uses the Java 2 Platform, Micro Edition (J2ME)[96] as provided by the Connected Limited Device Configuration (CLDC)[94] and Mobile Information Device Profile (MIDP)[95]. Only HTTP communication is used, as this is the only network connectivity that is guaranteed by the MIDP standard. The big differences between J2ME and other Java platforms mean that we unfortunately can not use the same implementation although many classes can still be reused. For example, the user interface had to be completely rewritten to suit the limited interfaces of MIDP. This system also provides a much cheaper way to send messages from phones than the, at least in Europe, extremely popular SMS services (see section 4.7 for a comparison).

Wearable computer This implementation is still experimental since the wearable computer it is designed for is still being built. Currently it uses an M2 head up display and different small keyboards, connected either to a laptop or an iPAQ PDA. The user interface uses low level Java graphics primitives to create a custom interface more suitable for a head up display.

4.4.2 Plugins

Three plugins exist today, an ICQ plugin, an IRC plugin and a plugin for the chat component in the MarratechPro E-meeting program. The ICQ plugin is written using the Licq[‖] plugin API in C++. The basic instant messaging features such sending and receiving messages, changing status, retrieving user information etc. are implemented.

[‖] http://www.licq.org/

The IRC[43] plugin is implemented as a Java program that shows the different IRC channels as users in the aggregated user list in MIM. Currently it is not possible to dynamically join and leave channels. Only the basic functions such as messages to channels and private messages to users are implemented.

MarratechPro is an e-meeting product by Marratech AB** which is designed for IP multicast but it also works with unicast. It has components such as audio, video, shared whiteboard and chat and is mostly written in Java. The chat plugin for MIM uses the chat agent from the MarratechPro application, so it is fully compatible. Different e-meeting groups are represented as different users in the MIM user list.

4.5 Related work

Frameworks developed in the fields of ubiquitous and pervasive computing are designed to solve many of the problems mobile applications face. For example in [79], approaches for systems for pervasive computing are presented: data and functionality need to be separated, force applications to explicitly acquire all resources, and to be prepared to re-acquire them, a common system platform allowing applications to run on wide variety of devices. These approaches are similar to the ones used in our system. The system built on these principles, *one.world*[34] could have been used as platform for our system, but like many other similar architectures it relies functionality (such as object serialization) which is not available on all of our target platforms.

Instead of having multiple concurrently running clients, we could use one client that migrates to the current device such as in the Sprite[23] or Emerald[41] systems, but this does not work if a client becomes disconnected from the network since we then do not have any way of migrating to another device. There are also problems with for example migrating a client using the Java AWT API to a device only supporting the Java MIDP API.

A remote desktop solution such as VNC[81] or Microsoft RDP[58] would solve some of our problems but would also introduce new ones. The available bandwidth can be very limited and the latency high, which would not provide a good user interface experience. Also, the user interface can not adapt to the new platform and we have no way of using the application with out network connectivity. For some platforms such as mobile phones these solutions are not practical at all.

4.6 Future work

Plugins for other systems will be implemented. One way to achieve this is to use one of the applications that already have support for all the major instant messaging systems. Not only instant messaging systems can be plugged in, other chat and messaging services can be used as well.

**http://www.marratech.com/

Since most wireless communication technologies can be used to position the user, positioning services can be plugged in to provide valuable information to other users, such as adding the user's current position or the last known position to the auto response message in ICQ. The Alipes platform [63] provides an architecture for location aware applications and can use several different positioning technologies and combine different positioning sources to one more accurate position.

One form of mobile platforms that need special user interface design is wearable computers, where the user interface can be limited to a low-resolution head mounted display and a chord keyboard or even only audio as the only means of input or output. See e.g. [85] for an example of the problems of user interface design for wearable computers.

4.7 Discussion

Using a system like MIM on a mobile phone to send short text messages can be a lot cheaper than using the traditional SMS service. Comparing current prices for the largest Swedish phone operator, we see that the difference is huge. A SMS message has a maximum length of 160 characters and sending a MIM message of 160 characters over GPRS from a phone uses 392 bytes, including HTTP headers and data for both the HTTP request and response. If this data exceeds free monthly included data volume the price is 0.0069 - 0.048 SEK depending on the subscription, while the price for one SMS message is 1.50 SEK. This is about 31-217 times cheaper.

We can also evaluate how our system relates to the different problems we stated in section 4.1. It can be seen that the MIM system handles all of the problems:

Wide variety of user interface possibilities Some platforms can be detected at runtime and the interface adapts to the special characteristics, but other platforms need specialized user interfaces.

Resource limitations State is kept at a server, not on each client reducing the storage needed. An IM client is not a CPU intensive application so there is no problem with the low CPU power of e.g. mobile phones.

Heterogeneous networks We use a low bandwidth protocol that can be used over HTTP, so it can easily be used on devices with limited network capabilities. Clients that can not maintain a continuous connection with the server can run in a polling mode instead. If we loose connection for some time we will regain all lost information when we reconnect again.

Mobility and state Any number of clients can be connected at the same time, and any client has access to the same information and messages. If messages are received and a new client later connects, it will receive these messages as well.

4.8 Conclusions

In this paper we have described a new instant messaging system, MIM, designed for mobile environments that do not suffer from problems current solutions do. The system supports any number of concurrently connect clients, which are synchronized so that a client can connect and disconnect at any time with out losing any messages. The system is designed to interoperate with different current instant messaging systems such as ICQ or MSN Messenger. It can run on different platforms with different capabilities such as PCs, PDAs, mobile phones and wearable computers and can work over temporary, low bandwidth connections.

It illustrates some of the design principles that are necessary in a mobile environment: minimize state kept at the clients, adapt the user interface to the different platforms and allow concurrently running clients.

Acknowledgements

This work was done within the RadioSphere project, which is supported by the Swedish Foundation for Strategic Research and the Objective 1 Norra Norrland - EU structural fund programme for Norra Norrland. Support was also provided by the Centre for Distance-spanning Technology (CDT).

Part 5

The MIM Web Gateway to IP Multicast E-Meetings

The MIM Web Gateway to IP Multicast E-Meetings

Roland Parviainen, Peter Parnes Department of Computer Science/Centre for Distance-spanning Technology
Luleå University of Technology, 971 87 Luleå, Sweden
Roland.Parviainen@cdt.luth.se, Peter.Parnes@cdt.luth.se

January, 2004

Abstract

As video conferencing and e-meeting systems are used more and more on the Internet and in businesses it becomes increasingly important to be able to participate from any computer at any location. Often this is impossible, since these systems requires often special software that are not available everywhere or impossible to install for administrative reasons. Many locations also lack the necessary network infrastructure such as IP multicast. This paper presents a WWW gateway system that enables users to participate using only a standard web browser. The design and architecture of the system are described and performance tests that show the scalability of the system are also presented.

Keywords: Multimedia gateways, Video conferencing, Multicast, RTP, HTTP, WWW

5.1 Introduction

Video conferencing and e-meetings systems are today commonplace on the Internet in several environments, especially in businesses that want to save money on travel costs or time spent on going to physical meetings. These systems are used for meetings, presentations and to continuously support for collaboration in a group setting, providing among other features text messaging, audio and video, and shared collaboration tools. Unfortunately it is impossible to participate in a video conference or an e-meeting session in many locations and occasions for various reasons. For example, many public access Internet terminals at e.g. web cafes, hotels or airport terminals offer only web access through a web browser and do not allow users to install custom software. Audio and video capture devices might also be missing. In other locations the access network might not support the necessary protocols such as IP Multicast.

One way to enable participation in e-meetings in a wide variety of different locations is by connecting e-meeting sessions to the World Wide Web. To be able to support as many systems as possible, the interface must be kept simple and limited to basic web standards such as HTTP/1.0[5] and HTML 3.2[77] or 4.01[78]. While this does not enable a full two-way participation as audio and video input are limited it still creates a great benefit to mobile users. This paper presents the architecture and evaluation of a web interface gateway system to e-meetings that enables users to participate in an e-meeting session from any place through

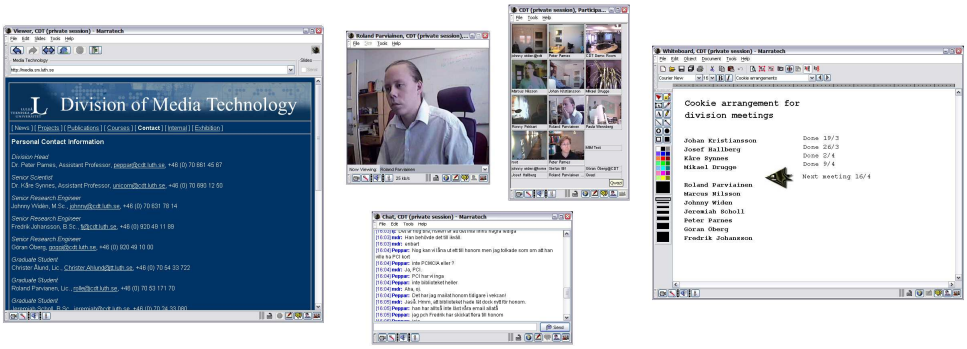

Figure 5.1: Screen-shot of the different tools of Marratech Pro

a standard web browser. The system is implemented using a commercial e-meeting tool, Marratech Pro.

5.2 Background

At Luleå University of Technology collaborative workspaces are used daily in a wide variety of situations. Example uses include allowing students to view classroom lectures from home, enabling members of discussion groups to interact from a distance, and providing members of projects and research divisions with increased presence of each other throughout their work day. The last mentioned case is referred to as the "e-corridor".

In the e-corridor members of the session can be both active or passive and the possible uses of the workspace range from giving or listening in a formal presentation to passive monitoring of video. Communicating with colleagues through the e-corridor often replaces other communication media such as e-mail, phone calls or personal visits. Users with broadband Internet access at home can choose to be part of the e-corridor, join meetings and follow presentations.

5.2.1 Marratech Pro

The software used for the e-corridor is Marratech Pro, a commercial application by Marratech AB[†], which is based on earlier research done by the Media Technology division[68]. Marratech Pro provides the users with the possibility to send and receive audio and video to and from other participants. In addition to audio, video and text messaging there is a shared whiteboard and a shared web browser. The chat, shared whiteboard and shared web browser tools all use proprietary protocols while the audio and video tools use standard protocols such as RTP as far as possible. It is designed to use either IP multicast or unicast. In the latter case traffic is tunneled through a media gateway called the "E-Meeting Portal". The

[†]http://www.marratech.com

video streams from participants in an e-meeting is presented in the "participants" window, which gives a thumbnail overview of all the video streams currently received from the group, while a "focus" window displays the video obtained from a single group member with a higher resolution and frame rate. All audio streams that are not manually muted by the user are mixed and played synchronized to the video streams. The shared whiteboard is mainly used to make quick drawings, sharing applications by incorporating screen shots and sharing Microsoft PowerPoint slides during presentations.

While the e-meeting system provides many benefits in the daily work, it is often impossible get these benefits when out traveling or visiting somewhere and the application is not already installed or cannot easily installed. The Marratech Pro system includes support for tunneling the traffic through firewalls and NAT network, but still requires the user to be allowed to install custom software and have the network capacity to receive the full session. Special versions that do not exist today would also be necessary on devices such as PDAs. A web interface is one solution to the problem of making e-meetings accessible from anywhere where we have a computer connected to the Internet. A web interface would also be useful as a light weight tool that can be used to get a quick overview of what's currently happening in an e-meeting.

5.3 Related work

The WebSmile[39] system is a gateway system for multicast video. It only supports the Motion JPEG codec and do not support any audio codecs. Apart from not having any Java video player applet, the system described in this paper uses the same ideas for video display. Mediascape[82] provides a similar system for video as WebSmile. In Mediascape users can also leave messages to other users through a system called "Post-it" and it is possible to start a direct video call with other users. NYNEX Portholes[24][47] is a web based group awareness tool which provides a web page with video snapshots and activity meters, which is also used in the MIM web interface.

5.4 E-meetings and the WWW

Although HTTP and HTML are commonly used for strictly asynchronous media, several extensions and plug-ins for synchronous media such as video and audio exist, for example streaming media players and Java applets. In this paper we will try to keep to the original features supported by HTML and HTTP as a way to get the widest possible support from the variety of web browser installations available.

A web interface or gateway to an e-meeting session can be implemented or deployed in two different modes:

Personal mode In the personal mode, the web interface is incorporated into the same application that the user uses normally, running on for example a desktop computer in the

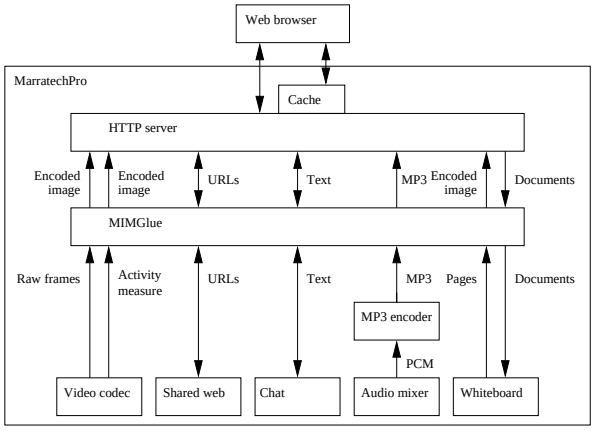

Figure 5.2: Architecture

office. Each user accesses his/her own application to view the web interface directly or through a proxy at the edge of the local network if necessary.

Public mode The web interface can also run at a central location. All users access the same application, often making user identification necessary.

The MIM web interface described in this paper can support both of these modes. It is also possible to divide possible uses of the web interface into different levels depending on the amount of participation:

1. Passively viewing an e-meeting now and then to get a sense of group awareness
2. Participate by chat and the shared web browser and whiteboard tools
3. Listen to audio and see live video
4. Send audio and video, see full frame-rate video and the ability of to edit whiteboard pages

In this paper support for levels 1-3 are implemented requiring only HTML and HTTP support and an MP3 player. Level 4 requires more advanced browser plug-ins.

In the following sections we describe each media of Marratech Pro and possible solutions in more detail.

5.4.1 Chat

Chat can create problems depending on if the gateway is running in a personal or public mode. In the personal mode, public and private chat messages pose no problem since there is only one user of the gateway and the correspondent always knows who the user is. In the public

mode, more than one user can send chat messages from the gateway. If user authentication is done by the gateway the user name can be added to all out going chat messages, making it possible to identify the sending user. This is sufficient for public chat messages, but there are still problems with private chat messages as it is impossible for the gateway to differentiate private messages from the e-meeting session to different users of the gateway if it is part of the session as one unique user. To solve this problem the gateway has to be able to join the session as different users, or the e-meeting software has to be modified to support this. Another low-weight solution is to create a convention where an e-meeting user adds the name of the recipient in private chat messages[‡].

5.4.2 Shared whiteboard

The shared whiteboard tool is one of the most complex parts of Marratech Pro. A user can draw text, lines and other geometrical objects, import text and Microsoft PowerPoint and Word documents, etc. Although supporting the full functionality of the tool in a web browser would be impossible without using advanced web tools such as Java script or a Java applet, only displaying the whiteboard pages are much easier. The pages can be rendered as images which are then viewed in the browser. As a form of input, documents such as images or Microsoft Office documents can be uploaded to the gateway and distributed from there to the session.

5.4.3 Shared browser

When a user distributes a web page the complete page is downloaded and distributed to all Marratech Pro clients from the local cache using a reliable multicast protocol. The built in browser in Marratech Pro displays the original URL in the location field in the GUI, creating the appearance that the page was downloaded from that URL. There are two ways to get access to the shared pages from the gateway system: either the web browser gets access to the page from the local cache in the gateway or the browser uses the original URL of the shared page.

The complete page is sent over reliable multicast to all clients in Marratech Pro due to scalability: just sending the URL would create a HTTP GET request "explosion" as all clients would simultaneously request the same web page at the same time. This problem does not occur in the same way for a gateway system, as users have to manually follow a link to a web page.

5.4.4 Audio

Audio often is the most important part of an e-meeting or video conference, but it is also hard to support with only basic web standards as it is a synchronous media that requires continuous streaming. Several audio formats can be streamed over HTTP, such as Windows media or MPEG-1. MP3, or MPEG-1 Layer III[60], is probably the most well know and

[‡]For example, "[TO Roland] Hi!"

widely supported format. Most browsers need external plug-ins or programs to play these formats. A phone gateway such as he SystemBase Dialgate-2010 or the Cisco VG248 Analog Phone Gateway products could also be used to provide audio support.

5.4.5 Video

Similar to audio video is a synchronous and continuous media, but for video single frames of a video stream can easily be incorporated in a web page. This requires that the gateway decodes the video stream and re-encodes frames in an image format that browsers support. An illusion of a video stream can be achieved by continuously refreshing the image regularly, either by reloading the HTML page that includes the image or by updating the image with push-technologies. The experimental *multipart/x-mixed-replace* MIME-type[7] can be used to push updates to for example images from the server to the client, but it is not implemented by all browsers[§].

To support full frame-rate video streaming additional tools is needed. As in WebSmile[39], a custom Java applet can be used to create a video player in a web page. Plug-ins such as the Window Media, Real Video or Quicktime plug-ins all support streaming video as well. In both cases, the gateway has to re-encode the stream into the new format and stream it over HTTP, unless original format is supported by the player.

Fortunately, we might not need to support the full frame rate of the live video streams: in [13] the required frame rate for video conferencing is reported as being no more than at least 5 fps. If we are only interested in providing awareness, [24] states that a frame rate as low as one snapshot every five minutes can provide group awareness in a work environment.

5.5 Implementation

Building on an existing application has several benefits such as:

- Easy access to already decoded and mixed audio and decoded video frames. All audio and video codecs supported by Marratech Pro are automatically supported by the web interface.
- Easy access to whiteboard pages as low level graphics objects that makes it easy to encode them to various image formats.
- The user only has to run one application.
- All input from the web interface to a session always comes from the same origin as if the user would use the normal tool.

On the other hand, this also means the gateway always requires the full user interface of Marratech Pro and can be hard to get access to from the Internet if the internal network is separated with firewalls and proxies.

[§]Unfortunately it remains unsupported in the most common browser today, Microsoft Internet Explorer.

