P2P Information Retrieval: A Self-Organizing Paradigm

Ka Cheung Sia
Department of Computer Science and Engineering
The Chinese University of Hong Kong
Shatin, N.T., Hong Kong SAR
kcsia@cse.cuhk.edu.hk

November 28, 2002

Abstract

In the light of image retrieval evolving from text annotation to content-based and from standalone applications to web-based search engines to increase popularity, we foresee the need for deploying content-based image retrieval (CBIR) into Peer-to-Peer (P2P) architecture. By doing so, we not only distribute the tasks of feature extraction, indexing and storage of image data into peers, we also introduce another aspect of searching in addition to filename-based method in prevalent P2P applications. Through the deployment of DISCOVIR, we introduce a model to improve query efficiency targeting on content-based search.

As clustering technique is widely used in database and information retrieval system for organizing data and improving retrieval efficiency. We surmise such functionality is valuable to a P2P distributed environment. In this paper, we introduce the concept of peer clustering at the level of overlaying network topology, and content-based query routing strategy to improve existing retrieval methods. We design and implement a DIStributed COntent-based Visual Information Retrieval (DISCOVIR) system with content-based query functionality and improved query efficiency. We demonstrate its scalability and efficiency through simulation.

1 Introduction

The appearance of Peer-to-Peer (P2P) applications in recent years have demonstrated the significance of distributed information sharing systems by offering the advantages of scalable storage space. However, since data storage is decentralized, data location process need to be reformulated in order to achieve efficient lookup. Current researches focus on improving data access under the filename-based lookup, i. e. filename is the only term to index a file. We argue that such lookup process is insufficient when the data collection is huge or distributed, especially in the P2P environment. Users might annotate the same file with different filenames, making the data location process error-prone and user dependent. We believe content-based information retrieval should be the ultimate goal of a P2P system.

Regarding to current content-based image retrieval (CBIR) systems, we envisage the potential use of P2P network in both scattering data storage and distributing workload of feature extraction and indexing. Through the realization of CBIR in P2P network, we can support enormous image collection without installing high-end equipment for the web server and incorporate individual user's contribution compared to prevalent centralized web-based search engine. Also, we make use of the computation power of peers for image preprocessing and indexing in addition to data storage.

By combing CBIR into a P2P environment, we **1)** revolutionize the way of searching in existing P2P architecture and **2)** overcome the bottleneck problem of running CBIR in a centralized fashion. In this paper, we introduce the design and implementation of DISCOVIR (DIStributed COntent-based Visual Information Retrieval) [4, 26], a CBIR software running on Gnutella [7]. In the notion of CBIR in P2P, current searching methodologies in P2P system might not be applicable or efficient as they focus on filename-based search. Therefore, we propose the Peer Clustering at the level of overlaying network topology, and Firework Query Model (FQM) [20] to facilitate content-based searching in P2P.

In the process of integrating CBIR into P2P and designing of algorithm for search efficiency improvement, we discover some resemblance between P2P and Neural Networks. We envision to model the P2P as Neural Network might be a potential to improve P2P system into a more intelligent environment rather than a data sharing network. We try to give some preliminary ideas at the end of this paper and set this as the future direction.

In the coming section, we will detail the related works in CBIR and P2P, and justify the motivation and necessity of combining them. In Section 3.1, we introduce the architecture of DISCOVIR and the functionality of its components, while in Section 3.2, we detail the Peer Clustering and FQM for improving content-based search in P2P. In Section 4, we present experiment result of simulation of FQM and give some analysis. We address on

how to extend current works in Section 5 and lastly, we give concluding remarks in Section 6.

2 Background

2.1 Content-based Image Retrieval

In early image retrieval system, it requires human annotation and classification on the image collection, the query is thus performed using text-based information retrieval method. However, there are several limitations for such implementation, they are **Human Intervention**, **Non-Standard Description** and **Linguistic Barriers**. These problems are due to the manual annotation on images, which is tedious, error-prone and subject to individual user perceptions and understandings.

