Chapter 6

Audio Storage and Transmission

This chapter shortly describes E-Chalk's audio part. It presents the evolution and problems that led to the design decisions for the current realisation. It provides an overview of the implemented components before Chapter 7 presents a detailed explanation of the recording methods.

6.1 Evolution of E-Chalk's Audio System

The ancestor of E-Chalk's audio system is the World Wide Radio system that dates back to early 1997. During almost 10 years, the system transformed itself to adapt to many technical changes and user demands. This section summarizes the evolution of E-Chalk's audio system by presenting the concepts and main design decisions of the different versions. The history of the system helps to understand some of E-Chalk's main design decisions as presented in Chapter 3.

The history of the audio system is primarily driven by the specialization from a generic Internet audio broadcasting system to a solution for transmission and archiving of the voice of an instructor lecturing with an electronic chalkboard. Thus looking at the history of the system also provides some hints about the differences between generic Internet audio broadcasting systems and the solutions needed for the remote lecturing.

6.1.1 The World Wide Radio System

The *World Wide Radio System (WWR)* [Friedland and Lasser, 1998], developed in 1997, aimed at providing a software solution for using the Internet as a broadcasting media for traditional radio stations. The idea behind the project was that the Internet would make it easier for more people to create their own radio station without the costs of leasing a radio frequency slot. Especially transmissions that were of interest only to a small group of people would now be possible. The main scenario was that of a small community radio station (for example for a school or a university) broadcasting audio content produced by a set of volunteers. In fact, the main users of the WWR system have been the

"Offener Kanal Berlin (OKB)"[1], the "Berliner Volksbühne", as well as several groups and organizations of high-school students.

The dominating technical problem were bandwidth restrictions in the Internet in these days. Analog modems constituted the number one way to connect to the Internet. CPU power was limited, especially for real-time signal processing so that most elaborated audio compression algorithms were not able to run in real time. In the given scenario, a modem connection could not only be found on the client side, but also on the server side. The WWR system therefore consisted of three parts: The server part, the client part, and a so-called *broadcaster*. The broadcaster was a kind of proxy server that received the compressed audio stream from a server and forwarded it to different clients (or other broadcasters).

Server

The WWR server was a small program (implemented for Linux and Sun Solaris) that was able to record audio directly from the soundcard and compress it down using a self-developed DCT codec (see [Friedland and Lasser, 1998] for details). The DCT codec was able to compress an 8 bit 8 kHz μ-law [ITU-T, 1988] mono audio signal (64 kBit) down to about 13 kbit/s which was enough for a 14.400 kbit/s modem dialup connection to receive the audio stream without interruptions. The server was able to stream from the input lines of the soundcard or from prerecorded files.

Broadcaster

Every connected client requires its own stream. The actual bandwidth requirements for the server therefore increases proportional to the amount of listeners. Since the World Wide Radio system also assumed a modem connection for the server, the solution was to send the compressed feed to an Internet provider with greater bandwidth and then distribute the signal from there to the individual clients. The program responsible for this distribution was called WWR broadcaster. Broadcasters get the compressed audio stream from a regular WWR server or from another broadcaster and send them to clients or other broadcasters. The result is a tree structure similar to the one that exists in the MBONE [85][2].

The multicasting system of WWR was transparent, i. e., the user did not know that he or she actually listened to a stream which came from a broadcaster server even though he or she originally connected to another server. A server automatically forwards clients if the number of simultaneous connections exceeds a threshold. In order to facilitate transparent forwarding of clients to receive the stream from different broadcasters – even while the client was running – a simple technique was used. Instead of the clients actively connecting to the server, the clients were accepting incoming connections from any WWR server or broadcaster. So a server could close the connection to a client any time and trigger a broadcaster or another server to connect to the client again.

[1] Literally translated: "open channel Berlin". A TV and radio station where everybody can broadcast his or her own TV or radio productions

[2] The reason for not directly using MBONE was that MBONE needs special router configurations and was not very popular outside the academic domain.

WWR differentiated two types of broadcasters: active broadcasters and passive broadcasters. Active broadcaster servers are always connected to its parent server and get the audio stream, even if no client is currently connected. Passive broadcaster servers only connect to its parent server if at least one client requests audio data. The second method results in a short delay for the first user and may lead to even further delays if the parent connection cannot be established (if this happens, the broadcaster redirects the client again).