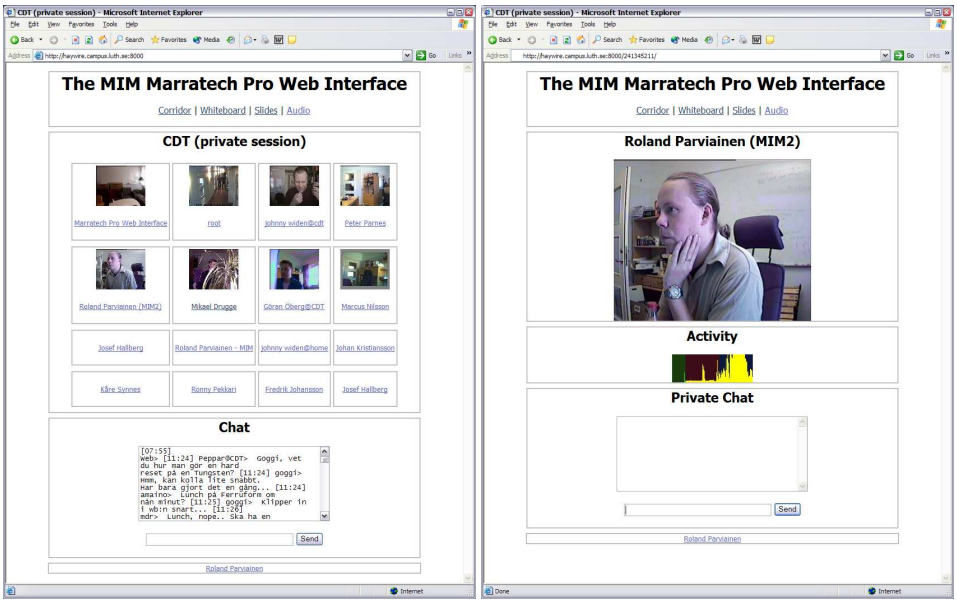

Figure 5.3: Simple web interface

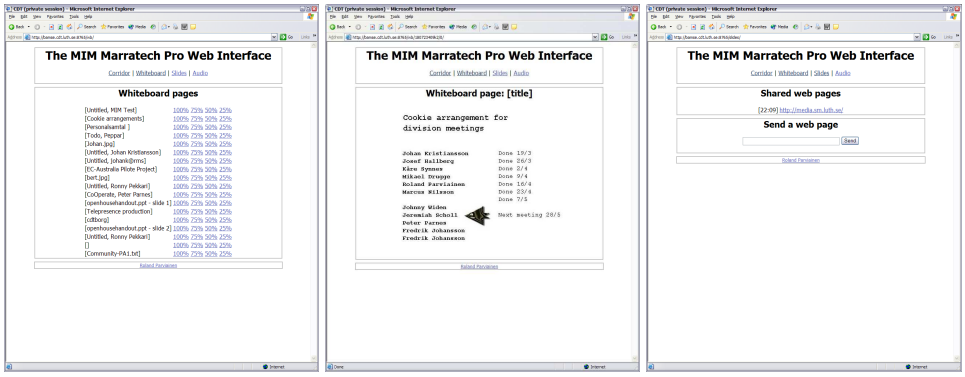

Figure 5.4: Simple web interface

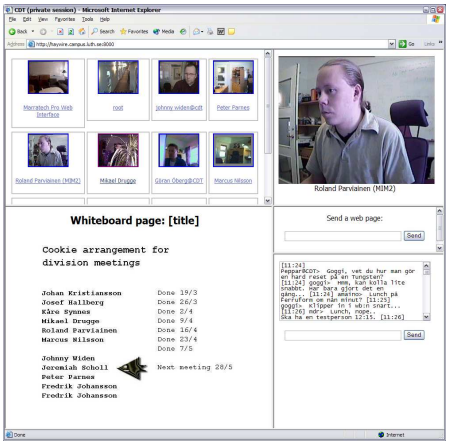

Figure 5.5: Advanced web interface

See figure 5.2 for an overview of the architecture of the system. The MIMGlue component is also responsible for communication with the MIM[70] system, where the name of the system originally comes from. All information to and from the HTTP server are handled by this component. In later section when it is said the HTTP server retrieves information from Marratech Pro it is implied that it goes through the MIMGlue component. Both the HTTP server and the MIMGlue component are running in the same Java process as Marratech Pro.

5.5.1 HTTP and HTML

The HTTP server that is used is included in Marratech Pro and originally based on an HTTP proxy by Diego Cassinera written 1996, although not much is left of that code now. It only supports HTTP/1.0 and each new request is handled by a new thread. Basic HTTP authentication was added to the server for the web interface.

Images are encoded as either JPEG or PNG images; for video images JPEG offers higher compression than PNG, while for whiteboard pages and activity indicators PNG often offer better compression depending on the contents. The defaults are set in the Marratech Pro preferences. The audio is encoded to mono 16Kbit/s, 22KHz MP3 by using a JNI interface to the Lame[¶] MPEG audio encoder.

Almost no HTML code is built in to the software; everything is handled by a template and tag system. HTML template files are stored on disk for the main starting page, different components such as user details, chat and whiteboard pages. Tags in the HTML are replaced with suitable HTML code before web pages are served to clients. This process can be recursive, i.e. HTML code inserted by a tag can contain new tags. Care has to be taken so the recursion is guaranteed to end. The template system means that the complete look and feel of the interface can be changed by just editing HTML code.

[¶] http://lame.sourceforge.net/

Two versions of the templates are currently used: one simple version which does not utilize HTML frames or automatic refresh of content and one advanced version that through frames creates a view that looks more like the normal mode of regular Marratech Pro. Different frames such as the video snapshot display and the chat are automatically reloaded to show new content by using a *<META>* tag in the HTML header. The *multipart/x-mixed-replace* push feature is not used by default due to the lack of support for it in Internet Explorer, but it can easily by added by a simple change to the templates. Figures 5.3 and 5.4 shows a simple version of the interface, while figure 5.5 shows the advanced version.

5.5.2 Activity indicators

As seen in figure 5.3, an activity indicator is added to the individual user views to make it easier to get an overview of the recent activity of a user since we cannot rely on the video stream to provide the information when video updates are far apart. The algorithm for detecting activity in video is similar to the one used in NYNEX Portholes[48]. Different activities such as sending audio or chat messages are also added to the indicator in different colors.

5.5.3 Different Marratech Pro media

Chat Chat messages are intercepted in the Marratech Pro application and forwarded to the HTTP server which stores all messages. When a request for the chat component is received, a HTML form is created and all messages are inserted. Chat messages to the session from the web interface are entered in another HTML form, and transmitted to the HTTP server with a HTTP GET request. The message is then forwarded to the Marratech Pro application which sends it to the session and adds it to the chat panel in the user interface. If the web interface is running in public mode and user authentication is enabled, the login name is prepended to all out going chat messages so remote users can identify the sender.

Shared whiteboard The shared whiteboard HTML component retrieves the list of available whiteboard pages from Marratech Pro. The individual pages can be retrieved by getting a reference to the Java graphics objects that are used to show the pages in the user interface from which images can easily be encoded in different sizes. Documents to be shared with a session are uploaded to temporary storage at the web interface, and the file name is sent to the Marratech Pro application which distributes the new pages in the normal fashion.

Shared browser As new shared web pages arrive from a session, the URLs are stored in the HTTP server and listed when the Shared web HTML component is requested. When URLs are sent from the web interface to the session, the URL is sent to the Marratech Pro application which downloads the web page to the local cache and then distributes it to the session participants through reliable multicast.

Audio Requests for a live audio stream are registered with the MP3 encoder, and the thread handling the request waits for data to arrive to a local queue. Decoded and mixed audio are

intercepted just before it is sent to the audio device. The raw audio data is placed in a queue, where the MP3 encoder retrieves it from. Encoded MP3 data is added to all currently registered HTTP audio request threads' local queue. As data arrives on the queue, it is transmitted over the HTTP connection. If queue grows too large data is discarded. If no audio is being received, silence is sent out to stop players from emptying their local buffers or rebuffering when there are short breaks in the audio.

Video When a request for a video snapshot is received, the data buffer of the last decoded video frame is retrieved, and encoded to the chosen image format which is sent back over the HTTP connection. The image can be scaled to show thumbnails as well.

5.5.4 Caching

To increase the performance, all requests to the HTTP server can be cached. Different types of objects such as video snapshots, whiteboard images or HTML pages can have different cache limits. To prevent web browser and proxies from caching the always changing information HTML <*META*> tags together with the *Pragma: no-cache* HTTP/1.0 header are used to state that the pages should not be cached. Different media have different needs when it comes to caching. As video snapshot often changes the cache should be short while the list of whiteboard pages does not change very often and can therefore be cached for much longer, although this is contrary to the performance gains of caching in the HTTP server as shown in section 5.6.

5.5.5 Privacy

If user authentication or identification is required, the Basic Authentication Scheme in HTTP/1.0 is used. If stronger authentication or encryption is needed, HTTPS (HTTP over SSL) would be the obvious choice. The HTTP server that is currently used does unfortunately not support HTTPS as of today, but SSL tunneling software such as Stunnel[‖] can easily be used together with the web interface.

5.6 Evaluation

The test system used to conduct the evaluation tests described in this section were two PC computers on a local 100 Mbit/s Ethernet network. The client computer was running Microsoft Windows XP Professional on an AMD Athlon XP 2100+ CPU[**] while the server computer was running Mandrake Linux 9.1 on an AMD Duron 800MHz CPU. The server was also connected to the Internet through a 10Mbit connection and the Marratech Pro instance running the web interface was part of an e-meeting session on this network interface

[‖] http://www.stunnel.org
[**] Running at 1733MHz

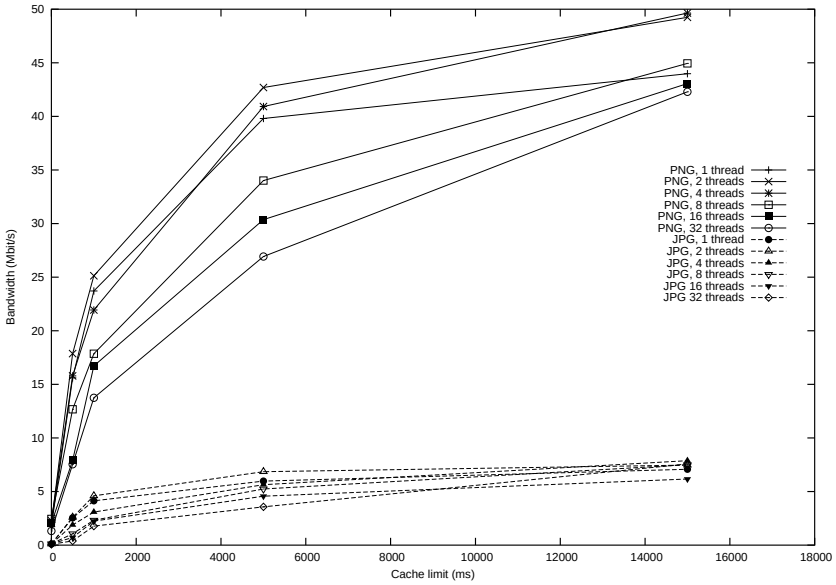

Figure 5.6: Bandwidth used for video images with different cache limits.

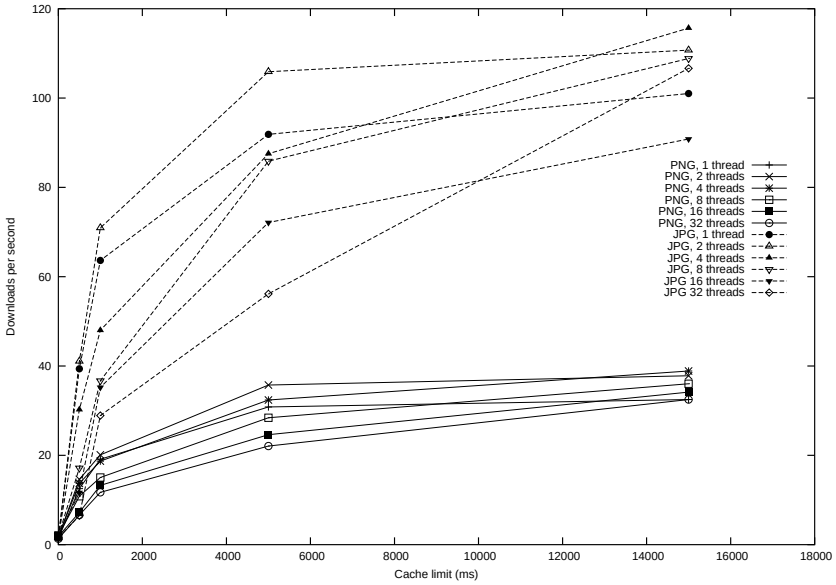

Figure 5.7: Number of successful image downloads per seconds with different cache limits.

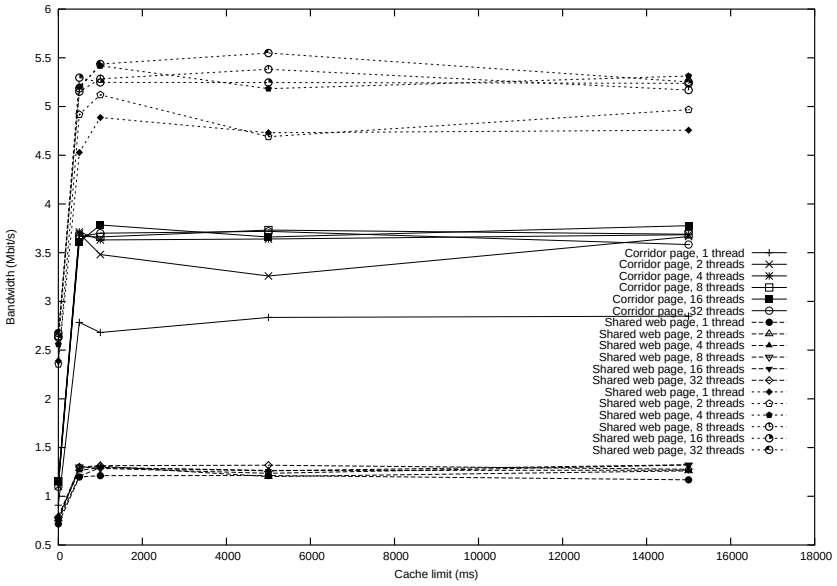

Figure 5.8: Bandwidth used for HTML pages with different cache limits.

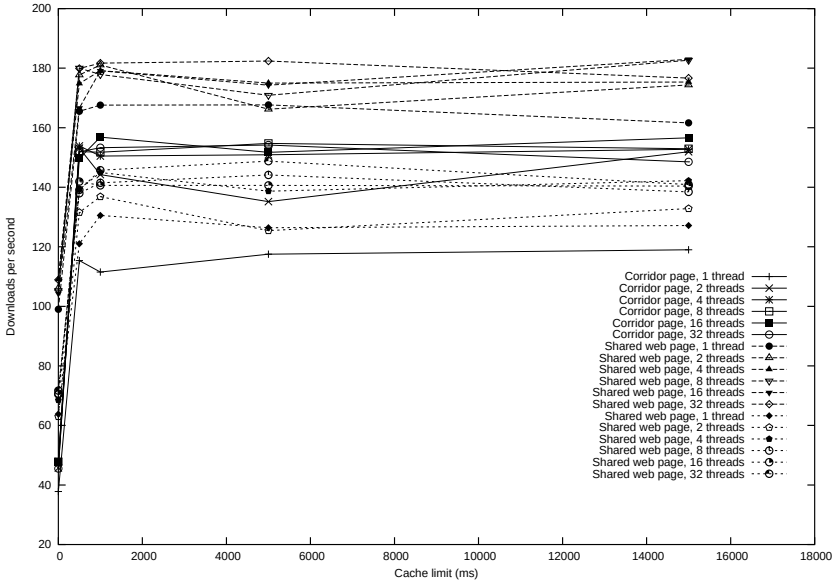

Figure 5.9: Number of successful HTML page downloads per seconds with different cache limits.

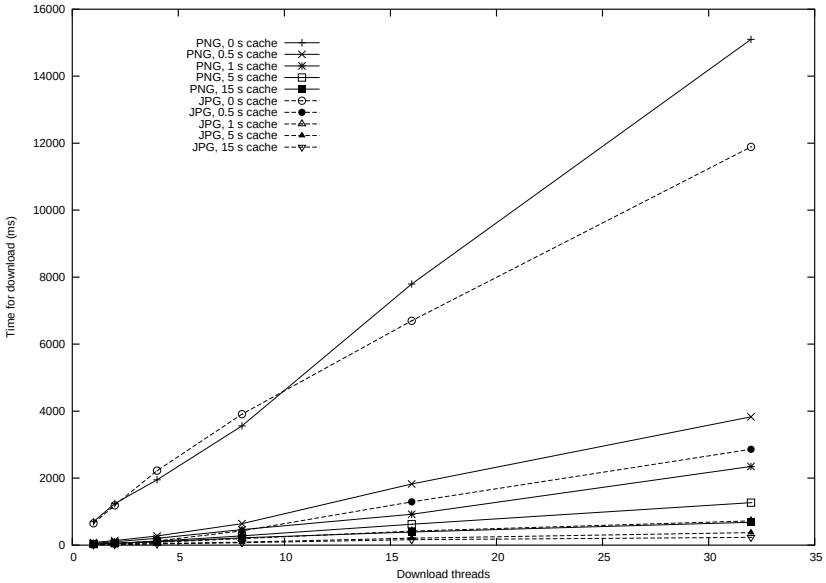

Figure 5.10: Time to download a video image with different number of download threads.

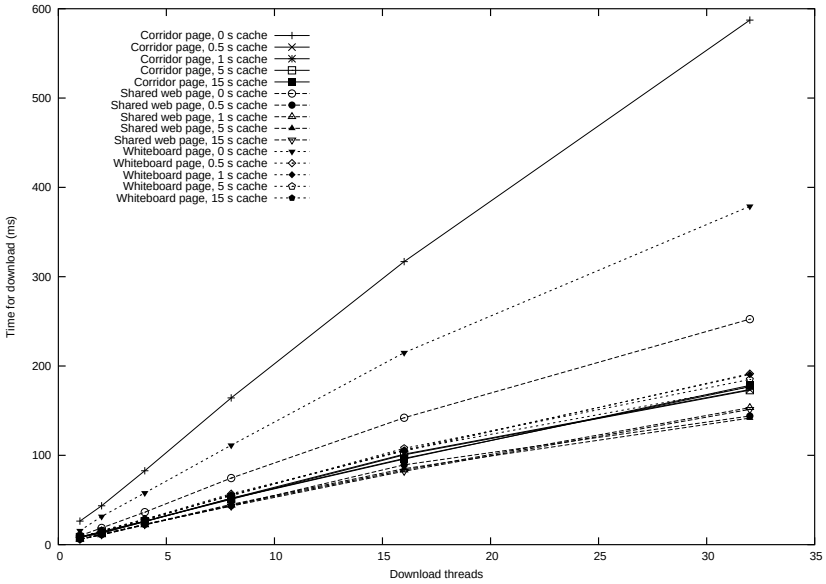

Figure 5.11: Time to download one HTML page with different number of download threads

with a total of 14 users, where 10 where sending video. There where 5 shared web pages and 18 shared whiteboard pages.

The client software was a custom Java application designed explicitly for these tests. Both the client application and the MIM Marratech Pro Web Interface used the Sun J2SE 1.4.1 Java runtime. The test application is given an URL and the number of threads to start at startup. Each thread continuously tries to connect to the web interface through HTTP/1.0, download the data and disconnect as fast as possible. Statistics are regularly forwarded to a manager thread.

5.6.1 Audio receivers

With regards to MP3 audio over HTTP receivers, we evaluate the minimum delay experienced by a user and the scalability of the system with regards to the maximum number of concurrent listeners.

The audio delay experienced by a user is mostly affected by the amount of buffering done by the MP3 player: for example, the default settings of the Winamp v2.81 MP3 player the end to end delay is around 30 seconds. Using the same player, with the minimum allowed settings for HTTP streaming buffer sizes the end to end audio delay was around 4 seconds. This includes the time for encoding to 8Khz 64 kbit/s PCM audio stream in a sender on the same local network, transmission over RTP, decoding to raw audio at 32 Khz, encoding to MP3 and finally download and playout over HTTP. The relatively large delay is caused by several parts: MP3 encoder delay, buffering before data is handed of to HTTP clients, network delay and buffering by the client. Minimization of the delay has not been a goal so far in this work and is left as future work.

Scalability was only tested with up to 500 concurrent receivers. The audio delay for a Winamp 2.81 client as described above was still 4s. CPU usage on the server computer roughly doubled from around 20% to 40%, while the bandwidth used was as expected close to 1Mbyte/s (500 receivers at 16 Kbit/s). This shows that the only limits on the scalability of the number of audio receivers are either the network bandwidth or the amount of concurrent open sockets that the Java virtual machine or operating system can maintain.

5.6.2 Web server performance

All tests where performed with different cache limits, i.e. the time objects in the cache are still considered valid. When objects are invalid, they are regenerated. Both PNG and JPEG image formats were tested, since they offer some tradeoffs: the PNG encoder is faster than the JPEG encoder but produces larger files in general. The PNG video images were approximately 210 Kbyte while the JPEG video images were approximately 10 Kbyte. Three dynamically generated web pages were used for testing: the Corridor page, which list the users of a session and shows a chat text are and was 4674 bytes large, the Shared web page, which list the currently shared web pages and was 4733 bytes and the Whiteboard page, which list the available shared whiteboard pages was 511 bytes large.

In figures 5.6 and 5.7 the effect of the cache for the download speed and the successful number of downloads for both PNG and JPEG images with different amount of download threads is shown. The increase in performance with a large cache limit is clearly visible. The large size of the PNG images results in higher bandwidth usage but smaller amount of successful downloads. The number of concurrent download threads has a noticeable effect on the download speed and rate of successful downloads, where more threads in general reduces the performance of the server. Figures 5.8 and 5.9 shows the same for the three different dynamically generated web pages. Here the beneficial effect levels out after a limit of 1s. The same performance drop when the number of current download thread increases is also visible.

