# CONTENT-BASED RETRIEVAL OF MUSIC IN SCALABLE PEER-TO-PEER NETWORKS

*Jun Gao*[1], *George Tzanetakis*[1], *Peter Steenkiste*[1,2]

[1]School of Computer Science [2]Department of Electrical and Computer Engineering
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA 15213-3891

**ABSTRACT**

A large portion of data exchanged in today's Peer-to-Peer (P2P) networks consists of music stored as MP3 compressed audio. Existing P2P systems typically are not scalable and only support primitive methods for the searching of music files, e.g., by looking up exact filenames or using simple metadata information such as artist or album name. In this paper, we present the design and evaluation of a scalable P2P system that uses Rendezvous Points (RPs) for music file registration and query resolution, and supports content-based Music Information Retrieval (MIR) of audio signals.

## 1. INTRODUCTION

A substantial percentage of the Internet traffic today consists of music files exchanged by Internet users over P2P networks. In such a system, each peer may contribute to the music collection by making a set of files on its local machine available to others, and the P2P protocol allows a peer to discover files stored on other peers. Unlike the traditional client-server based system, the decentralized and self organizing nature of P2P networks makes them a more suitable and powerful platform for resource sharing. However, the usefulness of existing P2P systems is often limited by their scalability and searching capability. For example, they only provide primitive searching methods, such as keyword searching, or searching based on simple metadata.

Centralized P2P systems such as Napster are not robust and may be vulnerable to Denial-of-Service attacks, since the central server forms the system's single point of failure. Such a system does not scale well as registration and query load increase. Distributed P2P systems, such as Gnutella and KaZaA, are more robust, but since peers do not explicitly register their shared files, a query may have to be broadcast throughout the network to get resolved. The potentially large number of messages involved for each query limits the system's scalability. Distributed Hash Table (DHT) [1, 2] based systems achieve good scalability by deploying a structured overlay P2P network. However, the basic set of applications built on top of DHT only supports exact file name look up and does not allow searching.

In addition to scalability, users of a P2P system desire more powerful searching capabilities such as searching based on the music content itself. For example, one may want to search for the album that contains an unknown song recorded from radio, and may want to find more songs that are "similar" to that one in terms of *tempo*. Recent advances in Music Information Retrieval (MIR) have enabled the analysis, indexing and retrieval of audio files based on musical content. Example work in MIR includes automatic audio classification [3] and beat detection [4]. By combining signal processing and machine learning algorithms, sophisticated models for audio retrieval and indexing can be built [5]. For example, [6] describes an automatic musical genre classification based on features automatically extracted from audio signals. [7] proposes the idea of performing MIR over P2P networks. However, their system only supports searching of symbolic data (MIDI) and not audio, and their P2P network architectures are either centralized or broadcast based.

In this paper, we present the design and evaluation of a scalable P2P system that supports content-based retrieval of music files as well as the traditional attribute-value based search using simple metadata. We ensure system scalability by employing a Rendezvous Points based architecture on top of DHT-based overlay network for music file registration and query resolution.

## 2. SYSTEM OVERVIEW

Figure 1 shows the software architecture on each peer node. A user may perform two types of operations: registering shared files and initiating searches. For registration, a shared audio file is represented with a Music File Description (MFD), which consists of a set of attribute value pairs (AV-pairs). The MFDs are either specified by the user or automatically generated by the Music Feature Extraction Engine (MFEE). We explain the MFEE component in more detail in Section 3. The criteria of a search is formulated as a query, which is also in the form of an MFD.

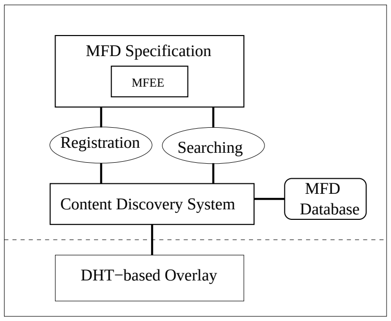

**Fig. 1**. Software architecture on a peer node.

The registration or query MFD is then passed to the Content Discovery System (CDS), which runs on top of a Distributed Hash Table (DHT) based P2P systems, such as Chord [1]. In a DHT, each peer is responsible for a region, represented with a node ID, in a contiguous $m$-bit virtual address space. A data item such as a file name, is associated with some value in this address space, e.g., by applying a uniform hash function to the data item, and stored on the peer whose region covers the value. Correspondingly, by applying the same computation to the data item, a peer can locate it from the same peer who stores it. We present the algorithm used by the CDS to distribute MFDs to peers in Section 4. The underlying mechanism of DHT ensures routing and message forwarding efficiency in such a system: in Chord, a peer only needs to keep information about $O(\log N_c)$ neighboring peers, and the number of overlay hops between two peers is $O(\log N_c)$, where $N_c$ is the total number of peers in the system.

