A Scalable Peer-to-Peer System for Music Information Retrieval

George Tzanetakis,* Jun Gao,† and Peter Steenkiste†‡
*Computer Science Department, Faculty of Engineering
University of Victoria
PO BOX 3055 STNCSC
Victoria, British Columbia V8W3P6 Canada
gtzan@cs.uvic.ca
†Computer Science Department
‡Department of Electrical and Computer Engineering
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, Pennsylvania 15213 USA
{jungao,prs}@cs.cmu.edu

Currently, a large percentage of Internet traffic consists of music files, typically stored in MP3 compressed audio format, shared and exchanged over peer-to-peer (P2P) networks. Searching for music is performed by specifying keywords and naïve string-matching techniques. In the past years, the emerging research area of music information retrieval (MIR) has produced a variety of new ways of looking at the problem of music searching. Such MIR techniques can significantly enhance the ways users search for music over P2P networks. For that to happen, there are two main challenges that need to be addressed: scalability to large collections and number of peers; and a richer set of search semantics that can support MIR, especially when the retrieval is content-based. In this article, we describe a scalable P2P system that uses rendezvous points (RPs) for music metadata registration and query resolution. The system supports attribute-value search semantics as well as content-based retrieval. The performance of the system has been evaluated in large-scale usage scenarios using "real," automatically calculated musical content descriptors.

One could argue that both the ideas of MIR and P2P became familiar to the general public with Napster (www.napster.com). Although crude both in terms of search capabilities and P2P performance, Napster provided for the first time an example of sharing vast amounts of musical data over large ad-hoc networks. Despite this early connection between MIR and P2P, there has not been much progress in combining these two areas. Although better P2P paradigms have been proposed, searching for music is currently still performed using traditional keyword-based text searching. Although a variety of novel ways of searching and retrieving music, especially in audio format, have been proposed, they have not found their way into P2P networks and remain largely academic exercises.

There are many advantages to P2P networks, such as distributed computing and storage power, fault-tolerance, and reliability. Owing to copyright restrictions, however, major recording labels have been reluctant to follow this paradigm, but the emergence of audio fingerprinting technology (Haitsma and Kalker 2002) is likely going to change this attitude. Although the intellectual property issues behind P2P networks are complicated, we believe that the use of techniques such as fingerprinting will allow digital music distribution while protecting copyrights. Although the main focus of the system is the use of P2P networks in MIR, the proposed architecture can be adapted for the requirements of fingerprinting. One of the greatest potential benefits of P2P networks is the ability to harness the collaborative efforts of users to provide semantic, subjective, and community-based tags to describe musical content. We believe this aspect of P2P networks provides a unique opportunity for changing the way music is produced and distributed.

Centralized P2P networks such as Napster are not robust and may be vulnerable to Denial-of-Service attacks, because 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 (www.gnutella.org) and KaZaA (www.kazaa.com), are more robust, but because peers do not

Computer Music Journal, 28:2, pp. 24–33, Summer 2004

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 limits the system's scalability and performance. Distributed Hash Table (DHT)-based systems, such as those described in Stoica et al. (2001) and Rowstron and Druschel (2001), achieve good scalability by deploying a structured overlay P2P network that supports efficient content location. However, the basic set of applications built on top of DHT only supports exact file name lookup and does not allow the rich search semantics desired for MIR.

In this article, we describe a robust, scalable P2P system that provides flexible search semantics based on attribute-value (AV) pairs and supports automatic extraction of musical features and content-based similarity retrieval. The system is shown to perform well under realistic loads consisting of features automatically extracted from a large database of audio recordings. The main contributions of this article are a general content-discovery mechanism that supports exact and similarity search based on AV pairs and its evaluation using a specific set of audio features computed on actual audio recordings. We believe that the proposed system provides the necessary flexibility and performance for effective use of MIR for searching in peer-to-peer networks.

