Poolcasting: a Social Web Radio Architecture for Group Customisation

Claudio Baccigalupo and Enric Plaza
IIIA - Artificial Intelligence Research Institute
CSIC - Spanish Council for Scientific Research
Campus UAB, 08193 Bellaterra, Catalonia (Spain)
Vox: +34-93-5809570, Fax: +34-93-5809661
{claudio,enric}@iiia.csic.es

Abstract

Poolcasting *is a social Web radio architecture in which groups of listeners influence in real time the music played on each channel. Poolcasting users contribute to the radio with songs they own, create radio channels and evaluate the proposed music, while an automatic intelligent technique schedules each channel with a group-customised sequence of musically associated songs. The benefits of this approach are multiple: on one hand music producers can increase the exposure of their songs to specific target audiences; on the other hand, music consumers can easily discover new songs that match their preferences and group with people with whom they share similar listening experiences.*

1 Introduction

In 2005, online radio listening increased 18 percent and free streaming of online music increased 37 percent [12]. About 52 million people tune in to Internet radios each month [3]. A recent survey revealed US consumers are moving to Web radios because [2]: they can listen to audio not available elsewhere, they can control/choose the music played, they get more music variety and more of new music. Users see Web radios as a *simple* way to listen for free to new and varied music; in the uncountable myriad of radio channels available on the Internet, they often find the station that best fit their musical tastes.

From the point of view of music providers and Web administrators, though, Web radios are not so *simple*. In first place, to set up a Web radio, a large library of songs in a digital format is needed, whether acquired by purpose or imported from a physical source. Additionally, a radio normally serves several channels at the time, and each channel needs dedicated bandwidth and storage space, and needs to be scheduled in advance, specifying which songs will be playing on each channel. After a Web radio has been set up, its library has to be continuously kept up to date, in order to guarantee new music to the audience and to avoid repetitions. Since competitiveness in the world of Web radios is very high, radios that do not conform to these points are soon abandoned, for changing from a station to another is as easy as clicking on a hyperlink. Indeed, this is the only action listeners can take when they do not like the songs played on a Web radio.

In this paper we present a network architecture that both simplifies the management of Web radios and improves the user listening experience and fidelity towards Web radio stations. This architecture, called *Poolcasting*, takes into account listeners not as a purely passive audience, but rather as a proactive public who is given the tools to interact and to collaborate, in order to adapt the music played on each channel according to their tastes. Poolcasting automates and facilitates the creation and management of a Web radio in several aspects. Firstly, no large library of songs is needed, for the streamed music comes from the personal digital libraries of those users who have agreed to share them. When a user shares her personal library, Poolcasting partly infers her musical preferences by analysing which songs and artists are contained in her library. Thus, for each channel, Poolcasting combines the individual preferences of all of its listeners to automatically create a customised sequence of songs that matches both the preferences of the audience and some musical associations criteria. Poolcasting proposes a new *social Web radio* experience, where the audience is able to influence in real time the music being played on each channel. At the same time, Poolcasting improves the administrators' work, who can spend less effort in set up and maintenance, and learn more about their listeners' musical preferences, to eventually focus on promoting new targeted music to their audience.

2 Social features in Poolcasting

A social revolution has recently shaken the Internet, with media content being more and more created and distributed by common users rather than by large corporations. In this scenario, Web radios seem to have remained quite unaffected. Similarly to terrestrial radios, Web radios still follow a classical one-to-many broadcast model, where listeners are not involved in the set up and programming of a music channel. Given that no generic Web radio can completely match the complex taste of a listener, users often switch from a station to another, always looking for a “temporarily better” Web radio. To get around this lack of user fidelity, several Web-based music recommenders have been implemented that provide users with personalised music streams. Pandora[1], for instance, prompts its visitors for a favourite song or artist, and produces a stream of related songs, while Last.fm[2] assembles listening experience data from the users’ media players to provide individually tailored music stations. These systems improve the user listening experience in terms of personalisation; however they completely ignore the social nature of radio to gather communities of related listeners. These recommenders *narrow*cast music, creating one personalised but isolated context for each user. On the contrary, Web radios as such are supposed to *broad*cast their signals to a large audience.

Poolcasting merges both approaches, providing public channels that everyone can listen to, where the musical content of each channel is customised in real time for the whole *group of listeners*. In this section we first describe how users tune in to a Poolcasting Web radio (Sect. 2.1); then we present the distinguished features that let the audience of a channel collectively influence the music played (Sect. 2.2); finally we show how both listeners and music providers can benefit from this social approach (Sect. 2.3).