In order to solve these problems, CBIR is proposed to pass such tedious task to computer. Since early 1990's, many CBIR systems have been proposed and developed, some of them are QBIC [5], WebSEEK [28], SIMPLIcity [30], MARS [16], Photobook [21], WALRUS [19] and other systems for some domain specific applications [12, 11]. In these systems, images are indexed by low-level feature like color, texture and shape information which can be extracted easily using computer. These systems are not designed to be distributed across different computers in a network. One of the shortcomings is that the feature extraction, indexing and also the query processing are all done in a centralized fashion which is computationally intensive and difficult to scale up. As indicated by several researchers [24, 27], one of the promising future trends in CBIR includes the distribution of data collection, data processing and information retrieval. By extending the centralized system model, we not only can increase the size of image collections easily, but we also overcome the scalability bottleneck problem by distributing the process of feature extraction and retrieval.

2.2 Peer-to-Peer

P2P network is a recently evolved paradigm for distributed computing. With the emerging P2P networks and various implementations such as Gnutella [7], Napster [18], LimeWire [13], YouServ [1], Freenet [6], Morpheus [17] and KaZaA [10], they focus on the retrieval of data based on filenames or metadata, such as ID3 tag of MP3, and offer the advantages of distributed resource [25], increased reliability [3] and comprehensiveness of information [14].

P2P systems were classified by [15] into three different categories, namely: Centralized, Decentralized Unstructured and Decentralized Structured.

Centralized - Napster [18] and SETI [25], to a certain extend, fall into this category, every peer in the P2P network keeps informing a central locations the information of its shared data, while queries are issued to this central locations. Though Napster was extremely successful, such scheme only utilize the data storage capability in P2P and subjects to legal problems and single point failure. Due to this reason, Researches proceed to the other two paradigms or work on content anti-censorship and user anonymity to bypass legal responsibilities.

Decentralized Unstructured - Gnutella [7] is an example of such designs. Figure 1 shows an example of a typical P2P network in this category. In the example, different files are shared by different peers, while there is **no index information** on where data are stored. When a peer initiates a search for a file, it broadcasts a query request to all its connecting peers. Its peers then propagate the request to their connected peers and this process continues. Unlike the client-server architecture of the web, the P2P network aims at allowing individual computer, which joins and leaves the network frequently, to share information directly with each other without the help of dedicated servers. In these networks, a peer can become a member of the network by establishing a connection with one or more peers in the current network. Messages are sent over multiple hops from one peer to another, known as flooding, while each peer responds to queries for information it shares locally. Although this designs are extremely resilient to nodes entering and leaving the network, they are unscalable and generates large loads for the network participants.

Decentralized Structured - These systems have no central directory server, thus are considered decentralized. While there is a significant amount of structure imposing on the P2P overlay network in order to improve searching in a decentralized fashion. Such highly structured P2P designs are quite prevalent in the research literature like CAN [22], Chord [29], Pastry [23] and Tapestry [32], they make use of hash function and scatter the storage of index in a controlled manner for efficient lookup of data.

Both CAN and Chord use Distributed Hash Table (DHT) to map the filename to a key, and each peer is responsible for storing certain range of (key, value) pairs. When a peer looks for a file, it hashes the filename to a key and ask the peers responsible for this key for the actual storage location of that file. Chord models the key as an m-bits identifier and arranges the peers into a logical ring topology to determine which peer is responsible for storing which (key, value) pair. CAN models the key as point on a d-dimension Cartesian coordinate space, while each peer is responsible for (key, value) pairs inside its specific region. Such systems take a balance between the centralized index and totally decentralized index approaches. They speed up and reduce message passing for the process of key lookup (data location); however, they incur a penalty for redistributing index storage when peers join and leave the network frequently, especially in a dynamic environment like the Internet, thus the applicability in an environment of extremely transient population of nodes is still an unknown.

2.3 Motivation

The development from Centralized to Decentralized Unstructured and to Decentralized Structured aims at improving data lookup efficiency by:

1. **Placement of index** - DHT-based algorithms improve the Decentralized Unstructured P2P by storing index distributively for use in lookup process.

2. **Replication** - Freenet [6] and some other researches [2, 15] propose to replicate data in other peers to improve the query efficiency in the Decentralized Unstructured fashion. This method is merely a tradeoff between data storage and lookup efficiency.