Related Work

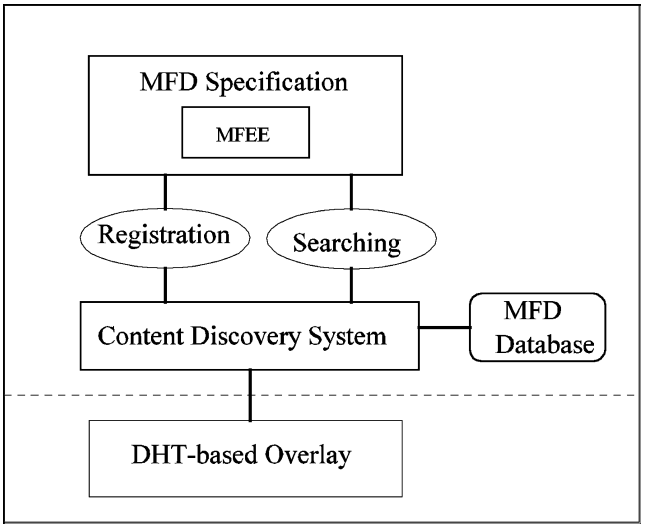
The main focus of this article is the retrieval of music in audio format over P2P networks. Much recent exciting work in MIR is relevant to the design of our system, and we review some representative publications. Although this article mainly describes similarity retrieval, the underlying framework of features and calculating distances forms the basis of a variety of audio-analysis algorithms such as musical genre classification (Tzanetakis and Cook 2002; Aucouturier and Pachet 2003), beat detection and analysis (Foote and Cooper 2002), similarity retrieval (Logan and Salomon 2001; Aucouturier and Pachet 2002; Yang 2002), audio fingerprinting (Haitsma and Kalker 2002), and clustering and visualization (Rauber et al. 2002). In addition to features computed from the automatic analysis of audio content, features computed based on text analysis of critics' reviews as well as P2P usage patterns have been shown to be effective for classification (Whitman and Smaragdis 2002). These articles are representative of each category. A general overview of the current status in MIR and an extensive bibliography can be found in Futrelle and Downie (2002); another good overview of the current state of the art and challenges in MIR is found in Pachet (2003).

DHT-based systems such as those described in Stoica et al. (2001) and Rowstron and Druschel (2001) solve some of the scalability problems of the well-known broadcast-based systems such as Gnutella and KaZaA. The content discovery system (CDS) proposed in this article is built on top of such a DHT-based system. The idea of using MIR over P2P was proposed in Wang et al. (2002); however, the proposed system's architecture suffers from scalability problems, and only the retrieval of symbolic data is examined. The potential of integrating MIR and the evolving semantic web was explored in Baumann and Kluter (2002). More recently, content-based retrieval over a P2P network using the JXTA programming framework was presented in Baumann (2003). The proposed system is mainly concerned with integrating feature representations into the P2P system rather than using the structure of the network to support exact and similarity search. An initial description of the system presented in this article can be found in Gao et al. (2003). The main differences of this article from previous work is the scalable and efficient support of both exact and similarity searching based on RPs, as well as the performance evaluation of using audio features computed from actual audio recordings rather than synthetic data.

System Overview

Following recent work in P2P networks, our system decouples the process of locating content from the process of downloading it. Each node in the network not only stores music files for sharing but

Figure 1. Software architecture of a peer node (DHT = distributed hash table, MFD = music file description, MFEE = Music Feature Extraction Engine).



also stores information about the location of other music files in the network. Therefore, when the user submits a search to the P2P system, the system returns a set of peers from which the music files that satisfy the search criteria can then be downloaded. Each file in the system is described by a music file description (MFD) which is essentially a set of AV pairs. For example, a possible MFD might be the following: {artist = U2, album = Rattle and Hum, song = Desire, . . ., specCentroid = 0.65, mfcc2 = 0.85, . . .}. Notice that the description contains both attributes that can be manually specified by the user, such as artist or album, as well as automatically extracted features for describing musical content (such as the Spectral Centroid). The Music Feature Extraction Engine (MFEE) calculates these features from the audio files. These automatically extracted features, in addition to being used for content-based similarity search, can also be used for various other types of audio analysis, such as musical genre classification.