Figures 5.10 and 5.11 the effect of the number of concurrent download threads on the total time for downloading one image or one HTML page. In both cases, the download time grows linearly as the number of threads increases. The benefit of even a short cache limit is clearly visible for both images and HTML pages, but as above longer cache limits than 1s does not enhance the performance in the case of HTML pages.

5.6.3 Availability

The default templates, as described above, have been tested with the most common browsers used today as well as text browser such as Lynx and Links. All features except the *multipart/x-mixed-replace* image push feature works in all tested browsers from Internet Explorer 3 and Netscape 3.04 and up, with the exception that images are not viewable in text browsers without an external helper program.

MP3 streaming over HTTP was tested with a variety of common players on different platforms, such as Windows Media player 7, XMMS, Winamp and iTunes. For some players the delay was substantial as the local buffer size could not be set to an appropriate value. While the delay would be to large for two way audio communication, it is often good enough for combined audio/chat communication.

5.7 Future work

The next step is to examine and evaluate technologies to receive live video streams as well as live audio, by using formats such as Windows Media, MPEG-1, Real Video or Quicktime. As shown in section 5.6, the audio delay is currently quite large and work in minimizing it is needed. The possible use of advanced web technologies such as Javascript, Java applets and Macromedia flash will also be examined. The activity indicator displays was originally designed for small devices such as J2ME capable mobile phones, and should be redesigned as there is much more space available to use on a web page.

5.8 Conclusions

In this paper the architecture and development of a novel Internet e-meeting gateway have been presented. The system fulfill a need for users to be able to participate in multicast e-meetings while using limited terminals in places such as web cafe's and airport terminals and on computers where the user is not allowed to install custom software. It enables users to participate using only a standard web browser and supports a wide variety of different media.

We have discussed two different uses of the system: as a generic gateway into one e-meeting and as a personal gateway to a specific instance of an e-meeting client, and presented an evaluation on the scalability of the former alternative. Using a flexible template system, the appearance and the design of the web pages implementing the e-meeting can easily be customized, with out requiring any change in the source code of the system. The evaluation shows that the system can scale to a large number of concurrent users if needed.

Part 6

A Web Based History tool for Multicast e-Meeting Sessions

A Web Based History tool for Multicast e-Meeting Sessions

Roland Parviainen, Peter Parnes Department of Computer Science/Centre for Distance-spanning Technology
Luleå University of Technology, 971 87 Luleå, Sweden
Roland.Parviainen@cdt.luth.se, Peter.Parnes@cdt.luth.se

June, 2004

Abstract

This paper presents a web based history tool for multicast based e-meeting systems. The tool allow users to see past events and activity graphs of current and past sessions. Video snapshots are stored when events are reported, making it possible to quickly get an overview of the history of a session. This tool has several uses such as a quick and easily accessible way to improve group awareness, monitoring and review of meetings and lectures.

6.1 Introduction

Video conferencing and e-meetings systems are today commonplace on the Internet in several environments, especially in businesses that want to save money on travel costs or time spent on attending physical meetings. These systems are used for meetings, presentations and to continuously support for collaboration in a group setting, providing among other features text messaging, audio and video, and shared collaboration tools and provides a sense of group awareness to members. Tools for reviewing the history of an e-meeting session have several uses:

Group awareness: If a user is not able to participate in a session, the history tool can be used to get a feeling of group awareness by enabling the user to get a quick overview of the history of a session and see who has been active at what times.

Reviewing meetings or lectures: To review meetings or lectures after they have occurred, for example to see which students where active during the lecture can be very important. If the actual content, e.g. what was said, viewing a playback of a recorded meeting would of course be better.

Monitoring and surveillance: For these uses it is very important the system really stores all important video snapshots, i.e. every time something changes in a video stream.

Research: A history tool is invaluable when doing research about the use of e-meeting tools themselves.

Several tools exist to record video conferences and e-meeting sessions, but there are some inherent problems: a need to remember to start the recording, the recording takes up a large

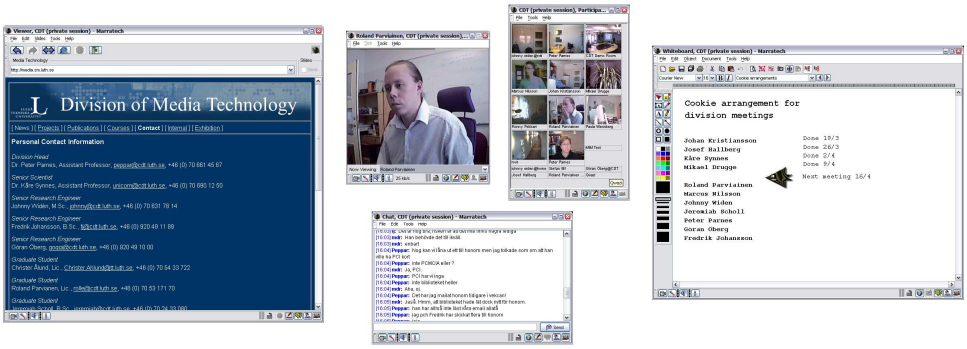

Figure 6.1: Screen-shot of the different tools of Marratech Pro

amount of disk space and a viewing a recording of a session can take a long time and can be cumbersome. To record continuous sessions that run 24 hours a day is not practical. Tools that create summaries of recorded sessions or video are also common[25][50], but while these tools can make it much easier to get a quick overview of a session they still need a complete recording of the session. The history tool described in this paper only store a minimum of information needed; the summaries are created at real time during the session and stored in a database for easy and flexible access to information.

As the storage of video snapshots and other session information can be sensitive the privacy issues have to be considered carefully. In this tool, users can themselves set what information the tool is allowed to use and store. The access to the tool is restricted through basic HTTP authentication and optionally protected through TLS/SSL encryption.

6.2 Background

At Luleå University of Technology collaborative workspaces are used daily in a wide variety of situations. Example uses include allowing students to view classroom lectures from home, enabling members of discussion groups to interact from a distance and providing members of projects and research divisions with increased presence of each other throughout their work day. The last mentioned case is referred to as the “e-corridor”.

In the e-corridor members of the session can be either active or passive and the possible uses of the workspace range from giving a formal presentation to passive monitoring of video. Communicating with colleagues through the e-corridor often replaces other communication media such as e-mail, phone calls or personal visits. Users with broadband Internet access at home can choose to be part of the e-corridor, participate in meetings and follow presentations.

The software used for the e-corridor is Marratech Pro, a commercial application by Marratech AB[†], which is based on earlier research done by the Media Technology research group[68]. Marratech Pro provides the users with the ability to send and receive audio and

[†]http://www.marratech.com

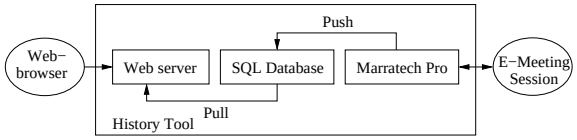

Figure 6.2: Architecture

video to and from other participants. In addition there are a shared whiteboard and a shared web browser. The chat, shared whiteboard and shared web browser all use proprietary protocols while the audio and video tools use standard protocols such as RTP. Marratech Pro can use either IP multicast or unicast. In the latter case traffic is tunneled through a media gateway called the "E-Meeting Portal". The video streams from participants in an e-meeting are presented in the "participants" window, which gives a thumbnail overview of all the video streams currently received from the group, while a "focus" window displays the video obtained from a single group member with a higher resolution and frame rate. All audio streams that are not manually muted by the user are mixed and played synchronized to the video streams. The shared whiteboard is mainly used to make simple drawings, application sharing and sharing Microsoft PowerPoint slides during presentations.

A web interface to Marratech Pro also exist[72]. It is mainly used for accessing the e-corridor when the regular application is not available or not usable. It only uses features available in HTML 4.01, except for audio where it requires a MP3 player that supports HTTP streaming. During the use of the Marratech Pro web interface, it became clear that it would be very useful to be able to see events in past as well, not just what is currently happening.

6.3 Architecture

The Marratech Pro application have been modified to process and send events and data from the currently joined session to an SQL database, from where a web interface retrieves and present information. See figure 6.2 for a description of the architecture. A web front end matches our needs and uses, such as good availability and the ability to get a quick overview of past events and a sense group awareness. Other uses such as analysis of tool use and statistics can use generic SQL query tools. The modified Marratech Pro can either be used as a non active participant in a session whose only task is to receive information or completely replacing the standard version or act as a personal history tool for a single user, recording events for all sessions the user participates in.

6.3.1 Processing of different media

While recording of the video streams of a session provides the complete video history, the storage needs can quickly grow beyond available resources. Only storing snapshots at regular intervals still may result in requiring to much storage space, while this also can miss important events if the interval is too large. Both methods stores large amount of redundancy: for

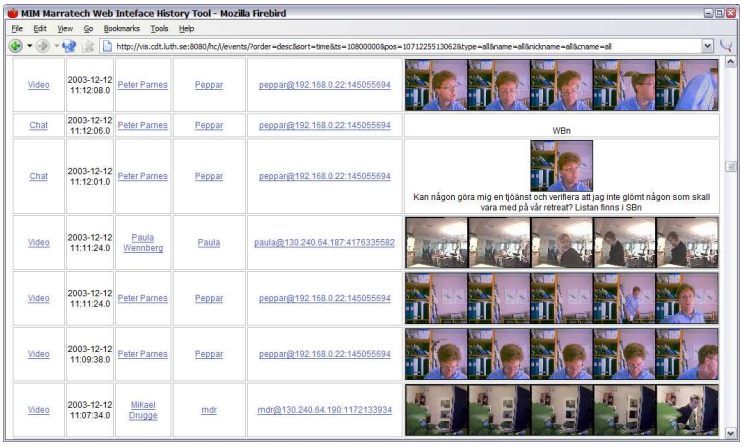

Figure 6.3: Showing events with snapshots

example, in the e-corridor, during night time when there usually is no activity it is unnecessary to store any video information. It can also be difficult to find interesting events. An alternative method which is used in the system presented in this paper is to store snapshots when activity is detected in the video streams. To make sure the event that generated the video activity is represented in the history snapshots are taken before and after activity are detected as well. The algorithm is described in more detail in section 6.4.

When a chat message is received, the text of the message, the sending user and a timestamp is stored in the database and if available, a current video snapshot is also stored. Information about private chat messages between users are not stored. Similarly, when a web page is distributed through the shared web browser the URL is stored along with the sending user, a timestamp and a video snapshot. Audio and whiteboard activity is directly reported as events, with a maximum of 6 events per minute per media. Mixed audio data can optionally be stored in the database, in the form of 16Kbit/s MP3 data.

6.3.2 Web front-end

The web front-end retrieves and presents the data from the database. Two types of displays are available, a list of events with snapshots and a list of activity indicators for each user:

Events with snapshots: The event list shows event type, timestamp, user, snapshot and event data if applicable, i.e. the event is a chat message or a URL. Users can select and sort events by user name, event type and timestamp. The time span to show on one page can also be selected. See figure 6.3 for an example.

Activity indicators: The activity indicator list shows activity indicators for each user which displays different activities in different colors. Each activity indicator shows either one hour, one day or one week of activity. Figure 6.4 shows an example of the activity of one day.

Figure 6.4: Activity indicators

In both views it is possible to retrieve any audio from the selected time interval as a HTTP stream.

6.3.3 Activity indicators

The purpose of the activity indicators is to, in a simple way, get a quick overview of all events in a longer time span that is possible with viewing video snapshots. Activity indicators are created from the data stored in the database when requested. The indicator graph is divided into bins corresponding to time intervals, and as events are added to the graph they are added to the bin matching the event timestamp. The sum of the values of each bin is drawn in the color representing the highest prioritized media type in the bin. The order of priority is as follows: video have lowest priority, then audio, whiteboard, chat and finally shared web browser events have highest priority. This ensures that e.g. the display of irregular and important events such as chat events are not drowned out by events that can occur often such as audio and video events.

6.3.4 User settings

To maintain users' privacy the history tool can filter out different type of events based on user preferences. These preferences are set through private chat messages to the history tool which are parsed and handled. Three messages are supported. First, **!deny [what]**, where **what** can either be a list of media types or "all", specifies what media the user does not want to be stored or processed. Similarly, **!allow [what]** specifies what media are allowed to be stored or processed. By default, all media are allowed. Finally, the **!show** message returns the current settings. The users are identified by the for the session unique RTCP CNAME parameter. Web based preferences was not included in the tool since it is difficult to map users in the session to users of the web based tool as the user parameters in a session can

be chosen arbitrarily by the user and also depends on what computer or network the user is currently using.

6.4 Implementation

Event processing and database insertion part is implemented in Java inside the Marratech Pro application. Events are forwarded from various parts of the application to the history subsystem, which processes event and stores video snapshots in an internal cache in a resolution of 96x72 pixels. As the video change detection algorithm detects video events, the five snapshot in the cache with a timestamp closest to the event timestamp are retrieved from the cache and encoded into JPEG images. The handling of video events are delayed so that snapshots both in the past and in the future relative to the timestamps can be used.

Video activity is detected using the following algorithm, similar to the one used in NYNEX Portholes[48]: the sum of the differences in luminance for each pixel between two consecutive frames is used as an activity value. If the difference between the current value and the average of the previous ten values are greater than a specified threshold a video event is reported.

Decoded and mixed audio is encoded as MP3 frames and are stored in the database together with a timestamp.

All HTTP requests to the web front end are handled in a similar way: the request parameters such as time span and sort order are parsed, a SQL query is formed and executed and the query results are formatted for HTML output or binary output.

The web front end is implemented as a Java Servlet, running under the Apache Tomcat 4.1 Servlet container, communicating with the database through JDBC. The MySQL database server is currently used the database management system.

6.5 Related Work

In [15] Chen describes a system for visualizing the activity in online lectures which includes a similar view as the activity graph view in our system, but a view comparable to the event view is not available. One method for creating summaries for recorded meetings is described in [25]. We employ similar algorithms in realtime in an on-going session to create the history directly. In [50] a system for browsing the history of conferences where spoken language is the main interaction mode is introduced, with some similar features.

6.6 Evaluation and future work

All performance measures in this section were done with the database and servlet engine running on a 2 GHz PC running Windows XP. The tests were run with data from one specific day (24 hours) with a total of 13 users, 1832 events and 8983 snapshots. The complete dataset

spanned a full month, with more than 30000 events and 135000 snapshots. The average snapshot size was 2430 bytes, while the average size of the activity graphs was 2350 bytes. Downloading the activity graphs view (HTML and all images, both dynamically generated) takes 1.1s with the download tool wget over a 10Mbit/s ethernet connection. Downloading the event list with snapshots takes between 3 and 6 minutes, depending on the tool used. The wide variation in time is due to differences in the number of concurrent download threads and efficiancy in rendering. The HTML page was generated and downloaded in 5s, the rest of the time was spent downloading the snapshot images. The total size of the page and images was 27.8 Mbyte.

Only informal studies of the usefulness of the tool have been done through interviews with users. The most requested feature was a search function, which will be implemented in the future. While privacy issues have not been a problem for users used to the concept of the e-corridor, other users have raised concerns over the possibility of the history tool being used as a surveillance tool. Researchers in other institutions have also shown interest in using the tool to study the use of Marratech Pro in a work group.

While the algorithm for detecting changes and choosing snapshots works adequately in most instances a better algorithm is needed for optimal performance.

While it is possible to replay audio, it can be hard to find a particular conversation. One possible way to help searching for spoken communication is to use speech recognition. One problem with speech recognition for our use is that the language spoken in session varies, sometimes even several languages are spoken during one conversation.

6.7 Conclusion

We have a presented a web based history tool which captures the history of a multicast e-meeting session. By storing events and video snapshots around critical moments in a database and providing a web interface to the data, a user can easily and quickly get a good overview of the history of a session from a web browser. Audio from selected time intervals can be streamed or downloaded as MP3 data over HTTP. The use of a database also makes it possible to easily calculate and present different statistics of the use of different media and tools in a session.

Part 7

Experiences of Using Wearable Computers for Ambient Telepresence and Remote Interaction

Experiences of Using Wearable Computers for Ambient Telepresence and Remote Interaction

Mikael Drugge, Marcus Nilsson,Roland Parviainen, Peter Parnes Department of Computer Science/Centre for Distance-spanning Technology
Luleå University of Technology, 971 87 Luleå, Sweden
mikael.drugge, marcus.nilsson, roland.parviainen, peter.parnes}@csee.ltu.se

October, 2004

Abstract

We present our experiences of using wearable computers for providing an ambient form of telepresence to members of an e-meeting. Using a continuously running e-meeting session as a testbed for formal and informal studies and observations, this form of telepresence can be investigated from the perspective of remote and local participants alike. Based on actual experiences in real-life scenarios, we point out the key issues that prohibit the remote interaction from being entirely seamless, and follow up with suggestions on how those problems can be resolved or alleviated. Furthermore, we evaluate our system with respect to overall usability and the different means for an end-user to experience the remote world.

Keywords: Ambient telepresence, mobile e-meetings, remote interaction, wearable computing.

7.1 Introduction

Wearable computing offers a novel platform for telepresence in general, capable of providing a highly immersive and subjective experience of remote events. By use of video, audio and personal annotations and observations, the user of a wearable computer can convey a feeling of "being there" even to those people who are not. The platform also enables a level of interaction between remote and local participants, allowing information to flow back and forth passing through the wearable computer user acting as a mediator. All in all, wearable computers emerge as a promising platform for providing telepresence, yet this statement also brings forward the following research questions:

- What form of telepresence can be provided using today's wearable computing technology?
- How can the telepresence provided be seamlessly used and employed in real-life scenarios?

- What is required to further improve the experience and simplify its deployment in everyday life?

In the Media Technology research group, collaborative work applications are used on a daily basis, providing each group member with an e-meeting facility from their regular desktop computer. In addition to holding more formal e-meetings as a complement to physical meetings, the applications also provide group members with a sense of presence of each other throughout the day. This latter case is referred to as the "e-corridor" — a virtual office landscape in which group members can interact, communicate and keep in touch with each other. As the e-corridor allows fellow co-workers to be together regardless of their physical whereabouts, it has become a natural and integrated part of our work environment.

As part of our ongoing research in wearable computing, we have had the wearable computer user join the e-corridor whenever possible; for example at research exhibitions, marketing events and student recruitment fairs. Since the members of our research group are already used to interact with each other through their desktop computers, we can build on our existing knowledge about e-meetings to study the interaction that takes place with a wearable computer user. This gives us a rather unique opportunity for studying the real-life situations that such a user is exposed to, and derive the strengths and weaknesses with this form of telepresence.

The key contribution of this paper is our experiences and observations of the current problems with remote interaction through wearable computing, and what obstacles must be overcome to make it more seamless. Furthermore, we propose solutions for how these shortcomings can be alleviated or resolved, and how that in turn opens up for further research in this area.

The organization of the paper is as follows: In section 7.2 we give a thorough introduction of our use of the e-corridor, serving as the basis for many of our observations and experiments. This is followed by section 7.3 in which we introduce our wearable computing research, and discuss how a wearable computer user can partake in the e-corridor. Section 7.4 continues by presenting our experiences of this form of telepresence, focusing on the shortcomings of the interaction, both from a technical as well as a social standpoint. The issues identified are subsequently addressed, followed by an overall evaluation of the system in section 7.5. Finally, section 7.6 concludes the paper together with a discussion of future work.

7.1.1 Related Work

Telepresence using wearable computers has been studied in a number of different settings. Early work by Steve Mann, et al., have explored using wearable computers for personal imaging [53, 55], as well as composing images by the natural process of looking around [54]. Mann has also extensively used the "Wearable Wireless Webcam"[†] — a wearable computing equipment for publishing images onto the Internet, allowing people to see his current view as captured by the camera. Our work is similar to this in that we use wearable computers to provide telepresence, yet it differentiates itself by instead conveying the experience into an e-meeting session.

[†] http://wearcam.org/

In computer supported cooperative work (CSCW), telepresence by wearable computers has often been used to aid service technicians in a certain task. Examples of this include [91] by Siegel et al. who present an empirical study of aircraft maintenance workers. This paper addresses telepresence that is not as goal-oriented as typical CSCW applications — instead, emphasis is laid on what ways there are to convey the everyday presence of each other, without any specific tasks or goals in mind. Roussel's work on the *Well*[83] is a good example of the kind of informal, everyday, communication our research enables.

