RJ 10221 (A0110-027) October 17, 2001
Computer Science

IBM Research Report

Virtual Jukebox: Reviving a Classic

Clemens Drews, Florian Pestoni
IBM Research Division
Almaden Research Center
650 Harry Road
San Jose, CA 95120

Research Division
Almaden - Austin - Beijing - Haifa - India - T. J. Watson - Tokyo - Zurich

Virtual Jukebox
Reviving a classic

Clemens Drews
IBM Almaden Research Center
San Jose, CA 95120
+1 408 927 1208
cdrews@almaden.ibm.com

Florian Pestoni
IBM Almaden Research Center
San Jose, CA 95120
+1 408 927 2626
fpestoni@almaden.ibm.com

Abstract

Recent advances in compression technology, combined with lower cost of storage and bandwidth, have made digital distribution of rich content including music not only technically feasible but also popular with a broad audience. However, limited progress has been made in the way this content is enjoyed by end users. In this paper, we focus on the problem of playing music in a shared space – e.g. office, home, car – such that all listeners who are present share a positive music experience.

Our scheme enables collaborative selection of content and pooling of content files. Users can express their preferences by contributing songs to be played and through a simple voting scheme. The system builds profiles and automatically selects content for playback, maximizing the match with the group's taste. As users vote, the system learns more about their collective preferences and can adjust the playlist accordingly, thus providing an incentive mechanism.

1. Background and Motivation

For as long as music has existed, it has been a collective experience. Until audio recording made it possible to separate the performance from the listening, music was enjoyed in concerts, large or small; even today, live musical performances continue to attract large audiences to venues ranging from smoky bars to stadiums, from Opera houses to private residences. Although some technologies – most notably the Walkman personal cassette player introduced by Sony in the 1980's – have indeed made possible and encouraged individual listening, music continues to be enjoyed mainly as a shared experience.

Over time, models for selection of recorded content have changed little and slowly. The introduction of audio broadcasting revolutionized the music industry, but since then few developments have significantly affected the way people choose what music to listen to. At one end of the spectrum, an individual may be in full control over what content to play for personal enjoyment or to be shared with others in the same location. At the other extreme, content selection is *delegated* to disc jockeys, programming managers, and advertisers; individuals are limited to selecting a particular radio station.

An icon of popular music tradition in some parts of the world, the *jukebox* was introduced in the late 1920's. It offered a simple and affordable model for content selection, a rare and early case of combining interactivity with a group selection process. The history of the jukebox is linked to the diffusion of popular styles of music that were under-represented in the established broadcasting networks. In 1927, The Automatic Music Instrument Company created the world's first electrically amplified *multi selection* phonograph. Underground tavern owners during Prohibition, who could not afford a live band, could instead attract customers by installing a jukebox, which was provided by an operator at no charge. [1]

As new technologies were introduced and social conditions changed, the importance of the jukebox faded but did not disappear completely. Today, jukeboxes containing hundreds of CDs can still be found in the back of some bars or providing a *retro* touch to the lofts of wealthy individuals.

However, many of the same problems that the jukebox model addressed remain. In a shared physical environment,

be it an office, car or home, how can the various individual preferences be satisfied? Given that audio can be quite intrusive due to the inability of humans to block sound (unlike images), what model can be used to minimize the negative effect that this intrusion may have on a group of listeners? We revisit the jukebox model of collective selection, eliminating many of the constraints and adding new levels of interactivity made possible by computer technology: inexpensive storage, simple and affordable local area networks, advanced audio compression algorithms and developments in user interface design.

2. The Virtual Jukebox Model

The technology we propose, which we have dubbed a *virtual jukebox*, is a networked music player that downloads music files on demand from distributed storage. It can be implemented on a custom or off-the-shelf embedded computing platform with compressed music decoding in hardware or software, and is connected to speakers or to a stereo system for audio output.

The virtual jukebox runs an embedded http server and supports a multi-modal user interface. Users may make selections from their web browsers, through speech commands or using a PDA. Each selection (essentially, a URL) is added to a FIFO queue that controls playback. Since files are stored on the network, the virtual jukebox needs only a limited amount of storage, enough to cache a few titles. Songs can be pre-fetched during playback of the current selection, to assure smooth transitions.

