TRANSPARENT TELETEACHING

Peter Ziewer & Helmut Seidl
Department of Computer Science
University of Trier, GERMANY
ziewer@uni-trier.de, seidl@uni-trier.de

Abstract

Today's teleteaching software restricts teaching in the way that it only supports a specific presentation application and a single platform. This is a major drawback and a reason why teleteaching is not that widespread as it could be. Therefore we have created a teleteaching environment, called TeleTeachingTool, which enables lecture recording and transmission to be done transparently to teaching. Our tool allows teachers to use arbitrary software including their preferred presentation tool running on their favorite platform. It combines desktop recording with real-time audio and video streams to a recordable and online presentable lecture. In this paper we will discuss the requirements of teleteaching, describe our teleteaching environment and compare it with other teaching software.

Keywords

lecture recording, screen recording, distance learning, Virtual Network Computing, multimedia, real-time

Introduction

Teleteaching, in our sense, is transmitting local lectures to online students or to an audience in lecture halls of collaborating universities. Additionally, the lectures are to be recorded. The environment needed to present a lecture in this teleteaching scenario certainly affects and restricts the teaching person. In general, teachers have developed their own style for presenting lectures. They are not willing to change the software for creating and presenting their lectures or the platform they are using, just because the teleteaching software only supports a specific platform and/or application. Some teleteaching software like Authoring on the Fly (AOF) (Ottmann & Lauer, 2002) even requires to create the presentations with its own presentation software, whose specific operation is to be learned by the teacher. Alternatively, existing slides must be converted before teleteaching can be done. This is a major drawback and certainly one reason why teleteaching is not that widespread as it could be. Our teleteaching environment, called TeleTeachingTool (TTT), introduced in this paper allows teachers to use their preferred presentation tool running on their favorite platform. Teachers may freely use arbitrary software during a lecture in addition to their presentation. The only restriction is to perform the visual part of the lecture completely on screen. In fact, for many teachers this is not a restriction, because notebook presentations shown to the audience using a beamer is a common way of teaching nowadays. This approach even admits doing some additional (online) explanations using a whiteboard, since there are electronic whiteboards with electronic pens available and more and more affordable. Accordingly, our environment supports teleteaching where the teacher does his/her computer based lectures as he/she did before. Broadcast and recording is completely transparent to teaching. Clearly somebody must manage the requirements and operate the software, but it is important that this is hidden from the teacher, who can create and present his/her presentation like he/she would do in a non-teleteaching environment. We expect the availability of one technical team that takes care for all teleteaching lectures of a whole university. If a teacher decides to do some lecture in teleteaching style, he/she should be able to do so in an easy way with as little preparation as possible and without any instructions, just by doing his/her lecture like he/she always has done.

In order to achieve the goal of creating such a teleteaching environment, where broadcast and recording of lectures is done transparent to the teacher, we use Virtual Network Computing (VNC) (Richardson, Stafford-Fraser, Wood & Hopper, 1998; AT&T Laboratories Cambridge, 2002) which is a remote display system. VNC allows to view and control a computing desktop environment not only on the machine where it is running, but from anywhere on the Internet and from a wide variety of machine architectures. For a functional teleteaching environment audio is obviously mandatory. You can take part in an online lecture if you receive the presentation (slides) and listen to the speaker. A video can be seen as a nice add-on. Because of todays technology only a low quality video of the teacher can be transmitted via ordinary network. We extended the VNC system by adding audio and video support using Java Media Framework (JMF) (Sun Microsystems, Inc., 2002), which offers the capability of capturing, playing, recording and transmitting audio and video with industry standard formats. Furthermore, we integrated functionality of transmitting VNC sessions to student's machines and recording them for later playback. Although VNC is able of sharing sessions, which means that multiple users have a view to (and control over) a single desktop, this feature is not suitable for a high number of simultaneous connections to students' machines, because there is one connection for each client. Therefore we extended and adapted the VNC protocol and software to be able to use the advantages of multicast transmissions to build up a scalable system. Multicasting is sending data in a way which ensures that any client who is interested in receiving the information, can receive it, but only those clients who are interested will receive it. Data is sent once per line at the most In order to make multicast work, we needed to replace the underlying network protocol TCP (Transmission Control Protocol) with UDP (User Datagram Protocol) (Network Sorcery, 2002).

Now this paper starts with an introduction to VNC and explains why it is a suitable basis to create a teleteaching environment. Afterwards we will describe our environment, called TeleTeachingTool, in detail and explain what was needed to create it. The following section is concerned with the differences of the network protocol between the VNC and our environment. Then we will discuss our experiences with the environment we gathered in daily use. The ‘Related Work’ section provides a comparison to other software related to teleteaching and the final section draws conclusions and presents perspectives.