A related example is the research done by Ganapathy et al. on tele-collaboration [29] in both the real and virtual world. This has similarities to our work, yet differs in that we attempt to diminish the importance of the virtual world, focusing more on bringing the audience to the real world conveyed by a remote user. The audience should experience a feeling of "being there", while the remote user should similarly have a feeling of them "being with him" — but not necessarily becoming immersed in their worlds.

In [31], Goldberg et al. present the "Tele-Actor" which can be either a robot or a wearable computer equipped human person at some remote location, allowing the audience to vote on where it should go and what it should do. A more thorough description of the "Tele-Actor" and the voting mechanism in particular can be found in [30]. The function of the "Tele-Actor" is similar to what is enabled by our wearable computing prototypes, but our paper focuses on providing that control through natural human to human interaction, rather than employing a voting mechanism.

As a contrast to using a human actor, an advanced surrogate robot for telepresence is presented by Jouppi in [40]. The robot is meant to provide a user with a sense of being at a remote business meeting, as well as give the audience there the feeling of having that person visiting them. The surrogate robot offers a highly immersive experience for the person in control, with advanced abilities to provide high quality video via HDTV or projectors, as well as accurately recreating the remote sound field. Besides our use of a human being rather than a robot, and not focusing on business meetings in particular, we investigate this area from the opposite standpoint: Given today's technology with e-meeting from the user's desktop, what kind of telepresence experience can be offered by a human user, and is that experience "good enough"?

In [6], a spatial conferencing space is presented where the user is immersed in the wearable computing world communicating with other participants. Another, highly immersive experience, is presented in [98] where Tang et al. demonstrate a way for two users to share and exchange viewpoints generated and explored using head motions. In contrast, our paper does not strive to immerse the user in the wearable computer, but rather provide the experience of an ambient, non-intrusive, presence of the participants. The motivation for this choice is that we want the participants to experience telepresence, and for that reason the remote user is required to remain focused on the real world — not immersed in a virtual world.

In [51], Lyons and Starner investigate the interaction between the user, his wearable computer and the external context as perceived by the user, for the purpose of performing usability studies more easily. Our paper reaches similar conclusions in how such a system should be built, but differentiates itself through our focus on telepresence rather than usability studies.

In [62], we present our experiences from sharing experience and knowledge through the use of wearable computers. We call this the *Knowledgeable User* concept, focusing on how information, knowledge and advice can be conveyed from the participants to the user of a wearable computer. In this paper, we instead discuss how this information can be conveyed in the other direction — from the remote side and back to a group of participants. Furthermore, we elaborate on this concept by discussing the current problems in this setup, our solutions to these, and how the end result allows us to achieve a more streamlined experience.

7.2 Everyday Telepresence

In the Media Technology research group, collaborative work applications are used on a daily basis. Not only are regular e-meetings held from the user's desktop as a complement to physical meetings, but the applications run 24 hours a day in order to provide the group members with a continuous sense of presence of each other at all times. In this section, we will discuss how we use this so called "e-corridor" to provide everyday telepresence.

The collaborative work application that we use for the e-corridor is called Marratech Pro, a commercial product from Marratech AB[‡] based on earlier research [68] in our research group. Marratech Pro runs on an ordinary desktop computer and allows all the traditional ways of multimodal communication through use of video, audio and text. In addition, it provides a shared web browser and a whiteboard serving as a shared workspace, as well as application sharing between participants. Figure 7.1 shows the e-corridor as a typical example of a Marratech Pro session.

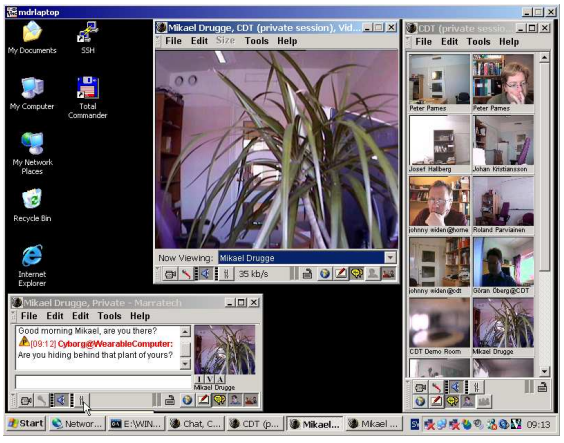

Figure 7.1: A snapshot of a typical Marratech Pro session.

The members of our research group join a dedicated meeting session, the e-corridor, leaving the Marratech Pro client running in the background throughout the day. By allowing

[‡]http://www.marratech.com/

those in the group to see and interact with each other, this provides the members with a sense of presence of each other. Normally, each member works from their regular office at the university, using the client for general discussions and questions that may arise. Even though most members have their offices in the same physical corridor, the client is often preferred as it is less intrusive than a physical meeting. For example, for a general question a person might get responses from multiple members, rather than just a single answer which a physical visit at someone's office may have yielded. Similarly, each member can decide whether to partake in a discussion or not, based on available time and how much they have to contribute. The ambient presence provided by running the client throughout the day allows members to assess their fellows' workload, glance who are present or not, and in general provide a feeling of being together as a group.

However, providing presence for people who are still physically close to each other is not everything; the true advantage of using the e-corridor becomes more apparent when group members are situated at remote locations. The following examples illustrate how the e-corridor has been used to provide a sense of telepresence for its members.

Working from home. Sometimes, a person needs to work from their home for some reason; maybe their child has caught a cold, or the weather is too bad to warrant a long commuting distance. In such situations, rather than becoming isolated and only use phone or email to keep in touch with the outside world, the e-corridor is used to get a sense of "being at work" together with their fellow co-workers.

Living in other places. In our research group, some members have for a period of time been living in another city or country, and thus been unable to commute to their regular office on a daily, weekly or even monthly basis. For example, one doctorand worked as an exchange student in another country for several months, while another person for over a year lived in a city hundreds of miles away. By using the e-corridor, the feeling of separation became significantly diminished; as testimonied by both the remote person as well as the local members remaining, it was sometimes difficult to realize that they were physically separated at all.

Attending conferences. As members of the research group travel to national or international conferences, they have been accustomed to enjoy their fellow co-workers' company regardless of time or place. For example, during long and tedious hours of waiting in the airport, members often join the e-corridor to perform some work, discuss some issue, or simply to chat with people in general. When attending the conference, the remote member can transmit speeches with live video and audio to the e-corridor, allowing people who are interested in the topic to listen, follow the discussion, and even ask questions themselves through that person. If the remote person is holding a presentation, it has often been the case that the entire research group has been able to follow it; encouraging, listening to, and providing support, comments and feedback to the presenter. In a sense, this allows the entire research group to "be there" at the conference itself, and it also allows the remote person to experience a similar feeling of having the group with him.

The seemingly trivial level of presence provided in ways like those described above should not be underestimated; even with simple means, this form of ambient everyday telepresence can have a strong influence on people and their work. Another testimony of the importance of this form of subtle, ambient, presence can be found e.g. in [73], where Paulos mentions similar awareness techniques for attaining user satisfaction.

Subsequently, by enabling a wearable computer user to join the e-corridor, the participants should be able to experience an encompassing form of telepresence. The remote user should similarly be able to feel the participants as "being with him", but not necessarily becoming immersed in the same way as they are.

7.3 Wearable Computers

In this section our wearable computer prototypes are presented, focusing on the hardware and software used to allow the prototypes to function as a platform for telepresence.

In terms of hardware, the wearable computer prototypes we build are based entirely on standard consumer components which can be assembled. The reason for favouring this approach, rather than building customized or specialized hardware, is that it allows for easy replication of the prototypes. For example, other researchers or associated companies who wish to deploy a wearable computing solution of their own, can easily build a similar platform.

The current prototype consists of a backpack containing a Dell Latitude C400 laptop with built-in IEEE 802.11b wireless network support. The laptop is connected to an M2 Personal Viewer head-mounted display, with a web camera mounted on one side providing a view of what the user sees. Interaction with the computer is done through a Twiddler2 hand-held keyboard and mouse, and a headset is provided for audio communication. Figure 7.2 shows the prototype when being worn by one of the authors. This setup allows the user of the wearable computer to interface with a regular Windows XP desktop, permitting easy deployment, testing and studying of applications for mobility.

To perform studies on remote interaction and telepresence, the platform needs suitable software — in our case, we have chosen to run the Marratech Pro client. Figure 7.3 shows the user's view of the application as seen through the head-mounted display.

There are both advantages and disadvantages with using an existing e-meeting application, such as Marratech Pro, for the prototype. The main advantage is that it provides a complete, fully working, product that our research group already uses on a daily basis. This is, naturally, a desirable trait rather than "reinventing the wheel" by developing an application for mobile communication from scratch. It should be noted that as the product is a spin-off from previous research, we have access to the source code and can make modifications if needed, adapting it gradually for use in wearable computing scenarios. The second, perhaps most important advantage, is that the client allows us to participate in the e-corridor. This makes studies, observations and experiments on wearable computing telepresence easy to deploy and setup.

The disadvantage that we have found lies in the user interface which, albeit suitable for ordinary desktop computing, can become very cumbersome to use in the context of wearable computing. This observation holds true for most traditional WIMP[§] user interfaces, for that matter; as noted e.g. by Rhodes in [80] and Clark in [18], the common user interfaces employed for desktop computing become severely flawed for wearable computing purposes. Although the user interface is not streamlined for being used in wearable computing, it remains useable enough to allow a person to walk around while taking part in e-meetings. Furthermore, the problems that emerge actually serve to point out which functions are required for wearable computing telepresence, allowing research effort to go into solving those exact issues. In this way, focus is not aimed at developing the perfect wearable user interface from scratch, as that can risk emphasizing functionality that will perhaps not be frequently used in the end. Rather, in taking a working desktop application, the most critical flaws can be addressed as they appear, all while having a fully functional e-meeting application during the entire research and development cycle.

7.4 Experiences of Telepresence

In this section, the experiences of using a wearable computer for telepresence in the e-corridor will be discussed. The problems that arose during those experiences will be brought forward, together with proposals and evaluations on how those issues can be resolved.

The wearable computer prototype has mainly been tested at different fairs and events, providing a telepresence experience for people within our research group as well as to visitors and students. The fairs have ranged from small-scale student recruitment happenings, medium-sized demonstrations and presentations for researchers and visitors, and to large-scale research exhibitions for companies and funding partners. The prototype has been used in the local university campus area, as well as in more uncontrolled environments — e.g. in exhibition halls in other cities. In the former case, the necessary wireless network infrastructure have been under our direct control, allowing for a predictive level of service as the user roams the area covered by the network. However, in the latter case, the network behaviour is often more difficult to predict, occasionally restricting how and where the user can walk, and what quality of the network to expect. Both these cases, and especially the latter, serve as valuable examples on the shifting conditions that a wearable computer user will, eventually, be exposed to in a real-world setting. We believe it is hard or impossible to estimate many of these conditions in a lab environment, warranting this kind of studies being made in actual real-life settings.

When using a wearable computer for telepresence, unexpected problems frequently arise at the remote user's side — problems that are at times both counter-intuitive and hard to predict. These need to be resolved in order to provide a seamless experience to the audience, or else the feeling of "being there" risk being spoiled. Below follows the primary issues identified during the course of our studies.

[§] Windows, Icons, Menus, Pointer.

7.4.1 User Interface Problems

As mentioned previously, the common WIMP user interfaces employed on the desktop does not work well in wearable computing. The primary reason for this is that the graphical user interface requires too much attention and too fine-grained level of control, thereby causing interference with the user's interaction with the real world. What may not be entirely apparent, however, is that these problems in turn can have severe social implications for the user, and those in turn interfere and interrupt the experience given to the audience.

As an example, the seemingly trivial task for the user to mute incoming audio will be given. This observation was initially made at a large, quite formal fair, arranged by funding partners and companies, but we have experienced it on other occasions as well. In order to mute audio, the collaborative work application offers a small button, easily accessible through the graphical user interface with a click of the mouse. Normally, the remote user received incoming audio in order to hear comments from the audience while walking around at the fair. However, upon being approached by another person, the user quickly wanted to mute this audio so as to be able to focus entirely on that person. It was at this point that several unforeseen difficulties arose.

The social conventions when meeting someone typically involves making eye-contact, shaking hands while presenting yourself, and memorizing the other person's name and affiliation. The deceptively simple task of muting incoming audio involves looking in the head-mounted display (preventing eye-contact), using the hand-held mouse to move the pointer to the correct button (preventing you to shake hands), trying to mute the incoming audio (currently preventing you to hear what the other person says). These conflicts either made it necessary to ignore the person approaching you until you were ready, or to try to do it all at once which was bound to fail. The third alternative, physically removing the headset from the ear, was often the most pragmatical solution we chose to use in these situations.

Although this episode may sound somewhat humorous, which it in fact also was at that time, there are some serious conclusions that must be drawn from experiences like this. If such a simple task as muting audio can be so difficult, there must surely be a number of similar tasks, more or less complex, that can pose similar problems in this kind of setting. Something as trivial as carrying the Twiddler mouse and keyboard in the user's hand, can effectively prevent a person, or at least make it more inconvenient, to shake hands with someone. As the risk of breaking social conventions like this will affect the experience for everyone involved — the remote user, the person approaching, and the audience taking part — care must be taken to avoid this type of problems.

The specific situation above has been encountered in other, more general forms, on several occasions. The wearable computer allows for the remote user to work even while conveying live audio and video back to participants. An example of when this situation occurs is when the remote user attends a lecture. The topic may not be of immediate interest to the remote user, thereby allowing her to perform some other work with the wearable computer in the meantime. However, those persons on the other side who are following the lecture may find it interesting, perhaps interesting enough to ask a question through the remote user. In this case, that user may quickly need to bring up the e-meeting application, allowing her to serve as an efficient mediator between the lecturer and the other persons. In our experience, this

context switch can be difficult with any kind of interface, as the work tasks need to be hidden and replaced with the e-meeting application in a ready state. The cost in time and effort in doing context switches like this effectively prevents a fully seamless remote interaction.

With the goal of providing a seamless and unhindered experience of telepresence, the user interface for the remote user clearly needs to be improved in general. Rather than trying to design the ideal user interface — a grand endeavour that falls outside the scope of this paper — we propose three, easy to implement, solutions to the type of problems related to the user interface of a wearable telepresence system.

- Utilize a "Wizard of Oz" approach [20]. It is not unreasonable to let a a team member help controlling the user interface of the remote user, especially not since there is already a group of people immersed in the remote world. We have done some preliminary experiments on using VNC[81], allowing a person sitting at his local desktop to assist the user of the wearable computer by having full control of her remote desktop. For example, typing in long URL:s can be difficult if one is not accustomed to typing on a Twiddler keyboard, but through VNC the assistant can type them on a keyboard on demand from the remote user. In a similar experiment, one person followed the remote user around, using a PDA with a VNC client running that allowed him to give assistance. It should be noted that this solution still offers some form of telepresence for the assistant, as that person can still see, via the remote desktop, a similar view as would have been seen otherwise.

- Automatically switch between real and virtual world. Even a trivial solution such as swapping between two different desktops — one suitable for the real world (i.e. the e-meeting application for telepresence), and the other suitable for work in the virtual domain (i.e. any other applications for work or leisure that the remote user may be running) — would make life simpler. By letting the switch be coupled to natural actions performed, e.g. sitting down, standing up, holding or releasing the Twiddler, the user is relieved of the burden of having to actively switch between two applications. The advantage may be small, but it can still be significant for efficiently moving between the real and virtual worlds.

- Reduce the need for having a user interface at all.[¶] Plain and simple, the less the remote user has to interact with the computer, the more he can focus on conveying the remote location to the audience. The hard part here is to find a proper balance, so that the remote user can still maintain the feeling of having his group present and following him.

7.4.2 Choice of Media for Communicating

For verbal communication, Marratech Pro offers both audio and text. Either media is important to have access to at certain occasions, as evidenced by our experiences described in [62]. As the wearable computer user is exposed to a number of different scenarios, being

[¶] If a user interface is still required for some reason, our research in the *Borderland* architecture [61] intends to provide ubiquitous access to the tools needed.

able to change between these media is a prerequisite for the communication to remain continuous and free from interruptions. For example, in the case discussed above, the remote participants' spoken comments interfered with the user's real world spoken dialogue. Rather than muting audio, a better solution would have been if the participants had instead switched over to sending their comments as text. This is something that is relatively simple to enforce by pure social protocols; as the participants are already immersed in the world that the user presents, they will be able to determine for themselves when it is appropriate to speak or not. However, although users can switch media at their own choice, this is not an ideal solution for seamless communication. For example, it requires participants to consciously care about which media to use, and does not take in account that they in turn may prefer one media over another for some reason.

To alleviate the problem of having all participants agree on using the same media, we have developed a prototype group communication system in Java that can arbitrarily convert between voice and text. Running the prototype, a user can choose to send using one media, while the receiver gets it converted to the other media. For example, a wearable computer user can choose to receive everything as text, while the other participants communicate by either spoken or written words. As speech recognition and voice synthesis techniques are well researched areas, the prototype is built using standard consumer products offering such functionality; currently the Microsoft Speech SDK 5.1[‖] is used.

The architecture for the system can be seen in figure 7.4. The system accepts incoming streams of audio or text entering through the network, which are then optionally converted using speech recognition or voice synthesis, before they are presented to the user. Similarly, outgoing streams can be converted before they reach the network and are transmitted to the other participants. In practice, the implementation cannot perform speech recognition at the receiving side, nor voice synthesis at the sending side, due to limitations in the speech SDK currently used. Both of these conversions are, however, fully supported at the opposite sides.

The prototype allows the choice of conversions being made to be controlled both locally and remotely. This means that participants can choose by what media communication from the remote user should be conveyed. For example, the remote user may lack any means for entering text, forcing her to rely solely on sending and receiving audio for communication. The participants, on the other hand, may prefer to communicate via text only. E.g. for a person attending a formal meeting, the only way to communicate with the outside world may be sending and receiving text through a laptop. The person in the meeting may therefore request the remote prototype to convert all outgoing communication to text. Similarly, the remote user has his prototype synthesizing incoming text into voice. In this way, a group of people can communicate with each other, with each person doing it through their preferred media.

The prototype runs under Windows XP serving as a proof of concept. Initial experiments have been performed using it for communication across different media. In the experiment, three persons held a discussion with each other, with each person using a certain media or changing between them arbitrarily. The results of these experiments indicate that this is a viable way of enabling seamless communication. Naturally, there are still flaws in the

[‖] http://www.microsoft.com/speech/

speech recognition, and background noise may cause interference with the speaker's voice. Nevertheless, as further progress is made in research on speech recognition, we consider a system like this will be able to provide a more streamlined experience of telepresence.

7.5 Evaluation

In this section we will give an overall evaluation of our wearable system for telepresence. Emphasis will be placed on evaluating its overall usability and the different means for how an end-user can experience and interact with the remote world.

7.5.1 Time for Setup and Use

The time to setup the system for delivering an experience depends on how quickly participants and wearable computer users can get ready. The strength of our approach of utilizing Marratech Pro and the e-corridor, is that the software is used throughout the day by all participants. This means that in all experiments we have performed, we have never had any requirements for persons to e.g. move to a dedicated meeting room, start any specific application, or to dedicate a certain timeslot to follow the experience. For them, the telepresence becomes an ambient experience that can be enjoyed as much or as little as desired, all from the comfort of their own desktop.

As for the user equipped with a wearable computer, the setup time is often much longer due to the reasons listed below.

- The backpack, head-mounted display, headset and Twiddler are surprisingly cumbersome to put on and remove. Even though everything is relatively unobtrusive once fully worn, the time to actually prepare it is too long; for example, the head-mounted display needs to be arranged properly on the user's head, and cables become intertwined more often than not. All this makes the wearable computer less used in situations that warrant its use within short notice.
- The batteries for the laptop and the head-mounted display needs to be charged and ready for use. As this can not always be done with just a few hours worth of notice, this effectively prevents rapid deployment of the wearable computer to capture a certain event.
- The time for the laptop to start, together with gaining a network connection and launching the e-meeting application, is about 5 minutes in total — this is too long to be acceptable.

These are relatively minor problems, yet in resolving these the wearable computer can become more easily used for telepresence experiences than it is today. We consider this to be a prerequisite before it will be commonly accepted outside of the research area as a viable tool for telepresence. Therefore, in order to overcome these limitations, the next generation wearable system we design shall exhibit the properties listed below.

