ENHANCED INTERACTION FOR STREAMING MEDIA

Wolfgang Hürst, Tobias Lauer
Institute of Computer Science, University of Freiburg, D-79098 Freiburg, Germany

Rainer Müller
imc AG, Office Freiburg, Georges-Köhler-Allee 106, D-79110 Freiburg, Germany

Keywords: Media delivery, multimedia streaming, lecture recording, interfaces, interaction, navigation.

Abstract: Streaming is a popular and efficient way of web-based on-demand multimedia delivery. However, flexible methods of interaction and navigation, as required, for example, in learning applications, are very restricted with streamed contents. Using the example of recorded lectures, we point out the importance of such advanced interaction which is not possible with purely streamed media. A new delivery method based on a combination of streaming and download is proposed which can be realized with Rich Internet Applications. It combines the advantages of streaming delivery with navigational and interactive features that are usually known only from locally available media.

1 INTRODUCTION

Generally, multimedia content is delivered to the users either via streaming or by download and local replay of the respective files. The decision which of these two approaches for media delivery is offered by the providers usually depends on the application and the data. However, there are situations where it is not obvious which approach should be used. Examples are teaching materials such as recorded lectures. Some institutions offer such recordings online as Webcasts where learners can access and review them via streaming. This has the advantage of giving the providers more control over the contents, and it does not require from the users to download huge amounts of data prior to viewing. However, many universities also allow students to download their lecture recordings as a whole for local replay (exclusively or in addition to streaming). Users sometimes prefer download over replay because it makes them less dependent on the provided service and online connection after a file has been downloaded once. In addition, manipulation of replay such as fast forward at different speeds, flexible navigation such as real-time scrolling along any direction of the time line, etc. are often cumbersome or even impossible to do with streamed data. However, such possibilities for an advanced navigation and interaction are particular important in learning applications.

In this paper, the delivery and access of recorded lectures is used as an example scenario for multimedia applications that require a high degree of user interaction. First, we describe the necessary basics for lecture recording and delivery. We identify enhanced interaction functionality as a key requirement in such a scenario (*Section 2*). Then, we present a synchronization model for lecture replay whose standard implementation offers this required interaction but lacks the possibility for streaming (*Section 3.1*). Motivated by this issue, we present a new method for the delivery of multimedia data which is based on a combination of streaming and download (*Section 3.2*). The proposed solution combines the advantages of streaming with interactive and navigational features that are usually only known from media replayed locally after they have been downloaded before.

2 DOWNLOAD VS STREAMING

Automatic lecture recording and delivery via Web casting has become a common trend at many universities: Live lectures and presentations are

Hürst W., Lauer T. and Müller R. (2006).
ENHANCED INTERACTION FOR STREAMING MEDIA.
In *Proceedings of the International Conference on Signal Processing and Multimedia Applications*, pages 216-223
DOI: 10.5220/0001570502160223
Copyright © SciTePress

captured and the recordings are automatically post-processed and published on the Web (Abowd 1999, Brusilovsky 2000). Initially originating from educational institutions, this approach is gaining increasing popularity in the industries as well. Users access and use these documents, for example, to review content, to look up specific information, or even as a substitute of the corresponding live event (Brotherton and Abowd 2004, Lauer et al. 2004).

One important issue for lecture Web casting is the final delivery of the documents, i.e. the question whether the respective data is sent to the viewers online via streaming-servers or if it has to be downloaded as a whole to the user's local machine. When comparing local replay vs. streaming, no absolute answer can be given as to which approach is generally preferable. For example, when asked about the importance of streaming vs. local availability (after downloading), the participating students of the study in Lauer et al. (2004) rated local availability much higher on a scale from 1 to 5 (Mean = 1.31, SD = 0.50) than streaming (Mean = 3.48, SD = 1.14). These subjective ratings were backed up by the server statistics: In all cases where both documents for local download and identical versions of the lecture in some streaming format were available, the overwhelming majority accessed (i.e. downloaded) the former one. However, the situation is different for corporate learners, who often do not have authorization for large downloads at all at their workstations. Since those computers are usually connected permanently to the corporate intranet, streaming is considered a better way of delivering the contents. This is especially true for companies with preconfigured client configurations, where the widespread "de-facto standards" (such as RealMedia (2006) or Windows Media (2006)) are often the only accepted formats, because no additional installation of software (such as a proprietary multimedia player) is permitted.