We then ask: are we able to formulate a P2P model that optimizes for the data location process, while introducing comparatively less penalty when peers join and leave the network? Are we able to perform content-based searching in this P2P network? Instead of distributing the storage of index into different peers, we allow a peer to index its own data collection while retaining the original Gnutella network topology but imposing a requirement on connections to serve as the global indexing structure. We name this architecture as **Decentralized Quasi-structured** and use **Self Orangization** to improve query efficiency. Moreover, we introduce the content-based image search functionality to illustrate the potential richness of queries in a P2P network.

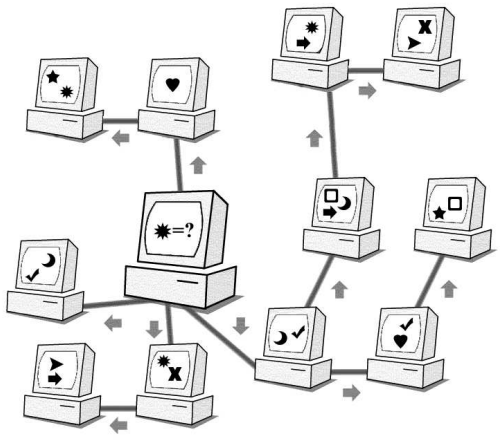

Figure 1: P2P information retrieval

3 DISCOVIR

3.1 System Architecture

In this section, we will describe the architecture of a DISCOVIR client and the communication protocol in order to perform CBIR over a P2P network. Through the DISCOVIR program, users can share images among peers around the world. Each peer is responsible for extracting and indexing the feature of the shared images, by doing so, every peers can search for similar images based on image content, like color, texture and shape among images shared by all DISCOVIR peer in the network.

Figure 2 depicts the key components and their interaction in the architecture of a DISCOVIR client. As DISCOVIR is derived from LimeWire [13] open source project, the operations of Connection Manager, Packet Router and HTTP Agent remain more or less the same with additional functionality to improve the query mechanism used in original Gnutella network. Plug-in Manager, Feature Extractor and Image Indexer are introduced to support the CBIR task. The User Interface is modified to incorporate the image search panel, Figure 3 shows a screen capture of DISCOVIR in the image search page. Here are the brief descriptions of each component:

- **Connection Manager** - It is responsible for setting up and managing the TCP connection between DISCOVIR clients.

- **Packet Router** - It masters the routing of message in DISCOVIR network between components and peers.

- **Plug-in Manager** - It coordinates the storage of different feature extraction modules and their interaction with Feature Extractor and Image Indexer.

- **HTTP Agent** - It is a tiny web-server that handles file download request from other DISCOVIR peers using HTTP protocol.

- **Feature Extractor** - It collaborates with the Plug-in Manager to perform various feature extraction and thumbnail generation of the shared image collection.

- **Image Indexer** - It indexes the image collection by content feature and carry out clustering to speed up the retrieval of images.

- **Bootstrap Server** - A server program maintaining an update list of currently available DISCOVIR peers.