- By using a vest instead of a backpack containing the wearable computer, the head-mounted display, headset and Twiddler can be contained in pockets. This way, they remain hidden until the vest is fully worn and the user can produce them more easily.
- By using an ordinary coat hanger for the vest, a "docking station" can easily be constructed that allows battery connectors to be easily plugged in for recharging. This also makes using the vest-based wearable computer more natural, and thus also more easily used and accepted by the general public.
- By having the wearable computer always on or in a hibernated state when not worn and used, it allows easy restoration of the e-meeting so that anyone can wear and operate it within short notice.

These properties will serve to make the wearable computer easier to wear and use, thereby making it possible for anyone to wear it in order to deliver an experience of telepresence.

7.5.2 Different Levels of Immersion

The e-corridor normally delivers a live stream of information (e.g. video, audio, chat, etc.) which the participants can choose to immerse themselves in. Typically, this is also the most common way of utilizing e-meeting applications like this. However, previous research in our group has added other ways of being part of an e-meeting; the first is a web interface [72], while the second is a history tool [71]. This gives us three distinct levels for how the telepresence can be experienced.

Marratech Pro. Using the e-meeting application, a live stream of video and audio allows the participants to get a first-hand experience of the event. The participants can deliver comments and instructions for the remote user, giving them a feeling of "being there" and allowing some degree of control of that user. Similarly, the remote user can deliver annotations and comments from the event, increasing the participants' experience further. What they say, do, and desire all have an immediate effect on the whole group, making the immersion very encompassing.

Web interface. Occasionally, persons are in locations where the network traffic to the e-meeting application is blocked by firewalls, or where the network is too weak to deliver live audio and video streams. To deal with such occasions, research was done on a web interface [72] that provides a snapshot of the current video, together with the full history of the text-based chat. The web interface can be seen as a screenshot in figure 7.5. Accessing this interface through the web, participants can get a sense for what is currently going on at the moment. Although they are not able to get a live, streaming, experience, the web interface has proven to work good enough to allow participants to control and follow the wearable computer user around.

For example, at one occasion, a person used to doing demonstrations of the wearable computer was attending an international conference, the same day as a large exhibition was to take place at his university back home. As he was away, another person had

to take on his role of performing the demonstration. Due to problems in the network prohibiting the regular e-meeting client to run properly, the web interface was the only possible way of joining the e-corridor. Nevertheless, this allowed him to follow that remote user during the demonstration — offering advice and guidance, and even being able to talk (through the remote user) to persons he could identify in the video snapshots. For this person, the web interface allowed him to "be" at the demonstration, while he in fact was in another country, and another time zone for that matter, waiting for the conference presentations to commence. This example serves to illustrate that very small means seem to be necessary to perform effective telepresence, and also how a user can seamlessly switch between different levels of immersion and still have a fruitful experience.

History tool. The history tool [71] is a research prototype that captures and archives events from an e-meeting session. A screenshot of the tool can be found in figure 7.6. The tool allows people to search for comments or video events, as well as browse it in chronological order to basically see what has happened during the last hours, days or weeks (e.g to see whether a meeting has taken place or not). Snapshots of the video for a particular user are recorded whenever a user enters a chat message, together with the text message itself and the time when it was written. Using motion detection techniques, snapshots are also taken whenever something happens in the video stream. E.g. when a person enters or leaves their office, video frames from a few seconds before and after the triggering event will be recorded, thus being able to see whether that person is actually entering or leaving the room. Naturally, this is mainly suitable and used for clients equipped with a stationary camera, because a head-mounted camera tends to move around a lot causing most video to be recorded. Furthermore, events related to a single person can be filtered out in order to follow that particular person during the course of a day, for example.

In terms of telepresence, the tool is, as the name suggests, a history tool and as such does not offer any means for interacting with the persons**. However, it serves as a valuable starting point for someone who has missed the beginning of e.g. the coverage of a certain exhibition, and who wants a summary and recap of the course of events so far. This may be done in order to prepare the user for becoming more immersed when following the rest of the coverage live, something which can be more easily done having first received the summary information as a primer.

The advantage of using the history tool, rather than letting the user watch a complete recording of the events so far, is that the tool often tends to manage capturing the events that are of key interest. For example, as something is seen by or through the wearable computer user, the amount of chat and conversations often rise, thereby capturing a large amount of video as well as audio clips around that point in time. In this way, the history tool serves as an efficient summary mechanism that implicitly captures events of interest; the more interest, the more conversations and actions, and the more will be archived and subsequently reviewed. After having gone through the history tool, the user can easily switch to more live coverage via the client or web interface. Thus, the

** For any interaction, either the Marratech Pro client or the web interface can be used.

history tool serves to make the transgression from the real world to the immersion in the remote world more seamless.

7.5.3 Appearance and Aesthetics

We have found that the appearance and looks of the wearable computer can dramatically influence the audience's experience of telepresence. What we mean by this statement is that the user of a wearable computer tends to stand out among a crowd, often drawing a lot of attention causing more people to approach the person out of curiosity — more so than what would have been the case without wearing the computer. Sometimes, people even become intimidated by being confronted with a "living computer" — again, causing people to react in ways they would not normally do. Although the effects are not always negative[††], it is important to be aware of the fact that they do exist and that they will, invariably, affect how the remote location is perceived. This becomes even more important to bear in mind considering that the audience may have no idea that this takes place, thereby being given a flawed or at least skewed perception of the remote location.

As telepresence should, in our opinion, be able to offer the participants a representation of a remote location that is as true and realistic as possible, measures need to be taken to ensure that the wearable computer will blend in with its user and the surrounding environment. For this reason, our next generation wearable computer will be smaller and designed to hide the technology as much as possible, according to the following criterias.

- A head-mounted display is difficult to hide and, due to its novelty, draws a lot of attention. With a smaller display, optionally mounted on a pair of glasses, it will be less noticed and easier to hide. At the same time, it becomes easier to motivate its use when people ask questions — motivating the use of a large, bulky display does not tend to sound credible to most people we have met. The less focus that is laid on the technology permitting telepresence, the more effective will it be.

- Eye-contact is very important; our experiences have shown that for efficient social interaction, both parties need to see both of each others' eyes. A semi-transparent head-mounted display allows the remote user to get eye-contact, yet one eye remains obscured from the other person's viewpoint. In this respect, the choice of a semi-transparent or opaque display has little impact on telepresence — the primary requirement is that it allows for eye-contact so that the experience delivered is not hindered.

- The camera is very important as it conveys video to the other participants. As discussed in [28], there are benefits and drawbacks with different placements, so a definite answer is hard to give for the case of providing good telepresence. Also, from a socio-technical standpoint, the question is whether the camera should be hidden well enough not to disturb the scene it captures, or if it should remain visible to let people know their actions are being conveyed to other people watching. For the time being, the camera for our wearable computer will remain head-mounted and visible to the spectators,

[††] On the contrary, wearable computers often generate much attention and numerous socially beneficial interactions with people.

since this allows us to effectively convey the scene with a relatively modest level of disturbance.

Referring to the previous discussion regarding eye-contact; in terms of allowing the audience to "meet" with a remote person seen through the wearable computer, they must be given the impression of eye-contact with that person. In [14], Chen presents a study of how accurately persons can perceive eye-contact. The results can be interpreted as suggesting the upper part of the head, rather than the areas in the lower part or shoulder areas, as the proper position for a head-mounted camera. Such placement, e.g. on top of the user's head or at the sides (as it is in our current setup), should provide a feeling of eye-contact for the audience, without drawing too much attention from the user. However, a more formal user study is required to validate this hypothesis of proper placement for eye-contact with a wearable camera.

- The Twiddler mouse and keyboard is currently a prerequisite for interacting with the wearable computer, yet as discussed before in section 7.4.1, it also interferes with the user's interactions in the real world. However, for the sole purpose of providing telepresence, the only interaction that is actually required on behalf of the remote user is when comments need to be entered as text. This observation means that if the participants can cope without such feedback, it will free the remote user's hands and allow for a more effective interaction with the remote environment. This, in turn, should make for a better experience that is not interrupted by the technology behind it. Of course, there is still the question whether this benefit outweighs the lack of textual comments, but that is likely to vary depending on the event that is covered. There may also be other types of keyboard which are less likely to cause this kind of problems, although we have only utilized the Twiddler so far in our experiments.

- Using a vest rather than a backpack to hold the computing equipment will enable the user to move around, and especially sit down, much more comfortably. With a backpack, the user lacks support for his back when sitting or leaning against objects, at the same time the added weight of the batteries and laptop cause fatigue in the area of shoulders and neck. This fatigue tends to reduce the physical movement of the remote user after long hours of covering an event, and this is detrimental for the audience and serves to reduce their motivation for following the event. Also, to allow for an immersive telepresence, the remote user should be able to partake in social activities — especially something as simple such as sitting down discussing with someone over a cup of coffee. Using a vest, the weight and computing equipment is distributed over a larger part of the user's body, thereby making it less obtrusive and permitting more freedom of movement and positions possible.

The above list constitutes our observations of using wearable computers in telepresence. Many of the problems are commonly known in the field of wearable computing, yet their actual implications for telepresence have not been emphasized. Motivated by the need for the experience to be as effective and unbiased as possible, our conclusion is that the appearance and aesthetics of a wearable computer must be taken in consideration when planning to use such a platform for telepresence.

7.5.4 Remote Interactions made Possible

The remote interactions that the system allows is currently limited mainly to unidirectional communication, coming from the persons at the remote side to the local participants who receive it. The people at the remote location currently have no way of seeing the participants, as the remote user is "opaque" in that sense. Participants who wish to speak with remote persons must do so through the user of the wearable computer, who serves as a mediator for the communication. This is further described in [62] where we utilize this opacity in the *Knowledgeable User* concept, where the remote user effectively becomes a representative for the shared knowledge of the other participants. Except for the option of adding a speaker to the wearable computer, thus allowing participants to speak directly with remote persons, we do not have any plans to allow for bidirectional interaction. Rather, we remain focused on providing an ambient sense of presence to the remote user as well as the participants.

7.5.5 Summary

We will summarize this evaluation of our wearable telepresence system in three statements, serving as advice for those who wish to reproduce and deploy a similar system.

- The time to prepare, setup and use the system will influence how much it will be used in everyday situations, warranting the design of a streamlined system if an investment in such technology is to be made.
- A participant can easily shift between different levels of immersion, and even with relatively unsophisticated means get a good experience and interact with the remote environment.
- The aesthetical appearance of the wearable computing equipment should not be neglected, as this may otherwise influence the people at the remote location for better or for worse.

7.6 Conclusions

We have presented our experiences of using a wearable computer as a platform for telepresence, conveying the presence of a remote locations to the participants of a continuously running e-meeting session. Experiences in real-life scenarios such as fairs, events and everyday situations, have allowed us to identify shortcomings and subsequently address these to improve the platform. We have evaluated the platform in terms of overall usability, and motivated what is of importance for the audience's experience to be as seamless as possible. In the introduction, we posed three research questions which we will now summarize our answers to.

- The form of telepresence that can be provided using today's wearable computing technology can be very encompassing; even with an ordinary e-meeting application at the

user's desktop, a fruitful experience can be delivered. For users who are already accustomed to enjoy the everyday presence of their fellow co-workers at their desktops, the step into mobile telepresence is a small one to take in order to extend its reach even further.

- To deliver a seamless experience of telepresence, the remote user must be able to freely interact with his environment, without social or technical obstacles that are not part of what should be conveyed. From a participant's point of view, having access to multiple interfaces (i.e. live, via the web, or via historical accounts) through which an event can be experienced, can be desirable in order for a seamless experience regardless of place and time.
- To simplify the deployment of wearable telepresence in everyday life, the remote user's equipment needs to be unobtrusive to handle and less noticeable, in order not to interfere with the remote environment. The user interface of the remote user must for this reason be highly efficient, while for participants an ordinary e-meeting application can serve to provide an experience that is good enough.

7.6.1 Future Work

We will redesign our current wearable computer prototype and fully incorporate the solutions suggested in this paper, in order to streamline the user's interaction with the wearable computer and the surrounding environment. The long term goal is to make remote interaction more efficient in general, allowing knowledge to pass back and forth between local and remote participants, either directly through the wearable technology itself or through the user of it acting as a mediator.

7.7 Acknowledgments

This work was funded by the Centre for Distance-spanning Technology (CDT) under the VITAL Mål-1 project, and by the Centre for Distance-spanning Health care (CDH).

Figure 7.2: The wearable computer prototype being worn by one of the authors.

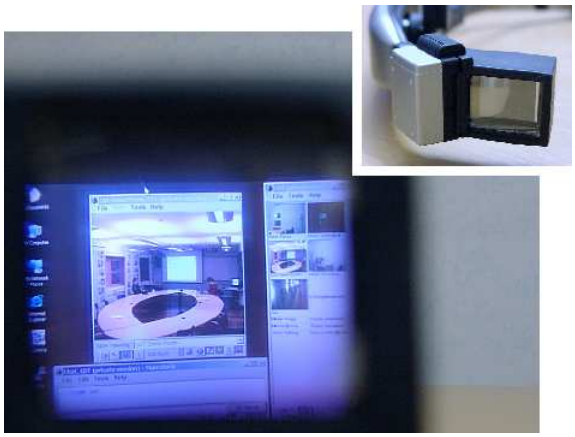

Figure 7.3: The Marratech Pro client as seen through the user's head-mounted display.

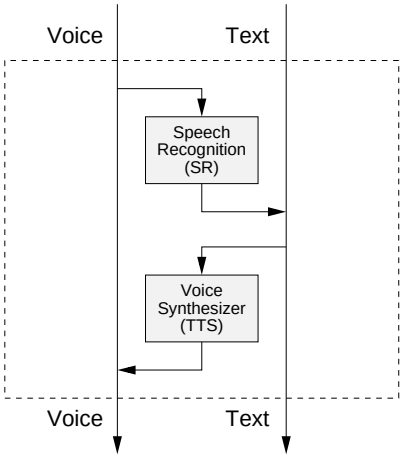

Figure 7.4: Architecture of the voice/text converter prototype, enabling communication across different media.

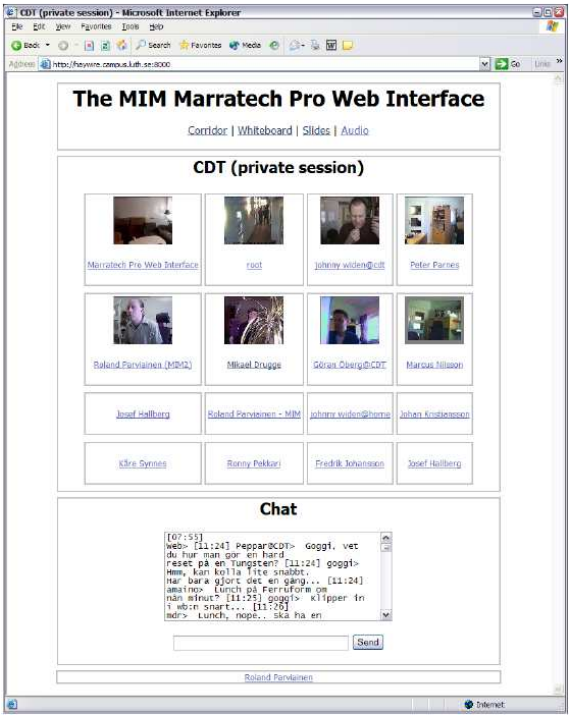

Figure 7.5: A screenshot of the Marratech Pro web interface, allowing access to e-meetings via web browsers.

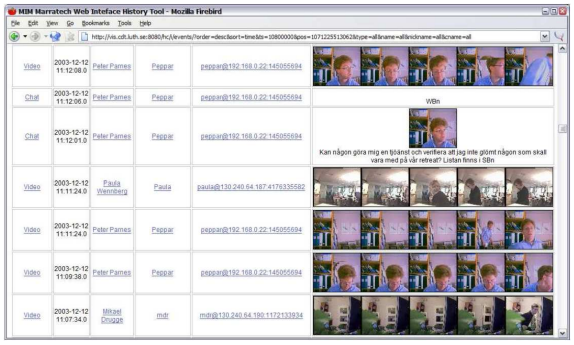

Figure 7.6: A screenshot of the Marratech Pro history tool, archiving events of interest.

Part 8

E-Meeting Services for Mobile Users

E-Meeting Services for Mobile Users

Roland Parviainen, Peter Parnes Department of Computer Science/Centre for Distance-spanning Technology
Luleå University of Technology, 971 87 Luleå, Sweden
Roland.Parviainen@cdt.luth.se, Peter.Parnes@cdt.luth.se

September, 2004

Abstract

This paper presents the Mobile E-meeting Services system, which is a collection of tools for improving the use of e-meeting and video conferencing tools in a mobile environment. The tools allow a mobile user to access an e-meeting session from a web browser or from a Java enabled mobile phone and make it possible to review missed events in the e-meeting through a history tool. The different tools are integrated into a commercial e-meeting tool, Marratech Pro.

8.1 Introduction

Video conferencing and e-meeting systems are today commonplace on the Internet in several environments, especially in businesses that want to increase productivity or save money on travel costs or time spent on attending physical meetings. These systems are used for meetings, presentations and to continuously support for collaboration in a group setting, providing among other features text messaging, audio and video, and shared collaboration tools and provides a sense of group awareness to its users.

However, the wide-spread adoption of video conferencing and e-meeting systems still faces several challenges. In [45] Kouadio and Pooch examines factors and challenges that affect the adoption of video conferencing, both in regards to technology and social issues. Two future challenges they identified are support for mobility and keeping history. Mobility is necessary as many users are mobile and at many instances away from their desktop PC and most of today's tools fail to take this into account. For example, the user might not have permissions to install the required software on the currently available PC or device, the software does not exist for the currently available platform such as Java 2 MIDP enabled mobile phone or PDA or the available network capabilities are not enough to support the applications. History is important in a mobile environment as there is often no possibility to review what has been missed in a session due to temporary lack of network connectivity.

In this paper we describe three tools of the Mobile E-meeting Services (MES) system. The MES system is designed to help mobile users to take part in an on-going e-meeting and consist of the following tools:

- A mobile phone client that can be used from any phone or device that supports Java J2ME MIDP applications.

- A web interface to an e-meeting session with the goal to make an e-meeting available anywhere where an Internet connection and a web browser are available.

- A history system that can be used to review past events in a session. This is important when a user can not continuously be part of session.

All these tools are based on the same principle of utilizing a running instance of a regular e-meeting client application to get access to the session. The first two tools are intended to cover a large majority of devices that a mobile user would need to be able to access an e-meeting from, while the last tool is designed to enable users to see what they have missed while not being able to access the session. A commercially available tool, Marratech Pro[†], was used to implement and test these tools.

As the MES system is supposed to enhance the current work environment and be used daily, there was two specific requirements for this work: firstly, it should be possible to access the e-meeting from as many locations as possible, on any device, as long as there is some Internet access, and secondly, the necessary tools should be deployable on today's devices and networks and not rely on future technology advances.

The usage of these tools can be divided into three different cases: when a user is using a computer were Marratech Pro is or can be installed and have necessary network capabilities (sufficient bandwidth, low latency and not to restrictive firewalls), on a computer where Marratech Pro can not be installed or used or on a mobile phone or PDA. In the first case, the standard Marratech Pro application is used to get access to the session, and the history interface can be used to see past events. In the second case, the web interface is used instead of Marratech Pro, and in the last case, we use the MIDLet client to get access to the session.

8.2 Background

At Luleå University of Technology collaborative workspaces are used daily in a wide variety of situations. Example uses include allowing students to view classroom lectures from home, enabling members of discussion groups to interact from a distance and providing members of projects and research groups with increased presence of each other throughout their work day. The last mentioned case is referred to as the "e-corridor".

In the e-corridor members of the session can be either active or passive and the possible uses of the workspace range from giving a formal presentation to passive monitoring of video. Communicating with colleagues through the e-corridor often replaces other communication media such as e-mail, phone calls or personal visits. Users with broadband Internet access at home can choose to be part of the e-corridor, participate in meetings or follow presentations.