In addition, there are general arguments in favor of streaming, not only in corporations but also for the distribution of online lectures at universities. For the learners, having to download large files (a 45-minute lecture including video may well amount to 500 MB or more) results in enormous preload times and may quickly fill up the local hard drives. Moreover, streaming servers provide solutions to handle large numbers of concurrent users and to adjust to changing bandwidth conditions. In addition, streaming technologies offer a certain degree of content protection. Since the streamed data are not stored permanently on the end user's system, the danger of unauthorized copies and their distribution is reduced.

On the other hand, local download and replay of the files has some significant advantages over streaming approaches. One of the reasons why students often prefer the former over the latter one is the pure desire to possess the files. Students often do not have a permanent high speed connection to the internet. Therefore, a long but one-time download is often accepted for the sake of complete independence of any network connection afterwards. The second and probably most important advantage of local replay is the ability to provide advanced features for interaction and navigation. In a survey (Lauer et al. 2004), we evaluated the importance of different interaction features which are illustrated in Figure 1. The results confirmed the assumption that advanced browsing and navigation functionality is essential when learning with lecture recordings. This observation is consistent with other studies, for example the one of Li et al. (2000), which evaluated different kinds of browsing approaches for digital video recordings (including but not limited to lecture recordings). Such subjective user ratings are also confirmed by studies observing the actual usage of different features in real-world situations. For example, Zupancic and Horz (2002) present a log file analysis indicating that users of recorded lectures make intensive use of such advanced browsing and navigation functionality when reviewing these documents.

It is obvious that the efficient use of the mechanisms presented in Figure 1 crucially depends on their responsiveness, i.e. the speed at which the resulting jumps in the document can be carried out. Even simple interactions, such as a slide-based navigation through a document, often result in a significant time delay when contents are streamed, thus disrupting the learning process and limiting the interaction with the data. Certain features, such as *random visible scrolling* (cf. (a) in Fig. 1) which was found to be very important according to Lauer et al. (2004), cannot be realized with streaming at all, because the ability to navigate along the timeline at any speed directly conflicts with the basic concept of streaming, as this feature requires real-time random access to any position within a document.

3 MODELS FOR DELIVERY

Based on the discussion in the previous section, we see ourselves confronted with two contradicting

Text search. Full-text search on the slides.

Enables direct access to particular positions in the document

Interaction features for slide-based navigation and browsing

b **Thumbnails stream.** Thumbnails of the slides with temporal information about their appearance. The thumbnail of the current slide is highlighted during replay. (Alternative representation: List of contents with slide titles)

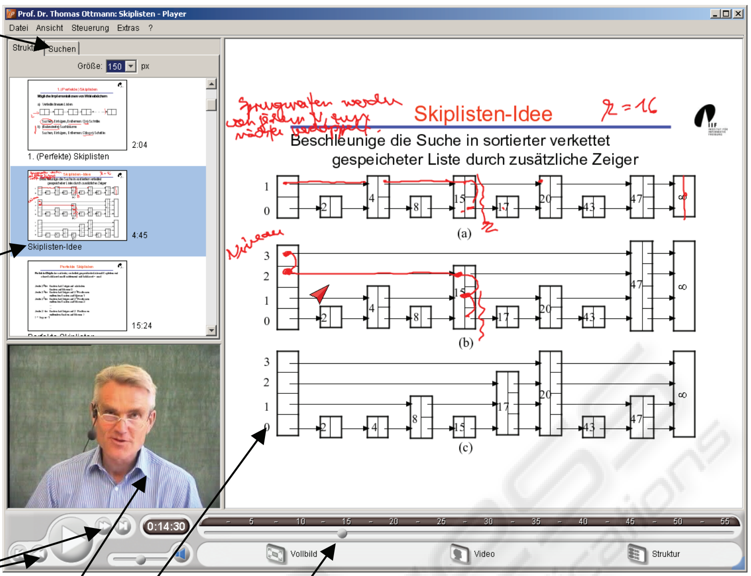
Clicking on a particular thumbnail forces replay to jump to the corresponding document position and updates the synchronized replay of the other media streams.

