Peer to Peer Systems and Applications
A peer-to-peer based online radio

Genc Mazlami, Luca Longinotti, Robert Richter, Daniele De Felice

May 31, 2012

Supervisor: Christos Tsiaras

Abstract

As the percentage of fully distributed systems in today's networks is growing continously, it is of very big interest to deepen the research in the field of peer-to-peer-systems. This paper presents a possible architectural, algorithmic and programmatic approach to design an Online Radio based on P2P infrastructure (e.g. Distributed hash table, tracker etc.) with an integrated recommender system choosing the songs according to the user's taste.

Contents

1 Introduction and theory **5**
1.1 Goals . . . 5

2 Planning **6**
2.1 Requirements . . . 6
2.1.1 Functional requirements . . . 6
2.1.2 Technical requirements . . . 6
2.1.3 Other requirements . . . 6
2.2 Use cases . . . 7
2.2.1 [UC] Join P2P network . . . 7
2.2.2 [UC] Create playlist . . . 7
2.2.3 [UC] Play playlist . . . 8
2.2.4 [UC] Step to next song in playlist . . . 8
2.2.5 [UC] Stop playing playlist . . . 8

3 Software architecture **10**
3.1 Architecture Overview . . . 10
3.1.1 User interface layer . . . 10
3.1.2 Application layer . . . 11
3.1.3 Domain layer . . . 12
3.1.4 Infrastructure layer . . . 12
3.2 Libraries . . . 12
3.2.1 P2P library . . . 12
3.2.2 MP3 Library . . . 12
3.2.3 ID3 Tag Library . . . 12

4 Recommendation system design **13**
4.1 Overview . . . 13
4.2 Replication and availability . . . 13

5 Peer-to-peer usage **15**

1 Introduction and theory

1.1 Goals

The goal of the project is to design and implement a online radio application based on peer-to-peer functionality. The system will be similar to a file sharing application in the sense that users will be able to share their own music to the platform.The user will give a starting point for the applicatoin by searching for an initial song, on which the system can base its further recommendations. Beside that, no other user interaction will be needed, since the software will act like a conventional radio in the sense that the application itself will choose songs to be played.
The choosing of the songs will be performed by a recommendation system working together with the DHT structure and executing algorithms to guess the user's music taste. The songs will be streamed / loaded directly from the other peers, as it is done in a conventional P2P system.

2 Planning

2.1 Requirements

2.1.1 Functional requirements

- The system must provide basic music player features: Play an music-file, stop playing, step to next song
- The system must be able to categorise songs in the P2P network (genre, style, artist) for example by reading ID3-tags
- The system must be able to find songs by means of given search terms
- The system must be able to create playlists based on the given user input
- The system must provide a user interface to allow operation of the software
- The user must be able to get a playlist by means of entered keywords
- The user must be able to add songs to the network
- The user interface must provide a function to visualize the underlying P2P mechanisms
- The user should be able to build blacklists

2.1.2 Technical requirements

- The system must be robust to some peers leaving the system
- The solution shall be based on pure P2P mechanisms, that is, the solution shall not contain any central elements (trackers or servers). Every peer will run the same software and have the same roles.
- The software should load the next songs in the background in order to avoid delays
- A configurable number of played songs should be stored one the users local system temporally, to increase their availability
- Since one of the functional requirements is the visualization of the underlying P2P mechanisms, there should be some monitoring callback interface (observer pattern).
- There should be some index for searching songs on the local disc by means of keywords

2.1.3 Other requirements

- The solution may use existing libraries and code, but those must be allowed to be published under GPL or another comparable open software license.