Virtual Network Computing

The VNC System

In the Virtual Network Computing (VNC) system (Richardson, Stafford-Fraser et al., 1998; AT&T Laboratories Cambridge, 2002), server machines supply an entire desktop environment that can be accessed from any Internet-connected machine using a thin software client. The technology underlying the VNC system is a simple protocol for remote access to a graphical user interface. It is called Remote Framebuffer (RFB) protocol (Richardson & Wood, 1998). Unlike other remote display protocols such as the X Window system, the RFB protocol is totally independent of operating systems, windowing systems and applications. It works at the framebuffer level. The display side of the protocol is based on a single graphics primitive: Put a rectangle of pixel data at a given x,y position. A framebuffer update represents a change from one valid framebuffer state to another. Updates can be incremental or non-incremental. There are various encoding schemes to compress the pixel data. The RFB protocol is demand-driven by the client. That is, an update is only sent by the server in response to an explicit request from the client. The input side of the RFB protocol is based on a standard workstation model of a keyboard and a multibutton pointing device like a mouse. The client sends input events to the server whenever the user pressed a key or pointer button or moves the pointing device. So all input events generated by the client are passed on to the applications running at the server side. The client only displays the framebuffer of the remotely controlled server. The RFB protocol is a thin client protocol that makes very few requirements of the client. In particular, the protocol makes clients stateless. A client can disconnect at any time and reconnect even from another machine and will find the graphical user interface in the same state as left.

VNC as suitable basis

VNC is a good choice as basis for building up a cross-platform environment, because implementations are freely available for common operating systems like, e.g., Linux, Windows, Macintosh and Solaris. As a client even a cross-platform Java implementation exists, which is the basis of our implementation. Another advantage of VNC is its lossless compression of framebuffers and thus also desktops with presentations. Most image and video compression methods are lossy to achieve smaller file size or lower bandwidth. A low quality video transmission of the speaking teacher is no problem. We recognize who is talking, his/her gestures and what he/she is doing even if some details are missing. But for the content of a slide a lossy compression can result in illegible text or indistinct sketches and tables (see figure 1). If the degree of loss is lower and the slides are only blurred instead of being unreadable, it is at least hard to follow the lectures: Reading low quality slides over a period of 90 minutes is awkward and tiring (even if the lecture is interesting).

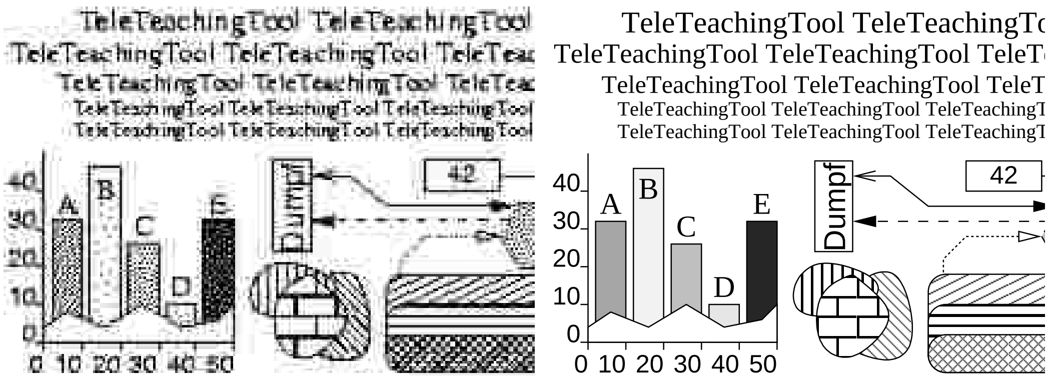

Figure 1: Lossy vs. lossless compression

The TeleTeachingTool Environment

In the TeleTeachingTool (TTT) environment we have a VNC server running the teacher's desktop. On the desktop, the teacher runs the needed applications like presentation software, editors or animations for the lecture. The next element in our environment is the TTT server, which is responsible for recording sessions/lectures and for delivering desktop, audio and video data to TTT clients. Furthermore, we have several TTT clients receiving these data and displaying them on students' machines. Alternatively, clients can also present earlier recorded sessions/lectures. Figure 2 displays an overview of our environment. In the online case, the client application is an extended VNC viewer with the ability to receive data via UDP instead of TCP (like normal vnc viewers do). The TTT client also can receive and play RTP (Real-Time Protocol) (Network Sorcery, Inc., 2002) audio/video streams. In the offline mode, our client reads the recorded session from local files. In this case, a control panel offers various vcr-like functions (play, pause, rewind, skip, ...) to let the user decide which part of the lecture he/she wants to see. This is helpful for learning. A student can repeatedly watch parts of the lecture until he/she understands the subject or skip parts he/she doesn't want to see. The online case obviously cannot offer the vcr-like controls, because the lecture is transmitted and displayed to all connected students in real-time. Transmissions are to be done in real-time to allow communication with online students and collaborating universities. Streaming servers are not applicable to do so, because of their delays caused by buffering techniques. The most popular streaming server, the RealPresenter (Real Networks, Inc.), for example has a delay of 30 seconds, roundtrip would even be 60 seconds. No online communication can by done if you have to wait at least one minute to get an answer. Via the Real-Time Protocol (RTP) it is possible to integrate media streams from various locations with only a short delay (approx 1-2 seconds) similar to that of satellite phones. These protocols are also commonly used by various conferencing tools like RAT or VIC (Clark, 1998; University College London [UCL], 2002).