[†] http://www.marratech.com/

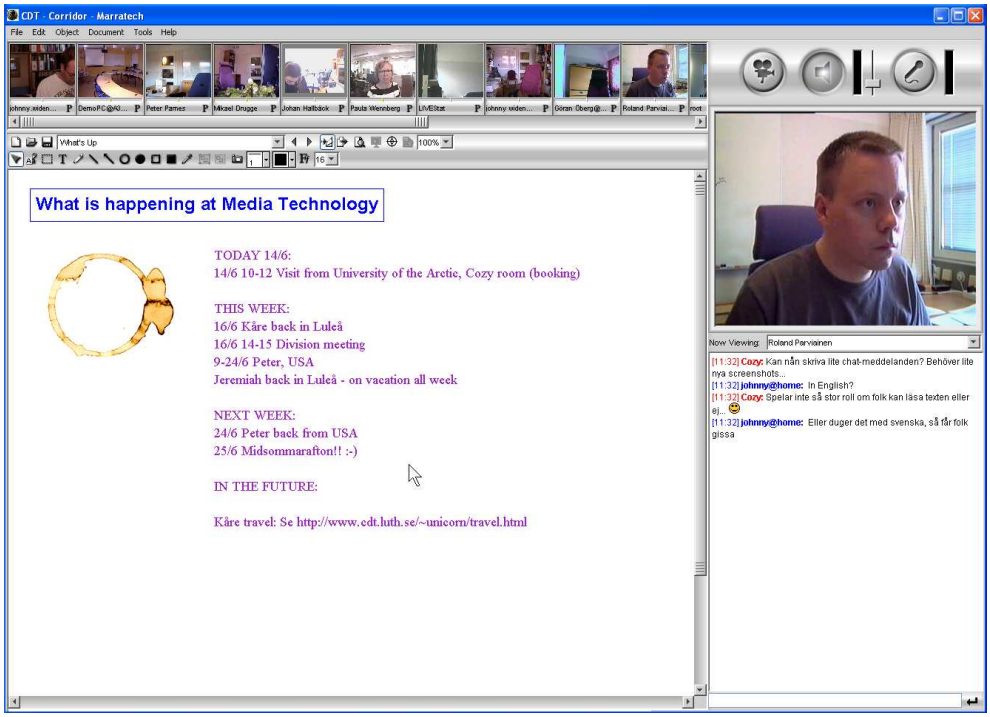

Figure 8.1: Screen-shot of Marratech Pro showing the e-meeting participants, video window, shared whiteboard and public chat. Private chat and the shared browser are not shown.

8.2.1 Marratech Pro

The software used for the e-corridor is Marratech Pro, a commercial application by Marratech AB, which is based on earlier research done by the Media Technology research group[68]. Marratech Pro provides the users with the ability to send and receive audio and video to and from other participants. In addition there is a shared whiteboard and a shared web browser. The chat, shared whiteboard and shared web browser all use proprietary protocols while the audio and video tools use standard protocols such as RTP. Marratech Pro can use either IP multicast or unicast. In the latter case traffic is tunneled through a media gateway called the “E-Meeting Portal”. The video streams from participants in an e-meeting are presented in the “participants” window, which gives a thumbnail overview of all the video streams currently received from the group, while a “focus” window displays the video obtained from a single group member with a higher resolution and frame rate. All audio streams that are not manually muted by the user are mixed and played synchronized to the video streams. The shared whiteboard is mainly used to make simple drawings, application sharing and sharing Microsoft PowerPoint slides during presentations. It is implemented mostly in Java 2 with some parts such as audio and video decoding in native C/C++ code.

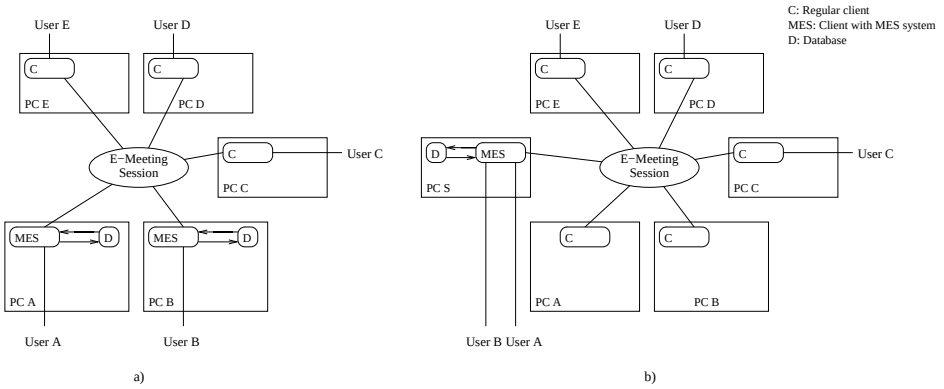

Figure 8.2: In a) users A and B uses their own instances of the mobility system running on their desktop PCs while in b) A and B uses the mobility system running on a centralized server.

8.3 Architecture

The general idea is to extend an existing e-meeting client application with mobility features. By building on an existing application development time decreases considerably as details such as session discovery and session joining, media coding and transport and user preferences are already implemented. Different e-meeting systems can also use proprietary protocols and solutions that would be difficult to support otherwise. If the application's e-meeting functionality is kept intact, the mobility enhanced client can replace a user's ordinary client seamlessly.

On the other hand, the final system can be hard to deploy centrally at a server as one application instance have to be able to handle and represent multiple users and the application might have to be able to run and function without a GUI.

A brief introduction to the different tools shows several similarities:

Web interface A web browser is used to access current data from an e-meeting. Information need to be retrieved from the MES system directly.

History tool The MES system stores events such as video snapshots, audio or chat text in a SQL server. A web interface is used to access the information stored in the SQL server.

MIDlet client A small J2ME MIDP application, called MIDlet, is used to access information from an e-meeting. This is similar to the web interface, except that the output is not formatted as HTML.

All the tools in the system requires a web server; the web interface and history tool are accessed through a web browser and the only network capabilities guaranteed by MIDP 1.0 is through HTTP. A web browser built in to the system eliminates the need for another communication protocol between the web interface and MIDlet client and the MES system.

The history tool only access information from the SQL database and can therefore be either built in or separate.

By implementing all tools as one separate part in the system, here called the *mobility manager*, we can exploit the similarities described above and reduce code complexity. We also achieve a degree of separation between the experimental new tools and the core parts of the e-meeting client which is necessary as different parts of the application is developed separately and the application should continue to function properly.

To be able to easily get access to session information, calls to the mobility manager is inserted into components such as the audio and video decoders The mobility manager intercepts data and provides access to it to the different tools, and can also insert data into the e-meeting. It is also responsible for processing events and inserting them into the history database. The different tools' only interface to the e-meeting is through the mobility manager.

The system can be deployed in several different modes, depending on the location of different services and the intended usage:

Centralized One central instance of each tool, which participates passively in the session, is made available to all users. Users access this instance from a web browser or a mobile phone. In this mode it is easier to configure NATs and firewalls, as only one host needs to be reachable from the outside. More advanced user management is needed by the mobility manager as one running instance need to be able to represent several different users.

Personal The instance of Marratech Pro that users actively use to participate in a session with, e.g. running at their office PC, also acts as a server for mobile access to the session. In this case, all users need to know a publicly routable host address of their own computer. It is also harder to configure NATs and firewalls. As each instance is used by only one single user, there is no need for advanced user management. The user's computer need to be turned on and Marratech Pro have to be running at all time.

Mixed A mixed mode is also possible, where different tools are at different places. For example, the history interface only need access to the history database and not direct access to the session, making it a good candidate to centralize. Users can also decide to use their own instance of different tools instead of utilizing the centralized service.

See figure 8.2 for examples of the different modes.

8.4 Implementation

Most of the MES system is implemented using Sun Java 2 version 1.4.2. Some parts necessary for handling audio, such as MP3 encoding and getting access to mixed and decoded audio, are implemented in native C/C++.

A web server, built into Marratech Pro, is used to for the web interface, the J2ME MIDlet client and optionally for the history interface as well. As most parts of Marratech Pro the web server is implemented in Java and is based on HTTP/1.0. The API is similar to Java Servlets,

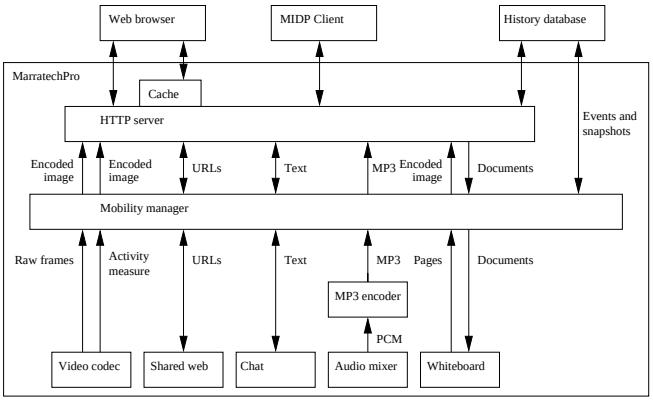

Figure 8.3: Overview of the implementation of the MES system. All data passes through the mobility manager.

making it easy to move services between the HTTP server and a standard servlet container if necessary. A standard SQL server, currently MySQL, running separately from Marratech Pro is used for the history database.

The only part of the system that is separate is the MIDlet client for mobile phones. The client is implemented using MIDP 1.0 as it is still most widespread standard in today's phones. Although some of the additional features in the still new MIDP 2.0 standard could be useful such better media support and TCP and UDP support. As MIDP 1.0 only guarantees networking through HTTP, a polling based protocol is used were HTTP GET requests to mobility manager in Marratech Pro returns messages in a custom binary format. Phones that include a web browser can also use the web interface as well, and the template system described in section 8.4.7 can be used to create web pages that are more suitable for a device with a small display and limited bandwidth.

In the following sections the different media of Marratech Pro and possible problems and solutions are discussed in more detail, followed by an presentation of each tool and how each media is handled specifically in that tool. In general, the different tools use simple and light weight presentation techniques in order to support as many client devices. See figure 8.3 for a general overview of the MES system implementation.

8.4.1 Chat

As chat is a textual, low bandwidth media it is easy to both display and input, even on limited devices such as a mobile phone. The handling of private messages poses more problems, including technical and policy issues. As private messages are sent to everyone in a session and filtered out by the clients, it would be technically possible to use information about both private messages and the actual text. Also, if the tool is running in a centralized mode one instance of the tool need to be able to act as more than one user. This is further described in section 8.4.6.

8.4.2 Audio

Audio is often the most important part of an e-meeting or video conference, but at the same time it is also hard to support with only basic web standards or Java 2 MIDP APIs as it is a synchronous media that requires continuous streaming. Input is especially difficult to handle. While Marratech Pro supports many different codecs, we can get access to already decoded and mixed raw audio to use in the different tools if necessary.

8.4.3 Video

Similar to audio, raw decoded video frames are available and since decoded video frames are used by all tools, the specifics of what video codec is used are not important. For many uses, there is no need to support the full frame rate of the live video streams: in [13] the required frame rate for video conferencing is reported as being no more than at least 5 frames per second. If we are only interested in providing awareness, [24] states that a frame rate as low as one snapshot every five minutes can provide group awareness in a work environment.

8.4.4 Shared web browser

When a user distributes a web page the complete page is downloaded and distributed to all Marratech Pro clients from the local cache using a reliable multicast protocol[65]. The built in browser in Marratech Pro displays the original URL in the location field in the GUI, creating the appearance that the page was downloaded from that URL. There are two ways to get access to the shared pages from the gateway system: either the web browser gets access to the page from the local cache in the gateway or the browser uses the original URL of the shared page.

The complete page is sent over reliable multicast to all clients in Marratech Pro due to scalability: just sending the URL would create a HTTP GET request "explosion" as all clients would simultaneously request the same web page at the same time. This problem does not occur in the same way for a gateway system, as users have to manually follow a link to a web page.

8.4.5 Shared whiteboard

The shared whiteboard tool is one of the most complex parts of Marratech Pro. A user can draw text, lines and other geometrical objects, import text and Microsoft PowerPoint and Word documents, etc. Although supporting the full functionality of the tool in a web browser would be impossible without using advanced web tools such as Java script or a Java applet, only displaying the whiteboard pages are much easier. The pages can be rendered as images that are then viewed in the browser or on a mobile form. As a form of input, documents such as images or Microsoft Office documents can be uploaded to the system and distributed from there to the session.

8.4.6 User management

Users of the different tools are authenticated against a basic user database, which is stored on disk with hashed passwords and managed from inside Marratech Pro. Other authentication methods can easily integrated as well.

When a tool is run in the centralized mode the same instance of Marratech Pro have to represent several different users, possibly at the same time. As a user is authenticated a virtual user is created by generating a new unique RTP SSRC and RTCP CNAME for the user and joining the necessary multicast groups. This virtual user is valid as long as the real user is logged in and active, making the other session members aware of the remote user. After 15 minutes of inactivity, the virtual user is removed from the session.

In the personal mode the same instance always represents one unique user, so there is no technical need to create a virtual user. However, it can be important for group awareness to be able to see when a mobile user accesses the session.

8.4.7 Web Interface

In many locations and on many devices it is impossible to participate in a video conference or an e-meeting. For example, many public access Internet terminals at e.g. web cafes, hotels or airport terminals offer only web access through a web browser and do not allow users to install custom software. Audio and video capture devices might also be missing. In other locations the access network might not support the necessary protocols such as IP Multicast.

One way to enable participation in e-meetings in a wide variety of different locations is by connecting e-meetings to the World Wide Web. To be able to support as many systems as possible, the interface must be kept simple and limited to basic web standards such as HTTP/1.0[5] and HTML 3.2[77] or 4.01[78]. While this does not enable a full two-way participation as audio and video input are limited it still creates a great benefit to mobile users.

Although HTTP and HTML are commonly used for strictly asynchronous media, several extensions and plug-ins for synchronous media such as video and audio exist, for example streaming media players and Java applets. However, in this prototype we will try to keep to the original features supported by HTML and HTTP as a way to get the widest possible support from the variety of web browser installations available.

Chat

As incoming chat messages are forwarded to the HTTP server by the mobility manager, they are timestamped and added to a stack. The messages are rendered in a HTML textarea tag when requested with the newest message first, to avoid the need for the user to always scroll to the bottom manually. Every virtual user, as described in section 8.4.6 has its own stack for private messages. Input is handled by a HTML textfield tag and sent to the server through an HTTP GET request.

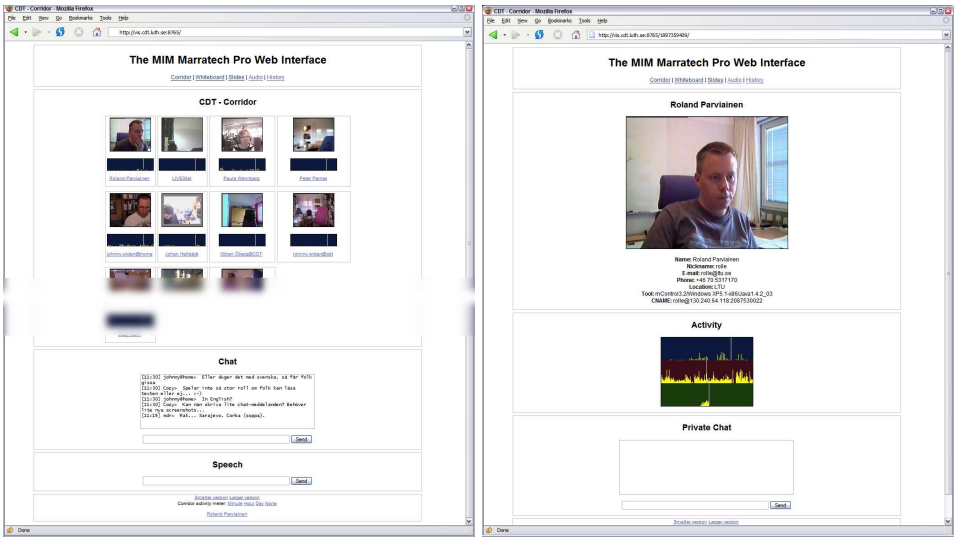

Figure 8.4: The simple web interface, showing the corridor view (left) and the user view (right). Parts of the left screen-shot have been cut.

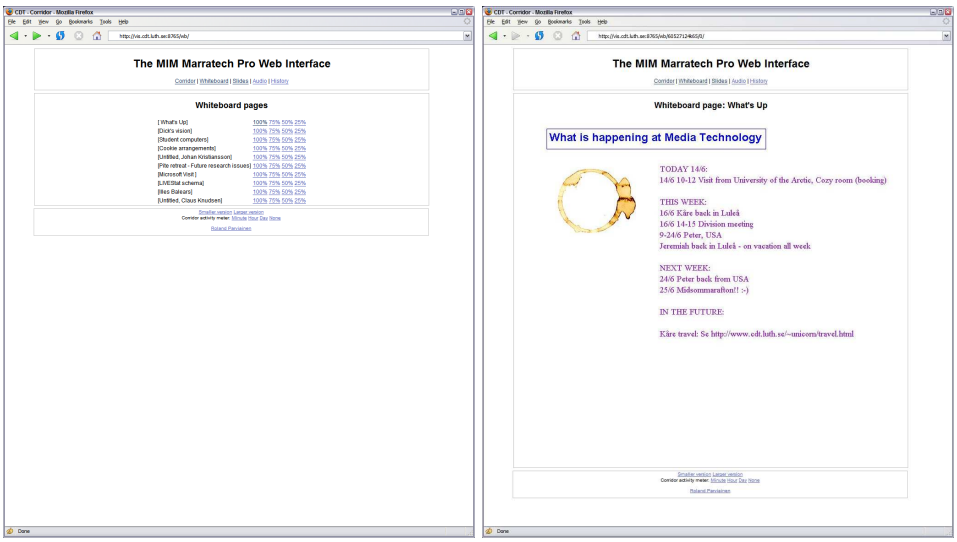

Figure 8.5: The simple web interface, showing the available whiteboard pages and one whiteboard page.

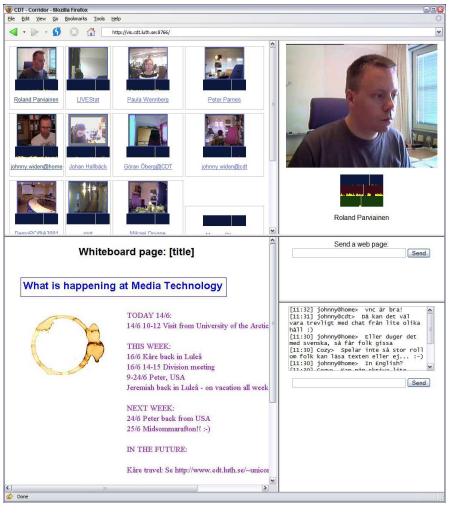

Figure 8.6: The advanced web interface, showing the corridor, video and activity indicator for one user, a whiteboard page and a public chat window.

Audio

Several audio formats can be streamed over HTTP, such as Windows media or MPEG-1. MP3, or MPEG-1 Layer III[60], is probably the most well know and widely supported format. Although most browsers need external plug-ins or programs to play these formats, they are usually included in modern operating system or browser installations. Decoded and mixed, raw PCM audio, are encoded into MP3 (16 Kbit/s, mono[‡]), which is made available as an HTTP stream. There is no support for sending audio from the web interface to the session. A user can also use a phone gateway, as described in [66] to both send and receive audio.

Video

Similar to audio, video is a synchronous and continuous media, but for video single frames of a video stream can easily be incorporated in a web page. This requires that the video frames are decoded and re-encoded into an image format that browsers support. An illusion of a video stream can be achieved by continuously refreshing the image regularly, either by reloading the HTML page that includes the image or by updating the image with push-technologies. The experimental *multipart/x-mixed-replace* MIME-type[7] can be used to push updates to for example images from the server to the client, but it is not implemented by all browsers[§].

To support full frame-rate video streaming additional tools is needed. As in WebSmile[39], a custom Java applet can be used to create a video player in a web page. Plug-ins such as the

[‡] Using the Lame encoder

[§] Unfortunately it remains unsupported in the most common browser today, Microsoft Internet Explorer.

Window Media, Real Video or Quicktime plug-ins all support streaming video as well. In both cases, the gateway has to re-encode the stream into the new format and stream it over HTTP, unless original format is supported by the player.