Each peer maintains a local MFD database to store the MFDs it receives. When receives a query, a peer examines its database and returns the set of MFDs that match the query to the query initiator. The matched MFDs may contain sufficient information that the query initiator is looking for, e.g., the song name of an unknown piece of music. To further retrieve the actual music file, the query initiator may connect to the peer who owns the song for downloading.

## 3. MUSIC FEATURE EXTRACTION

The MFEE component takes as input an audio file in compressed format, such as MP3, the MPEG audio compression standard, and outputs a feature vector, also known as the content-based vector, of AV-pairs that characterizes the particular musical content of the file. In our system, we use the feature set proposed in [6] for the purpose of musical genre classification. This feature vector captures aspects of instrumentation and sound texture (what instruments are playing and their density distribution over time), rhythm (fast-slow, strong-weak), and pitch content (harmony) and has been shown to be an effective representation for the purposes of classification and retrieval of music. Examples of such features include *tempo, beat strength* and *degree of harmonic change*. The different types of information represented by the feature vector combined with the query flexibility of the system supports a rich variety of queries. For example, a user can search only on the basis of rhythmic content while ignoring other similarity aspects.

We use a standard linear quantization and normalization to transform the dynamic ranges of the continuous features into discrete values necessary for searching based on AV-pairs. Linear quantization was chosen so that the statistics of the distribution of the features do not change. In our system, each feature is quantized to 100 discrete values. Experiments comparing automatic classification of the original features and the quantized features showed no significant differences in the results.

The results of the MFEE component together with manually annotated metadata such as artist and album name are combined to form a Music File Description (MFD), which is a collection of AV-pairs. As an example, $MFD_1 : \{a_1 = v_1, ..., a_n = v_n\}$ consists of $n$ AV-pairs, where $a_i, i = 1..n$ can be either a manually annotated attribute or a content-based feature attribute. For query, MFDs are similarly formed to represent the search criteria. In particular, the MFEE is used to generate a query MFD when the user provides a sample piece of music. Using named AV-pairs in MFDs allows more powerful queries than keyword searching. For example, depending on a user's need, rather than specifying $\{$*U2, Zooropa*$\}$, the user can search for $\{$*artist = U2, albumName = Zooropa*$\}$ or $\{$*artist = U2, songName = Zooropa*$\}$.

## 4. SCALABLE CONTENT DISCOVERY

Unlike centralized systems where files are registered at a single place or broadcast-based systems where a query may potentially be sent to all peers in the system, the CDS in our system takes on a scalable approach where it uses Rendezvous Points (RPs) for registration and query resolution.

### 4.1. MFD registration

To register an MFD such as $MFD_1$ shown above, the CDS applies a uniform hash function $\mathcal{H}$ such as SHA-1 to each AV-pair in $MFD_1$ to obtain $n$ node IDs: $\mathcal{H}(a_i = v_i) \rightarrow N_i, i = 1..n$, where $N_i$ is the ID of a peer in the system. The MFD is then sent to each of these peers, and this set of peers is known as the Rendezvous Points(RP) set for this MFD. Upon receiving an MFD, the peer inserts it into its database. Hence each peer is responsible for the AV-pairs that are mapped onto it. For example, node $N_1$ will receive all MFDs that contain $\{a_1 = v_1\}$.

Since the number of AV-pairs in an MFD is typically small (e.g., $< 50$), the size of the RP set for an MFD is small and registrations can be done efficiently. Different MFDs will have different corresponding RP sets, which naturally separates the system's registration load. Registering each AV-pair of an MFD individually allows the MFD to be searched using any subset of its AV-pairs.

### 4.2. MFD searching

We classify searches conducted by a user into two categories: exact searches and similarity searches. In an exact search, the user is looking for MFDs that match all the AV-pairs specified in the query simultaneously, and any extra AV-pairs that may be in the MFDs but not in the query are ignored. Suppose the query is $Q : \{a_1 = v_1, ..., a_m = v_m\}$. Since the MFDs that match $Q$ are registered at RP peers $N_1$ through $N_m$, where $N_i = \mathcal{H}(a_i = v_i)$, the CDS can send a single query message to any of these $m$ peers to have the query fully resolved. For efficient resolution, the CDS chooses a peer that has the smallest MFD database. Once a query is received, the peer conducts a pairwise comparison between the query and all the entries in its database to find the matching MFDs.