Online Lectures

The teacher's desktop can run on any standard VNC server. It offers the desktop environment which is familiar to the teacher. In our installation, we use a VNC server running on the teacher's machine in his office. This has the advantage that the teacher can run all software he/she uses in his/her daily business without transporting a notebook to the lecture hall. In the lecture hall he/she can use any computer running a VNC client. Because of the stateless client design of VNC, the state of his/her graphical user environment will be the same when he/she disconnects and reconnects later. This offers the ability to prepare a lecture and start the presentation software even a day (or more) before the lecture is done. In order to do the presentation the teacher just starts his/her presentation using a fullscreen VNC viewer, which lets him/her remotely control his/her machine. The TTT server is connected to the teacher's VNC server just as another (shared) VNC client. But one, however, which does not send any input events, as the teacher should have exclusive access to his/her presentation. The VNC server sends framebuffer updates to both, the teacher's VNC client and the TTT server. So both see the same desktop. The TTT server also grabs audio supplied by the teacher's and/or audience’s microphone(s) and the signal(s) from

(a) video camera(s) and combines audio, video and desktop to a complete lecture. Furthermore, the TTT server listens for connecting TTT clients. A short initialization is done via TCP. It consists of the initialization of the RFB protocol and the multicast addresses for the lecture transmission. Now the TCP connection ends and the TTT client joins the multicast groups to receive and display the desktop, audio and video data (UDP transmissions). The online students are now part of the audience. (Some technical details on the transmission are provided in the next section.)

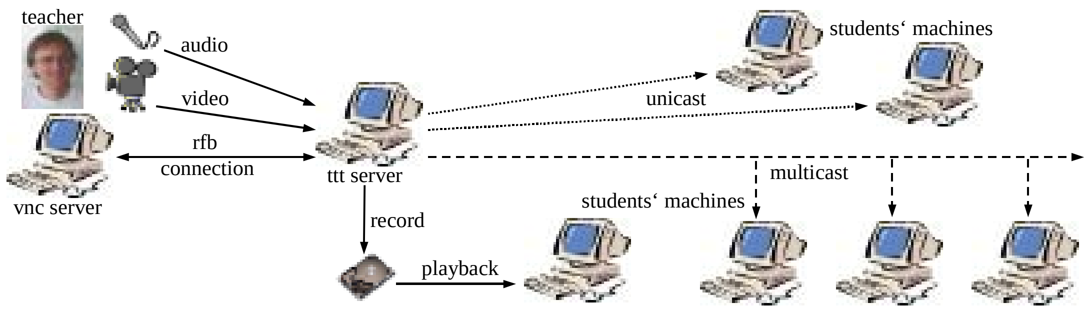

Figure 2: TeleTeachingTool environment overview

Recording and Playback

The TTT server does the session/lecture recording as well. All RFB messages received from the teacher's VNC server are written to a file. Additionally, a timestamp is added to each message to determine the delay since the beginning of the recording and the receipt of the message. For playback the messages are read sequentially from the file and are delayed according to their timestamps. There are several works related to the topic of recording rfb messages (Li & Hopper, 1998; Li, Spiteri, Bates & Hopper, 2000; Li, Stafford-Fraser, & Hopper, 1999). But for our purpose the display of the desktop recording must be also synchronized to the audio/video stream(s). Therefore the timestamp of each message is compared to the actual time of the audio/video stream(s) and delayed until both times match. Now the message can be displayed synchronously. In order to improve performance of jumping around during playback, the size of each message is also added beside the timestamps. So it is possible to skip messages without parsing them. A file pointer to the previous message, as introduced in (Li & Hopper, 1998), is not needed, because the read size information can be cached and so previous messages can be easily located within the file. In order to improve performance our player copies all messages from the file to memory anyway and so all messages can be easily accessed.