In addition to these capabilities, which bring the concept of the jukebox up to date with current technologies, we propose a voting scheme that adds a whole new level of interactivity. As a song is playing, users can vote for or against this title. If enough negative votes are received, the song is skipped and the next song in the queue is played. The system collects votes and builds a profile representing the collective preferences of the community of users.

When no selections are pending, the system can automatically make selections on behalf of its users. On a traditional jukebox, this was typically implemented by playing random songs from the list of titles locally available. On the virtual jukebox, we use the history of votes to select not just any song that has been played, but those that are popular within the *specific* group of users that usually connect to this player. A time limit can be set to make sure that the same song is not played too often and avoid repetition.

The system we propose addresses the problem of selecting music in a shared space. For example, in an office environment, audio can help set the mood and may improve productivity. However, musical preferences are quite idiosyncratic, and the wide variety of styles available can lead to significant disagreement within a group as to what particular content is collectively acceptable.

By enabling users to contribute their own music collections, we can guarantee that the distributed content database will *include* a representative sample of the *union* of all titles that are of interest to the community that develops around it. Voting, on the other hand, can be used to *exclude* those titles that are not in the *intersection* of individual preferences. This rule can be relaxed to allow some flexibility, for example playing songs that several members of the community feel strongly in favor of but for which other members have a slight negative bias.

Our proposed scheme can help build consensus and a stronger sense of community. The voting scheme can highlight commonalities among users and the method of pooling resources can help expose listeners to content they may not have experienced before. Because the system learns from its users, it encourages a more interactive experience. However, when selection of music takes a low priority for listeners, the system can automatically take over this role on behalf of its community of listeners and taking into consideration their collective preferences.

2.1. Content Selection and Playback

The *virtual* jukebox is not limited to a small collection of titles that can fit in it, like its physical predecessors were. Instead, it can use any compatible content available on the (local) network. In the simplest selection model, a user specifies the URL for a compressed music file, which is downloaded to the jukebox as needed for playback.

By adopting this *distributed content* approach, not only can the selection be unlimited, but also new content can be easily introduced. A user's personal collection may be incorporated instantly to the repertoire of songs the jukebox can play by just providing the URL in which it can be found. This greatly simplifies pooling of content among members of the listening community.

To ease the navigation of possibly large collections of titles, we propose a simple crawling mechanism. If content is organized hierarchically (as would be the case for content stored in the familiar directory structure used by modern computer systems) or in other linked list fashion (e.g. hyperlinks in HTML) then a crawler simple enough to run in a embedded system can follow the links and build a list of music files from a base URL. Moreover, the crawler can read the metadata contained in each file and maintain a small database of titles that can be be browsed and sorted by different criteria, such as author, album or year. Note that this requires very limited persistent storage, as the title themselves are not copied over to the embedded system.

The implementation of metadata may vary with the encoding of the content; for example, in the MP3 format, ID3 tags are stored in the last 128 bytes of a file. Thus, a crawler using a protocol that supports downloading byte ranges such as HTTP could read just the appropriate portion of the file to make building this index of titles more efficient. Additionally, every time an individual title is selected, it can be checked against the database and, if it happens to be a new URL, it can be automatically added to the list of crawled titles.

The virtual jukebox implements a multi-modal user interface for content selection, including Web browsing, speech and handheld computers. No single UI is absolutely superior to the others, since each one has relative advantages depending on the access mode and the users' preferences and abilities.

The metadata collected by the crawler can be used as the foundation for a speech-based user interface. We combine speech input (recognition) and output (synthesis) to enable sophisticated navigation of data. By restricting the vocabulary to only those words contained in the metadata, the performance of speech recognition can be improved. This may require multiple input/output interactions, for example listing first a list of authors and then, once the user makes a selection, listing albums or titles by the chosen author.

Speech input is especially well suited for direct access. For example, a user may say "Play *My Way* by Frank Sinatra". The system would then try to find a phonetic match within its vocabulary. The match may not be unique, i.e. there may be several possibilities (e.g. there may be both a live and a studio version, the recognition of the spoken command may be uncertain or the request may be an incomplete specification); in such a case, the system may present a reduced set of options to the user for selection, or play one at random.

The current state of the art in speech technology does not guarantee flawless interactions. For example, proper names or creative spellings (*Lynyrd Skynyrd* comes to mind) may be incorrectly rendered by the speech synthesis system. In addition, speech recognition engines are usually sensitive to ambient noise, and therefore the playback of music during speech-based selection may introduce further errors. While we have considered possible solutions to some of these problems (see Conclusion and Future Work), this is outside the scope of this paper. Note that from an implementation point of view, the speech processing need not be performed on the jukebox itself: network protocols such as VoiceXML [2] allow for interaction via speech with remote systems.