2.1 Basic interaction: passive listeners

Listening to Web radios is very simple. Users just have to navigate with a Web browser to the radio home-page, click on one of the available channels, and an appropriate streaming media player (e.g., iTunes, VLC, Winamp) opens up playing the music broadcast from the radio station. In Poolcasting, this process has not been changed. Figure 1 shows an example of the Web interface of a Poolcasting radio with three active channels; clicking on a Listen now button opens a media player that immediately starts playing music from the selected channel.

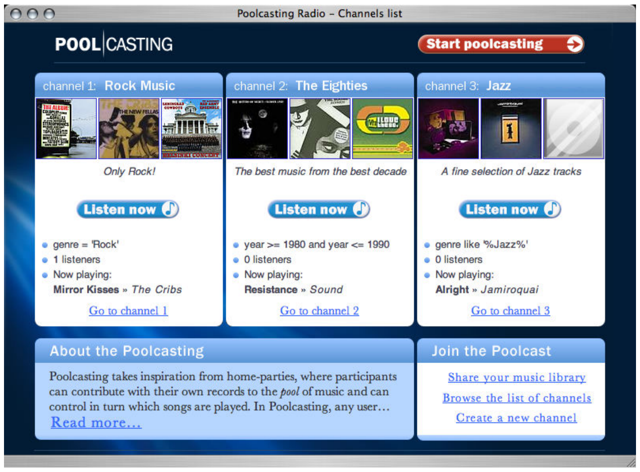
2.2 Social interaction: active participants

Tuning in to a Poolcasting Web radio channel is a concise and intuitive process, so that users who just want to passively listen to music can do so. However, users who want to actively participate in the selection of the music played can do so swiftly as well, with different functions offered by the Web interface.

Sharing personal music libraries. Web radios usually broadcast songs from a centralised music library that is set up and maintained by the administrators, who spend time and effort in acquiring new songs and keeping the collection continuously up-to-date. Still, some listeners may find the library poor or inappropriate; for instance it may contain too many mainstream tracks, or too many obscure songs, or it may be not sufficiently updated.

Poolcasting introduces the ability for listeners to share their personal music libraries, that is, to contribute with songs they own to the collection of music the radio can broadcast. To share one’s music library and become an *Active Participant*, a user has to click on the Share your music library link and indicate the path to the folder where the library is stored (see Fig. 2). Right now, sharing is implemented only for users whose library is managed using Apple iTunes and is stored in a folder accessible via HTTP. Poolcasting accesses this folder and retrieves the index file where library information is stored (a file called iTunes Music Library.xml for iTunes-managed libraries). Once this file has been loaded, all the songs in the library are virtually included in the *Music Pool* of the radio. This means that, as long as the user is connected, Poolcasting can pick from her library any song to be played on any radio channel.

The advantages of a *distributed* collection of songs over a classical *centralised* music library are numerous. Firstly,



Only Rock!

- genre = 'Rock'
- 1 listeners
- Now playing:
  Mirror Kisses » *The Cribs*

The best music from the best decade

- year >= 1980 and year <= 1990
- 0 listeners
- Now playing:
  Resistance » *Sound*

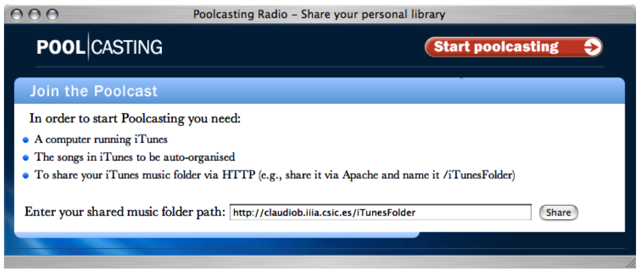
A fine selection of Jazz tracks

- genre like '%Jazz%'
- 0 listeners
- Now playing:
  Alright » *Jamiroquai*

Poolcasting takes inspiration from home-parties, where participants can contribute with their own records to the *pool* of music and can control in turn which songs are played. In Poolcasting, any user… Read more…

Figure 1. A Poolcasting Web radio.