Shared browser

As new shared web pages arrive from a session, the URLs are stored in the HTTP server and listed when the Shared web HTML component is requested. When URLs are sent from the web interface to the session, the URL is sent to the Marratech Pro application that downloads the web page to the local cache and then distributes it to the session participants through reliable multicast.

Shared whiteboard

The shared whiteboard HTML component retrieves the list of available whiteboard pages from Marratech Pro. The individual pages can be retrieved by getting a reference to the Java graphics objects that are used to show the pages in the user interface from which images can easily be encoded in different sizes. Documents to be shared with a session are uploaded to temporary storage at the web interface, and the file name is sent to the Marratech Pro application that distributes the new pages in the normal fashion.

Speech synthesis

Speech synthesis is implemented in the web interface as an additional input method. Text is entered similar to chat messages and then synthesized to speech as a raw PCM stream by the mobility manager. The PCM stream is then fed into the audio encoder and sent to the net in the same way as spoken audio. A chat message containing the entered text and a note specifying that this text was synthesized is also sent. Speech synthesis is done by the Festival speech system. Different voices, such as male, female, English or Swedish, can be chosen by the user.

Template system

Almost no HTML code is built in to the software; everything is handled by a template and tag system. HTML template files are stored on disk for the main starting page and different components such as user details, chat and whiteboard pages. Tags in the HTML code are replaced with suitable HTML code before web pages are served to clients. This process can be recursive, i.e. HTML code inserted by a tag can contain new tags. Care has to be taken so the recursion is guaranteed to end. The template system means that the complete look and feel of the interface can be changed by just editing HTML code. CSS style sheets are also supported.

Two versions of the templates are currently used: one simple version that does not utilize HTML frames or automatic refresh of content and one advanced version that through frames creates a view that looks more like the normal mode of regular Marratech Pro. Different

frames such as the video snapshot display and the chat are automatically reloaded to show new content by using a <*META*> tag in the HTML header. The *multipart/x-mixed-replace* push feature is not used by default due to the lack of support for it in Internet Explorer, but it can easily by added by a simple change to the templates. Figures 8.4 and 8.5 shows the simple version of the interface, while figure 8.6 shows the advanced version.

Activity indicators

As seen in figure 8.4, an activity indicator is added to the individual user views to make it easier to get an overview of the recent activity of a user since we cannot rely on the video stream to provide the information when video updates are far apart. The algorithm for detecting activity in video is similar to the one used in NYNEX Portholes[48]. Different activities such as sending audio or chat messages are also added to the indicator in different colors.

8.4.8 MIDlet client

While Internet-connected computers are common, they are not everywhere. If it would be possible to access the e-meeting from a device such a mobile phone, the range of locations from where a mobile user can be part of a session increases greatly. Mobile phones become more and more powerful both in regards to networking support and to physical features such as CPU and memory size. More and more phones that are shipped today support both custom Java applications and 2.5G or 3G technologies. Since the main focus is on devices and network technologies available to consumers today, J2ME MIDP 1.0[95] is an obvious choice as a platform. A recent report by the Zelos Group[56] states that the annual sales of Java-capable phones will grow from 39 million to 63 million between 2004 and 2009 in the US alone. The list of J2ME devices from Sun Microsystems[¶] lists more than 135 J2ME devices, from 19 different manufacturers. Of these, 7 support MIDP 2.0 while the rest are listed as supporting MIDP 1.0.

For some of the devices it is also possible for developers to create applications using other platforms than J2ME, such as designing applications to run directly on top of the Symbian OS on certain Nokia or Sony Ericsson phones. These platforms might have fewer limitations than the J2ME platform, but on the other hand the applications become much more platform dependent. Most device manufacturers also adds extra, proprietary Java APIs extending the required API in MIDP; developing for these APIs have the same problem of platform dependency.

The limitations of any solution for our target devices consist of several different parts: the limitations of the APIs provided by the standard that are available to an application developer, of the network technology used by service provider and of the physical limitations of the devices such as display size and CPU power.

In the MIDP 1.0 specification the minimum required display for a device is a screen size of 96x54 pixels with a display depth of 1 bit. Memory resources are also very limited; the MIDP 1.0 specification requires 32 kilobytes of volatile memory for the Java run-time.

[¶] http://wireless.java.sun.com/device/

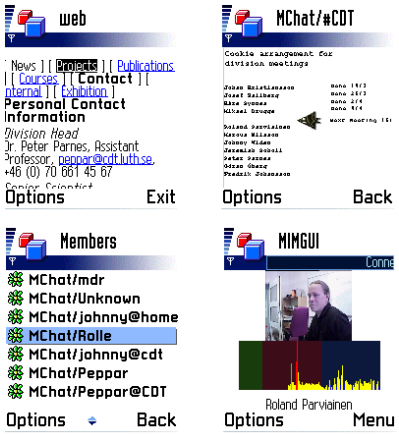

Figure 8.7: Examples of the shared web browser, whiteboard, user list and combined activity and video display running on a Nokia 7650 phone

The only networking capabilities that a MIDP 1.0 implementation must support are through HTTP 1.1; in particular the HEAD, GET and POST requests must be supported. None of the standard network technology solutions for mobility and group communication such as IP multicast, media and transcoding gateway or mobile IP works.

Although some phones have digital still image or video cameras there are no standard APIs for accessing the capture device directly or for accessing the images after they are captured, making it very cumbersome to support any image input to a Java program. While image capture has not been technology commonly associated with mobile phones until the recent popularity of digital camera phones, audio capture is of course one of the main points of any phone. Still, there exist no standard API for accessing the microphone, the speaker or the camera. This leaves the keypad as the only choice for input.

In Sweden, data transmission over the mobile phone network is supported over either GSM data, GPRS or UMTS. GPRS has a theoretical maximum data transmission speed of 171.2 Kbit/s utilizes all 8 available time slots. The number of timeslots available to a GPRS terminal is limited both by limitations in the terminals and in the network. Typically 1, 2 or 3 timeslots are available. UMTS, a 3G solution, supports data transmission speeds from 64 and 144 Kbit/s in large macro cells, 384 Kbit/s in micro-cells and small macro-cells and up to 2 Mbit/s in pico-cells. In other countries several other technologies exist such as Cdma2000 in the US, Korea and Japan and FOMA, a 3G W-CDMA service in Japan.

Architecture

The same HTTP server that is used for the web interface is also used to provide services to the MIDlet clients. All requests from clients are handled similarly to requests from browser

clients, except that the returned information is formatted in a binary message based format instead of HTML. The binary message based protocol is the same as in MIM[70], which also included a limited MIDlet client.

As a new connection between a MIDlet client and the mobility manager is established the client registers its display properties and memory capacity. This information can be used by the mobility manager to determine correct image sizes to minimize bandwidth and memory usage.

The user can also set the polling interval, allowing the user to make a compromise concerning the speed of updates, battery consumption and network usage.

Chat

Incoming messages are stored in a stack for each connected MIDlet client in the HTTP server and forwarded to the client at the next update. Chat messages from the phone to the session are sent as HTTP GET request, which the message encoded into the URL. Only new messages since the last update are sent by default. Similar to the web interface, the user can chose to have text messages to the session synthesized to speech.

Audio

Due to limitations of the current APIs available, it is not possible to listen to streaming audio in the MIDlet client. A phone gateway can also be used, if the phone and network technology allows for phone calls at the same time as the MIDlet application are connecting the Internet. For example, this is not possible on most phones using GSM and GPRS. The possibilities of speech recognition are currently being investigated.

Video

Similar to audio, video is hard to support due to its high bandwidth and continuous nature. While all current Java phones can display images, some only have a black and white or a gray-scale display or low resolution display. The current standards do not directly support any video format.

The limited display size makes it impossible to show more than one video frame at the same time, reducing the bandwidth that is needed. Similar to the web interface, requests for video snapshots are forwarded to the mobility manager, which retrieves the current thumbnail image and encodes it as a PNG image. This image is then sent back to the requesting client. Snapshots for different users can be requested at the same time, making it possible to retrieve the complete thumbnail view in one request. The thumbnail image size is 88x72 pixels for PAL video streams.

If the client device has enough memory more than one snapshot can be requested at the same time, reducing the potentially very long round trip times that would accumulate if a user would like to browse through all snapshots.

Shared browser

Although browsers such as Opera[‖] exist for many types of phones, MIDP does not include any way to control external browsers. Distributed URLs can either be displayed so that the URL can manually be entered into the browser, or the ReqwirelessWeb[**] HTML component can be used to display the web page if the user has a license. URLs can be sent to the Marratech Pro instance and distributed similar to the web interface described in section 8.4.7.

Shared whiteboard

While supporting the full functionality of the whiteboard in a mobile device would be very hard, only displaying the whiteboard pages are much easier. The pages can be rendered as images that are then transferred and viewed on the device. Depending on the memory available on the device, the initial image could be either a high resolution version of the whole page or a scaled down version.

The client requests a scaled down image of the complete whiteboard page from the Marratech Pro MIM plug-in in which the user can zoom in on selected parts. Scaling is always done by the plug-in, which means that each zoom or scroll request means a new request to the MIM Server.

Activity indicators

As updates are even more sparse and random than in the web interface, activity indicators are just as important in the MIDlet client. If the connected client has register a display size activity indicator images are generated with a suitable geometry, otherwise a standard size is used.

8.4.9 History Tool

As a mobile user travels or moves between devices and networks, there will be times, sometimes quite long, were it will be impossible or not practical to access an e-meeting even with the previously described tools. To be able to see what was missed during these times, a history tool was developed. Tools for reviewing the history of an e-meeting have several uses in a mobile environment such as enhancing group awareness and reviewing past events. If a user is not able to participate in a session at all times, the history tool can be used to get a feeling of group awareness by enabling the user to get a quick overview of the history of a session and see who has been active at what times. It is also possible to review meetings or lectures after they have occurred, for example to see which students where active during the lecture. A history tool also has other uses, not specific to mobility, such as a tool for monitoring and surveillance or research of e-meeting use.

Several tools exist to record video conferences and e-meeting sessions, but there are some inherent problems: a need to remember to start the recording, the recording takes up a large

[‖] http://www.opera.com/products/smartphone/

[**] http://www.reqwireless.com/

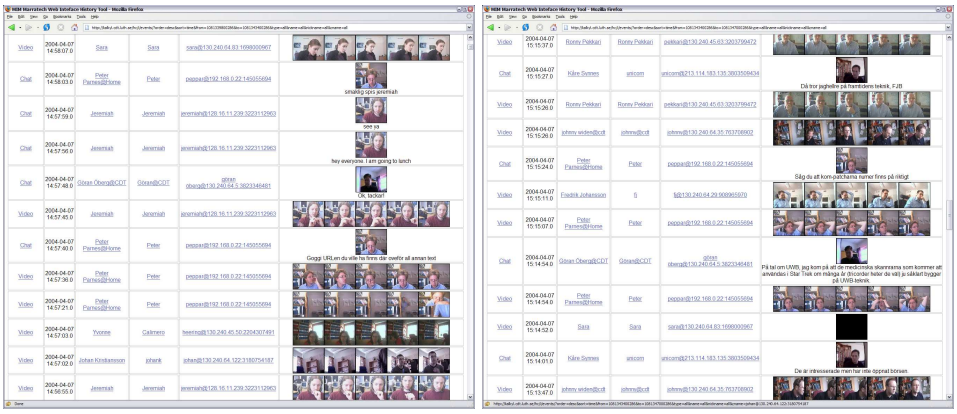

Figure 8.8: Two examples of the event view of the history tool.

amount of disk space and a viewing a recording of a session can take a long time and can be cumbersome. To record continuous sessions that run 24 hours a day is not practical. Tools that create summaries of recorded sessions or video are also common[25][50], but while these tools can make it much easier to get a quick overview of a session they still need a complete recording of the session. The history tool described here only store a minimum of information needed; the summaries are created at real time during the session and stored in a database for easy and flexible access to information.

The mobility manager added to Marratech Pro process sends events and data from the currently joined session to a SQL database, from where a web interface retrieves and present information. A web front end matches our needs and uses, such as good availability and the ability to get a quick overview of past events and a sense of group awareness. Other uses such as analysis of tool use and statistics can use generic SQL query tools.

The web front-end retrieves and presents the data from the database. Two types of displays are available, a list of events with snapshots and a list of activity indicators for each user:

Events with snapshots The event list shows event type, timestamp, user, snapshot and event data if applicable, i.e. the event is a chat message or a URL. Users can search and sort events by user name, event type and timestamp and the time span to show on one page can also be selected. See figure 8.8 for an example.

Activity indicators The activity indicator list shows activity indicators for each user that displays different activities in different colors. Each activity indicator shows either one hour, one day or one week of activity. Figure 8.9 shows an example of the activity of one day.

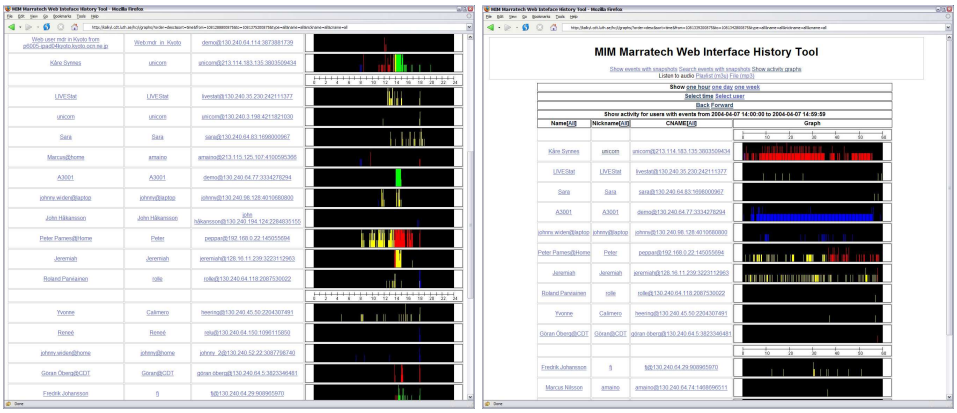

Figure 8.9: The activity indicator view of the history tool. The left example shows one day while the right example shows one hour, 14.00-15.00, of the same day.

Chat

When a chat message is received, the text of the message, the sending user and a timestamp is stored in the history database and if available, a current video snapshot is also stored. Information about private chat messages between users is not stored for privacy reasons. See also section 8.4.9 for more about privacy.

Audio

When audio data is received from a user, audio activity is reported and inserted into the database, with a maximum frequency of 6 events per minute.

As MP3 data encoded for the web interface can be inserted into the database as well. This audio data is a single mixed stream and no information of the current speaker or speakers are stored apart from the audio events described above. Although the bit rate is low, storing all audio data increases the storage requirements. The data management tools described in section 8.4.9 can be used to limit this.

By selecting a link in the event list view the audio data for the currently selected time span is retrieved from the database and served as a MP3-file over HTTP. The data can also be streamed if the player supports it. The length of the audio data can be much shorter than the selected time span as silence is neither stored nor inserted into the stream.

Video

While recording of the video streams of a session provides the complete video history, the storage needs can quickly grow beyond available resources. Only storing snapshots at regular intervals still may result in requiring too much storage space, while this also can miss important events if the interval is too large. Both methods stores large amount of redundancy:

for example, in the e-corridor, during night time when there usually is no activity it is unnecessary to store any video information. It can also be difficult to find interesting events. An alternative method that is used in the system presented in this paper is to store snapshots when activity is detected in the video streams. To make sure the event that generated the video activity is correctly represented in the history, snapshots are taken before and after activity are detected as well.

To be able to store snapshots from a time before events happens, a buffer of snapshots is maintained. When an event happens, the five snapshots with a timestamp closest to the event timestamp is chosen after a short delay to ensure future snapshots are also represented.

Shared browser

The URL of distributed web pages are stored in the database in the same way as chat messages. One problem with this strategy is that the web pages can change between the time they were distributed and the time a user follows a link in the history tool, and thus the displayed history no longer represent the true events. One solution to this would be to download and store the complete web page in the database and make it available.

Shared whiteboard

As a user is editing a whiteboard page, activity is recorded with a maximum frequency of 6 events per minute. Pages are stored as images when they have changed more than a predefined threshold or after a timeout, whatever happens first.

Privacy

The mobility manager maintains a virtual user that represents a history collection agent that maintains privacy preferences for all users. These preferences are set through private chat messages to the virtual user that are parsed and handled. Three messages are supported. First, **!deny [what]**, where **what** can either be a list of media types or "all", specifies what media the user does not want to be stored or processed. Similarly, **!allow [what]** specifies what media are allowed to be stored or processed. By default, all media are allowed. Finally, the **!show** message returns the current settings. While this does not achieve a high level of security for the user as it is easy to spoof policy messages it is deemed sufficient as it would be just as easy for the malicious user to directly store the sensitive information as the decoded (and possibly decrypted) information is available to all participants of an e-meeting.

Data management

Although care has been taken to reduce the data storage needs, the database size will grow linearly over time. To manage the storage space, a simple program is available that can reduce the size through several options such as:

- Reducing the size and/or quality of old images

- Moving old data to another server

- Deleting old data

In the latter two cases, different media can be treated differently; e.g. audio can be deleted while snapshots are moved to another database server.

8.5 Related work

Trying to solve network heterogeneity through active media processing in media gateways is a common approach, see for example [1][67] and [99]. The WebSmile[39] system is an example of a combination of a media gateway and a web interface. It is designed as a www gateway system for multicast video. It only supports the Motion JPEG codec and do not support any audio codecs. Apart from not having any Java video player applet, the web interface described in this paper uses the same ideas for video display. Mediascape[82] provides a similar system for video as WebSmile. In Mediascape users can also leave messages to other users through a system called "Post-it" and it is possible to start a direct video call with other users. NYNEX Portholes[24][47] is a web based group awareness tool which provides a web page with video snapshots and activity meters, which is also used in the web interface described here.

There are several products that try to utilize web browser for conferencing, such as WebEx and Microsoft Office Live Meeting (formerly Placeware). Their version of web conferencing usually combines an audio conference with a slide presentation or software demonstration through a web browser.

Mobile People Architecture (MPA)[52] aims to enable users to be able to contact each other and communicate independently of their location and their current device type. A personal layer is added above the application layer and all communication is routed through each user's personal proxy. Different application drivers handle media conversion and communication with different session layers such as IMAP or ICQ. The MES system working in the personal mode is similar to the MPA architecture. Session and conversion drivers are implemented directly inside Marratech Pro. ICEBERG[100] have similar goals as MPA, but works on the network layer instead.

In [15] Chen describes a system for visualizing the activity in online lectures that includes a similar view as the activity graph view of the history tool, but a view comparable to the event view is not available. In [25] Erol et. al. describes on method for creating summaries for recorded meetings. In the history tool similar algorithms are used in real time in an ongoing session to create the history directly. In [50] a system for browsing the history of conferences where spoken language is the main interaction mode is introduced, with some similar features.

8.6 Evaluation

The unmodified commercially available version of Marratech Pro 3.4 was compared against the mobility enhanced version described in this paper. Both version run on the same computer, a 1GHz PIII running Windows XP, in the same session with 19 members, including 13 video senders. The total bandwith average for the session was 800 Kbit/s. Table 8.1 shows that CPU usage increases substantially; the main cause is the video activity detection algorithm that compares video frames for all video streams. When audio is received the CPU usage increases further due to the need for encoding audio to MP3.

Version	No audio	Audio
Unmodified	10%	11%
Mobility enhanced	38%	55%

Table 8.1: CPU usage

The response time of the web interface was tested during the same session as above. The complete corridor view was downloaded using the wget tool[††] over a 10Mbit/s network and the time was measured using the time tool. The total page size was on average 330Kbyte (varied slightly due to images changing and causing different compression ratios), including all images, CSS style sheets and the complete chat text (1285 lines, 82Kbytes). No caching was enabled in the web server. Table 8.2 shows the minimum, average and maximum times over 50 runs. Even though the download time increases when audio is received and MP3 encoding is enabled it is still adequately low. With modern browser that can download required data such as images concurrently the response time is even faster. More performance tests of the web interface are available in [72].