Slide backward/forward buttons.

Forward and backward navigation through the slides in their temporal order in the document

Video stream. Video capture of the instructor (talking head shot)

Presentation graphics stream. Slides, graphics, annotations, etc. presented to the audience



Prof. Dr. Thomas Ottmann: Skiplisten - Player
Datei Ansicht Steuerung Extras ?
Struktur Suchen
Größe: 150 px
2:04
1. (Perfekte) Skiplisten
4:45
Skiplisten-Idee
15:24
Skiplisten-Idee
Beschleunige die Suche in sortierter verkettet gespeicherter Liste durch zusätzliche Zeiger
(a)
(b)
(c)
0:14:30
Vollbild
Video
Struktur

Interaction features for flexible skimming at different speeds and granularities

a **Random visible scrolling.** Time-based slider in combination with real-time random access to the data streams. Moving the slider along the timeline at any speed and in either direction results in an immediate update of all media streams.

Enables flexible and interactive navigation in the file in order to skim its visual content at different granularity levels, e.g. to get a quick overview of a document, to quickly skip parts of minor interest, to go back a few sentences (or even just a few words) in order to reset replay to a position of particular interest, etc.

Figure 1: Different interaction functionalities of a media player replaying a recorded lecture.

demands: On the one hand, there are good reasons why the recordings should be kept centrally and delivered to the users with streaming technologies. On the other hand, advanced interaction, which is a key requirement in order to achieve high acceptance and successful learning, usually requires local availability of the data. In the following, we first describe our established replay model which offers this requested interactivity but whose previous implementation relies on local replay (*Section 3.1*). Then we introduce a new realization of this concept which makes use of the underlying replay and synchronization model in a distributed scenario, thus enabling streaming as well as interactive navigation (*Section 3.2*)

3.1 Generic Model for Local Replay

Multi-stream recording and delivery. Most approaches for automatic lecture recording use a *multi-stream capturing* approach, i.e. they capture each media stream separately and later synchronize them during replay or in an additional post-processing step. This way, each stream can be provided in the best possible quality. In addition, the final delivery can be adapted to the available resources (bandwidth, storage, etc.) depending on the relevance of the respective media channel. For example, the less critical, but very data intensive video of the instructor is often downsized, replaced with a few snapshots, or even removed completely in environments with limited resources. Often, only the critical media streams of a (tele-)presentation (Gemmell and Bell 1997), i.e. the audio with the lecturer's narration and the presentation graphics (slides, annotations, etc.) are delivered. On the other hand, additional but less data intensive streams containing meta-information about the content and structure may also be synchronized with the recordings (cf. the thumbnails stream depicted in (b) in Fig. 1). As discussed in Section 2, such meta-information can be very important for navigation and interaction.

Synchronized replay. In order to realize a reliable *synchronized replay* of such an arbitrary number of streams we need a generic synchronization model. In addition, we have to support advanced navigation functionality in order to provide high *interactivity* to the users, as specified above. For this reason, we introduced a multi-stream synchronization model in Hürst and Müller (1999). In the following, we summarize this model insofar as it is important for the understanding of the remainder of this paper. Readers interested in the technical details are referred to Hürst and Müller (1999).

The basis for our *synchronization model* is an open intermediate format where the different media streams are kept separately, only implicitly coupled by the time. The streams are structured in a flat hierarchy oriented at one designated audio stream. In our case, the audio stream covers the presenter's narration. This flexible structure of the individual documents (i.e. audio stream, presentation graphics, and any other media channels that should be integrated for the final replay) is mapped to a generic replay architecture with separate processes, threads, or instances (*slaves*), each replaying one particular media stream. They are coupled with the particular process (*master*) replaying the designated audio stream. The master sends out timestamps to the slaves in order to guarantee synchronized replay. This allows for a tight temporal synchronization and quality assurance for the most critical media type. The generic replay architecture does not restrict the flexibility of the document structure and can be realized in nearly all target technologies and formats (including standard formats such as Flash, QuickTime, or SMIL).