Two operations are supported by the system, and both take MFDs as arguments. In registration, a new music file is made available for sharing, and its associated MFD is registered to the P2P system. During searching, the user query is converted into an appropriate MFD that is then used to locate the nodes that contain files that match the search criteria. Once the nodes are located, they are contacted directly to start the actual downloading of the file. The main concern in the design of our system is the efficient content discovery rather than the actual downloading mechanism.

Figure 1 shows the software architecture on each peer node. The MFD of either registration or search query is passed to the content discovery system (CDS), which runs on top of a DHT-based P2P system, such as Chord (Stoica et al. 2001). 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. Therefore, a DHT implements just one operation, *lookup(key)*, which yields the network location of the node currently responsible for a given key.

To implement a DHT, a lookup algorithm must address the following issues: mapping keys in a load-balanced way, forwarding a lookup for a key to an appropriate node, and building routing tables adaptively as nodes enter and exit the system. Correspondingly, by applying the same computation to the data item, a peer can locate it from the same peer who stores it. A good introduction to looking up data in P2P systems is found in Balakrishnan et al. (2003). We present the algorithm used by the CDS to distribute MFDs to peers in the Scalable Content Discovery Section. 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 $O(\log N)$ neighboring peers, and the number of overlay hops between two peers is $O(\log N)$, where N is the total number of peers in the system. Each peer maintains a local MFD database to store the MFDs it is assigned by the CDS. Upon the arrival of a query, each peer examines its local MFD database and returns the set of MFDs that match the query to the query initiator. Subsequently, the query initiator can download the actual music file from the peer that owns the music.

Music Feature Extraction

The MFEE component takes as input an audio file in either pulse-code modulation (PCM) format or a

compressed format (such as MP3) and produces 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 Tzanetakis and Cook (2002) for the purpose of musical genre classification. This feature vector statistically 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). It has been shown to be an effective representation for the purposes of classification and retrieval of music. Features based on the Short Time Fourier Transform as well as Mel-Frequency Cepstral Coefficients are used to represent sound texture, and features based on Beat and Pitch Histograms are used to represent rhythm and pitch content. More specifically, the means and variances of the Spectral Centroid, Rolloff, Flux, and Zero Crossings, as well as the first five Mel-Frequency Cepstral Coefficients (MFCC) over a 1-sec texture window using a 20-msec analysis window are calculated to represent Spectral Texture. For the Beat Histogram calculation, a Discrete Wavelet Transform filterbank is applied, and autocorrelation-based envelope periodicity detection is performed. For the Pitch Histogram calculation, the multiple pitch detection algorithm described in Tolonen and Karjalainen (2000) is used. The different types of information represented by the feature vector, combined with the query flexibility of the system, support a rich variety of possible query specifications. For example, a user can search only on the basis of rhythmic content while ignoring other aspects of music similarity or combine metadata and content-based similarity search.

We use 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 100×1 discrete values. Experiments comparing automatic classification of the original continuous features and the quantized features showed no significant differences in the results. Using the features and dataset (ten genres) described in Tzanetakis and Cook (2002) and a Gaussian classifier, we obtained 57.5 percent classification accuracy using the unquantized features and 58 percent accuracy using the quantized version.

The results of the MFEE component, together with manually annotated metadata such as artist and album name, are combined to form the MFD, which is a collection of AV pairs. For example, MFD_1:$\{a_1 = v_1, \ldots, a_n = v_n\}$ consists of n AV pairs, where a_i, $i = 1 \ldots n$, can be either a manually annotated attribute or a content-based feature attribute. For specifying queries, 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. Any subset of the MFD can be used for query specification, and using named AV pairs in MFDs allows more possibilities than traditional keyword-based searching. Some examples of possible queries of increasing complexity are the following:

- Search for {artist = U2}
- Search for {artist = U2, year = 1985, tempo = 80 beats-per-minute (BPM) – 100 BPM}
- Search for {10 most similar to *x.mp3* (content-based similarity search)}
- Search for {10 most similar to *x.mp3*, artist = U2}

These are only symbolic pseudo-representations of the queries. In an actual implementation, a variety of user interfaces for query specification would be provided by the system (some interesting possibilities are described in Tzanetakis et al. 2002). For queries with content-based parts such as the last two examples, the audio file *x.mp3* is first converted using the MFEE to numerical features that describe musical content that are subsequently used for similarity search. This mechanism allows any audio file to be used in the system, even if it does not have any metadata information associated with it (for example, a file recorded off a radio broadcast).

Scalable Content Discovery

Unlike centralized systems in which files are registered at a single place, or broadcast-based systems

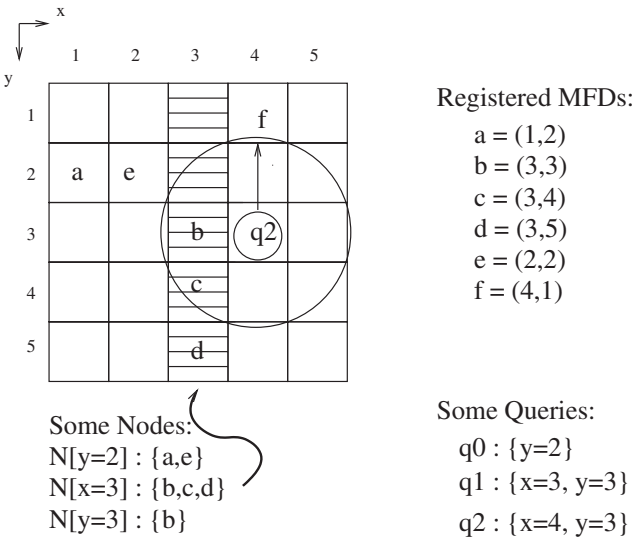
where a query may potentially be sent to all peers in the system, CDS uses a scalable approach based on RPs for registration and query resolution. Essentially, the P2P network is structured to efficiently represent the space of AV pairs for the tasks of searching and retrieving music.

MFD Registration

To register an MFD, the CDS applies a uniform hash function such as SHA-1 to each AV pair in the MFD to obtain n node IDs: $H(a_i = v_i) \rightarrow N_i$, 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 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 $\{a_1 = v_1\}$. Figure 2 shows a made-up small-scale example of how the system works. The MFD is two-dimensional, with the horizontal coordinate corresponding to the first attribute and the vertical coordinate corresponding to the second attribute. (This is only done for purposes of illustration. The actual system can handle multidimensional data as well.) One can visually observe the distribution of the MFD (a,b,c,d,e,f) and how they are assigned to nodes such as $N[x = 3]$. In two dimensions, each node is responsible for a row or a column of the AV grid. Obviously, this results in significant overlap between the entries of the local databases on each node, which enhances the robustness of the system. Even in this contrived example, it can be seen that the load on each node depends on the distribution of specific AV pairs such as $x = 3$. This issue is addressed by the load-balancing scheme described later.

Because the number of AV pairs in an MFD is typically small (e.g., less than 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. An important point is that registering each AV pair of an MFD individually allows the MFD to be searched using any subset of its AV pairs, which is important to allow the rich set of possible query semantics we desire for MIR.

Figure 2. Illustration example of the Content Discovery System.



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:$\{a_1 = v_1, \ldots, a_m = v_m\}$. Because the MFDs that match Q are registered at RP peers N_1 through N_m, where $N_i = H(a_i = v_i)$, the CDS can send a single query message to any of these peers to have the query fully resolved. For efficient resolution, the CDS chooses the peer that has the smallest MFD database. Once a query is received, the peer conducts a pair-wise comparison between the query and all the entries in its database to find the matching MFDs. An example might be {artist = U2, year = 1985, tempo = 100 BPM}, which means the matched MFD must have the above three AV pairs in their description. Most likely, the node that contains the locations of all U2 songs will have the smallest local MFD database, so it will be contacted and locally searched for MFDs with the correct year and tempo.