The controls of the TeleTeachingTool (in playback mode) allow the user to jump to any point within the recording to offer him/her more flexibility while learning. JMF offers methods that start playback of audio and video by giving a timestamp where to start. The VNC design of incremental framebuffer updates is hardly designed for this purpose: Any update message since the beginning of the recording can be part of the framebuffer for a given timestamp. But the straight forward solution of calculating the correct framebuffer by combining all updates to the specified timestamp, which is easy to implement, is not efficient. For example, if you jump to minute 70 of a lecture, you have to go through approx 10 Mbyte of framebuffer update messages, but only 100 kbyte of this data will make up your final framebuffer. A second possibility is to go backwards and see which updates are needed to cover a full framebuffer and only consider these messages. The older the message, the smaller is the probability of a contribution. This should be more efficient, but is not that easy to implement. Therefore we have chosen another solution. We add some non-incremental updates to our recording, which cover a full framebuffer at least every two minutes. Thus, it suffices to take two minutes of a recorded VNC session into account for calculating the needed complete framebuffer. From this startpoint we can continue with sequential playback. Also, the non-incremental updates are used to deal with packet loss during transmission as discussed in the next section.

Lecture Transmission

Using Multicast: Benefits and Restrictions

The TeleTeachingTool server combines the stream of framebuffer updates, receiving from the teacher's VNC server, and the audio/video streams from its capture devices to an online lecture and delivers it to the students in real-time. This means that the same data has to be transmitted to a lot of clients. It is an one-to-many connection and therefore the typical multicast situation. Using multicast makes the TeleTeachingTool a scalable system for transmitting online lectures, but it also gives us some restrictions we have to deal with. Multicast is based upon UDP, which is a connectionless transport-layer protocol. Data is packed into UDP packets and a best effort transmission is done. There is no reliability and the maximum packet size is restricted. The RFB protocol of VNC, however, needs a reliable streaming data transfer. It uses two streams, one transfers the client-to-server messages, while the other deals with the opposite direction. The sender feeds the stream and the receiver consumes bytes from the stream as needed to parse messages. The size of a message is not known (except for fixed size messages), because messages are queued in a stream and therefore can be transferred without information about their sizes. The VNC server writes a framebuffer update message while encoding is in progress. There is no need to compute the total message before the transmission can start. The receiving VNC client also starts displaying the update while decoding it, although it has not received the full message (maybe the server is still encoding!). In order to do a multicast transmission the RFB messages have to be packed into UDP packets. This means that each message has to be totally computed before the transfer process can start. A solution to this problem is buffering messages before transmitting them.

Splitting huge messages

Additionally it has to be taken into account that the size of each UDP packet is restricted. Most of the RFB messages fit into one UDP packet, but a framebuffer update can be too huge and will result in truncated messages if packed into UDP. The maximum size depends on system properties and varies between platforms, but 64 kbytes should be supported by all clients. The size restriction can be resolved by splitting huge framebuffer updates into smaller ones. Because of the possibility of packet loss and non-sequential transmission, each of the pieces of the origin update must be a valid RFB message. Therefore a header is generated for each part and the data of the framebuffer update message has to be adapted. All updates are built up of one or more rectangles. If an update message contains multiple rectangles it can be split into updates containing only a part of these rectangles. If a single rectangle is too huge to fit into an UDP packet, it must be split into smaller rectangles, which can be packed and transmitted. The hextile encoding (Richardson & Wood, 1998) allows an easy way to do so. Hextile encoded rectangles are divided up into tiles of 16x16 pixels, which makes them easy to split up without completely decoding and encoding the data again. The already encoded data can be used with little adaption needed. We take the amount of rows with a height of 16 pixels which fits into one UDP packet. By taking a complete row the result will always be a rectangle with the same width as the original one, but with lower height. If a single row of hextiles is still too huge it can be further partitioned into a lower number of hextiles. Framebuffer update messages containing only a single hextile always fit into one UDP packet. The colors need not be specified for each tile. In general they are specified for the first tile and for later tiles only if the colors change. The protocol definition says that the first non-raw tile must specify the background color. If we split an existing update message, this condition can only be guaranteed for the first of the newly created update messages, which contain the first part of the original message. For all following splits the needed color informations have to be gathered and added to the first tile of the new rectangle (part). Color informations can be easily determined while parsing the former rectangle (part), which has to be done anyway to calculate the size of the message. Figure 3 displays an example. The lower part of the figure displays the newly generated framebuffer update messages, created by splitting the origin update message (displayed at the top of the figure) and adding a new header and an additional color tag to each new message. By now, there is no repacking algorithm for other encoding schemes, but it is always possible to compute the raw data represented by the encoded rectangle, split the rectangle and encode this in any wanted encoding. Anyway, there is no need for the TeleTeachingTool to support arbitrary RFB encodings. In difference to the VNC environment, where the client selects the encoding and is individually supplied, the TTT server specifies the encoding to be used. This is because in the TTT scenario the same data is sent to multiple clients. They all must receive the same framebuffer update messages and therefor use the same encoding.