Navigation in the document is supported through backchannels from the slaves to the process replaying the master stream. For example, if a user navigates through the slides by clicking on an icon in the thumbnail stream (cf. (b) in Fig. 1), a timestamp is sent over its backchannel to the master which synchronizes its own replay and guarantees an alignment of the other slaves. When a user is dragging a slider along the timeline, basically the same activity takes place but at a much finer granularity: Timestamps indicating the current position of the slider thumb on the timeline are continuously sent to the master stream which guarantees the synchronization of all other slaves for visual media and thus realizes the random visible scrolling feature (cf. (a) in Fig. 1). As with common media players, audio replay is paused while a user is dragging the slider thumb but starts replay at the corresponding position as soon as the slider is released. The content of raster-based visual streams, such as videos, is updated immediately if the respective data is encoded in a format which supports real-time random access to each frame. If the respective content is not considered important for browsing, the video stream can alternatively be paused during scrolling similarly to the audio feedback. This synchronization model guarantees high flexibility and quality because it enables the synchronized replay of arbitrary streams and it supports advanced navigation and browsing of the respective files.

Implementation. In our implementation, the stream containing the presentation graphics is captured in a symbolic representation and stored in an XML-like format. Such *object-* or *vector-based* recording has several advantages over *raster-based* capturing (i.e. a stream of bitmap-based frames). For example, the recordings can be scaled to fit any window size during replay without significant loss of quality thus enabling high quality replay on different platforms and devices. In addition, an object-based representation can easily be transferred to other data formats, while the transformation of raster graphics into a non-bitmap format may lead to a decreased quality (or might not be possible at all without unreasonable effort). Other advantages include a (usually) smaller data volume, the option to post-edit the contents if necessary (e.g. to remove errors, misspellings etc. from slides), easier analysis and index generation, (e.g. to enable full-text search on the slides), or the possibility to implement better, advanced interaction functionalities, such as the ones described above.

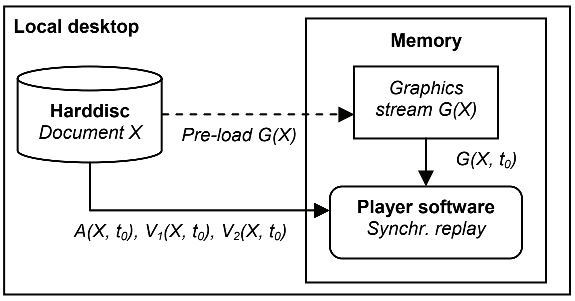


Figure 2: Illustration of the implementation of the replay model (example with audio stream *A*, presentation graphics *G*, and two video streams V_1 and V_2).

Figure 2 illustrates the standard implementation. All vector graphics of the document (slides, annotations, etc. as well as metadata such as thumbnails and content lists) are kept in the system's main memory due to their reasonable size even for long documents. Media channels such as audio and raster-based streams are played as streams from the hard drive (or other local storage), using a buffer which only stores the data to be displayed next. The central player module replays the master stream (audio), guarantees the synchronization of the other media streams, and handles the graphical user interface which offers different functionalities for document browsing. Hence, the player software is realized as a standalone desktop application that relies on the local availability of the respective documents. This stands in direct conflict with our aim discussed in Section 2: to support streaming media as well.

3.2 Integration of Streamed Replay

The synchronization model described above enables generic integration and processing of different media streams, guarantees a high quality, synchronized replay, and supports advanced features for navigation and browsing. The experience and feedback we gained with the usage of its different implementations since we originally introduced it, confirm these statements. For example, our evaluations showed that students strongly prefer the respective recordings over the ones provided with standard streaming technologies, although this involved the installation of proprietary player software and large file downloads (Lauer et al. 2004).