Another alternative for navigation of content is through a Web browser. The jukebox is addressable via HTTP (i.e. runs an embedded HTTP server) and serves HTML pages. Using transcoding techniques, any HTML page on the network that contains links to music files or to other pages can be processed by the jukebox (acting as a proxy) such that relative references are made absolute and are included as a parameter to a URL that points to the jukebox. Thus, users can seamlessly access any page,[1] such as the web pages served by Apache to represent the local filesystem available via HTTP or FTP. Based on the content type, music files can be identified as such and treated differently: when the user clicks on a link pointing to a music file, this is interpreted as a selection.

With more advanced transcoding, the jukebox can support navigation from PDA's. Naturally, PDA's would have to be connected to the network, either directly (e.g. wireless card) or through a proxy (e.g. USB to a PC on the network.) Due to the more limited graphics capabilities of most current PDA's, it may not be possible to access any Web page, as was the case for a full-fledged Web browser. However, the database of crawled content can be presented for selection, possibly with multiple levels of interactivity.

A user-initiated selection is not always played instantaneously. The virtual jukebox, just as its more traditional cousins, is in essence a FIFO queue of musical selections. Unlike in a traditional jukebox, however, the content is not locally available. Files could be streamed from their source, i.e. they could be fetched in real time as they are being played. However, due to latency problems on some networks and the uncertain delivery time for packages on TCP/IP, this may lead to interruptions and reduce sound quality.

An alternative mechanism, that requires some (non-persistent) storage is to use a cache. The queue and the cache can work together to pre-fetch files that are coming up for playback. The cache needs only be large enough to contain a few (2-5) songs. With reasonable bandwidth and compression schemes, playback of a song should, on average, take longer than the actual download, thereby guaranteeing that as long as there are enough selections on the queue, transition from one song to the next will be smooth and without interruptions. Of course, if the queue is empty and a new selection is made, there will be a delay before actual playback begins, as this first song is being downloaded.

Figure 1 shows these components as well as the user interface modules.

The queue manager, the cache and the audio player must be synchronized to guarantee smooth transitions. Each URL on the queue represents a selection. When the current selection is finished playing, the audio player notifies the cache and the queue manager of this event. The recently played file, which was stored on the cache, is

[1] Pages with JavaScript, applets or some CGI scripts may not be transcoded correctly.

marked as *unlocked*, indicating that it is eligible for deletion. All entries in the queue are bumped up one position, as the top one is removed. The next URL (now the top element on the queue) should already be available in the local cache, and the audio player is pointed at it so that playback can commence. The file is marked as *locked*, to indicate that it is in use.

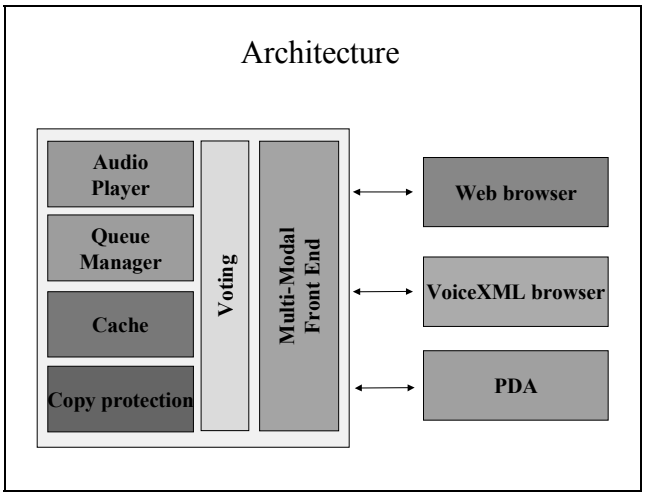

Figure 1- Architecture

The URL corresponding to the next song in the queue which has not yet been downloaded is passed to the cache. Depending on the capacity of the cache and the number and size of the music files are actually stored in it, the next song may be second in the queue or deeper. The cache first checks the size of the file to be downloaded and makes room if necessary by deleting the oldest eligible file and continuing as needed with the next eligible file until all have been deleted.