Client

The WWR client was a small program compiled for Linux, Sun OS, Solaris, FreeBSD, Windows (16 bit), Windows NT, Mac OS 9, Next Step, Irix, and other operating systems. The clients consisted of a single executable (about 30-60 kB, depending on the platform) that could be started right off the download, without the need for any installation. The work flow for receiving a radio program for the first time consisted of viewing the web site of the radio station, downloading the client software, then starting the client software, and pressing a button on the web site to make the server connect the running client software. A process considered too difficult by many users.

The client received the compressed signal via TCP from a server, decompressed it, converted it to the format capabilities of the underlying sound card, and played it back. Buffering strategies were used to guarantee uninterrupted playback. A buffer length of about seven seconds was needed in order to achieve satisfying results.

6.1.2 World Wide Radio 2

The *World Wide Radio 2 (WWR2)* project (beginning of development in 1999 together with Bernhard Frötschl) was initially meant to be a Java rebuild of the WWR system. There were several reasons for rebuilding the WWR system in Java. Of course, maintaining server, broadcaster, and client implementations for so many different platforms was very cumbersome. When using pure Java, however, programs are automatically small and portable. Most importantly, Java offered the possibility to embed the client directly into a website without the user needing to download and install client software. Embedding the client as an Applet in a web page also revealed new possibilities. Applets on the same web page are able to communicate. This made it possible to synchronize the audio stream with different applications. A feature that made WWR2 a perfect candidate to take over the audio streaming part of E-Chalk (see Chapter 5 for details). However, WWR2 was also used as a stand-alone program by different institutions, among them Uniradio Berlin [58], the project GIOVE [22], and "Berliner Gruselkabinett" [10]. An independent evaluation of the system has been performed by the *Funkschau magazine* [Manhart, 1999].

Using Java as a client platform also introduced several problems which were mainly due to the restricted access rights of Applets and Java's initial performance. For years (until the introduction of Java 1.3) Java Applets running in state-of-the-art web browsers could not playback sound with a sampling frequency higher than 8999 Hz [53]. Fortunately, this was sufficient for recording speech. The basic input format for WWR2 was again 8 kHz, 8 bit μ-law. The first idea to compress these data was to use the low bandwidth percep-

tual CODEC developed in [Friedland and Lasser, 1998]. However, the adaptive compression that was used there could not be implemented in Java for efficiency reasons. We found that it was not possible to implement even a subset of MPEG-audio encoding in these days, since any fast DCT-based approach was too slow to run in real time under Java. In the end, we implemented several codecs for different bandwidth, CPU speeds, and audio quality levels into WWR2 by modifying older compression standards. WWR2 contained a simple and fast codec, which used no compression but the Java built-in GZIP [P. Deutsch, 1996] algorithm. The resulting stream needed about 50 kbit/s. To achieve better compression, WWR2 also contained the 4 bit version of the μ-law codec, adapted from [ITU-T, 1988]. This codec, together with GZIP, compresses down to about 20 kbit/s. The sound quality was adequate. To achieve a good trade-off between sound quality, compression, and execution speed, we modified the ITU ADPCM standard [ITU-T, 1990]. The results were 4 bit, 3 bit, and 2 bit modified-ADPCM codecs that, combined with GZIP, gave an effective average compression of 30 kbit/s, 22 kbit/s, and 15 kbit/s.

The WWR2 server was also able to replay files. Programming of sequences and loops could either be done off-line via a configuration file or on-line via a telnet command-line interface. For this purpose, the WWR2 server had a built-in macro language. While the World Wide Radio system had only been able to transmit live, the WWR2 system also supported on-demand listening. In on-demand mode, no server is used, and a client reads a WWR2-encoded audio signal directly over an HTTP stream (see Chapter 5). While live transmission had been considered the most important feature in the original WWR system, in the E-Chalk system it lost more and more importance. Although E-Chalk still supports live transmissions, the main use of the system is on-demand. Along with other technical improvements, such as using a 16 kHz sampling rate, the WWR2 system specialized more and more on on-demand replay. In the end, the resulting system was called *E-Chalk Audio*, which is described in the next section.