In a similarity search, the user is trying to find music files that have a similar feature vector to what is specified in the query. Suppose the user has a clip *unknown.mp3* with an extracted feature vector $\{f_1 = v_1, ..., f_m = v_m\}$, and wants to find 10 songs that are most similar to the clip. Using the same technique as above, the CDS may select a pair, e.g., $\{f_1 = v_1\}$ and send the query to the peer $N(= \mathcal{H}(f_1 = v_1))$. This peer, instead of conducting a pairwise equality test, computes the "distance" between the query vector and each MFD in its database. In our current system, Manhattan distance defined as $d(f, f') = |v_1 - v'_1| + ... + |v_m - v'_m|$ is used, where $v_i$'s and $v'_i$'s are the values of vector $f$ and $f'$ respectively. More sophisticated way of computing distance, such as "cosine distance" may also be used. The distances are then ranked and the 10 MFDs that have the smallest distance are returned to the user.

However, sending the query to peer $N$ alone will fail to discover the MFDs that slightly differ from the query in $f_1$, but are similar or identical regarding other features, because those MFDs are not registered with $N$. This is undesirable especially when $N$ does not have enough matches. In this case, our system uses a limited expanding ring search to gather more results: in addition to $N$, the query is sent either by $N$ or the query initiator to peers that correspond to values that are near $v_1$, e.g., $f_1 = v_1 \pm 1, f_1 = v_1 \pm 2, ....$ Accordingly, these peers will carry out the distance computation and return any results.

### 4.3. Load balancing

By using Rendezvous Points, network-wide message flooding is avoided at both registration and query times. However, in practice, some AV-pairs may be much more common or popular in MFDs than others. It has been observed that the popularity of keywords in Gnutella follows a Zipf-like distribution [8]. Such a distribution will cause a few peers being overloaded by registrations or queries, while the majority of peers in the system stay underutilized. To improve system's throughput under skewed load, the CDS deploys a distributed dynamic load balancing mechanism [9], where multiple peers are used as RP points to share the heavy load incurred by popular AV-pairs. When an AV-pair appears in many MFDs, instead of sending all the MFDs to one peer, the system partitions them among multiple peers. Similarly if there are a large number of queries for the same AV-pair, the system allows the original peer who is responsible for this pair to replicate its database at other peers. The partitions and replicas corresponding to one AV-pair are organized into a logical matrix, the Load Balancing Matrix (LBM), and the matrix automatically expands or shrinks based on this pair's current registration and query load. LBMs help to eliminate hot spots in the system under skewed load, and the system can maintain high throughput in processing registrations and queries [9].

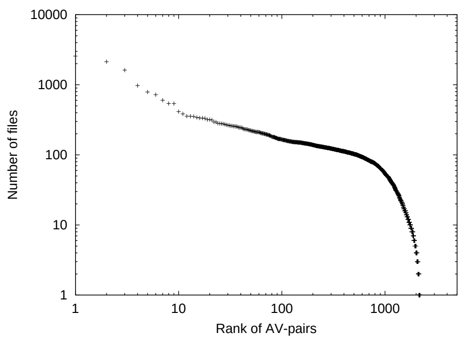

**Fig. 2**. Popularity distribution of feature attributes.

## 5. SYSTEM EVALUATION

The MFEE is built using Marsyas [10], a free software framework for audio analysis. We evaluate our system using an event-driven simulator [9]. For our experiments, we set up a P2P network that has 10,000 peers, and each peer is configured with DSL-level link bandwidth ($\sim 500Kbps$). As MFDs, 30 feature attributes were extracted from 4,816 MP3 files representing a variety of genres and styles. Figure 2 is the log-log plot of the AV-pair distribution in these files. There are 2,178 distinct AV-pairs, and the distribution is highly skewed: the most common AV-pair (ranked 1) appears in 53% of the MFDs and 41 AV-pairs only appear in 1 MFD.

