Lessons from *LyricTime*^{TM}: A Prototype Multimedia System

Shoshana Loeb, Ralph Hill, Tom Brinck
Bell Communications Research,
445 South Street, Morristown, N.J. 07960-1910
shoshi@bellcore.com, rdh@bellcore.com, hammer@bellcore.com

The *LyricTime*^{TM} system is a personalized music system that currently runs on SUN SPARC*station*^{TM} workstations. Songs are automatically selected for listeners by a filter that adapts to them according to feedback they provide. The songs are played at a listener's workstation using its built-in audio capability. At the same time, a still image from the album cover for the song is presented on the display. The listener is free to stop and start playing at any time, step forward and backward through the list of selected songs, change the volume, tell the filter what mood he or she is in, and provide evaluative feedback on the current song.

The *LyricTime* prototype system was developed as part of an investigation of adaptive information filtering at Bellcore. This investigation explores techniques for selectively extracting and delivering information for specific users from large information sources, and includes the study of:

- adaptation mechanisms that tune the filter to the needs of the user based on feedback from the user,

- the demands that customized multimedia information delivery place on communications networks, and

- the design of control interfaces that allow casual users to easily use customized information delivery services.

We constructed the *LyricTime* system listener interface with the RENDEZVOUS™ programming language which has been developed at Bellcore. This is an experimental object-oriented programming language that supports declarative programming with constraints to simplify construction of multimedia and multi-user interfaces to computer applications. The RENDEZVOUS language has been used to build a variety of multi-user text and graphic interfaces for multi-person games and education.

The *LyricTime* research prototype is a personalized music system, similar in spirit to a jukebox -- it has a collection of songs that can be played for the listener. At its simplest level, the *LyricTime* prototype selects songs from a database and plays them for the listener. More specifically, the *LyricTime* prototype uses an information filter [Loeb 1991] to select songs from the database, using descriptions of the songs, a listener profile, and feedback from the listener. The listener can step through the selected songs, looking at title and artist information, or have the *LyricTime* prototype play them. The listener profile provides listener specific preference information to the filter. Listeners can have different profiles for different moods. Listener feedback is used to update the profile based on the listener's opinion of songs that have been played.

The current interface of the *LyricTime* prototype was designed to demonstrate the concept thus, it may not be optimal. The interface resembles a physical jukebox in appearance and behavior. A hand-drawn color image of a jukebox is the backdrop for the interface. The drawing helps establish the listener's conceptual model of the *LyricTime* prototype.

The listener interface to the *LyricTime* prototype is divided into three main units. The upper portion shows information on the selected songs and the song currently being played. This includes a still image representing the current song, some text indicating how many songs have been selected by the filter, and the title and artist information for, from left to right, the previous, current, and next songs. The middle portion of the interface contains buttons for controlling the playing of songs. These buttons allow the listener to step backward and forward through the list of selected songs, play the current song, and stop playing. There is also a button to quit the

LyricTime is a trademark of Bell Communications Research.

SPARCstation is a trademark of SPARC International.

RENDEZVOUS is a trademark of Bell Communications Research.

program. Once the listener clicks on the play button, the songs that have been selected by the filter are played, in order, until the listener clicks on another control button. If the listener causes playing to be stopped in the middle of a song the volume is gently brought to zero before stopping. This avoids abrupt stops that can startle listeners. The lower portion of the interface has three groups of five buttons. The upper group controls audio volume. The middle group allows the listener to select a mood, indicating which listener profile to use (there is one profile per listener, per mood). The last group of buttons allow the listener to express an opinion of the song that is currently being played. This information is used to update the current listener profile.

There are seven main components in the *LyricTime* prototype, which could be distributed in a network, and (ultimately) could even be supported by different vendors. These components are:

(1) The listener's terminal which must have audio and graphic output capability and support pointing input (e.g., with a mouse or touch screen).

(2) The user interface which presents information to the listener, does low-level updates of the graphics (such as highlighting buttons as they are clicked) and informs the controller of user actions.

(3) One or more database(s) of songs. Each song has four elements: digital audio, text for the listener (title, artist, liner notes), images to be shown while the song is played, and a description used by the filter.

(4) The controller interfaces between the user interface and the filter. It passes listener feedback to the filter, and tells the interface where to find the songs.

(5) The adaptor which uses listener feedback information to update the listener profiles.

(6) The filter which uses the listener profiles and song database to select songs for the listener.

(7) The listener profiles which provide information to the filter on the type of songs the listener likes and dislikes.

The *LyricTime* prototype presents a structurally simple, high-level abstraction to the listener. This frees the listener to use the *LyricTime* prototype without having to understand how it is provided. The *LyricTime* prototype interface is characterized by simplicity and directness of control. Every button says what it does, and does exactly one thing. Buttons always react to being clicked on, unless it is obvious from the behavior of the interface that it makes no sense to click on them. (There are only two examples of this in the *LyricTime* prototype: clicking on the Play button while a song is being played, and clicking on the Stop button when nothing is being played. In both cases the buttons are grayed out to indicate that that are not available, but they are still visible, so the user knows that the functionality is there, but not available at this time.)

We claim that the *LyricTime* prototype is a good platform for studying delivery of continuous media to users across a network. The two principle types of continuous media are full-motion video and audio. From the perspective of a communications network, the problems of delivery of audio and video are qualitatively similar. However, compared to video, acceptable quality digital audio can be delivered very inexpensively, with low-cost equipment, because of its lower bandwidth requirement.

The *LyricTime* prototype raises issues of delivery and interruption of continuous media (with play and stop) and synchronization of different media types (e.g., changing the ``album cover'' picture as a new song begins). These are important issues for all multimedia systems that deal with continuous media. The *LyricTime* prototype provides both an example of an service that raises these issues, and an inexpensive platform with which they can be studied. Any bandwidth independent solutions that are found in the domain of audio (e.g., various forms of pre-fetching and local caching based on predicted user behavior) should be transferable to the video domain. Thus, the *LyricTime* prototype allows most problems with continuous media to be studied, in the context of a complete service, without incurring the high cost of setting up and maintaining a video network and library.

We believe that the *LyricTime* prototype provides an example of a compelling multi-media application. Any audio (or information) system based on a personal library suffers from several limitations. A personal library is limited in size to a set of albums purchased, and by buying whole albums, it is more difficult to selectively purchase only those songs one is interested in. A personal library is more difficult to keep up-to-date, but with the *LyricTime* prototype, new releases could become immediately available.