To further illustrate this process, in Figure 2, the query q_0:{$y = 2$} will go to the corresponding node $N[y = 2]$ and directly return the correct answer {*a, e*}. The query q_1: {$x = 3$, $y = 3$} will check the database size of nodes $N[x = 3]$ and $N[y = 3]$, select the smallest $N[y = 3]$, and then return the correct answer {*b*}. Note that the answer is the location of the music file that can then be subsequently downloaded.

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 {$f_1 = v_1, \ldots, f_2 = v_2$} and wants to find ten 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 ($= H(f_1 = v_1)$). This peer, instead of conducting a pair-wise equality test, computes the ''distance'' between the query vector and each MFD in its database. Our current system uses Manhattan distance, defined as $d(f, f') = |v_1 - v_1'| + \ldots + |v_m - v_m'|$, where all the v_i and v_i' represent the values of vectors f and f', respectively. More sophisticated ways of computing distance, such as ''cosine distance,'' may also be used. The distances are then ranked, and the ten top 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 peer 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., $f_1 = v_1 \pm 1$, $f_1 = v_1 \pm 2$, etc. Accordingly, these peers will carry out the distance computation and return any results. Figure 2 shows schematically this expanding ring search for query q_2:{$x = 3$, $y = 4$} as two concentric circles.

Of course, it is also possible to combine exact AV search and content-based similarity search. This last point is important and directly influenced the design of our system. Although it is possible to use more elaborate data structures for multidimensional nearest neighbor search (the similarity search) such as KD-trees (Bentley 1975) and try to distribute them over the P2P network, these structures do not directly support searching for arbitrary subsets of AV pairs as our system does. Supporting such searches is important for MIR, because we would like to combine both metadata and content-based retrieval in the same query. Finally, range queries such as {tempo = 80–120 BPM} are resolved by issuing multiple queries corresponding to the values within the range. We are experimenting with a more efficient multi-resolution approach to range searching where not only values but also ranges of values are assigned to nodes. That way, the number of nodes that must be visited to resolve a range query is reduced to $O(\log L)$, where L is the length of the range. This method is described in more detail in Gao and Steenkiste (2003).

Load Balancing

By using RPs, 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 (Sripanidkulchai 2001). Such a distribution will cause a few peers to be overloaded by registrations or queries and thus create bottlenecks, while the majority of peers in the system stay underutilized. Figure 3 shows a distribution of AV pairs computed from automatically extracted musical content features described in more detail in the System Evaluation section. To improve the system's throughput under skewed load, the CDS deploys a distributed dynamic load-balancing mechanism described in Gao and Steenkiste (2004), where multiple peers are used as RP points to share the heavy load incurred by popular AV pairs.

When a node corresponding to a data item (e.g., an AV pair) becomes overloaded by registration or queries, it recruits more nodes to share the load instead of rejecting the registrations or queries. When an AV pair appears in many MFDs, instead of send-

Figure 3. Popularity distribution of feature attributes.