[1] Available at: http://www.pandora.com
[2] Available at: http://www.last.fm



Poolcasting Radio – Share your personal library

POOL|CASTING

Start poolcasting

Join the Poolcast

In order to start Poolcasting you need:

- A computer running iTunes
- The songs in iTunes to be auto-organised
- To share your iTunes music folder via HTTP (e.g., share it via Apache and name it /iTunesFolder)

Enter your shared music folder path: http://claudiob.iiia.csic.es/iTunesFolder Share

Figure 2. Sharing a personal library.

a distributed collection of songs is more dynamic: when a new user shares her library, the songs contained in her library enter the Music Pool; when she leaves, these songs are removed; in general the set of songs in the Music Pool continuously varies over time. Next, a distributed collection is more up-to-date: while an administrator would usually update the radio library once in a while (e.g., once a week), individual users add new songs to their libraries as soon as they acquire them, so these instantly enter the Music Pool. A distributed collection also contains more of the longed for "audio not available elsewhere": in fact personal libraries often include songs not publicly distributed (e.g., personal audio material, uncommon records, alternative versions of known themes). Finally, a distributed collection contains a finer selection of songs: while a centralised library is normally a massive heterogeneous collection of albums, not filtered by any quality criteria, personal libraries usually contain themes the owner has selected and appreciates.

Notice that Web radios copyright and license policies are not affected by the the fact that Poolcasting retrieves songs from personal libraries. In fact, whether a radio station pays a fee for on-air time, for number of streamed songs or for number of listeners, the fee still applies; whether the broadcast songs are copyright-free, no fee has to be paid; when the personal libraries contain DRM-protected songs, these are not broadcast.

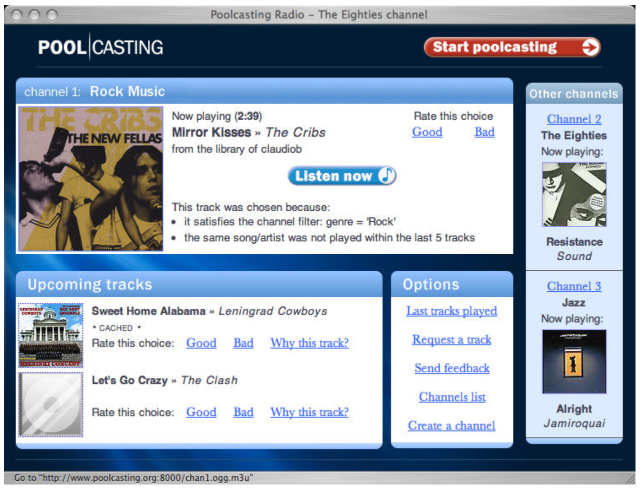
Influencing the music selection of the channels. Another advantage of having users share their personal libraries is that Poolcasting is able to *implicitly* infer the users' musical preferences by analysing *listening experience* data stored in the library index file. For instance Apple iTunes stores the play count of each song (how many times a song was played) and the explicit rating (1 to 5 stars) assigned by the user. Exploiting these data, Poolcasting builds a partial *preference model* for each active participant, under the intuitive assumption that the higher the rating assigned to a song and the higher its play count, the more the user likes that song.

The Poolcasting Web interface also allows users to *explicitly* state their preferences towards any song played or scheduled on a channel. For each channel, Poolcasting provides a channel-specific Web page (see Fig. 3) with real-time information (current track playing, history of played tracks, upcoming tracks) and links to explicitly evaluate and request songs. When a participant assigns a rating to a song, Poolcasting accordingly updates her preference model for that song. When a participant requests a track to be played on channel, Poolcasting updates her preference model as well, assigning a high preference to that track.

Preference models (made of both implicit and explicit musical preferences) are internally used by Poolcasting to play, on the channel a participant is listening to, more songs she probably likes (e.g., songs from her best-rated artists) and less songs she probably dislikes. Notice that Poolcasting does not immediately and surely play a requested song, for this would infringe most Web radios copyright policies.

Creating radio channels. Commonly Web radio provide their audience with a fixed set of channels. Some of these may have no listeners at all; still the administrators will have spent time in their creation, and money in songs acquisition, storage space and bandwidth. Large as the number of channels can be, some listeners may still be unsatisfied for they won't find *the* channel that best fits their taste.

Poolcasting introduces the ability for listeners to create channels. Every listener can browse the collection of active channels and listen to one of them, or else create a new one by clicking on the Create a new channel link and defining its name, description and *Channel Pool*, that is, the subset of songs that is allowed to play on that channel (see Fig. 4). The Channel Pool is defined in terms of metadata restrictions; for instance "Jazz from 1967" or "Electronic Dance at 135 BPM". Once a channel is created, it appears in the list of the available channels. Poolcasting automatically



Poolcasting Radio – The Eighties channel

POOL|CASTING

Start poolcasting

channel 1: Rock Music

Now playing (2:39)
Mirror Kisses » *The Cribs*
from the library of claudiob

Rate this choice: Good Bad

Listen now

This track was chosen because:
- it satisfies the channel filter: genre = 'Rock'
- the same song/artist was not played within the last 5 tracks

Upcoming tracks

Sweet Home Alabama » *Leningrad Cowboys*
• CACHED •
Rate this choice: Good Bad Why this track?