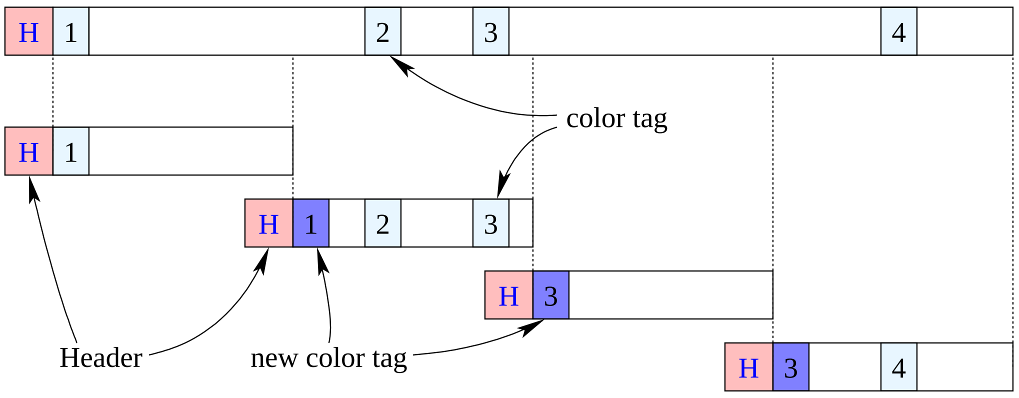

Figure 3: Splitting and adapting update messages

Unreliable Transmission

Another restriction of UDP is, that it does not offer reliable transmission. Packets can be lost and the sequence of packets can be changed. Therefore missing packets or wrong sequence numbers must be tolerated. In order to deal with unreliable UDP transmission, we first consider the effect of packet loss. The loss of RFB messages of the kind Bell, ServerCutText or SetColourMapEntries message can be ignored, because they are not essential. The SetColourMapEntries message is not supported by the X-based server anyway. So we only need to deal with loss of framebuffer update messages. If an update is not received it is not displayed. It thus can happen that some parts of the screen are not redrawn and therefore show outdated image data. That is no problem as long as most of the displayed framebuffer is correct. But the amount of outdated image data can add up, because of the use of incremental updates. In order to compensate for this effect, we use additional non-incremental updates. Non-incremental updates are large. Thus, to decrease network traffic, no full non-incremental images are transmitted, but only parts of it. The size of these parts and the frequence of their transmission determines the time within which all image data can be received as additional non-incremental update. In our implementation we send 1/12 each 10 seconds such that we have sent a full non-incremental update every 2 minutes. This does not mean, that after this delay clients have updated all image data, because non-incremental updates are also transmitted via UDP and therefore can be lost as well. But it increases the chance of viewing a completely correct framebuffer. If the rate of lost packets is very high this method is obviously not practicable, but audio and video transmission will suffer also from packet loss and result in very low quality either. In this case a useful transmission is not possible anyway. The recording of messages does not suffer from packet loss, because the TTT server is connected to the VNC server with a reliable TCP connection.

Additional Unicast Support

Currently not all networks support multicast transmissions, but we expect that this will change with the spreading of the new version 6 of the internet protocol (IPv6) (Network Sorcery, Inc., 2002). For now, our tool additionally supplies a limited amount of unicast clients. These clients receive the same data (packed in UDP packets) as the multicast clients do, but instead of being part of a multicast group, each UDP packet is sent individually to each client's IP address. In order to bound the network traffic produced by sending each packet several times, the number of clients can be limited by the TTT server and video transmission can be turned off. Also there must be a possibility to detect if unicast clients are still interested in receiving packets. In the case of multicast transfer this is done automatically by the network, but with unicast the UDP packets are transmitted even if the TTT client is no longer running. Therefore, we demand disconnecting TTT clients to send a short UDP message to the TTT server before they exit. This would work well if there weren't such things like abnormal program termination, system shutdown or network trouble. Therefore each unicast TTT client also periodically sends a short alive message to the TTT server. The server administers a list of unicast clients, where clients are inserted when they connect and removed when they send a disconnect message as described above or when their alive messages are missing for some time. This guarantees that (most of the time) only needed unicast traffic is produced.