6.1.3 The E-Chalk Audio System

In 2002, a complete rebuild of the audio system was performed (internally called *World Wide Radio 3*). The challenges of the system were quite different than a few years ago. In the meantime, analog modems had become faster and were not the only possibility to connect to the Internet from home. The Java Virtual Machine as well as processors had improved and real-time decoding of more complicated codecs did not constitute a problem anymore. Now, a wide range of commercial Internet broadcasting systems were available and codecs had long surpassed high fidelity[3]. The design decisions for the new system were therefore dominated by new aims which in turn where a result of user demands from the universities and other schools that had been using E-Chalk.

Mainly three problems seemed to demand a solution in the new E-Chalk audio system. The integratability and combinability of E-Chalk audio with other software, as well as extensibility of E-Chalk Audio itself; an infrastructure that is able to handle E-Chalk's audio format, e. g., to allow editing, automatic checking and repairing, as well as converting E-Chalk lectures and in particu-

[3]Originally specified in DIN 45500 in the early seventies, now revised into DIN EN 61035.

lar E-Chalk's audio format; and the improvement of the subjective quality of E-Chalk's audio recordings.

The interfacing possibilities of E-Chalk audio, as well as the possibility to replace E-Chalk's audio core with completely different software, is facilitated by using SOPA as the underlying architectural layer of E-Chalk. The E-Chalk Audio system consists of a set of independent nodes that are managed by SOPA. Developers can easily substitute nodes with their own versions or add more functionality by implementing additional nodes. The life-cycle and deployment of the nodes is automatically managed as described in Chapter 4.

E-Chalk's audio system features a set of tools for converting between E-Chalk Audio and different other formats. An Exymen plug-in developed by Mary Ann Brennan allows the conversion between E-Chalk Audio and the audio formats supported by the Java Media Framework [84], which include "wav" and "aiff" containers with different codecs as well as "mp3" [54]. Because Exymen plug-ins can also be used as SOPA nodes, E-Chalk lectures can also be recorded using these formats, as long as encoding works in real time and a player for synchronized playback with the chalkboard data is available. E-Chalk Audio recordings can be edited using an Exymen plug-in (see [Friedland, 2002a]).

Although many codecs could now be integrated into E-Chalk and the audio replay capabilities of the browsers had enhanced dramatically, many E-Chalk lectures were not recorded with audio. Because the audio track is a vital part of any E-Chalk lecture, we looked at the recording archives of other lecture recording systems. Interestingly enough, the archives of other projects that do not require technical operators for camera and audio recording control also contain many lectures without audio track (compare for example the lecture archives of the e-class project [74] or the archives of the Classroom Presenter [64]). Several regular E-Chalk users were interviewed informally to analyze the reasons for this problem. Often, audio was omitted because instructors did not want their voice to be archived. Their fear was that their reputation as professors could be harmed when errors, comments, or small jokes from the classroom lecture appeared in the replay. People could recite their utterances word by word. Of course, this problem cannot be solved by software other than by providing the user with the choice not to record audio.

Many of the lectures, however, do not have audio because of usage errors during the recording. Often, the lecturers just forgot to switch on the microphone or the operating system's mixer settings were not correct. Sometimes, audio recording was omitted or subsequently removed because professors and students complained about clipping or noise in the recordings. Mostly, the cause of these problems were again operational errors that were caused by the fact that the instructor had to concentrate on giving the lecture.

As described in Chapter 3, E-Chalk's philosophy does not allow overhead resulting from the use of the system. Consequently, the system must help the instructor in handling the task of recording audio properly, i. e., E-Chalk Audio had to do more than simple recording, encoding, sending, and archiving the incoming sound signal. Hence the notion of Active Recording was introduced in E-Chalk Audio. Chapter 7 is devoted to this approach.

6.2 E-Chalk's Default Audio System

As described in the previous section, the E-Chalk audio system consists of many components and is highly adaptable and configurable thanks to the SOPA system. Therefore, the audio system runs in different configurations at different locations. This chapter therefore describes the components of the E-Chalk Audio system as they are shipped with the default installation. E-Chalk's default audio system consists of a set of about 20 SOPA nodes. Half of them are devoted to the classical recording, compression, and transmission tasks and the other half constitutes the active recording system. This section describes the recording, encoding, and transmission core briefly and the next chapter describes Active Audio Recording.