Let's Go Crazy » *The Clash*
Rate this choice: Good Bad Why this track?

Options

Last tracks played
Request a track
Send feedback
Channels list
Create a channel

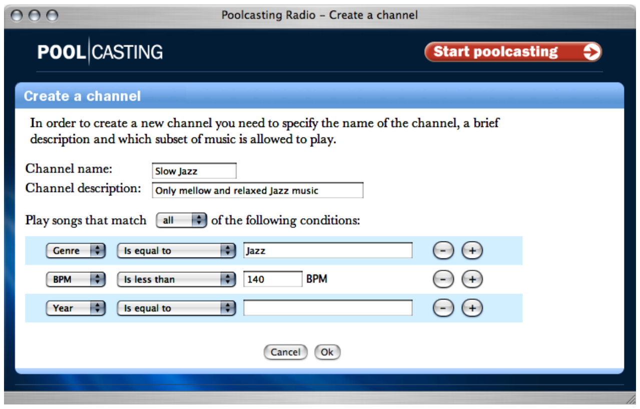
Other channels

Channel 2
The Eighties
Now playing:
Resistance
Sound

Channel 3
Jazz
Now playing:
Alright
Jamiroquai

Go to "http://www.poolcasting.org:8000/chan1.ogg.m3u"

Figure 3. Detail of a Web radio channel.


Poolcasting Radio – Create a channel

Create a channel

In order to create a new channel you need to specify the name of the channel, a brief description and which subset of music is allowed to play.

Channel name: Slow Jazz
Channel description: Only mellow and relaxed Jazz music

Play songs that match all of the following conditions:

Genre	Is equal to	Jazz
BPM	Is less than	140 BPM
Year	Is equal to	

Figure 4. Creating a Web radio channel.

provides a music stream for that channel and programmes it with music that fulfils the specified restrictions.

2.3 Examples of user interaction

We present some situations where the social features of Poolcasting improve the listening experience when compared to a common Web radio.

Example 1. Paul likes Soundtrack music (music *composed* for movies), and is disappointed for most "Soundtrack" Web radios mainly broadcast popular songs inspired by or included in movies. John enters a Poolcasting Web radio, shares his library (containing many Soundtrack themes), creates a new channel as "genre = Soundtrack" and starts listening to it. Initially, the Channel Pool only contains themes from his library; this means John will hear a sequence of his own songs, automatically selected and ordered without the need for him to manually compile a playlist. John promotes the channel on a Soundtrack-related Web forum, inviting people to join the channel and share their own themes. As the collection grows, different Soundtrack-lovers from different places will listen together to a unique stream of music, automatically ordered and generated to match their tastes. John, for instance, will listen to some songs he does not own, but all the same appreciates, mixed with other songs he already knows and likes. At one moment, John recalls a Soundtrack theme from his library that he supposes other listeners will like to hear, so he requests the theme to be played. The audience likes the proposed song and rates it positively. This implicitly increases John's reputation as a good human recommender in the context of Soundtrack music. Music labels also benefit from this architecture: to promote new soundtracks they just have to join the Poolcasting radio and share their records. Soundtrack themes will then be automatically scheduled on the appropriate channel and broadcast to the specific group of listeners that like that genre.

Example 2. Several friends own large music libraries in their offices, but do not have any portable device where to copy them on. One day they decide to set up a home party where the music played will be "Dance from the Nineties". Before leaving their offices, they all enter a Poolcasting radio, share their libraries and create a channel defined as "genre = Dance and year $>$ 1990 and year $<$ 2000". Once at the party, the host connects to the same Poolcasting radio channel. The media player will then start playing music tailored for the party, where the sequence of songs, automatically ordered and group-customised, will fairly satisfy all the participants, without the need for a human DJ.

Example 3. A German musician performs as a DJ in a night club and needs to stay continuously up-to-date with the last "Progressive Trance" hits (a subset of electronic dance music where BPM is within 125 and 135). He enters Poolcasting and creates a niche-targeted "Progressive Trance" channel. He also shares his library, which contains published tracks, copyright-free unpublished tracks produced by himself and live performances. As more people from all over the world join this channel and share their music, he suddenly gets to listen to Progressive Trance tracks that are completely unknown in Germany but might be appropriate for his upcoming performances. Meanwhile, his own tracks are broadcast on the channel, and are listened by a specific niche of public that will probably appreciate and evaluate his works better than a generic audience.

3 Poolcasting architecture

This section explains in detail the architecture of a Poolcasting radio, how the social features we have described are integrated and how a sequence of songs is customised for a group of listeners/participants.