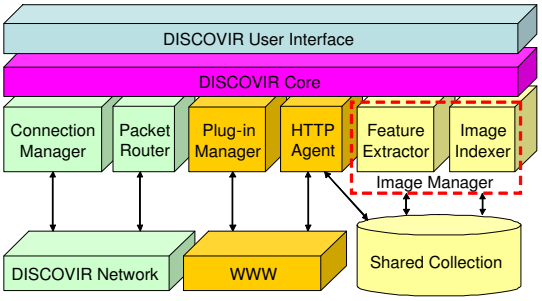

Figure 2: Architecture of DISCOVIR

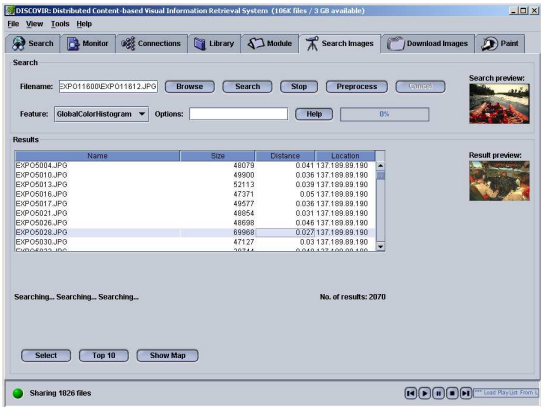

Figure 3: Screen Capture of DISCOVIR

3.1.1 Flow of Operations and Functionality of DISCOVIR Components

The following is a scenario walk-through to demonstrate the interaction between components. When a user chooses to preprocess his image collection, the *Feature Extractor* collaborates with *Plug-in Manager* to extract content feature and generate thumbnails from all images in the shared directory. The result is passed to *Image Indexer* for indexing and clustering purposes. In case when a new DISCOVIR client wants to join the network, the *Connection Manager* asks a *Bootstrap server* for a list of currently available DISCOVIR clients in order to hook up to the network. Once a user initiates a query for similar images, the *Feature Extractor* extracts feature from the example image instantly, *Packet Router* is responsible for assembling an ImageQuery message and sending out to the DISCOVIR network. For instance, when an ImageQuery message is received from other peers, the *Packet Router* checks for any duplication and propagates to other peers through DISCOVIR network. Meanwhile, it passes the message to *Image Indexer* for searching similar images. Upon similar images are found, an ImageQueryHit message is assembled and passed to *Packet Router* for replying the initiating peer. When ImageQueryHit messages return to the initiating peer, its *HTTP Agent* downloads thumbnails or full size images from other peers upon receiving user request from the user interface.

Preprocessing

Plug-in Manager is responsible for contacting web-site of DISCOVIR to inquire the list of available feature extraction modules, it will download and install selected modules upon user request. Currently, DISCOVIR supports various feature extraction methods in color and texture categories such as AverageRGB, GlobalColorHistogram, ColorMoment, Co-occurrence matrix, etc [9]. All feature extraction modules strictly follow a predefined interface in order to realize the polymorphic properties of switching between different plug-ins dynamically, see Fig. 4.

Feature Extractor will extract feature and generate thumbnails for all images in the shared collection by using a particular feature extraction module. Let I represent a raw image data, f be the feature extraction method, the feature extractor perform the task illustrated in Eq. 1,

$$f : I \times \theta \rightarrow \vec{I} \tag{1}$$

where θ is the feature extraction parameter and $\vec{I}$ is the extracted feature vector. Image Indexer will then index the image collection using the multi-dimensional feature vectors $\vec{I}$ in order to answer an incoming query. It also clusters the set of feature vector for the sake of improving query efficiency by acquiring statistical distribution information of the local image collection.

Compared with the centralized web-based CBIR approach, sharing the workload of this computational costly task among peers by allowing them to store and index their own image collection helps solving the bottle-neck problem by utilizing distributed computing resources.

Connection Establishment

For a peer to join the DISCOVIR network, it connects to the Bootstrap Server using the Connection Manager. The Bootstrap Server is responsible for storing a finite list of IP address of peers currently in the DISCOVIR network and randomly picks an IP address to return to the peer. Once the IP address is received, the peer is able to hook up to the DISCOVIR network by connecting to the selected peer. In order to provide the Bootstrap Server with update list of IP address, the Connection Manager of each peer sends an alive message to the bootstrap server periodically after it has connected to the DISCOVIR network.

Query Message Routing

After a peer joins the DISCOVIR network, it may initiate a query for similar images. The Feature Extractor processes the query image instantly and assemble an ImageQuery message to be sent out through Packet Router. Likewise, when other peers receive the ImageQuery messages, they need to perform two operations, *Query Message Propagation* and *Local Index Look Up*.

- **Query Message Propagation** - In order to prevent query messages from looping forever in the DISCOVIR network, two mechanisms are inherited from Gnutella, namely, the Gnutella replicated message checking rule and Time-To-Live (TTL) of messages. The replicated message checking rule can prevent a peer from propagating the same query message again. The TTL mechanism constrains the horizon of a query message able to reach. After these two checkings, the query message will be propagated to linked peers through Packet Router.