6.2.1 Encoding

The incoming audiodata from the active recording system is a 16-bit mono signal with a sampling rate of 16 kHz. It is compressed down to one of the following four bandwidth 40 kbit/s, 32 kbit/s, 24 kbit/s, and 16 kbit/s (upper limits) using a variant of an ADPCM codec inherited from WWR2. ADPCM is a very simple and computationally efficient lossy compression method that gives usable results for speech [ITU-T, 1990]. ADPCM stands for Adaptive Differential Pulse Code Modulation and is a waveform-quantization method. Given a few samples of the waveform from the past, the next sample is predicted by a heuristics. The same prediction heuristics works in both the encoder and the decoder, so that only the difference between the prediction and the measured sample has to be transmitted from the encoder to the decoder to reconstruct the original signal. If the prediction is good, the variance of the prediction error should be much smaller than the signal variance, reducing the amount of bits that have to be transmitted. In practice, the transmitted difference values are quantized by defining a step-size table of a certain size. The table contains predefined difference values. The difference is then encoded by a reference into the step size table. Because the size of the table is set to a fixed size, the amount of bits that have to be transmitted per sample is also fixed. By changing the size of the table it is easily possible to control the bandwidth-quality tradeoff from lossless to unbearable lossy. To achieve further compression, especially in regions with no signal, the signal is packetized and compressed with the Java built-in ZIP algorithm [P. Deutsch and J-L. Gailly, 1996]. The E-Chalk Audio format is described in detail in Appendix D.

6.2.2 Live Transmission and Archiving

The encoded and packetized audio signal is then ready to be streamed over the Internet. E-Chalk's audio server component is a straightforward implementation that waits for a connection of a client and then streams the encoded data packet-by-packet over TCP/IP. Each connection is managed in a separate thread with a large buffer (by default 256 kB), to compensate for connection bandwidth variations or even temporary stalls. A self-developed benchmark program that simulates incoming connections in a short time period was used to test the robustness of the approach and to determine the maximum amount of manageable connections. With state-of-the-art computers, however, memory

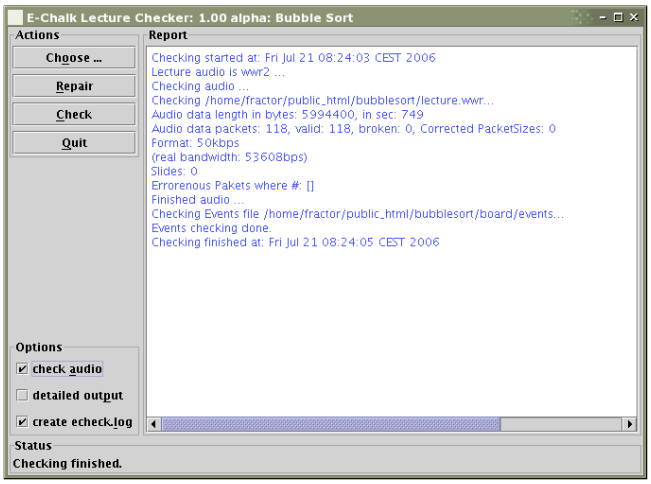

Figure 6.1: A screenshot of the E-Chalk lecture checker and repair tool. Audio recordings are scanned for broken packets and the file is repaired if possible.

and CPU power are sufficient to handle several hundreds of connections, even if the video server and the chalkboard server are running on the same machine. The server node can be used to stream any audio or video format.

During the transmission, the encoded version of the audio stream is also saved to a file along with an index file that contains a list of offsets pointing to the beginning of each packet. During on-demand replay, both files are streamed over HTTP and the index file is used by the client to accelerate random seek. For a description of the audio client refer to Section 5.2.2).

In order to simulate bandwidth variations, self-developed traffic shapers were used that randomly limit the connection speed. This is useful to determine the required buffer size at the receiving end. A large receiving buffer results in a large time-shift between sender and receiver. A small buffer might be unable to compensate for normal bandwidth variations. I found that using a client buffer size of about 3.2 seconds suffices to allow a 90-minute uninterrupted transmission of a typical chalkboard lecture combined with 40 kbit/s audio over ISDN (64 kbit/s). In the typical scenario, where a local student follows a university lecture at home, the buffer is almost only needed when an image that has been put on the board has to be transmitted. The delay between recording and playback, however, is unacceptable for bidirectional transmissions. It is therefore necessary to adjust the size of the buffer when synchronizing E-Chalk with a video conferencing system.

6.3 Tools

Numerous small helper applications are part of E-Chalk's infrastructure and allow the handling of the audio format both stand-alone and as part of a lecture with board content and video (compare also [Friedland, 2002a] and [Knipping, 2005] Chapter 6).