The components of a Poolcasting Web radio are (see Fig. 5): a *Database*, which contains all the data regarding songs, channels, listeners, participants; a *Web Interface*, which allows users to open the radio streams, create channels, share libraries and evaluate songs; *Active Participants*, users who share their personal libraries using the Web Interface; the *Music Pool*, containing references to every song shared by every participant; the *Library Analyser*, a module that builds individual preference models analysing the listening experience data of the personal libraries; *Preference Models*, describing the musical preferences of each participant, either inferred from personal libraries or from users' explicit ratings; a *Musical Associations* model, containing a musical association degree for each pair of songs and for

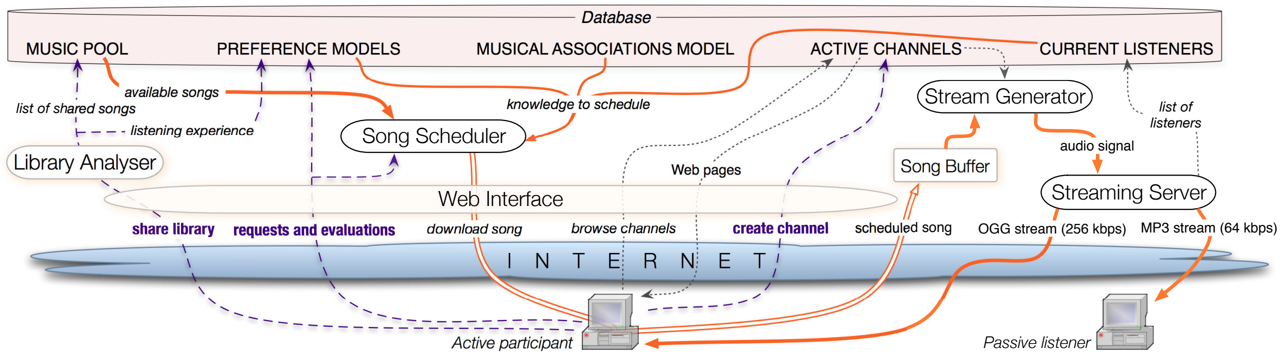


Figure 5. Architecture of Poolcasting.

each pair of artists in the Music Pool; *Channels*' references, containing for each channel its name, its description, its Channel Pool (definition of which songs can be played) and it list of current listeners; the *Song Scheduler*, a module that selects in real time which songs to play on each channel; a *Song Buffer*, where songs that are to be played are temporarily cached; the *Stream Generator*, a module that loads songs from the Song Buffer and transforms them into uncompressed audio signals; the *Streaming Server*, that converts the uncompressed audio signals to music streams and broadcast them to the listeners; *Listeners*, who employing an appropriate media player connect via the Internet to the Streaming Server to listen to their favourite channel.

3.1 Music Pool and Preference Models

When a participant shares her library (see Fig. 2), the Library Analyser retrieves and parses the index file from the user library and tries to identify each track of the library by matching its title and artist against a public large music database. In our implementation, we use the catalogue of the Web-based music community MyStrands[3] which contains over 6 million songs and offers a Web service called OpenStrands that automates the matching queries. After this process, every song from the user library is virtually included in the Music Pool and, as long as the user is connected, can be scheduled to play on any channel. No audio file is retrieved at this step.

The Library Analyser also performs another task: analysing the listening experience data included in the retrieved index file, builds a Preference Model of the user. A Preference Model describes the degree in which each participant likes each of the songs included in the Music Pool. Namely, we define for each participant P and for each song S a *preference degree* $g(P,S)$ with values in $[-1,1]$, where -1 means P hates S, 1 means P loves S and 0 represents indifference. The technique used to assess the preference degree of P for a song S in her library considers the rating P assigned to S and the play count of S in the library of P, both *normalised* according to the average listening behaviour of P. In other words, if P has given S a rating higher than (resp., lower than, equal to) the average rating assigned by P, then the *normalised rating* $n(P,S)$ of P for S is higher than (resp., lower than, equal to) 0. The same happens with the *normalised play count* $m(P,S)$ of S for P. For any non-rated song, $n(P,S) = 0$; for any unheard song, $m(P,S) = 0$. For any song S in the library of P the *preference degree* of P for S is defined as: $g(P,S) = \theta n(P,S) + (1-\theta)m(P,S)$, with $0 \leqslant \theta \leqslant 1$ (currently $\theta = 0.5$). For songs not included in the personal library of P, we define $g(P,S)$ following the assumption that if P does not own song S but owns other songs *from the same artist* of S, then her preference for S can be estimated as her average preference for those songs.