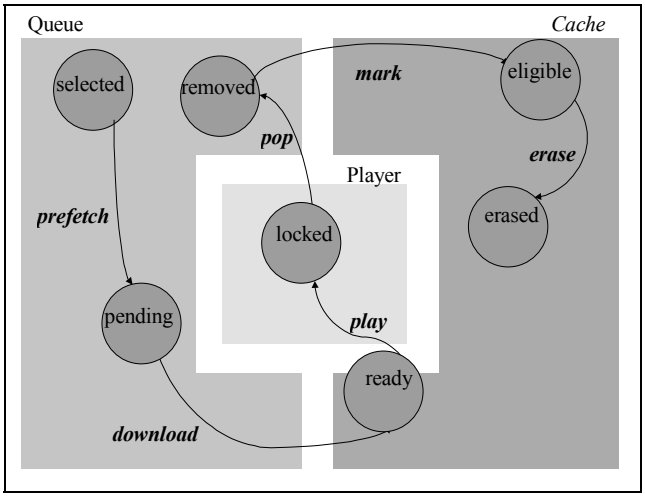

Figure 2- Finite state machine

It is never optimal to delete a file that is newer than the file that is currently playing. Each file in the cache has an *age* associated with it. In the case in which no files older than the current selection can be removed from the cache, and the next file to be downloaded still will not fit, the cache must wait until the current files are processed. Should a file that is still in the cache be selected for playback, no download will be necessary, but its age and state must be adjusted accordingly.

Once there is enough room in the cache, the next file is downloaded. As soon as it is fully downloaded, it is marked as *ready*. A conservative policy for the audio player is to never start playing a file that is not yet ready; a more aggressive policy may start playing the song back right away, hoping that data will be ready when needed. Figure 2 summarizes the discussion above.

2.2. Voting and Autoplay

Since musical taste can vary widely among the members of the audience of the jukebox, it is necessary to give listeners a way of controlling the playback in case they find it too annoying. Certain selections may offend some parts of the audience, or may not be suitable for the current situation. We chose a democratic approach of resolving these different interests: whenever a majority of listeners dislikes the current playback, it is stopped immediately, and the queue proceeds to the next song.

The audience can give feedback about the song that is being played. This feedback is used in two ways: to influence the playback of a song and to gain knowledge about the taste of the audience.

During playback of a song, negative, positive or no feedback about this particular song is given by each listener. Negative feedback is used to stop a song from being played if a majority of listeners express negative feedback. Positive feedback cancels out negative feedback. This feedback is also used to learn about the preferences of the listening community; the use of these preferences is described in the following section.

To stop a song and skip to the next one, the net negative votes must exceed a threshold, e.g. 50% of the total audience. For the proposed scheme it is not particularly important to know the exact number of listeners. This value can be set as a parameter, which can be updated as members join or leave the community. (See also discussion of user tracking in the Conclusion and Future Work section.)

In settings where it is inconvenient for the audience to actively select content, there needs to be a mechanism for the jukebox to select and play content automatically. The first approach that comes to mind is to just randomly select songs over the entire list of songs the jukebox knows

about. A shortcoming of this approach is that, since the content is made up by contribution of its audience, a random selection will be biased towards the taste of the person that contributes the most content. Furthermore, the audience may only like a small subset of the content available.

This problem has traditionally been tackled in different ways. Radio stations work in an autoplay-like mode, although they cater to an audience that is several orders of magnitude larger than a typical jukebox audience. They try to play "hits" most frequently, while playing new content at longer intervals. The feedback to radio stations, though, has a long latency and is usually (at least for popular music) determined by the sales figures published by the record labels.

An alternative is the model used in dance clubs, where a disc jockey selects content on behalf of the audience. Here, there is a more immediate feedback mechanism: the DJ can gauge the audience's reaction based on their behavior. Are they dancing? Do they look bored? Should the tempo be more upbeat?

However, for very small audiences such as in the case of the virtual jukebox, neither of these models is ideal. In such a small setting, an individual would expect to have a greater influence on the content selection.

Our goal was to come up with a system that learns about the taste of its audience by observing its users' feedback. When not filled with user requests, the system would play songs that are popular among its listeners. We wanted the system to mix less popular content with more popular content, so it would introduce content to the audience that is new or different than the mainstream the audience is normally exposed to.