Experiences

At the University of Trier we use the TeleTeachingTool since October 2001. We have transmitted and recorded four courses with a total of over 70 lectures by now, which are publicly available at http://TeleTeaching.uni-trier.de. There were three computer science and one media science courses. One of our courses was done in co-operation with the Saarland University. The teachers alternated in doing the lectures and the lecture was transmitted to the other university using the TeleTeachingTool. Additionally, we built up a real-time backchannel, to allow interaction between the teacher and the audience from the other university. The teacher got a video of the other lecture hall and there was a microphone for students to ask questions. The recording could be done on either side, because a TTT server can connect to any VNC server anywhere in the world and audio and video can be recorded from a local capture device or from a received RTP transmission. Also, two lectures of the media science course were done by a teacher from another university who's presentation was transmitted to a lecture hall in Trier and could be seen by our audience. The backchannel will become a part of the TeleTeachingTool in the near future instead of being an add-on solution.

Formats, File Sizes and Bandwidth Usage

The sizes of a recorded lecture of approx 90 minutes including only desktop and audio (22050Hz, mono, mpeg encoded) are between 20 and 30 Mbytes (30-40 Mbytes unpacked). So a course of around 20 lectures will fit on a single compact disc. The size of the video file depends on encoding and resolution. With Quicktime/h.263 encoding we achieve 60-100 Mbytes with a resolution of 176x144 and over 200 Mbytes with 352x288. This will result 7 or less lectures to fit onto one compact disc. Because of its low quality, the video is less important for the recording and can be omitted to reduce bandwidth or storage capacity if needed. The video can be stored to an AVI (Audio Video Interleave) format file using the same h.263 encoding, but due to a bug in JMF the synchronized playback is only possible for less than 36 minutes. Therefore we have chosen the Quicktime format. The transmissions are done using the ulaw (audio) and mjpeg (video) encodings and the RTP protocol and therefore are compatible to the conferencing tools RAT and VIC (Clark, 1998; University College London, 2002). The bandwidth used for audio transmission is approx 50-60 kbps. For the desktop transmission the bandwidth usage is hard to determine, because it varies a lot depending on the content of the desktop. Also the rate is not constant, because there are peaks, e.g. for new slides. The average over a 90 minute lecture is approx 50-60 kbps, too. Considering the peaks a value of 100 kbps is more realistic. The biggest part of bandwidth is seized by the video. Depending on encoding and quality the video transmission can use some hundreds or more than 1000 kbps. Changing the desktop encoding to ThightVNC (Kaplinsky, 2002) or zlib compression and using the gsm audio format should make it possible to limit the bandwidth usage to 128 kbps. Thus, should let our tool support low bandwidth networks like ISDN. Choosing a suitable format and encoding for video is still part of our discussion. IBM has developed a mpeg4 encoder and decoder, but due to legal issues the decoder was taken off their website for some time and re-released only recently. The encoder is not publicly available at all. Maybe an offline transcoding of the video recording will be an alternative, because it offers the ability to use better encodings, which cannot be performed in real-time or cannot be performed by JMF at all. Surely there have to be decoders within JMF for all used audio and video encodings to display the recorded sessions/lectures with a java application. For the broadcast scenario only real-time encodings are usable. Several RTP formats (e.g. audio:GSM, MPEG, .. video:H.263, MJPEG, ..) are supported by JMF. See the JMF Homepage for details (Sun Microsystems, 2002).

Problems and Solutions

Although the JavaMediaFramework offers the ability to add audio and video support to java applications in short time, we determined some problems. Their are major compatibility troubles between JMF and linux sound drivers. In particular the combination of JMF and ALSA (Advanced Linux Sound Architecture) (ALSA-Project, 2002) does not work very well by now. Another problem that impaired our work was that sending audio interfered with recording. Small periods of sound are missing and result in a shortened audio recording. Although the missing sound parts are only a view milliseconds in time, they add up to some minutes over a lecture duration. Therefore synchronization is lost to the end of the recording. We have solved this problem by using two machines, both running a ttt server. One server does the recording and the other the online transmission. We expect to get rid of these problems with newer versions of JMF, improved linux sound drivers and higher system performance in future.