3.2 Musical Associations Model

In order to provide a satisfactory listening experience to the radio audience, we want to ensure the property that each song played on a channel is musically associated with the song it follows. This is intended to emulate the behaviour of terrestrial radios where human DJs programme sequence of songs that musically "flow" one after the other. While a human DJ knows from experience which songs are associated, Poolcasting needs an automatic technique to obtain a model of good musical associations between songs or artists.

For this purpose, we have elaborated a pattern mining technique [5] that, by analysing the co-occurrences of songs and artists in a large collection of human-built playlists, estimates a *song association degree* for each pair of songs and an *artist association degree* for each pair of artists in the Music Pool. Briefly, this technique evaluates that two songs or artists are strongly associated (or else, musically "flow" one after the other) when they occur together in many human-built playlists, few songs separate them in

[3]Available at: http://www.mystrands.com

each of these playlists and they do not occur separately (that is, only one of the two occurs) in many playlists.

Other techniques can be implemented to build a musical associations model. We chose playlists as a source of knowledge because human-built playlists include cultural information that cannot be extracted from audio signal, and also contain songs in a specific order, which can be preserved when programming a radio channel.

3.3 Group customisation and satisfaction

The Song Scheduler is responsible for programming in real time a sequence of songs for each channel. Using the Musical Associations model and the individual Preference Models, we want the Song Scheduler to arrange a sequence of songs that a) are musically associated and b) match the preferences of current audience. Since listeners of the same channel can have diverging preferences, a critical challenge is how to combine different individual preferences to satisfy the group as a whole. Different strategies can be used to face this issue. A *Plurality Voting* strategy, for instance, would compile a sequence by accumulating songs that match the individual Preference Models of the *majority* of the current listeners. Though apparently democratic, such an approach does not consider the *degree of satisfaction* of listeners over time. For instance, it could compile a sequence in which each song satisfies the *same* majority of listeners, eventually leaving a large minority of the audience totally unsatisfied with the sequence played.

To guarantee more fairness among listeners, we have developed a different strategy [4]. Rather than equally considering all the listeners' Preference Models, we assign a higher weight to the preferences of the listeners that were less satisfied with the last songs played, so that eventually every participant gets to listen to some songs she likes. For example, let $P1$, $P2$ and $P3$ be three listeners of a Poolcasting channel, let X be the song currently playing, and let Y be the next song scheduled to be played. Let the individual musical preferences for song Y be: $g(P1, Y) = 1$, $g(P2, Y) = 1$ and $g(P3, Y) = -0.5$, that is, both $P1$ and $P2$ like song Y while $P3$ quite dislikes it. Since $P1$ and $P2$ are somehow "favoured" in this selection against $P3$, the musical preferences of $P3$ will have a higher weight in determining which song to play on the channel after Y.

3.4 Song Scheduler

The Song Scheduler is a module integrated in the Poolcasting architecture that uses knowledge from both the Preference Models and the Musical Associations model to select in real time which song to play on each channel. The task of the Song Scheduler is defined as follows. Let H be a channel, let Y be the last song scheduled on channel H and let $\phi(H)$ be the Channel Pool of H, that is, the subset of songs of the Music Pool that fulfils the restrictions of channel H (e.g., "genre = Soundtrack"). Before the streaming of song Y terminates, the Song Scheduler has to schedule the next song to play on channel H after Y, where this song should be a) included in the Channel Pool $\phi(H)$, b) musically associated with Y and c) satisfactory for the current audience of channel H.

There are several ways to programme a Song Scheduler in order to achieve these properties, depending on which property is considered more relevant. As a matter of fact, the precise implementation is not critical for the Poolcasting architecture: the Song Scheduler only represents a module that takes in input knowledge about Music Pool, Preference Models and Musical Associations and returns in output the next song to schedule on each channel. For instance, a Song Scheduler could be programmed to play some group-preferred songs even if they do not strictly satisfy the Channel Pool; or else it could assign more importance to musical associations than to the Preference Models; or else it could completely ignore musical associations and be guided only by the preferences of the audience. In our current implementation we have opted for a Case-Based Reasoning (CBR) technique [4], which is in our expectations the most efficient approach for scheduling many channels in real-time, balancing the contributions of both musical associations and listeners' preferences. This CBR approach works in three consecutive steps:

- (*Retrieve Process*) when a new song has to be scheduled on a channel, the Musical Associations model described in Sect. 3.2 is used to retrieve from the Channel Pool a subset of candidate songs that are musically associated with the last scheduled song;

- (*Reuse Process*) Preference Models are used to sort the candidate songs and to pick the one that best matches the group preference of the current listeners, according to the strategy described in Sect. 3.3;

- (*Revise Process*) participants can evaluate the proposed song before it is streamed; whether user feedback is strongly negative, the song is substituted in real-time for the next best candidate of the Reuse step.