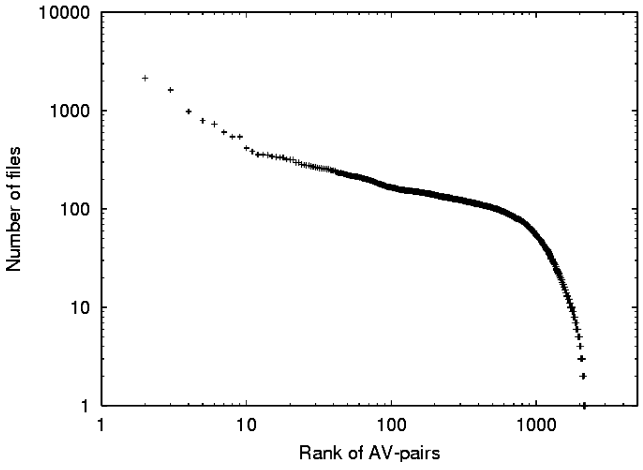
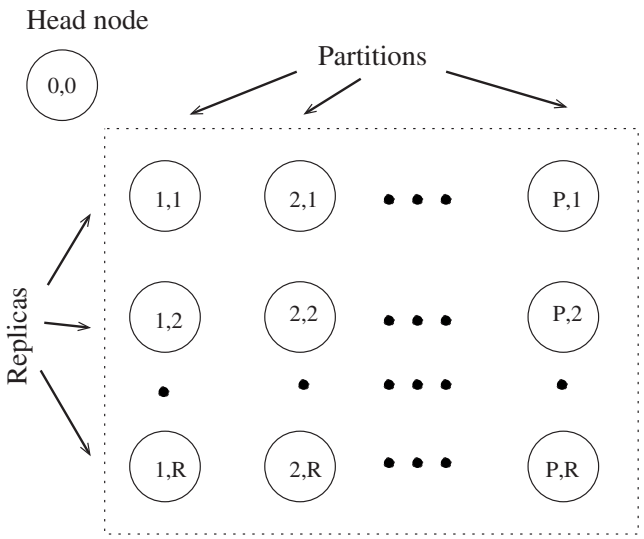


Figure 4. Load-balancing matrix.



ing 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.

As a concrete example, some particular AV pair such as {tempo = 100 BPM} might be very common. Partitioning the set of songs that have that particular AV pair among multiple RP points balances the registration load. Another type of load that is possibly independent of the registration load is the popularity of queries. For example, a new release {artist = Madonna, year = 2003} will initially only be registered with a few RPs, although many queries will be requesting it. Replicating their information will balance the query load. The partitions and replicas corresponding to one AV pair are organized into a logical matrix, the load-balancing matrix (LBM). Each column of the matrix holds a partition of the MFDs that contain this AV pair (which helps spread registration load) and nodes in different rows within a column are replicas of each other (which helps spread query load). An LBM starts with one node and expands and shrinks depending on the registration and query load. The number of partitions is proportional to the registration load, and the number of replicas is proportional to the query load corresponding to this data item.

Figure 4 shows the layout of the matrix for a data item. Each node in the matrix has a column and row index (p, r), and node IDs are determined by applying the hash function H to the data item (an AV pair in this case) and the column and row indices together: $N_i(p, r) = H(data, p, r)$. To register a data item, the registering node selects a random partition from the matrix and registers with each node in that partition. To retrieve all possible matches, a query must be sent to all existing partitions in the matrix; however, only one node is selected from each partition owing to replicas. An important property of the LBM is that both the registration load and query load are spread evenly within the matrix.

As an optimization, each matrix may use a head node, with ID $= N_i(0, 0)$ $(= H(data, 0, 0))$ to store its current size and coordinate the matrix's expansion and shrinking. In Gao and Steenkiste (2004), fault-tolerant mechanisms for restoring the size information when the head node fails are discussed. An end point can retrieve the size of the matrix from the head node. To ensure that the head node will not become a performance bottleneck, endpoints can also obtain the size efficiently by conducting binary probing along the two dimensions, caching the size for future use. To summarize, LBMs help eliminate "hot spots" in the system under skewed load, and the system can maintain high

throughput in processing registrations and queries (Gao and Steenkiste in press).