As registration workload, we generated 100,000 MFDs by replicating each of the 4,816 files approximately 20 times, and assigned them to random peers. Each peer registers the files it has with the system. Due to the skewed feature distribution, registrations of common AV-pairs result in multiple partitions. For query load, 100,000 queries were generated following a Zipf distribution independent from the above distribution. Each query corresponds to the features of one music file. We do so to emulate the behavior of a user who submits a music clip and looks for similar music. The most popular MFD occurs in over 10% of the queries, and the majority of the MFDs only occur in a few queries. A query's initiator is randomly picked from all peers, and for simplicity, only exact matches are returned. A peer rejects a query and generates a failure when the peer's link utilization has reached 100% due to simultaneous queries.

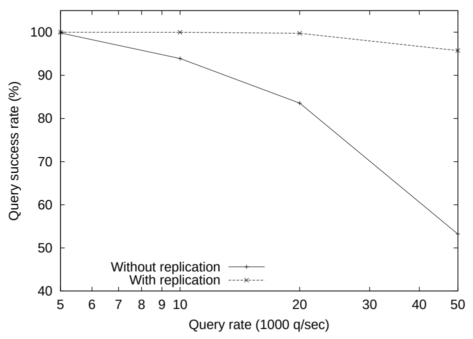

**Fig. 3**. Query success rate comparison.

Figure 3 compares the query success rate as a function of query arrival rate (Poisson arrival) to the system under two scenarios. In the first scenario, when reaching link capacity, a peer simply rejects new queries that arrive at it without replicating its content at other peers. Since for each query the CDS has 30 candidate AV-pairs, query load can spread well among peers even without replication. Therefore the system achieves a high success rate under high load, e.g., the success rate is 94% for a query rate of $10,000q/s(queries/s)$. However, as load increases further, peers corresponding to popular queries will be saturated, and the success rate drops quickly. In the second scenario, by using the dynamic replication mechanism, peers who observe high load will replicate their databases at other peers to dissipate concentrated query load. As a result, we observe that with replication, the system can sustain a much higher query rate while keeping the success rate above 95%.

## 6. CONCLUSIONS

In this paper, we described a scalable and load-balanced P2P system that supports a rich set of music search methods. In particular, our automatic music feature extraction technique enables sophisticated music content based searches such as similarity searches. The RP-based registration and query scheme ensures system scalability by avoiding network wide message flooding encountered in current P2P systems. We evaluated the system using a realistic registration load obtained from a large set of music files. Our dynamic load balancing mechanism allows the system to maintain high throughput under skewed Zipf query load.

## 7. REFERENCES

[1] I. Stoica, R. Morris, D. Karger, F. Kaashoek, and H. Balakrishnan, "Chord: A Scalable Peer-To-Peer Lookup Service for Internet Applications," in *Proceedings of SIGCOMM 2001*, San Diego, CA, August 2001, pp. 149–160.

[2] A. Rowstron and P. Druschel, "Pastry: Scalable, Distributed Object Location and Routing for Large-scale Peer-to-Peer Systems," in *Proceedings of Middleware 2001*, Heidelberg, Germany, November 2001.

[3] Eric Scheirer, *Music-Listening Systems*, Ph.D. thesis, MIT, 2000.

[4] Masataka Goto and Yoichi Muraoka, "Music Understanding at the Beat Level: Real-time Beat Tracking of Audio Signals," in *Computational Auditory Scene Analysis*, David Rosenthal and Hiroshi Okuno, Eds., pp. 157–176. Lawrence Erlbaum Associates, 1998.

[5] Malcolm Slaney, "Mixtures of Probability Experts for Audio Retrieval and Indexing," in *Proc. Int. Conf. on Multimedia and Expo (ICME)*. IEEE, Aug. 2002.

[6] George Tzanetakis and Perry Cook, "Musical Genre Classification of Audio Signals," *IEEE Transactions on Speech and Audio Processing*, July 2002.

[7] C. Wang, J. Li, and S. Shi, "A Kind of Content-Based Music Information Retrieval Method in a Peer-to-Peer Environment," in *Proceeding of ISMIR 2002*, Paris, France, Oct. 2002, pp. 178–186.

[8] K. Sripanidkulchai, "The Popularity of Gnutella Queries and Its Implications on Scalability," http://www.cs.cmu.edu/ kunwadee/research/p2p/gnutella.html.

[9] Jun Gao and Peter Steenkiste, "Rendezvous points-based scalable content discovery with load balancing," in *Proceedings of NGC 2002*, Boston, MA, October 2002.

[10] George Tzanetakis and Perry Cook, "Marsyas: A framework for audio analysis," *Organised Sound*, vol. 4(3), 2000.