3.5 Poolcasting at works

The architecture described in this paper has been implemented and an experimental Poolcasting Web radio is currently running within the internal network of our Research Institute. The radio runs on an Intel-based Mac Pro server and was developed with open source software and standard formats: MySQL for the Database, liquidsoap for the Stream Generator, icecast for the Streaming Server, Apache for the Web Server, Ruby for both Song Scheduler and Library Analyser, Rails for the Web framework, and XHTML/CSS for the Web Interface. A private Web Interface allows the administrators to start/stop the server, list the active channels, browse the listeners' Preference Models and update the Musical Associations Model. Each channel is provided in both MP3 and OggVorbis encoding formats. Hereafter we will explain the cycle of life of a radio channel.

A Poolcasting Web radio is idle until a participant (or an administrator) creates a channel using the Web Interface. When this occurs, the Streaming Server opens a new Internet stream for the channel and waits for the Stream Generator to fill it with music. The Stream Generator, in turn, asks the Song Scheduler to pick from the participants' libraries one song to be played on the channel. The Song Scheduler checks the Channel Pool and the current audience and selects a song X, then connects via HTTP to the personal library where X is stored and downloads the file into the local Song Buffer. In turn, the Stream Generator transforms that file into an uncompressed audio signal that the Streaming Server finally broadcasts to the connected listeners. Once the channel is up and running, the Stream Generator has to continuously feed the Streaming Server with music; any delay would cause an annoying interruption in the stream broadcast to the audience. For this reason, the Stream Generator works "two songs in advance": while song X is being broadcast to the public, and the next song Y has been cached into the Song Buffer, ready to be played, the Stream Generator asks the Song Scheduler to select a song Z that will play on the channel after Y. Then, when X ends, Y is loaded to the Streaming Server, Z replaces Y in the Song Buffer, and a new song is selected by the Song Scheduler. This approach prevents delays that may occur when a personal library becomes unavailable because a user has suddenly disconnected from the system. In this way the Web radio can serve for each channel an uninterrupted music stream, without gaps between songs. The Streaming Server also maintains in the Database an updated list of the IP addresses of the listeners of each channel; channels that have not had any listener are automatically disabled after some time, in order to save bandwidth.

Our future work includes: testing listeners' satisfaction and loyalty, introducing a "reputation degree" for each participant, improving the Preference Models, allowing users to share only part of their personal libraries, introducing the use of tags to define channels and better characterise songs, implementing library sharing also for libraries not managed with Apple iTunes and deploying a comprehensive Poolcasting open source package.

4 Related Work

Previous work has addressed the generation of a sequence of music to maximise the satisfaction of a group of listeners. MusicFX [10] is a system that works in a gym centre to select, from an existing collection of radio stations, the one that maximises the "mean happiness" of the public. For this purpose, users have to *manually* indicate their preferred genres, while Poolcasting *infers* Preference Models from the listeners' personal libraries. AdaptiveRadio [8] is a group-based Web radio where listeners can show discontent for a song, so that no other song from the same album will be played. Thus, user interaction is limited to veto song selections while in Poolcasting songs can also be promoted. Moreover, Poolcasting defines musical associations not just for songs from the same album, but exploiting co-occurrence analysis of a large collection of playlists. Flycasting [9] is an online radio system where channels are group-personalised based on users' explicit requests, while Poolcasting creates a customised and ordered musical sequence combining users' requests, evaluations, implicit preferences and musical associations to match the overall satisfaction.

Other network architectures have been proposed to provide personalised music listening experiences following a distributed approach. r-MUSIC [13] describes a (local) wireless network architecture where people share music from their handheld devices and vote which songs should be streamed from a centralised speaker; however Poolcasting proposes a (global) Internet radio architecture, in which individual preferences are not obtained with a voting mechanism but combining explicit and implicit listeners' musical preferences. StreamOnTheFly [1] is a peer-to-peer network for local radios to publish their audio material on the Internet, where users find shared radio programs either browsing the content archives or constructing complex queries. The novelty of Poolcasting is that participants do not only share audio content, but also their listening experience; in fact each Poolcasting channel virtually gathers people with similar tastes that listen at once to the same music content. AXMEDIS [6] proposes a multi-channel architecture with a peer-to-peer platform to reduce costs for content production and distribution in the digital-content market. Poolcasting also addresses the issue of reducing the costs of music distribution on the Internet, and proposes a Web radio architecture that both preserves protection policies for musical content and increases user accessibility to online music.