On the other hand, as discussed before, there are various reasons for document delivery via streaming technologies. The recent trend of so called *Rich Internet Applications* (RIAs) now enables us to combine the best of both approaches. RIAs run in a Web browser but resemble traditional desktop applications by providing a similar look and feel as well as functionalities comparable to local desktop applications. This is achieved by transferring some of the processing from the server to the client side (i.e. the Web browser) using Ajax (Asynchronous JavaScript and XML) technologies, such as Dynamic HTML, XML, Cascading style sheets, DOM, and JavaScript.

Distributed implementation for streamed replay. The basic idea of our new approach is to transfer our synchronization model to a distributed client-server architecture that enables us to access the data-intensive streams (i.e. the raster-based and acoustic data) of a document via streaming. However, less data-intensive streams (i.e. the presentation graphics as well as some potential metadata streams) are stored and managed on the local client application. The realization of this distributed implementation is a direct transfer of the local architecture as it is illustrated in Figure 2. Figure 3 shows the respective example for the distributed case. It should be noted that the actual synchronization model is still the master-slave synchronization where the audio with the instructor's voice is the designated master stream and synchronization of all other media streams is controlled by the RIA implementation.

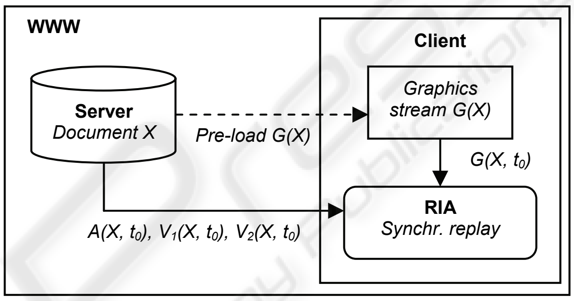


Figure 3: Illustration of the distributed implementation of the model (cf. Fig. 2) using RIA technology.

This approach is based on the observation that the most important media stream for browsing and navigating, i.e. the presentation graphics stream has a relatively low data volume (especially if the recording was done in a vector-based description as described above). Additional streams for navigation, for example, the thumbnail stream or any other stream featuring metadata for navigation purposes, are normally even smaller. They can easily be transmitted as a whole when the user starts to view a lecture and will then be available locally on the client throughout the session. Thus, navigation within the graphics stream can be carried out with all the desired features mentioned above, including visible scrolling. Locating a certain part of the talk in the slides stream is very quick and requires no buffering, since none of the streamed media is involved. Once the position has been found, replay resumes there: the audio and video streams are requested from the streaming server and delivered via the respective streaming protocols. Hence, the situation is similar to the local case with the exception that the video stream is not updated during scrolling. However, this is not a problem for our application scenario, since the video recording is not considered critical (Gemmell and Bell 1997). During normal replay, the streamed audio signal is still used as the master stream synchronizing all other media channels. Thus, the best possible replay quality of the two critical media (i.e. audio and presentation graphics) is still guaranteed.

Client-server communication. Figures 4 till 6 visualize the event traces for the communication between user, RIA, and server. The grey area marks the data stored and the RIA application running on the client machine. When the user accesses the URL of a lecture document X (Fig. 4), the RIA is transmitted to the client. The RIA's program logic then downloads the document description, which contains information about the different streams in X (in our example, one audio stream A, the graphics stream G, and two video streams V_1 and V_2). The RIA requests the graphics stream $G(X)$ (i.e. slide and annotations). $G(X)$ is transmitted completely and managed locally in a database that is part of the RIA. Upon completion of this step, the lecture document X is ready for replay.

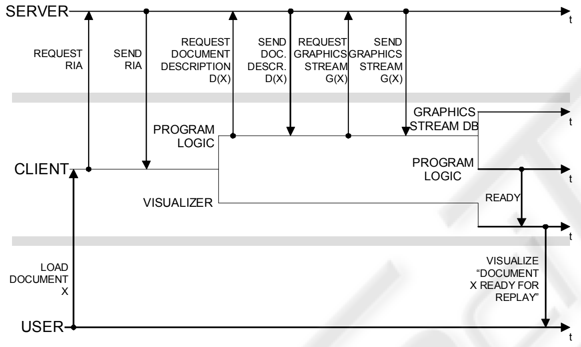


Figure 4: Event traces at startup of the RIA.

