Radiommender: P2P On-line Radio with a Distributed Recommender System

Fabio V. Hecht, Thomas Bocek, Nicolas Bär, Robert Erdin, Beat Kuster, Marium Zeeshan, Burkhard Stiller
University of Zurich, Department of Informatics (IFI), Zurich, Switzerland
Email: {hecht,bocek,stiller}@ifi.uzh.ch, {nicolas.baer,robert.erdin,beat.kuster,marium.zeeshan}@uzh.ch

***Abstract*—Radiommender is a fully-distributed, peer-to-peer on-line radio. Users can share their music collection and explore music collections of other users. The difference to current file sharing systems is that songs do not need to be searched individually – a distributed recommender system is used to estimate user preference. Recommendation is done with a combination of an implicit voting system that assigns songs to search terms and an affinity network that correlates user interest. The demonstration shows the software functionality and includes the visualization of affinity graphs built by the recommender system.**

I. Introduction

Peer-to-peer (P2P) architectures for media distribution attract a large user base, saving bandwidth costs and profiting from increased scalability in, e.g., BitTorrent [1] and eDonkey [2] networks. These systems require that users search for media files, download them, and play them later. Online radio systems, such as Spotify [3], work differently by performing download and playback of songs with minimal user intervention, used at parties, while driving, or working. Spotify, however, is a client/server application that uses a P2P architecture to offload the central server by downloading songs from other peers whenever possible.

This paper introduces Radiommender, an Internet radio system that benefits from a fully-distributed P2P architecture. Similarly to a file-sharing application, users share music files; however, no user interaction is required while the radio is playing. In order to start playback, users must only input a *search term* T (e.g., genre or artist), based on which a play list is populated with highly relevant songs by a distributed recommender system. During playback, interaction between the user and the system is minimal – only skip and stop buttons are available, besides performing a new search. The main challenge is on predicting the user preference with a high probability in a fully-distributed environment.

II. Related Work

P2P on-line radios that perform application-layer multicast for live audio streaming exist since several years, but the idea of a distributed recommender system that builds a play list out of existing songs is novel. The demonstration developed is based on a collection of works on P2P distributed recommender systems, as described in Section IV.

III. Radiommender Design

Figure 1 displays the high-level architecture of Radiommender. The Player and Controls components play back audio and interact with the user, respectively. The Play List component keeps and selects songs to be played, by using the two recommender systems, SearchTerm and Affinity Network, which are described in Section IV.

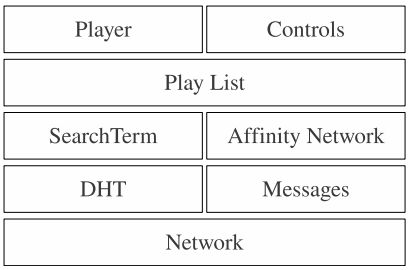

Fig. 1. Radiommender high-level architecture

In order to benefit from fully-distributed properties, Radiommender uses the TomP2P [4] as both a DHT substrate, and to send direct messages between peers. TomP2P is an advanced DHT and tracker that allows multiple values to be stored under a single key, and is able to return a subset of values with specified criteria, order, and size.

The Radiommmender uses a distributed tracker-based architecture to keep track which peer has which song. A song s is associated with a 20-byte $songID$ and is defined as a quadruple $(genre, artist, album, title)$. A song that a user chooses to share is added to the tracker using $songID$ as key and its peer address as value. In order to download a song, a peer first searches for the tracker using the $songID$, then requests a list of peers that can provide the song.

IV. Distributed Recommender System

Designing a distributed recommender system imposes inherent challenges compared to a centrally hosted system, such as lack of global state, limited possibility to pre-compute information and possibility of bogus meta information. Thus, a combination of two recommender systems – SearchTerm and Affinity Network – is proposed. While SearchTerm searches for songs with a high probability of being related to the search term based on user-ratings, the Affinity Network explores and suggests new songs based on similar music taste of users, independently of the search term entered by the user.

A. SearchTerm

The SearchTerm recommender system is based on a combination of item-based collaborative filtering [5] and content-

based recommender systems [6], relying on user votes stored in the DHT. Every song recommended (i.e. played) receives an implicit vote by the user – if the song is played through, it is considered that the song is related to the search term (*like*); if the user presses the skip button, the vote is cast against the relationship (*dislike*). Thus, search accuracy is trained by vote information from users without relying entirely on meta data. Explicit voting and rating were considered as options, but minimal user interaction during playback was favored.

The rating function R for song s and given search term T is defined in 1, where l_s^T and d_s^T represent, respectively, the total number of likes and dislikes of the song s under T for songs that have at least one vote.

$$R(T,s) = \frac{l_s^T}{l_s^T + d_s^T} \in [0,1]; l_s^T, d_s^T \in \mathbb{N} \quad (1)$$

To avoid a song being rated 1 after one vote only, a threshold τ (currently set to 5) is defined on the minimum number of votes necessary for the song to receive its full rate. Up to the threshold, the maximum rating of a song is 0.5, which still allows it to be discovered and further rated. The values of l_s^T and d_s^T are added to the DHT under different keys $k(T_s, like) \rightarrow l_s^T$ and $k(T_s, dislike) \rightarrow d_s^T$.