5 Conclusions

Nearly one-fourth of frequent online music users say the ability to share music with others in some fashion is an important criteria when selecting an online music service [11]. Exchanging mix-tapes used to occur with friends in social environments, but with online music sharing much of this sociality is stripped away [7]. Peer-to-peer networks have made exchanging music files very easy, though a small effort has been devoted to help people share their listening experiences; that is, to provide a framework where people with similar tastes can virtually gather together to listen at once to the music they like. Current Web radios ignore the fact that people are more and more eager to collaborate on the Internet, and offer free music streams which are not customised to the listeners' preferences. In this paper we have presented Poolcasting, a social Web radio architecture where a) users can create channels that best match their tastes, b) user preferences are inferred from the listening experience data contained in their personal libraries and c) users can actively participate in the music scheduling process by sharing their personal libraries and by requesting or evaluating the scheduled songs.

Poolcasting represents an innovative and powerful distribution medium that mediates between music providers and music consumers. Any musician or music label, by registering to Poolcasting, can submit to the *Music Pool* her produced music, that is later distributed along the different channels. Since the content of each channel is defined by the users themselves, target audiences naturally emerge from this consumer-driven process. Poolcasting helps producers increase *exposure* to the public, while customised channels prevent consumers from "information overload", that is, being exposed to an immense, unfiltered, unorganised and unmanageable amount of songs. Poolcasting enables a streamlined process that facilitates the matchmaking between music producers and communities of similar listeners, organised around group-customised channels. In the future, we plan to expand the social interaction among users with the possibility to share not only listening experiences, but also profiles, opinions and other group-related activities.

In terms of copyright issues, any public Poolcasting radio should pay the same license fees applied to current Web radios to stream over the Internet. These licenses change from country to country; for instance, SGAE manages music digital rights in Spain, where in 2007 the license fee for a small non-commercial streaming-only Web radio has a monthly cost of 53.25 euros[4].

6 Acknowledgements

This research is partially supported by the MID-CBR (TIN2006-15140-C03-01) project and by a MyStrands scholarship.

[4]More details on SGAE fees at: http://www.sgae.es

References

[1] R. Alton-Scheidl, A. Micsik, M. Pataki, W. Reutz, J. Schmidt and T. Thurner. StreamOnTheFly: A Peer-to-Peer Network for Radio Stations and podCasters. In: *Proc. of the AXMEDIS '05 Conference*, 2005.

[2] Arbitron-Edison Media Research. Reasons Why US Consumers Listen to Web Radio. May 2005.

[3] Arbitron-Edison Media Research. Web Radio Listening Up 50%. April 2006.

[4] C. Baccigalupo and E. Plaza. A Case-Based Song Scheduler for Group Customised Radio. In: *Proc. of the ICCBR 2007 Conference*, LNCS (LNAI), vol. 4626, 2007.

[5] C. Baccigalupo and E. Plaza. Mining Music Social Networks for Automating Social Music Services. In: *Workshop Notes of the ECML/PKDD 2007 Workshop on Web Mining 2.0*, 2007.

[6] P. Bellini and P. Nesi. An Architecture of Automatic Production of Cross Media Content for Multi-channel Distribution. In: *Proc. of the AXMEDIS '05 Conference*, 2005.

[7] B. Brown, A. Sellen and E. Geelhoed. Music Sharing as a computer supported collaborative application. In: *Proc. of the ECSCW '01 Conference*, 2001.

[8] D. L. Chao, J. Balthrop and S. Forrest. Adaptive Radio: Achieving Consensus Using Negative Preferences. In: *Proc. of the GROUP '05 Conference*, 2005.

[9] D. B. Hauver and J. C. French. Flycasting: Using Collaborative Filtering to Generate a Playlist for Online Radio. In: *Proc. of the WEDELMUSIC Conference*, 2001.

[10] J. F. McCarthy and T.D. Anagnost. MusicFX: An Arbiter of Group Preferences for Computer Supported Collaborative Workouts. In: *Proc. of the CSCW '98 Conference*, 1998.

[11] M. McGuire and D. Slater. Consumer Taste Sharing Is Driving the Online Music Business and Democratizing Culture. 2005.

[12] The NPD Group, Inc. US Consumers 13+ Music Listening. May 2005.

[13] U. Wolz, M. Massimi and E. Tarn, r-MUSIC, A Collaborative Music DJ for Ad Hoc Networks. In: *Proc. of the WEDELMUSIC Conference*, 2004.