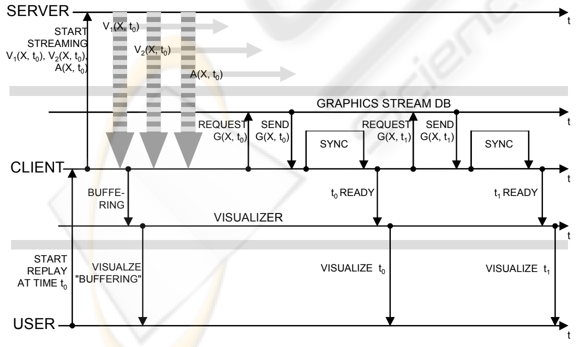


Figure 5: Event traces during replay.

When replay is started at t_0 (e.g. by the user pressing the "play" button in the application, cf. Fig 5), the audio stream $A(X, t_0)$ and video streams $V_j(X, t_0)$ are requested from the streaming server. The appropriate part of the graphics stream $G(X, t_0)$ is retrieved from the local database and synchronized with the continuous streams. Note that all synchronization is done by the RIA program logic on the client. Thereby, the RIA takes buffering and pausing into account, and the server is not involved in the synchronization process. In each synchronization cycle (at time-stamp t_i of the master stream A), the respective part $G(X, t_i)$ of the graphics stream is requested locally, synchronized with the streamed media and displayed to the user.

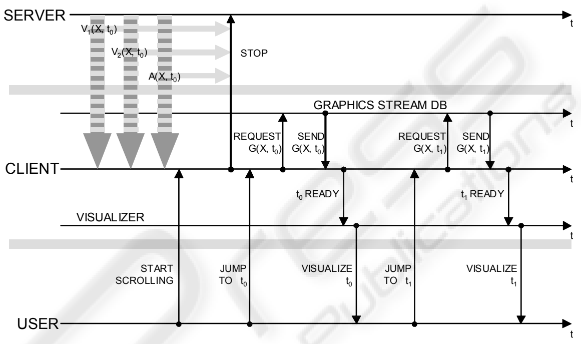


Figure 6: Event traces during random visible scrolling.

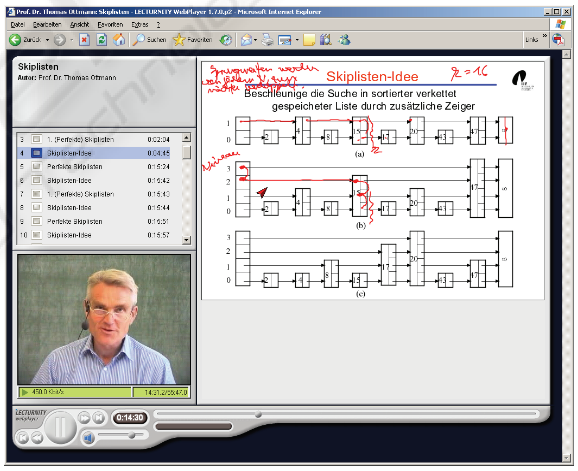
The most interesting cases in our scenario are the different forms of user interaction, e.g. dragging the slider in order to locate some specific part of the document (cf. Fig. 6). When the user starts scrolling (by holding down the mouse button on the slider), the server is notified to stop streaming the continuous media. As long as the user is scrolling, every new position on the slider is mapped to the corresponding time-stamp t_i in document X and the respective part of the graphics stream $G(X, t_i)$ is retrieved and displayed immediately. Note that no server access is required since the complete graphics stream is available locally and no synchronization with other streams is necessary. Thus, we can achieve real-time visible scrolling on the most important streams for browsing, i.e. slides and annotations as well as meta data. Thereby, the RIA enables the user to get a quick overview of the contents and to easily locate known sections in the recorded document without any network latency. The RIA does not reconnect to the server for streaming until the user releases the slider. Then, replay is resumed with all involved streams just as described above and shown in Fig. 5.