Conventional Teaching vs. TeleTeaching

What are the restrictions for a teacher using the TeleTeachingTool ? We determined that a teacher can do his/her presentation as usual. He/she uses the computer to present his/her slides and does his/her talk and the TeleTeachingTool does the lecture recording transparent for the teacher. But there is a main difference between listening to a local lecture or to a recorded lecture. In a local lecture the audience is focused to the teacher and his/her motion and gestures. Surely the audience takes a look at the slides, but most of the time their eyes are aimed at the teacher. This is different for recorded or online lectures. The main window shows the presentation and the video of the teacher is only in a small additional window. Although the video data needs most of the total amount of data, its size and quality is still very poor compared to a conventional movie. In order to reduce bandwidth or safe storage capacity the video can even be turned off. Than only the desktop presentation and the audio is left. Therefore the students focus their attention onto the desktop while listening to the teacher's voice. This gets boring very quickly if the presentation only consists of slides, which are shown for a couple of minutes without change. There is no relation between the voice and a static slide as you have between the voice and the gestures of the teacher in a local lecture. Therefore some action should be integrated in the (recorded and transmitted) presentation. One thing is to do presentations with overlays, which increases the frequency of changing slides (if you count each overlay as a single slide). For example, not a complete listing is shown, but only the part up to the line the teacher talks about. If he/she continues to the next point he can change the slide and the next line of the listing is shown. His/her talk is always related to the last line shown on the screen. Another possibility to add action/motion to a presentation is using a pointer. Obviously this must be a computer based pointer and cannot be a stick or a laser pointer, which are only visible to the local audience, but not on the recorded desktop (or just in the small, low quality video). A usual mouse pointer, though, is computer-based and can be recorded. Using a huge mouse pointer will attract more attention than a standard one. Another possibility is to use features like freehand drawing or underlining, which are offered by many presentation softwares (and will be a feature of TTT in future). In our lectures, we have used a huge mouse pointer together with a LCD graphics tablet and an electronic pen allowing words to be underlined or bordered and some sketches or notes to be added while the lecture is in progress. This increases the quality of a recording considerably and even local presentations benefit from it.

Besides the teachers, the students are also users of the TeleTeachingTool. In the passed months we discovered, that many students had problems installing the needed software, although only Java, JavaMediaFramework and TeleTeachingTool is needed to receive online lectures or watch recordings. One reason was that due to a bug in Sun's installation routine of Java 1.3 for Windows, starting a Java application by doubleclick fails, if its path includes a blank. Using command line installation, on the other hand, cannot be expected of most Microsoft Windows users. Some users even tried to open Java class files with their Windows Media Players. Therefore a more user friendly installer or technologies like Java Web Start must be used in near future to address a larger number of students. Lectures could be packed with a player to provide an executable to let students easily download and start watching lectures.

Related Work

Our Requirements

The idea of lecture recording and network transmission is not new. Various other teleteaching softwares are around which, however, hardly fulfill our requirements. We wanted a platform independent solution that allows us to transmit our lectures to collaborating universities and to students, who are not part of the local audience. Furthermore the lectures should be recorded and archived for later use. A restriction concerning the choice of presentation software was not accepted. Also we wanted to be able to use multiple applications instead of only one presenter, because we wanted to show animations or edit files while the lecture is in progress. Many of today's teleteaching systems do only support Microsoft Windows and therefore are often not applicable for universities using Unix or Linux like we do.

Other Systems

In an earlier stage of research, lectures have been recorded to vhs-tapes, but the quality suffers from light reflections upon the blackboard and the restricted resolution of the vhs-system. Also distributing recordings was problematic, because of copy mechanism and prices. Even using document cameras instead of filming the blackboard hardly brought improvement. Applications like VIC, RAT, WB and SDR developed by the Networked Multimedia Research Group at University College London (Clark, 1998; UCL, 2002) are a collection of tools rather than a single solution. They are mainly conferencing applications, which work well for audio and video transmission, but offer only a restricted whiteboard functionality instead of a free choice of presentation applications.

Other lecture recording applications, like Authoring on the Fly (AOF) (Ottmann & Lauer, 2002), have an extended whiteboard, which offers more features. But all lectures must be compatible with these systems. Slides have to be adapted or newly generated with a new application, whose operation is to be learned by the teacher, who is, maybe, not willing to do so. There is no free choice of application and the use of multiple software for a single lecture is very restricted or not possible at all. Additionally, there are some compatibility issues to deal with and online transmission is not very reliable. These systems are well suited for recording slides-only presentations, if the teacher is willing to create his/her slides with the specific software. An advantage are the good editing options provided by this systems.

In todays Internet most audio and video transmissions are done with streaming servers like RealNetworks' RealPresenter or Microsoft's WindowsMediaProducer. Unfortunately these are commercial products and the transmissions of the second one can only be received on Microsoft platforms. The transmission is always peer-to-peer which makes them less scalable. No multicast is supported. Another drawback are the lossy compression codecs used for video transmission (see figure 1). A lossless transmission of slides cannot be achieved. Finally, these streaming servers do not offer real-time transmission, which makes them unusable for communication between persons in different locations.