2.2 Use cases

2.2.1 [UC] Join P2P network

- **Actors:**
  - User A
  - User B
  - System
- **Precondition:**
  - Address/ID of bootstrap node is known (e.g. User B).
- **Description:**
  - User A knows the address of User B. User A enters User B´s address into the System.The System tries to establish a connect to User B. If a connect has been established successfully, the DHT is updated.
- **Postcondition:**
  - DHT contains ID of new joined User
  - Connection between User A and B is available

2.2.2 [UC] Create playlist

- **Authors:**
  - User A
  - Users B..X
  - System
- **Precondition:**
  - User A is connected to the P2P network
- **Description:**
  - User A enters a keyword into the System.
  - The System is looking for corresponding songs on the users local disk, as well as in the P2P network. To offer the user a high variety, songs from the P2P-network should be favored over local songs.
  - If there are corresponding songs in the network, the first one is selected. The System establishes a connection between User A and the source of the song - i.e. User B.
  - After the first song is loaded, the System continues to load corresponding songs in the background, until a defined number of songs is loaded.
- **Postcondition:**
  - Playlist with similar songs is created

2.2.3 [UC] Play playlist

- **Authors:**
  - User A
- **Precondition:**
  - Songs are loaded from the network to the users local machine
  - Playlist is created out of loaded songs
  - OR: existing playlist was stopped or paused
- **Description:**
  - User A triggers play procedure on the playlist
  - If this is a new play, or a play after a stop [UC 3.4] the system starts playing a random song of the playlist
  - If it is a play from a pause, the playlist is played from the position where it was paused
  - When a song was completely played, the application chooses another one (randomly) till the complete playlist is played
- **Postcondition:**
  - Songs in the playlist are being played

2.2.4 [UC] Step to next song in playlist

- **Authors:**
  - User A
- **Precondition:**
  - Playing of the songs is in progress
- **Description:**
  - The current song in play is skipped.
  - The next song of the playlist starts playing immediately
- **Postcondition:**
  - Next song in playlist is being played

2.2.5 [UC] Stop playing playlist

- **Authors:**
  - User A
- **Precondition:**
  - Playing of the songs is in progress

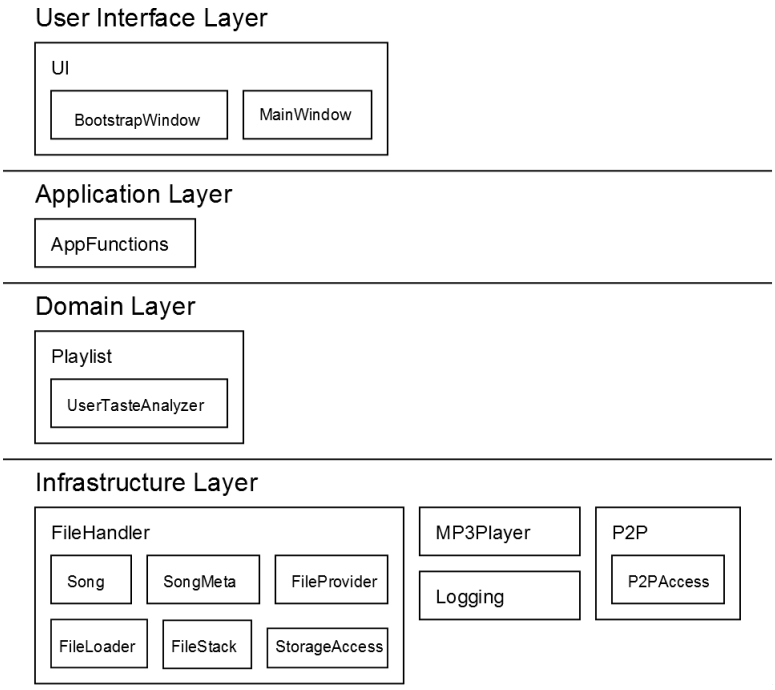
- **Description:**
  - The User triggers the stop procedure, which stops the playing of the playlist and sets the playing position to the beginning again
  - The stopped playlist can not be continued at the place where it was stopped.
- **Postcondition:**
  - No song is being played.