All other forms of navigation (such as slide forward/backward, clicking a thumbnail in the table of contents, etc.) use the same basic mechanism. The contents of all object-based streams are pre-loaded at startup to the client. Any interaction that forces the player to jump to a new position within the document is displayed in those streams without any delay by the respective slaves. Streaming of all other media streams (usually audio and video) is resumed at this new position as soon as normal replay continues.

Object- vs. raster-based recording. It is important to note that the individual presentation streams must be available separately in order to permit this type of media delivery. In addition, the visual graphics streams required for navigation (most importantly the presentation graphics) must be small enough to allow for a quick transmission to the client at the start-up of the RIA. It now becomes obvious that presentation recording approaches producing integrated documents cannot support the proposed distributed replay architecture. This is especially true for the two extreme approaches, streaming-only and download-only. Screen grabbers (e.g. Camtasia (2006)) generate videos, which can be streamed from the server. There is no further interaction or functionality provided on the client side except the usual start, pause and stop of streaming video. The other extreme are approaches producing, e.g., Flash or QuickTime documents (Macromedia 2006). These documents often provide convenient interaction facilities client-side, but have to be preloaded in a progressive download mode from the server without any later client-server communication. Hence, our claim that an object-based recording should be preferred is not only motivated by the arguments given in Section 3.1 but also to support better interaction with the documents. With massive binary information involved in the preloaded streams (as is the case for sampled audio and video), transmission time increases sharply, which quickly turns the advantage of the approach into the opposite. Thus, if such data is to be used, the compression rate and, consequently, the quality of the represented contents, is a crucial factor.

Implementation. Although the implementation of the proposed method is not trivial and involves a lot of small technical difficulties it can be done with common internet technologies usually applied for the realization of Rich Web Applications. A browser-based implementation can be realized using (D)HTML, JavaScript, and streaming technologies such as Windows Media (2006) or RealMedia (2006). In the latter case, synchronization can also be done using SMIL. Our own implementation, for example, produces Windows Media-, Flash-, and RealMedia-based RIAs. These can be executed in any common web browser. Figure 7 shows an example of the RealMedia implementation which provides the same look and feel as the desktop player illustrated in Figure 1. Further examples can be accessed from our website (E-lectures 2006).

Providing desktop look and feel via templates. For reasons of recognition and convenience, a RIA's graphical user interface is normally adapted to the desktop counterpart, i.e., the stand-alone media player. Most of the GUI components should behave as expected from "normal" applications, following common desktop interaction paradigms. This allows the users to switch seamlessly between desktop and Web-based applications. For example, assume a student who uses a desktop player in combination with previously downloaded lecture recordings at home but a Web browser to access these recordings on a laptop via wireless LAN while on campus. From the user's perspective, these two applications should look and feel almost identical. When comparing the local desktop media player and its Web-based counterpart shown in Figure 1 and 7, respectively, it can be seen that this demand is clearly fulfilled by our implementation.


Prof. Dr. Thomas Ottmann: Skiplisten - LECTURNITY WebPlayer 1.7.0.p2 - Microsoft Internet Explorer
Datei Bearbeiten Ansicht Favoriten Extras ?
Zurück
Suchen
Favoriten
Links
Skiplisten
Autor: Prof. Dr. Thomas Ottmann
Skiplisten-Idee
Beschleunige die Suche in sortierter verkettet gespeicheter Liste durch zusätzliche Zeiger
(a)
(b)
(c)
0:14:30

3	1. (Perfekte) Skiplisten	0:02:04
4	Skiplisten-Idee	0:04:45
5	Perfekte Skiplisten	0:15:24
6	Skiplisten-Idee	0:15:42
7	1. (Perfekte) Skiplisten	0:15:43
8	Skiplisten-Idee	0:15:44
9	Perfekte Skiplisten	0:15:51
10	Skiplisten-Idee	0:15:57

Figure 7: RIA implementation of a media player similar to the one depicted in Figure 1.