In our proposed algorithm, the popularity of all content known to the jukebox gives a measure of the current taste of the audience. The jukebox will statistically select popular songs more often than less popular ones. It will still play the least popular songs, although less frequently.

A virtual jukebox that does not have any pending requests from users will select a song that is most likely to appeal to a majority of the audience. This is determined solely based on the history of positive and negative votes that songs have received. We chose not to overload the user interface with more controls to let users determine the popularity level of a song. Also having the audience go through a questionaire, as in [5], to determine what the audience does or does not like seemed too unflexible for an audience whose taste changes over time.

Our approach makes it possible for the jukebox to adapt to the changing taste of its audience. It closes the feedback loop between the users and the system with very little time lag. While giving preference to popular songs, the autoplay mechanism does not lock out less popular songs, and thus avoids repeating "hits" over and over again.

An integer is used as the measure of a song's popularity. Each song that is new to the jukebox is assigned a default popularity, e.g. ½ of the maximum priority. The popularity of a song can change if the song receives:

1. A majority of *negative* votes during a playback cycle;
2. A majority of *positive* votes during a playback cycle;
3. No votes during playback over an extended period of time.

In the first case, the popularity value is decreased; in the second case, it is increased; and in the last case, the popularity will converge towards the default popularity over time.

The proposed scheme relies on the fact that users will express their preference. If this is not the case, the jukebox will basically do a random selection over the content known to it. The proposed scheme accounts for taste that varies over time, by converging towards neutral the popularity of a song that is neither voted for nor against.

2.3. Copyright Protection

The increased availability of compressed digital formats for audio files has had a positive effect, exposing consumers to this new technology and creating a critical mass for its widespread adoption. However, owners of copyrights on recordings have expressed their concerns, in the press, in Congress and in the courts of law, that their rights are being violated. While unprotected content will continue to be available for a long time – after all, CDs are unprotected and ripping the tracks has become an easy task – it is to be expected that more content will be made available in encrypted form over the next few years.

Several copy protection or digital rights management schemes have been introduced, which aim to restrict how content can be (re-)distributed. These schemes can be described as *binding* the content to an entity; different models exist, including binding content to media or to a PC. Some schemes allow for protected content to be passed from one device to another, in some cases requiring an additional *license*. (This is sometimes referred to as superdistribution.) In general, only compliant players can play copy-protected content.

As far as the use of copy protection in the context of virtual jukebox is concerned, barring circumvention of these legal and technological restrictions,[2] the jukebox

[2] The Digital Millenium Copyright Act and similar legislation in other countries have made it illegal to manufacture, sell or use circumvention devices.

would need to be compliant with the requirements of a certain scheme in order to play content encrypted under it. In some cases, this may be incompatible with the proposed use: for instance, if content is bound to the PC used to download it, and superdistribution rights are not granted, then the jukebox would not be able to legally obtain a decryption key to unlock the content.

New models for copy protection are emerging that attempt to find a better balance between the needs of copyright owners and the desire of consumers to have compelling applications for the content they have legally acquired. We hope that this paper will help influence decision-makers in these matters to include enough flexibility in the systems they adopt to allow for the advantages we have described.

3. Prototype

We built a prototype that demonstrates the feasibility of the proposed ideas above. Our virtual jukebox is based on an off-the-shelf single-board system [3] that includes a x86 CPU, a sound chip and an Ethernet network interface. We added external RAM for runtime and transient storage, and flash memory for the system image, applications and persistent storage.

The system has a minimal amount of memory to pre-cache digital audio content and store data about the content. The amount of storage needed is determined only by the maximum number of songs that are desirable to pre-cache and by the maximum length of a song. An average MP3 song has about 1 MB per minute play time. The average song is 3.5 minutes long, and it is safe to assume that among popular music not many songs are longer than 45 minutes (the length of one side of an old fashioned record), so a 45MB cache would be suitable for most if not all cases of popular music songs.

For packaging, we chose a scaled-down version of a Wurlitzer jukebox. The company, which is still in business, sells merchandise; we found a "piggy bank" that, at 9" tall and 5" wide, almost perfectly matched the size of our system board. A little mechanical abuse allowed us to insert the hardware in this encasing. This familiar look immediately gives the audience a visual cue that suggests the device's functionality. In addition, most people think "it's cute."