3 Software architecture

3.1 Architecture Overview

Being inspired by the design of todays network infrastructures, which give a beautiful layered system architecture, the P2P online radio was designed using a layered approach too. The idea was to reduce or minimize the collaboration complexity of our modules, by defining layers - each having its own tasks - and describing interfaces betweend the layers.
The system is constructed out of four layers:

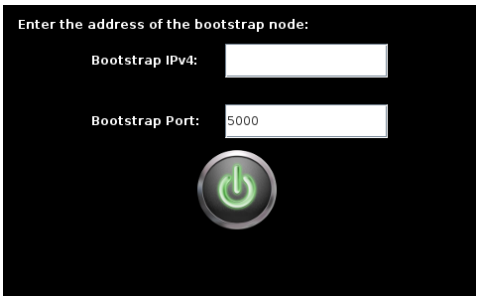
- User interface layer
- Application layer
- Domain layer
- Infrastructure layer



3.1.1 User interface layer

The user interface layer provides - as its name tells you - the graphical front end for user interaction. Each one of the listed use cases for the application can be started by the user using the components of this layer. It consists mainly of the BootstrapWindow and the MainWindow.

BootstrapWindow:



MainWindow:

The BootstrapWindow serves as a starting point for the network. The user enters an identification of a known peer (e.g. an IPv4 address and a port) to which he wants to bootstrap. On successful bootstrap, the MainWindow appears, which allows the user to share its own songs to the network or initialize the internet radio by searching for a keyword.

3.1.2 Application layer

The application layer shall not contain any underlying infrastructure logic (such as connection services, device storage services etc.). It serves as a mediator layer

that mapps UI interaction to calls of underlying mechanisms.The AppFunctions module handles these tasks by calling the respective methods (like nextSong(), connect()) from the underlying layers. The only effective functionality this layer provides is the playing of instances of the type Song, which are defined in layers below. It is achieved by the module MP3Player which serves as a main sound playing engine for the whole client. It supports the conventional music playing functionality like play(Song), stop(). Further than that, it has to ability to recognize how long the current song was played, and register it into the songs attributes. These information will then be used by lower layers to decide if the played song was or was not lying in the user's taste.

3.1.3 Domain layer

The domain layer contains all functionality related to the creation of playlists, since this is the central domain of the project. It achieves its goals through combination and calling of infrastructure layer functionality. It also provides mechanisms to analyze the user's behaviour with respect to the played songs, and thus gather usefule information for the recommendation system.

3.1.4 Infrastructure layer

All fundamental and technical functions (like IO transactions, network connectivity, bootstrapping, file transfer and streaming) are provided by the infrastructure layer. It provides its services through usage of system calls, native language calls and calls to the external libraries (such as TomP2P).

3.2 Libraries

3.2.1 P2P library

At the beginning of the porject an important decision was the choice of an appropriate library supporting peer-to-peer functionality, DHT functions, redundancy,recovering from peer failure etc. Some of the candidate libraries were: OpenKad,JDHT, Bamboo-DHT, OpenChord and TomP2P. While Bamboo-DHT and OpenChord have quite old releases and are not fully supported anymore, JDHT is maybe too simple for the purpose of this project and OpenKad is more oriented towards key-based routing. As a conclusion of the evaluation, the most appropriate library to use was TomP2P.

3.2.2 MP3 Library

Since the radio does not need very sophisticated playing functions - it will act like a radio, e.g. no fast forward or fast backward functionality, no pausing functionality - only a very basic library supporting simple MP3 decoding and playing functionality was needed. We found the Javazoom JLayer-MP3 Library to be such a library.