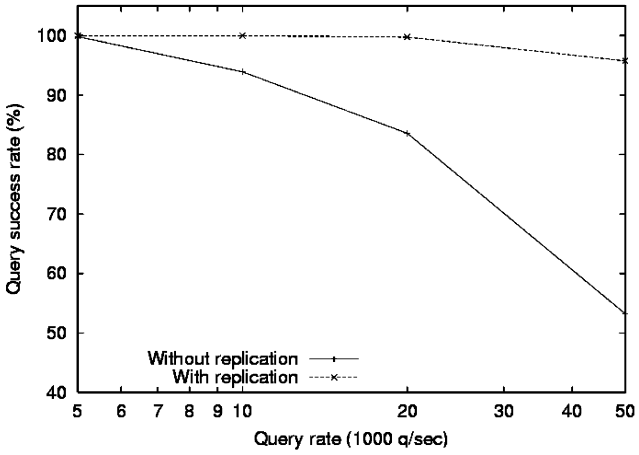
System Evaluation

The MFEE is built using Marsyas (Tzanetakis and Cook 2000), a free software framework for audio analysis, and we evaluate our system using an event-driven simulator (Gao and Steenkiste 2004). For our experiments, we set up a P2P network that has 10,000 peers, and each peer is configured with DSL-class link bandwidth (approximately 500 Kbps). As MFDs, 30 music content-based features are used as attributes. They were automatically extracted from 5,000 MP3 files representing a variety of genres and styles. Figure 3 shows 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 percent of the MFDs, and 41 AV pairs only appear in 1 MFD.

For registration workload, we generated 100,000 MFDs by replicating each of the 5,000 files 20 times and assigning them to random peers. Each peer registers the files it is assigned with the system. Owing to the skewed feature distribution, registrations of common AV pairs result in multiple partitions. For query load, 100,000 queries were generated following a query-popularity Zipf distribution independent from the AV pair distribution shown in Figure 3. Each query corresponds to the features of one particular music file. This is done in order 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 percent 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 percent owing to simultaneous queries.

Figure 5 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

Figure 5. Query success rate comparison.



rejects new queries that arrive at it without replicating its content at other peers. Because for each query the CDS has 30 candidate AV pairs, query load can spread evenly among peers, even without replication. Therefore, the system achieves a high success rate under high load: for example, the success rate is 94 percent for a query rate of 10,000 queries per sec (q/sec). However, as load increases further, peers corresponding to popular queries will become 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 percent.

Conclusions

In this article, 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 content-based similarity retrieval. The RP-based registration and query specification 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. It is our hope that the design of our system will inspire additional research in the interesting area of bringing state of the art MIR algorithms to the increasingly popular P2P networks. Another important aspect of the proposed system is that new attributes can be incorporated into the system with minimal effort.

We are currently refining the design of our system to handle similarity searches more efficiently and plan to further evaluate our system with traces acquired from real users. Further experiments are necessary for a more detailed evaluation of system performance. In addition, user studies need to be conducted to explore how users interact with the system and their typical queries. Duplicate copies of the same audio content can be detected by adapting an audio-fingerprinting technique to work with our P2P system.

Acknowledgments

We gratefully acknowledge the support of the National Science Foundation under grant ISS-008594. This research was also sponsored in part by the Defense Advanced Research Project Agency and monitored by ARFL/IFGA, Rome, New York 13441-4505 USA, under contract F30602-99-1-0518. Additional support was provided by Intel.

References

Aucouturier, J.J., and F. Pachet. 2002. ''Music Similarity Measures: What's the Use?'' *Proceedings of the International Conference in Music Information Retrieval.* Paris: IRCAM, pp. 157–163.

Aucouturier, J.J., and F. Pachet. 2003. ''Representing Musical Genre: A State of the Art.'' *Journal of New Music Research* 32(1):83–93.

Balakrishnan, H., et al. 2003. ''Looking Up Data in P2P Systems.'' *Communications of the ACM* 46(2):43–48.

Baumann, S. 2003. ''Music Similarity Analysis in P2P Environment.'' In *Proceedings of the 4th European Workshop on Image Analysis for Multimedia Interactive Services.* London: Word Scientific, pp. 314–319.

Baumann, S., and A. Kluter. 2002. ''Super-Convenience for Non-Musicians: Querying MP3 and the Semantic Web.'' *Proceedings of the International Conference on Music Information Retrieval.* Paris: IRCAM, pp. 297–298.

Bentley, J. 1975. ''Multidimensional Binary Search Trees Used for Associative Searching.'' *Communications of the ACM* 18(9):509–517.