6.3.1 Lecture Repair Tool

Among the most important tools is the *E-Chalk lecture checker* (compare Figure 6.1). Initially, the tool was created as a reaction to problems reported by Technische Universität Berlin. One of their computers running E-Chalk in the classroom crashed regularly due to a hardware problem. The tool that is part of the E-Chalk distribution now gives users more security not to lose their work, for example due to a crash of the computer or a power shortage during the lecture (this typically happens in a laptop presentation without stationary power supply). When a computer crashes during a lecture, the last few audio packets are sometimes missing and/or the the last packet has only been written partly. Because the lecture recording has not been terminated properly, no HTML files for archived lecture replay have been saved into the output directory, so that the lecture cannot be replayed in the browser. In most cases, the classroom lecture is continued after a reboot of the machine. Often instructors then just continue the crashed E-Chalk recording. E-Chalk then appends the continued lecture to the files already written without checking if the already saved parts of the lecture haven been terminated properly (such a check would take too long). So in this case, the broken packet is in the middle of the recording (and in the case of multiple computer crashes there might even exist several broken packets). The E-Chalk lecture checker scans E-Chalk lecture recordings for corruption and optionally repairs them. The board event file is scanned line by line and corrupt events are removed. Corrupt audio packets are found by unpacking each packet in the archive. If a packet cannot be unzipped entirely and/or the entries given in the index file do not correspond to the actual offset of the packet headers the archive needs repair. Corrupt packets are replaced by correct ones containing silence, trying to maintain the time synchronization of the tracks. Of course, the lecture checker is also able to generate a new index file. Video files are repaired similarly by replacing corrupt packets with packets that contain idle frames (compare Chapter 8). However, if several entire packets have been lost during recording, the archived chalkboard, audio, and video tracks may need to be re-synchronized manually using Exymen. In the end, the lecture checker generates new HTML pages and puts a current Java replay client into the lecture directory.

6.3.2 Audio Format Converter

The *WWR2 to WWR3 converter* works as wizard and as command-line tool. The command line tool is used internally by the E-Chalk Startup Wizard if a lecture in the old format is to be continued using a newer E-Chalk version. In the beginning, the purpose of the converter consisted exclusively of this syntax transformation from one format to the other. However, when active recording (Chapter 7 explains Active Recording in detail) was introduced in E-Chalk, users asked for the possibility to improve old E-Chalk recordings using the noise and humming fingerprints and/or the equalizer settings created with the newer E-Chalk versions. Of course, this is only possible when the old lectures and the new fingerprints have been produced with the same hardware setup, i. e., if only the E-Chalk version has changed. By using the GUI wizard version of the WWR2 to WWR3 converter, it is possible to upgrade the syntax while applying a recently recorded fingerprint.

6.3.3 E-Chalk Broadcaster

E-Chalk audio also features a transparent multicasting mechanism similar to the one explained in Section 6.1.1. The mechanism also uses a simple broadcaster tool that works similar to the one already introduced in WWR(1). The difference is that the transparent forwarding of clients works by an event mechanism that is integrated into the E-Chalk audio format (compare Appendix D). When a server or broadcaster wants to forward a connected client to another server or broadcaster (for example, if the maximum number of connections is exceeded), it sends the URL pointing to the web page of the new server or broadcaster encoded in the WWR stream. The client Applet then replaces its own host web page by opening the URL. Because the board client expects to receive the entire event file at the start of transmission and the rest of the events line-by-line, the E-Chalk events are broadcasted by a small Unix shell script [4]. Since most of the live transmissions are performed from within the network of a university, the broadcaster application is used very rarely[5]. For this reason, a broadcaster for the video stream has not been implemented yet, although it would easily be possible.[6]

Several other tools, for example a recording monitor, an automatic mixer control, and a noise fingerprint recorder, have been built as part of the active recording component to enhance the recording quality. These tools are described in detail in the following chapter.

[4] A Java version of a broadcaster for E-Chalk board content was created by Christian Burger from elfzehn GbR.

[5] The broadcaster application was only used for special events at locations where only a modem connection was available, for example an arts event where an electronic chalkboard was used as drawing device in September 2004.

[6] The audio broadcaster application cannot be used because the video broadcaster would have to generate the initial header for each connecting client (compare Appendix F).