The most promising competing application for lecture recording is Camtasia. Camtasia a commercial product for audio and screen recording offered by TechSmith (TechSmith Corporation, 2002). They have developed a lossless compression codec called TSCC (TechSmith Screen Capture Codec) which fits into industry standard, but only supports Microsoft Windows. In order to enable playback on other platforms, the recording must be encoded by other lossy video codecs. Camtasia offers a lot of different and interesting annotation methods to highlight words, areas or single windows. There is also a good editor for postprocessing of the recorded lectures. Camtasia's main attention lies on the screen recording. Although it offers a feature called "Live Output", which lets the Camtasia recorder appear as a standard video capture device, which can be used as a video source by applications like streaming media encoders, it has no built-in functionality for network transmission or integrating real-time audio or video streams supplied from other locations via network. Camtasia uses a static frequency for screen grabbing. If the frequency is low the recording looks jerky , but if it is increased it will result in larger files. In contrast, the RFB protocol of VNC is demand driven, which means that framebuffer updates are only send by the server in response to an explicit request from the client. The server can grab framebuffers with a high frequency, but framebuffer updates are only transmitted to a client if the framebuffer really has changed and if the client is ready to process the update. The result for our environment is a smooth recording (and later playback) without increasing the size of the files or the bandwidth used.

Conclusion and Future Work

Our teleteaching environment offers live transmission and recording to be done transparently to teaching. The minor restrictions put onto the presenting teacher allows to record and transmit any computer-based lecture in an easy way. Therefore our environment is a good choice for building up large libraries of recorded lectures in short time, as needed to create virtual universities like ULI-Campus (www.uli-campus.de) or VIROR (www.viror.de). In order to improve our system we have to ease the handling for the students some further. We have created a more user friendly installer to do so. Right now we are integrating a screendraw functionality, including rectangle, line and freehand-drawing, to allow application independent highlighting. Additionally, the used encodings could be improved to reduce network traffic to support low bandwidth networks. Also the limited editing options (cutting beginning and end of a recording) have to be improved to offer more flexible postprocessing. An editor enabling cutting, splitting, merging of lectures, replacing or re-recording of audio, video and desktop stream(s) and building up an index, like described by Li et al. (2000), is desirable. The backchannel add-on we have used to enable lectures with cooperating universities is the first step to create an online teaching scenario, including support for real-time communication between student(s) and teacher(s). The TeleTeachingTool and the recorded lectures are available at our website: http://TeleTeaching.Uni-Trier.de.

References

ALSA-Project. (2002). Advanced Linux Sound Architecture [Computer software and manual]. Retrieved September 17, 2002, from http://www.alsa-project.org

AT&T Laboratories Cambridge. (2002). Virtual Network Computing [Computer software and manual]. Retrieved September 17, 2002, from http://www.uk.research.att.com/vnc/

Clark, L. (1998). Introduction to multimedia conferencing and the SHRIMP tools. *The JNT Association.* Retrieved September 17, 2002, from http://www.ja.net/development/video/archive/shrimp/shrimp-userguide-intro.pdf

Kaplinsky, C.V. (2002). TightVNC: Enhanced VNC distribution [Computer software and manual]. Retrieved September 17, 2002, from http://www.tightvnc.com

Li, S.F., & Hopper, A. (1998). What you see is what I saw: Applications of stateless client systems in asynchronous CSCW [Electronic Version]. *Proceedings of the Fourth Joint Conference on Information Sciences (JCIS'98), Research Triangle Park, NC, 3,* 10-15.

Li, S.F., Spiteri, M., Bates, J., & Hopper, A. (2000). Capturing and indexing computer-based activities with Virtual Network Computing [Electronic Version]. *Proceedings of the 2000 ACM Symposium on Applied Computing, Como, Italy, 2,* 601-603.

Li, S.F., Stafford-Fraser, Q., & Hopper, A. (1999). Frame-buffer on demand: Applications of stateless client systems in web-based learning [Electronic Version]. *Proceedings of the 5th International Conference on Information Systems Analysis and Synthesis (ISAS'99), Orlando, FL, 2..*

Network Sorcery, Inc. (2002). RFC Sourcebook [Internet standards and protocols]. Retrieved September 17, 2002, from http://www.networksorcery.com/enp/default.htm

Ottmann, Th., & Lauer, T. (2002). Authoring on the Fly [Computer software and manual]. *University of Freiburg.* Retrieved September 17, 2002, from http://ad.informatik.uni-freiburg.de/aof/

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

Richardson, T., & Wood, K.R. (1998). The RFB protocol. Version 3.3. *AT&T Laboratories Cambridge.* Retrieved September 17, 2002, from http://www.uk.research.att.com/vnc/rfbproto.pdf

Sun Microsystems, Inc. (2002) JavaTM Media Framework API, Version 2.1.1 [Computer software and manual]. Retrieved September 17, 2002, from http://java.sun.com/products/java-media/jmf/

TechSmith Corporation. (2002). Camtasia [Computer software and manual]. Retrieved September 17, 2002, from http://www.techsmith.com/products/camtasia/camtasia.asp

University College London, Department of Computer Science, Network and Multimedia Research Group. (2002). Mbone conferencing applications [Computer software and manual]. Retrieved September 17, 2002, from http://www-mice.cs.ucl.ac.uk/multimedia/software/