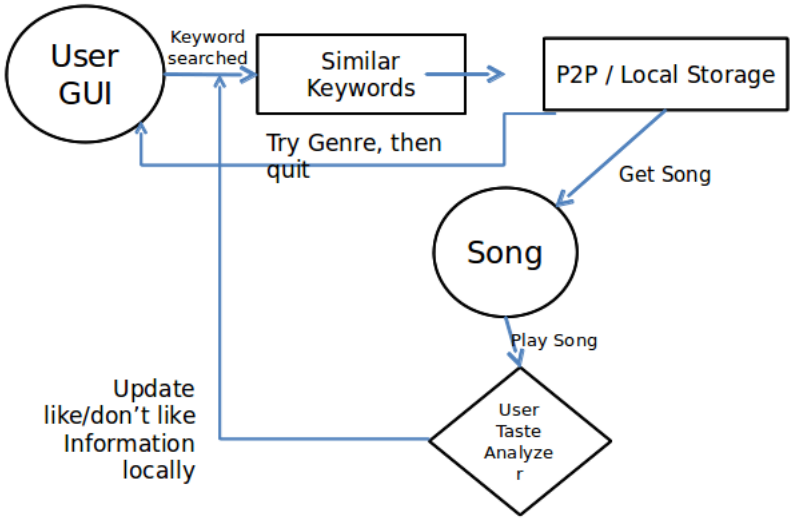
3.2.3 ID3 Tag Library

Since it is one of the most complete and easy to use tagging libraries, we decided to use JAudioTagger as a Library to deal with the reading and the management of MP3-audio- tags.

4 Recommendation system design

4.1 Overview

We decided to stick with a keyword based approach to generate good recommendation for the users.In our case the DHT will have keywords as keys and peer-addresses as respective values. A peer holding a song associated with a keyword will be put into the distributed hash table with the respective keyword as key.



A user initiates the recommendation system by entering an initial keyword to search songs for. The system first gets similar keywords to the one entered and then performs a lookup-procedure by getting the addresses of peers holding songs with the given keyword. If no peer has shared a song associated with the given keyword, the system checks if any such song is already available locally, if not, it retries the above with the genre as keyword and if even that gives no matches it stops and notifies the user that nothing corresponding was found.
When a song is played, the UserTasteAnalyzer module of the domain layer comes into account. It checks whether the song has been played for a predefined amount of time. If so, the system interprets this evidence as a 'likeliness relation' between the played song and the given keyword. Thus, the songs keywords are added to 'likeliness-list' of the given keyword. If on the other hand the user quickly changed the song, this song's title is added to an 'unlikeliness-list' to prevent it from being ever played again in connection with the current given keyword. It is of high importance to notice that the relation information for each keyword remains stored locally, as it is not a global property of a song to be shared, but rather a personal decision of each user.

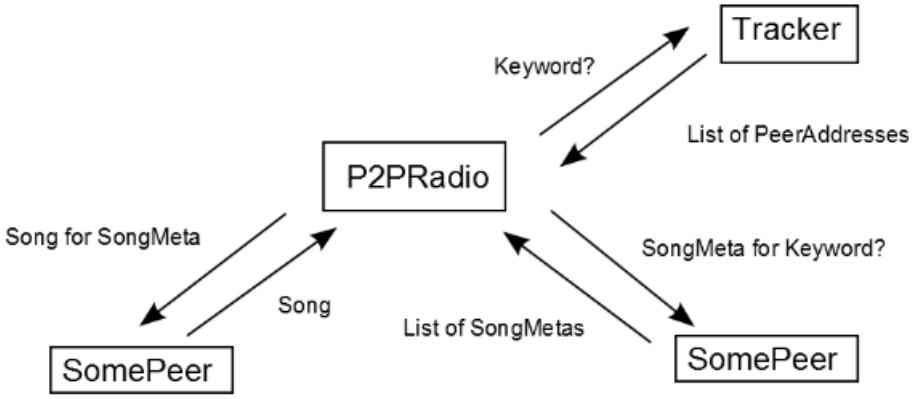
4.2 Replication and availability

Replication of the data on the DHT is automatically achieved by TomP2P.
Further, replication of the individual songs is assured by having each peer that

is interested in a particular song have its own local copy of the song, which it then announces on the DHT too.

5 Peer-to-peer usage

In our application each peer announces to the DHT the songs he has available locally, by putting his IP-address as a value with the song's keywords (e.g. artist, title, genre) as keys. This is also done upon downloading songs from the peer-to-peer network.



When searching for new songs, our system asks the DHT for a specific keyword. It gets back a list of IP-addresses of peers that have content matching that keyword. It then randomly selects one of those peers and asks for metadata about all the songs the peer possess that do match the given keyword. We then get the song data itself for all songs matching the metadata on the remote peer.