Our prototype runs a custom image of the Linux operating system. The application is implemented in Java, including a bare bones http server; the synchronized queue, cache manager and audio player; and support for remote front ends (HTML, VoiceXML, etc.) Audio decoding is performed in software using the xaudio library. [3]

Speech input (recognition) and output (synthesis) was actually performed on a remote system running IBM's Viavoice and VoiceXML browser. The virtual jukebox generates VoiceXML documents on the fly and serves them over HTTP to support navigation of the content database by artist and genre. Because each "page" implicitly defines a limited grammar, reasonable recognition performance can be achieved without training.

Figure 3 - Virtual Jukebox Prototype

Voting from a web browser was implemented with an applet that displays the current selection and has buttons for positive ("*rocks*") and negative ("*sucks*") votes. Buttons are disabled after a vote and until the next song starts playing to prevent duplicate votes. For access devices with limited display capabilities such as PDA's, a stripped-down HTML interface displays the state of the queue, including the current selection, and allows navigation by artist through a hierarchical representation and pagination (a limited number of songs can be displayed at any time.)

Three prototypes have been built. One is used at the Almaden Research Center for demos, usability studies and further research. The two other units are installed at IBM's Industry Solution Labs in Hawthorne, NY and Zurich, Switzerland, where they are used to showcase non-traditional computing platforms, embedded systems and novel user interfaces.

Our experience with this prototype indicates that the use of the device as a music player, with content distributed throughout the network, is easy to understand for users who have had prior experience with MP3s. Indeed, for less technically-inclined users, this device had the appeal of simplifying the use of compressed music files, although several had reservations about the use of speech as an interface.

The use of voting was less intuitive to most users. However, we have developed a small community of users of this and other systems derived from it, that makes heavy use of voting and autoplay. One interesting use that we had

not envisioned beforehand was using voting to cull out poorly encoded files: whenever such a title was played, a collective and instantaneous response guaranteed enough votes to skip the song and exclude it from the high popularity lists.

4. Conclusion and Future Work

We have shown a way to share what is possibly the most intrusive multimedia content: audio. Users can share very large amounts of audio content with a relatively inexpensive embedded device that has very limited storage and processing capabilities. We have introduced a selection process that is convenient for users and allows them to specify the content they want to listen to. As an alternative, such as when casual background music is desired, users can let the system choose the content, which will be customized to their collective preferences.

In our prototype, we implemented a simple voting algorithm. However, with the knowledge available to the system, more complex algorithms as described in [6] could potentially make the automatic selection of content an even more positive experience for its audience.

Further work will have to address a way to share control over the volume of the jukebox. For places like office environments, where frequent interruptions grasp the attention of parts of the audience, there needs to be a way to adjust the volume based on ambient noise and external events. In addition, it may be possible to improve the recognition of spoken commands even while the system is playing music by applying noise-canceling techniques in which the output signal is "subtracted" from the input signal.

In its current form, the virtual jukebox assumes users are not abusing the system by e.g. flooding the system with their requests, and therefore dominating the music. As long as an audience shares the same space, there will be sufficient social interaction among the audience to prevent this from happening. Based on our work on the virtual jukebox, we have extended the concept to allow an audience to share a virtual space by streaming the content instead of playing it back at a central location. In this case, normal social interaction may be insufficient for self-regulation, and the virtual jukebox will need to introduce play quotas.

To make the music selection even more appealing, it is important to have means for detecting who is part of the audience. The jukebox could be extended to track who is physically present through several mechanisms, partly determined by the interface used to interact with the jukebox. If the HTTP interface is used, simply using cookies will let the system identify active listeners. With voice commands, one could use an RF-ID/Active RF type approach to identify users of the jukebox. The selection of content can then be targeted by only considering the preferences of users who are present.

5. References

[1] *"History of the Jukebox,"* http://www.history-of-rock.com/history_of_the_jukebox.htm

[2] VoiceXML Forum, http://www.voicexml.org

[3] *"PCM-5823 NS Geode with Dual Ethernet, Audio, and VGA/LCD,"* http://www.advantech.com/products/PCM-5823.asp

[4] *"MP3 and Digital Audio Solutions for All Platforms and CPUs,"* http://www.xaudio.com

[5] *"MusicFX: An Arbiter of Group Preferences,"* J. McCarthy and T. Anagnost, ACM 2000 Conference on Computer Supported Cooperative Work, 2000.

[6] *"Social Information Filtering or Music Recommendation,"* Upendra Shardanand, Final Thesis, MIT, September 1994