In order to support a convenient production process and to guarantee a similar look and feel, we developed and implemented a template-based approach (illustrated in Figure 8) which enables document authors to choose from a variety of basic templates (or create new custom templates). These specify the interaction, layout, and design of the RIA. An example would be the choice whether a table of contents should be included, what it should look like, and where it should be placed. The selected template can then be customized further, for example, to include corporate logos. Authors or producers can also customize the quality of the individual media streams, i.e. audio and video encoding, screen resolution, supported bandwidths, the versions of the supported player plug-in and streaming server, or choose to include metadata required for compatibility with document exchange standards such as SCORM (2006). A complete automation of the production process can be achieved if all the above selections (the fields connected with "Author" in Fig. 8) are stored in a profile that is processed by the RIA generator. The realized generation and publishing component uses XML as description language.

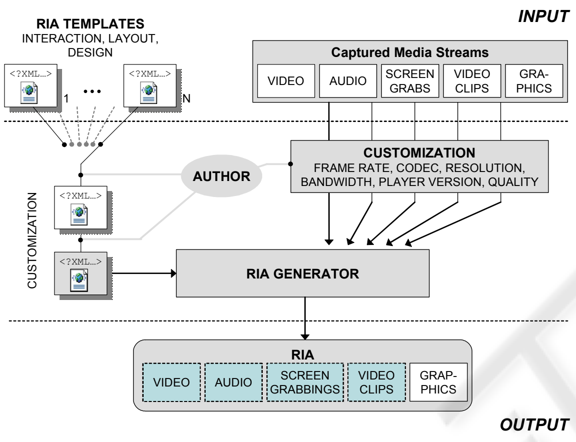


Figure 8: Automation of the RIA generation process.

4 CONCLUSION

While pure streaming solutions have advantages for both users and providers, they do not support certain navigational features required in the context of learning, especially with regard to efficient browsing within a document. We presented a distributed replay architecture which makes it possible to preload those media streams which are the most important ones for navigation, while streaming the other, more voluminous media. This allows learners to use features known from local applications, such as random visible scrolling, while avoiding the disadvantages of downloading large multimedia documents. The concept has been implemented as a Rich Internet Application using standard internet technologies.

The proposed replay model is not restricted to learning applications but can be transferred to other scenarios where Web-based delivery of multimedia contents consisting of more than one stream is required and certain navigational features are desired. The basic prerequisite is that the stream(s) directly involved in the navigation process are small enough (in terms of data volume) to be transferred as a whole at the initiation of a session. This requirement is usually met when the data consists of mostly symbolically represented information or drastically compressed binary data.

REFERENCES

Abowd, G.D., 1999. Classroom 2000: an experiment with the instrumentation of a living educational environment. *IBM Systems Journal, Special issue on Pervasive Computing* 38, (4).

Brusilovsky, P. Web lectures: electronic presentations in web-based instruction. *Syllabus* 13 (5), 18-23.

Macromedia, 2006. http://www.macromedia.com/software/breeze

Brotherton, J. and Abowd, G.D., 2004. Lessons learned from eClass: assessing automated capture and access in the classroom. *ACM Transactions on Computer-Human Interaction* 11 (2), 121-155, ACM Press.

Camtasia, 2006. http://www.camtasia.com/

E-lectures, 2006. E-Lecture portal of the University of Freiburg. http://electures.informatik.uni-freiburg.de

Gemmell, J. and Bell, G., 1997. Noncollaborative telepresentations come of age, *Communications of the ACM* 40 (4), ACM Press.

Hürst, W. and Müller, R., 1999. A synchronization model for recorded presentations and its relevance for information retrieval. In *Proceedings of ACM Multimedia 99*, ACM Press.

Li, F., Gupta, A., Sanocki, E., He, L. and Rui, Y., 2000. Browsing digital video. In *Proceedings of the ACM conference on Computer Human Interaction (CHI)*, ACM Press.

Lauer, T., Müller, R., and Trahasch, S., 2004. Learning with lecture recordings: key issues for end-users. In *Proceedings of ICALT 2004*, IEEE Press.

RealNetworks, 2006. http://www.realnetworks.com/

SCORM, 2006. http://www.adlnet.org/

Zupancic, B. and Horz, H., 2002. Lecture recording and its use in a traditional university course. In *ACM SIGCSE Bulletin* 34(3): 24-28, ACM Press.

Windows Media, 2006. http://www.microsoft.com/windows/windowsmedia/