	No audio	Audio
Minimum	0.8s	1.0s
Average	1.1s	3.2s
Maximum	1.7s	4.6s

Table 8.2: Download time

The response time of the history tool was tested with the database and servlet engine running on a 2 GHz PC running Windows XP. The tests were run with data from one specific day (24 hours) with a total of 13 users, 1832 events and 8983 snapshots. The complete dataset spanned a full month, with more than 30000 events and 135000 snapshots. The average snapshot size was 2430 bytes, while the average size of the activity graphs was 2350 bytes. Downloading the activity graphs view (HTML and all images, both dynamically generated) takes 1.1s with the download tool wget over a 10Mbit/s ethernet connection. Downloading the event list with snapshots takes between 3 and 6 minutes, depending on the tool used. The wide variation in time is due to differences in the number of concurrent download threads and efficiency in rendering. The HTML page was generated and downloaded in 5s, the rest of the

[††] using the wget option –page-requisites

time was spent downloading the snapshot images. The total size of the page and images was 27.8 Mbyte. By limiting the number of events shown per page, i.e. viewing a shorter time span on each page, the download time can be reduced.

The MIDLet client was tested using a Nokia 7650 mobile phone and GPRS. The minimum round-trip time for this setup is around 1s. If the GPRS connection was active, the time for requesting and downloading one snapshot or activity indicator was 11s. The average size of both image types in the test was the same: 2400 bytes. Requesting both a snapshot and an activity indicator took 21s. If the connection was not active, an additional 5s was used for connection setup.

8.6.1 User experiences

The web interface have mostly been used as an awareness tool: to quickly see who is in their office, see the recent chat conversation, checking a whiteboard page, or participating in chat discussions when traveling. It has also been used together with the regular client over low bandwidth modem connections, where the standard Marratech Pro client was used for audio and other media except video and the web interface was used to get video snapshots of the other participants. Other situations were the web interface have been used are on networks were UDP traffic has been filtered by firewalls and on operating systems where Marratech Pro is not supported. The audio support has not been frequently used, as chat and video are the main communication media, not audio, most of the time in the e-corridor. During formal meetings when audio is the main media, users have either managed to install and use the regular client instead or not participated at all.

The history tool has been used to see what have happened at a certain time, view chat history, searching for specific chat messages and seeing when someone was in their office. Some user interface problems have been identified by users and resulted in new features that are currently being tested such as limits on the number of events listed per page to reduce download time and better search capabilities. Most of the current users have a long experience with e-meeting and video conferencing tools and are used to cameras and have not had any specific privacy concerns regarding the storing of session data, although some users have said that it is important to have to the access logs for both the history tool and the web interface.

The MIDLet client has not seen wide use. Some drawbacks that have been mentioned are the slow connection times, large delays and the cumbersome interface on mobile phones. The few times when it has been needed it has been invaluable though.

8.7 Future work

The main focus for future work is on input from the mobile user to the e-meeting session and better support for audio. Work is currently underway to incorporate speech synthesis and speech recognition into the system in order to make the content of audio conversations available and as a way of audio input. This can be used in all three tools. A MIDlet client to the history database is also being considered.

8.8 Conclusions

This paper have presented the Mobile E-meeting Services system, which includes different tools for supporting mobile users of e-meetings and allows a user to access an e-meeting session from a web browser or a Java-enabled mobile phone and thus extending the e-meeting to devices and locations where it previously was hard to participate. A history tool enables users to easily view past and missed events, making it possible to catch up with events in a session after disruptions in network connectivity.

The main current drawbacks of the system are the limited support for audio and input from a remote participant to an e-meeting session. Still, the tools have been found to be useful and an important addition to the daily work environment, especially as a way of improving groups awareness.

Bibliography

[1] E. Amir, S. McCanne, and H. Zhang. An application level video gateway. In *Proceedings of ACM Multimedia 1997*, 1997.

[2] R.J Anderson and C. Manifavas. Chameleon – A New Kind of Stream Cipher. In *Fourth Workshop on Fast Software Encryption*, 1997.

[3] M. Baugher, D. McGrew, M. Naslund, E. Carrara, and K. Norrman. The Secure Real-time Transport Protocol (SRTP), 2004. IETF RFC3711.

[4] V. Bellotti and S. Bly. Walking away from the desktop computer: Distributed collaboration and mobility in a product design team. In *Proceedings of the ACM Conference on computer Supported Cooperative Work*, 1996.

[5] T. Berners-Lee, R. Fielding, and H. Frystyk. Hypertext transfer protocol – HTTP/1.0, 1996. IETF RFC1945.

[6] M. Billinghurst, J. Bowskill, M. Jessop, and J. Morphett. A wearable spatial conferencing space. In *Proc. of the 2nd International Symposium on Wearable Computers*, 1998.

[7] N. Borenstein and N. Freed. MIME (Multipurpose Internet Mail Extensions) Part one: Mechanisms for specifying and describing the format of Internet message bodies, 1993. IETF RFC1521.

[8] B. Briscoe. Marks: Zero side-effect multicast key management using arbitrarily revealed key sequences. In *First International Workshop on Networked Group Communication (NGC)*, 1999.

[9] B. Briscoe and I. Fairman. Nark: Receiver-based multicast non-repudiation and key management. In *ACM Conference on Electronic Commerce*, 1999.

[10] I. Brown, C. Perkins, and J. Crowcroft. Watercasting: Distributed watermarking of multicast media. In *Proceedings of the First International Workshop on Networked Group Communication (NGC)*, 1999.

[11] R. Canetti, J. Garay, G. Itkis, Daniele Micciancio, M. Naor, and B. Pinkas. Multicast security: A taxonomy and efficient constructions. In *IEEE Infocomm'99*, 1999.

[12] I. Chang, R. Engel, D. Kandlur, D. Pendarakis, and D. Saha. Key management for secure internet multicast using boolean function minimization technique. In *IEEE Infocomm'99*, 1999.

[13] M. Chen. Achieving effictive floor control with a low-bandwidth gesture-sensitive videoconferencing system. In *Proceedings of ACM Multimedia 2002*, 2002.

[14] M. Chen. Leveraging the asymmetric sensitivity of eye contact for videoconference. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, 2002.

[15] M. Chen. Visualizing the Pulse of a Classroom. In *Proceedings of ACM Multimedia 2003*, 2003.

[16] B. Chor, A. Fiat, and M. Naor. Tracing Traitors. In *Advances in Cryptology—CRYPTO '94*, 1994.

[17] Y. Chu, S.G. Rao, S. Seshan, and H. Zhang. Enabling conferencing applications on the internet using and overley multicast architecture. In *Proceedings of SIGCOMM*, 2001.

[18] A. Clark. What do we want from a wearable user interface. In *Proceedings of Workshop on Software Engineering for Wearable and Pervasive Computing*, 2000.

[19] D. Clark and D. Tennenhouse. Architectural Considerations for a New Generation of Protocols. In *Proceedings of SIGCOMM*, 1990.

[20] N. Dahlbäck, A. Jönsson, and L. Ahrenberg. Wizard of oz studies: why and how. In *Proceedings of the 1st international conference on Intelligent user interfaces*, 1993.

[21] S. Deering. *Multicast Routing in a Datagram Internetwork*. PhD thesis, Stanford University, 1991.

[22] T. Dierks and C. Allen. The TLS protocol, 1999. IETF RFC2246.

[23] F. Douglis and H. Ousterhout. Transparant process migration: Design alternatives and the Sprite implementation. *Software – Practice and Experience*, 21(8):757–785, 1991.

[24] P. Dourish and S. Bly. Portholes: supporting awareness in a Distributed Work Group. In *Proceedings of CHI 92*, 1992.

[25] B. Erol, D. Lee, and J. Hull. Multimodal summarization of meeting recordings. In *Proceedings of the IEEE International Conference on Multimedia & Expo (ICME 2003)*, 2003.

[26] S. Floyd, V. Jacobson, S. McCanne, C. Liu, and L. Zhang. A reliable multicast framework for light-weight sessions and application framing. In *Proceedings of SIGCOMM*, 1995.

[27] A More Loss-Tolerant RTP Payload Format for MP3 Audio. draft-ietf-avt-rtp-mp3-01.txt, work in progress.

[28] S.R. Fussell, L.D. Setlock, and R.E. Kraut. Effects of head-mounted and scene-oriented video systems on remote collaboration on physical tasks. In *Proceedings of the conference on Human factors in computing systems*, 2003.

[29] S.K. Ganapathy, A. Morde, and A. Agudelo. Tele-collaboration in parallel worlds. In *Proceedings of the 2003 ACM SIGMM workshop on Experiential telepresence*, 2003.

[30] K. Goldberg, Song D., and A. Levandowski. Collaborative teleoperation using networked spatial dynamic voting. *Proceedings of the IEEE*, 91:430–439, 2003.

[31] K. Goldberg, D. Song, Y. Khor, D. Pescovitz, A. Levandowski, J. Himmelstein, J. Shih, A. Ho, E. Paulos, and J. Donath. Collaborative online teleoperation with spatial dynamic voting and a human "tele-actor". In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA'02)*, 2002.

[32] J. Gosling, B. Joy, and G.L. Steele. *The Java^{TM} Language Specification*. Addison-Wesley, 1996. ISBN 0-201-63451-1.

[33] A.M. Graziano and M.L. Raulin. *Research Methods: A Process of Inquiry*. Allyn & Bacon, 4th edition edition, 1999.

[34] R. Grimm, T. Anderson, B. Bershad, and D. Wetherall. A System Architecture for Pervasive Computing. In *Proceedings of the 9th ACM SIGOPS European Workshop*, pages 177–182, 2000.

[35] M. Handley. *On Scalable Internet Multimedia Conferencing Systems*. PhD thesis, University College of London, 1997.

[36] D. Hoffman, G. Fernando, V. Goyal, and M. Civanlar. RTP payload format for MPEG1/MPEG2 video, 1998. IETF RFC2250.

[37] Request for proposals - Embedded signalling systems issue 1.0. International Federation of the Phonographic Industry, 54 Regent Street, London W1R 5PJ, 1997.

[38] ISO/IEC 9075:1992. Information technology – Database languages – SQL, 1992.

[39] M. Johanson. A RTP to HTTP video gateway. In *Proceedings of the tenth international conference on World Wide Web*, 2001.

[40] N.P. Jouppi. First steps towards mutually-immersive mobile telepresence. In *Proceedings of the 2002 ACM conference on Computer supported cooperative work*, 2002.

[41] E. Jul, H. Levy, N. Hutchinson, and A. Black. Finegrained mobility in the Emerald system. *ACM Transactions on Computer Systems*, 6(1):109–133, 1988.

[42] M. Kakihara and K. Sørensen. Mobility: An extended perspective. In *Proceedings of the 35th Hawaii International Conference on System Sciences*, 2002.

[43] C. Kalt. Internet Relay Chat: Client Protocol, 2000. IETF RFC2812.

[44] S. Katzenbeisser and F.A.P. Petitcolas, editors. *Information Hiding: Techniques for Steganography and Digital Watermarking*. Artech House, 2000.

[45] M. Kouadio and U. Pooch. Techonology on social issues of videoconferencing on the Internet: a survey. *Journal of Network and Computer Applications*, 25, 2002.

[46] S. Kristoffersen and F. Ljungberg. Representing modalities in mobile computing, a model of IT use in mobile settings. In *Proceedings of Interactive Applications of Mobile Computing (IMC'98)*, 1998.

[47] A. Lee, A. Girgensohn, and K. Schlueter. NYNEX Portholes: Initial User Reactions and Redesign Implications. In *GROUP'97, Proceedings of the International ACM SIGGROUP Conference on Supporting Group Work*. ACM Press, 1997.

[48] A. Lee, K. Schlueter, and A. Girgensohn. Sensing activity in video images. In *CHI 97 Extended Abstracts*. ACM Press, 1997.

[49] T.J. Lehman, S.W. McLaughry, and P. Wyckoff. TSpaces: The Next Wave. In *Proceedings of the Hawaii International Conference on System Sciences (HICSS-32)*, 1999.

[50] S. Luz and D. Roy. Meeting browser: A system for visualising and accessing audio in multicast meetings. In *Proceedings of the IEEE Workshop on Multimedia Signal Processing*, 1999.

[51] K. Lyons and T. Starner. Mobile capture for wearable computer usability testing. In *Proceedings of IEEE International Symposium on Wearable Computing (ISWC 2001)*, 2001.

[52] P. Maniatis, M. Roussopoulos, E. Swierk, K. Lai, G. Appenzeller, X. Zhao, and M. Baker. The Mobile People Architecture. *Mobile Computing and Communication Review*, 1999.

[53] S. Mann. Wearable computing: A first step towards personal imaging. *IEEE Computer*, 30:25–32, 1997.

[54] S. Mann. Personal imaging and lookpainting as tools for personal documentary and investigative photojournalism. *ACM Mobile Networks and Applications*, 4, 1999.

[55] S. Mann and R.W. Picard. An historical account of the 'wearcomp' and 'wearcam' inventions developed for applications in 'personal imaging'. In *IEEE Proceedings of the First International Conference on Wearable Computing*, 1997.

[56] S. McAteer. Wireless Application Platforms: Java Versus Brew, 2004. Zelos Group, <URL:http://www.zelosgroup.com/>.

[57] S. McCanne, M. Vetterli, and Van Jacobson. Receiver-driven Layered Multicast. In *Proceedings of SIGCOMM*, 1996.

[58] Remote Desktop Protocol (RDP) Features and Performance, White Paper, 2000. Microsoft Corporation.

[59] MPEG Group. <URL:http://cselt.stet.it/mpeg/>.

[60] MPEG Group. ISO/IEC International Standard 11172; coding of moving pictures and associated audio for digital storage media up to about 1,5 mbit/s, 1993.

[61] M. Nilsson, M. Drugge, and P. Parnes. In the borderland between wearable computers and pervasive computing. Research report, Luleå University of Technology, 2003. ISSN 1402-1528.

[62] M. Nilsson, M. Drugge, and P. Parnes. Sharing experience and knowledge with wearable computers. In *Pervasive 2004: Workshop on Memory and Sharing of Experiences*, 2004.

[63] J. Nord, K. Synnes, and P. Parnes. An Architecture for Location Aware Applications. In *Proceedings of the Hawaii International Conference on System Sciences (HICSS-35)*, 2002.

[64] P. Parnes. Scalable Reliable Real-time Transport Protocol - SRRTP. Work in progress*, 1996.

[65] P. Parnes. *An IP-Multicast based Framework for Designing Synchronous Distributed Multi-User Applications on the Internet*. PhD thesis, Luleå University of Technology, 1999. ISSN 1402-1544 / ISRN LTU-DT–99/31–SE / NR 1999:31.

[66] P. Parnes. Secure Inclusion of Phones into Online E-Meetings. In *6th IFIP/IEEE International Conference on Management of Multimedia Networks and Services (MMNS)*, 2003.

[67] P. Parnes, K. Synnes, and D. Schefström. Lightweight application level multicast tunneling using mTunnel. *Journal of Computer Communication*, 1998.

[68] P. Parnes, K. Synnes, and D. Schefström. mStar: Enabling collaborative applications on the Internet. *Internet Computing*, 4, 2000.

[69] R. Parviainen. Multicast Interactive Radio. In *Proceedings of the Practical Application of Java*, pages 137–153, 1999.

[70] R. Parviainen and P. Parnes. Mobile Instant Messaging. In *Proceedings of the 10th International Conference on Telecommunications ICT'2003*, 2003.

[71] R. Parviainen and P. Parnes. A Web Based History tool for Multicast e-Meeting Sessions. In *Proceedings of the IEEE International Conference on Multimedia and Expo (ICME'2004)*, 2004.

[72] R. Parviainen and P. Parnes. The MIM Web Gateway to IP Multicast E-Meetings. In *Proceedings of the SPIE/ACM Multimedia Computing and Networking Conference (MMCN'04)*, 2004.

* <URL:http://www.cdt.luth.se/~peppar/docs/rtp_srm/>

[73] E. Paulos. Connexus: a communal interface. In *Proceedings of the 2003 conference on Designing for user experiences*, 2003.

[74] M. Perry, K. O'Hara, A. Sellen, B. Brown, and R. Harper. Dealing with Mobility: Understanding Access Anytime, Anywhere. *ACM Transactions on Computer-Human Interaction*, 8, 2001.

[75] C. Phanouriou. *UIML: A Device-Independent User Interface Markup Language*. PhD thesis, Virginia Tech, 2000.

[76] J. Postel. User Datagram Protocol, 1980. IETF RFC768.

[77] D. Ragget. HTML 3.2 Reference Specification, 1997. W3C Recommendation.

[78] D. Ragget, A.L. Hors, and I. Jacobs. HTML 4.01 Specification, 1999. W3C Recommendation.

[79] R.Grimm, J.Davis, B.Hendrickson, E.Lemar, T.Anderson, B.Bershad, G.Borriello, S.Gribble, and D.Wetherall. Systems Directions for Pervasive Computing. In *Proceedings of the 8th Workshop on Hot Topics in Operating Systems (HotOS-VIII)*, 2001.

[80] B.J. Rhodes. WIMP interface considered fatal. In *IEEE VRAIS'98: Workshop on Interfaces for Wearable Computers*, 1998.

[81] T. Richardson, Q. Stafford-Fraser, K.R. Wood, and A. Hopper. Virtual Network Computing. *IEEE Internet Computing*, 2(1), 1998.

[82] N. Roussel. Mediascape: A Web-based media space. *IEEE MultiMedia*, 6(2), 1999.

[83] N. Roussel. Experiences in the design of the well, a group communication device for teleconviviality. In *Proceedings of the tenth ACM international conference on Multimedia*, 2002.

[84] L.A. Rowe and R. Jain. ACM SIGMM Retreat Report on Future Directions in Multimedia Research, 2004.

[85] A. Schmidt, H.W. Gellersen, M. Beigl, and O. Thate. Developing user interfaces for wearable computers: Don't stop to point and click. In *International Workshop on Interactive Applications of Mobile Computing (IMC2000)*, 2000.

[86] H. Schulzrinne. RTP profile for audio and video conferences with minimal control, 1996. IETF RFC1890.

[87] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. RTP: A transport protocol for real-time applications, 1996. IETF RFC1889.

[88] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. RTP: A transport protocol for real-time applications, 2003. IETF RFC3550.

[89] H. Schulzrinne and J. Rosenberg. An RTP payload format for generic forward error correction, 1999. IETF RFC2733.

[90] J. Schwenk, T. Martin, and R. Schaffelhofer. Tree-based multicast key agreement. In *Communications and Multimedia Security 01*, 2001.

[91] Jane Siegel, Robert E. Kraut, Bonnie E. John, and Kathleen M. Carley. An empirical study of collaborative wearable computer systems. In *Conference companion on Human factors in computing systems*, 1995.

[92] M. Steiner, G. Tsudik, and M. Waidner. Key agreement in dynamic peer groups. *IEEE Transactions on Parallel and Distributed Systems*, 11(8), 2000.

[93] G.J. Sullivan and T. Wiegand. Video compression – from concepts to the h.264/avc standard. *Proceedings of the IEEE*, 93(1), 2005.

[94] Connected, Limited Device Configuration Specification 1.0a, 2000. JSR-000030, Sun Microsystems, Inc.

[95] Sun Microsystems. Mobile Information Device Profile Final Specification 1.0a, 2000. JSR-000037, Sun Microsystems, Inc.

[96] Java^{TM} 2 Platform: Micro Edition, A White Paper. Sun Microsystems, Inc.

[97] The JavaSpaces Service Specification, v1.2.1, 2002. Sun Microsystems, Inc.

[98] F. Tang, C. Aimone, J. Fung, A. Marjan, and S. Mann. Seeing eye to eye: a shared mediated reality using eyetap devices and the videoorbits gyroscopic head tracker. In *Proceedings of the International Symposium on Mixed and Augmented Reality (ISMAR2002)*, 2002.

[99] T. Turletti and J. Bolot. Issues with multicast video distribution in heterogeneous packet networks. In *Proceedings of the Sixth International Workshop on Packet Video*, 1994.

[100] H.J. Wang, B. Raman, C. Chuah, R. Biswas, R. Gummadi, B. Hohlt, X. Hong, E. Kiciman, Z. Mao, J.S. Shih, L. Subraimanian, B.Y. Zhno, A.D. Joseph, and R.H. Katz. ICEBERG: An Internet-core Network Architecture for Integrated Communications. *IEEE Personal Communications*, 2000.