Playback is always started by a user search. The search term T is looked up on the DHT under the key $k(T) = hash(T)$. The peer responsible for storing $k(T)$ returns a set of recommendations (songs) $s \in S_T$ in decreasing order of rating $R(T,s)$. For each recommended song s, the search is extended by looking up its artist $artist(s)$ and genre $genre(s)$. These are used as search terms to obtain an additional song set S'_T, which allows the possibility of other artists from the same genre to be recommended. Ratings for songs in S'_T are multiplied by the rating of the original song (e.g. $R(T,s') = R(T,s) \times R(artist(s), s')$). This broadens the number of related songs to be considered for playback. Recommended songs obtained using the algorithm described above are added to the play list with probability defined by their rating. SearchTerm stops when there are enough good recommendations (currently 50 songs with $R > .95$).

B. Affinity Network

SearchTerm only recommends songs that have received ratings. New songs, though, need to be played occasionally in order to receive ratings. Since a random search for songs may perform badly, and relying solely on meta data would be problematic for songs with incomplete or incorrect meta data, the Affinity Network, inspired by [7], is defined to provide a heuristic for random elements to train SearchTerm.

The Affinity Network is a weighted undirected graph, where nodes consist of a set of users $U = \{u_1, u_2, \ldots, u_n\}$, and edges are calculated based on the affinity function. Considering S_i the set of songs shared by user u_i, the affinity function is defined in 2.

$$A(u_i, u_j) = \frac{\Sigma_{s_i \in S_i} \Sigma_{s_j \in S_j} m(s_i, s_j)}{4 \times |S_j|} \in [0,1] \quad (2)$$

TABLE I
SONG SIMILARITY

Match	*Similarity* $m(s_i, s_j)$
genre, artist, album, song	4
genre, artist, album	3
genre, artist	2
genre	1
none	0

The song similarity function $m(s_i, s_j) \in \mathbb{R}^+$ is defined by using a method loosely inspired by the Borda count [8], an election method under which each candidate is ranked by voters and assigned an amount of points inversely proportional to its ranking. Each candidate $c \in C$ is assigned a weight $w = |C| - k$, where k is its ranking. In the scenario considered, the degrees of matching between songs in S_i and S_j are used as candidates, as in Table I.

Counting Bloom filters [9] are used to reduce meta data traffic between peers. Four filters are sufficient to perform the calculation in 2, one for each non-zero value of $m(s_i, s_j)$ in Table I.

Initially, a peer calculates its affinity to a random set $n \in N$ of other peers obtained from the DHT. The peer with highest affinity is selected and its affinity to other peers M_n is requested. The affinity to peers in $m \in M_n$ is estimated transitively by multiplying the affinity of n with m, since calculating the affinity to another peer is a costly ($O(i \times j)$) operation. Peers with higher affinity values are more likely to be asked to recommend a random song.

V. Demonstration Scenario

The demonstration scenario is composed of several peers running the Radiommender software and sharing short copyright-free songs. Users can search and play songs and see how the recommender system works. Affinity graphs and SearchTerm logic are displayed in large screens to provide a visualization of the distributed recommender system used.

References

[1] "BitTorrent," http://www.bittorrent.com/, last visited: June 2012.

[2] O. Heckmann, A. Bock, A. Mauthe, and R. Steinmetz, "The eDonkey File-Sharing Network," *Proceedings of the Workshop on Algorithms and Protocols for Efficient Peer-to-Peer Applications*, vol. 51, pp. 2–6, 2004.

[3] G. Kreitz and F. Niemela, "Spotify – Large Scale, Low Latency, P2P Music-on-Demand Streaming," in *2010 IEEE Tenth International Conference onPeer-to-Peer Computing (P2P)*, aug 2010, pp. 1–10.

[4] "TomP2P, a P2P-Based Key-Value Pair Storage Library," http://tomp2p.net/, last visited: June 2012.

[5] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Item-based Collaborative Filtering Recommendation Algorithms," in *10th Intl. Conference on World Wide Web*. New York, NY, USA: ACM, 2001, pp. 285–295.

[6] M. J. Pazzani and D. Billsus, "Content-based Recommendation Systems," in *The Adaptive Web: Methods and Strategies of Web Personalization*, ser. LNCS, vol. 4321. Springer, 2007, pp. 325–341.

[7] G. Ruffo and R. Schifanella, "A Peer-to-peer Recommender System Based on Spontaneous Affinities," *ACM Trans. Internet Technol.*, vol. 9, no. 1, pp. 4:1–4:34, Feb. 2009.

[8] D. Black, "Partial Justification of the Borda Count," *Public Choice*, vol. 28, pp. 1–15, 1976, 10.1007/BF01718454.

[9] L. Fan, P. Cao, J. Almeida, and A. Z. Broder, "Summary cache: a scalable wide-area web cache sharing protocol," *SIGCOMM Comput. Commun. Rev.*, vol. 28, no. 4, pp. 254–265, Oct. 1998.