Foote, J., and M. Cooper. 2002. ''Audio Retrieval by Rhythmic Similarity.'' *Proceedings of the International Conference on Music Information Retrieval.* Paris: IRCAM, pp. 265–266.

Futrelle, J., and J. S. Downie. 2002. ''Interdisciplinary Communities and Research Issues in Music Information Retrieval.'' *Proceedings of the Third International Conference on Music Information Retrieval.* Paris: IRCAM, pp. 215–221.

Gao, J., and P. Steenkiste. 2004. ''Design and Evaluation of a Distributed Scalable Content Discovery System.'' *IEEE Journal on Selected Areas in Communications* 22(1):54–66.

Gao, J., and P. Steenkiste. 2003. ''An Efficient Scheme Supporting Range Queries in DHT-based Systems.'' Carnegie Mellon University Technical Report CMU-CS-03-215.

Gao, J., G. Tzanetakis, and P. Steenkiste. 2003. ''Content-based Retrieval of Music in Scalable Peer-to-Peer Networks.'' *Proceedings of the International Conference on Multimedia and Expo.* Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, pp. 309–312.

Haitsma J., and T. Kalker. 2002. ''A Highly Robust Audio Fingerprinting System.'' *Proceedings of the Third International Conference on Music Information Retrieval.* Paris: IRCAM, pp. 107–115.

Logan, B., and A. Salomon. 2001. ''A Music Similarity Function based on Signal Analysis.'' *Proceedings of the International Conference on Multimedia and Expo.* Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, n.p.

Pachet, F. 2003 ''Content Management for Electronic Music Distribution.'' *Communications of the Association for Computing Machinery* 46(4):71–75.

Rauber, A., E. Pampalk, and D. Merkl. 2002 ''Using Psychoacoustic Models and Self-organizing Maps to Create a Hierarchical Structure of Music by Sound Similarity.'' *Proceedings of the Third International Conference on Music Information Retrieval.* Paris: IRCAM, pp. 71–80.

Rowstron, A., and P. Druschel. 2001. ''Pastry: Scalable Distributed Object Location and Routing for Large-Scale Peer-to-Peer Systems.'' *Lecture Notes in Computer Science* 2218:329–350.
Sripanidkulchai, K. 2001. ''The Popularity of Gnutella Queries and its Implications on Scalability.'' Available online at www.cs.cmu.edu/~kunwadee/research/p2p/gnutella.html. Accessed 11 February 2004.
Stoica, I., et al. 2001. ''Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications.'' *Proceedings SIGCOMM.* New York: Association for Computing Machinery, pp. 149–160.
Tolonen, T., and M. Karjalainen. 2000. ''A Computationally Efficient Multipitch Analysis Model.'' *IEEE Transactions on Speech and Audio Processing* 8(6):708–716.
Tzanetakis, G., and P. Cook. 2002. ''Musical Genre Classification of Audio Signals.'' *IEEE Transactions on Speech and Audio Processing* 10(5):293–302.
Tzanetakis, G., A. Ermolinskyi, and P. Cook. 2002. ''Beyond the Query-by-Example Paradigm: New Query Interfaces for Music Information Retrieval.'' *Proceedings of the 2002 International Computer Music Conference.* San Francisco: International Computer Music Association, pp. 177–183.
Wang, C., J. Li, and S. Shi. 2002. ''A Kind of Content-Cased Music Information Retrieval Method in a Peer-to-Peer Environment.'' *Proceedings of the Third International Conference in Music Information Retrieval.* Paris: IRCAM, pp. 178–186.
Whitman, B. and P. Smaragdis. 2002. ''Combining Musical and Cultural Features for Intelligent Style Detection.'' *Proceedings of the Third International Conference on Music Information Retrieval.* Paris: IRCAM, pp. 47–52.
Yang, C. 2002. ''The MACSIS Acoustic Indexing Framework for Music Retrieval: An Experimental Study.'' *Proceedings of the Third International Conference on Music Information Retrieval.* Paris: IRCAM, pp. 52–62.