- **Local Index Look Up** - The peer searches its local index of shared files for similar images using the Image Indexer. Once similar images are retrieved, the peer delivers an ImageQueryHit message back to the requester through Packet Router. Since images indexing is performed on the peer in preprocessing stage, the searching time can be speeded up.

Query Result Display

When an ImageQueryHit message returns to the requester, user will obtain a list detailing the location and size of matched images. In order to retrieve the query result, the HTTP Agent will download thumbnails or full size image from the peer using HTTP protocol. On the other hand, HTTP Agent in other peers will serve as a web server to deliver the requested images. This HTTP Agent is essential for integration to WWW which will be described later in detail.

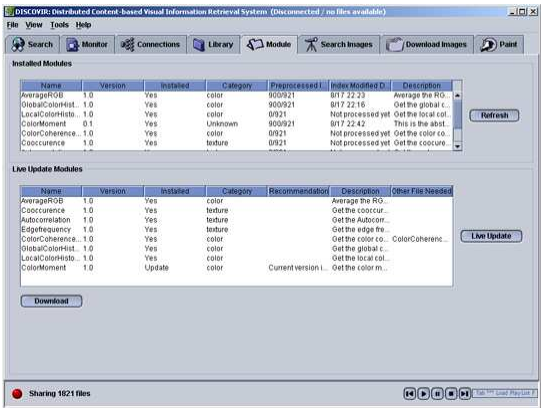

Figure 4: Downloading plug-in module for DISCOVIR

3.1.2 Gnutella Message Modification

The DISCOVIR system is built compatible to the Gnutella network. In order to support the image query functionalities mentioned above, two types of messages are added. They are:

- **ImageQuery** - Special type of the Query message. It is to carry name of feature extraction method and feature vector of query image, see Fig. 5

- **ImageQueryHit** - Special type of the QueryHit message. It is to respond to the ImageQuery message, it contains the location, filename and size of similar images retrieved, and their similarity measure to the query. Besides, the location information of corresponding thumbnails are added for the purpose of previewing result set in a faster speed, see Fig. 6

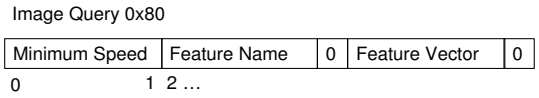

Figure 5: ImageQuery message format

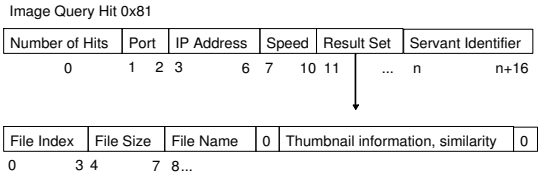

Figure 6: ImageQueryHit message format

3.2 Peer Clustering and Firework Query Model

Our design goal for DISCOVIR is to improve data lookup efficiency for content-based search in a decentralized and unstructured P2P environment, while keeping a simple network topology and number of message passing to a minimum. The DISCOVIR is built compatible to the Gnutella network. There are two types of connections in DISCOVIR, namely ***random*** and ***attractive***. Attractive connections are to link peers sharing similar data together. We perform peer clustering at the level of overlaying network topology instead of locally shared data, thus content-based query routing is realizable to improve query efficiency. As a result, it manages to be scalable when network grows. We have implemented a prototype version of DISCOVIR, built on top of LimeWire [13] with content-based image searching capability.

3.3 Peer Clustering

With the inherent nature of DISCOVIR network, we apply notation in graph theory to model it, see Table. 1. For the sake of generality, we try to keep this in high level of abstraction. In the actual realization, we choose the vector model in information retrieval literature as the underlying data structure for representing data. Here are some definitions:

Definition 1 *We consider information shared by a peer can be represented in a multi-dimension point based on its content, and the similarity among files is based on the distance measure between data points. Consider*

$$f \quad : \quad c_v \rightarrow \vec{c_v} \tag{2}$$

$$f \quad : \quad q \rightarrow \vec{q} \tag{3}$$

f is the mapping function from file c_v to a vector $\vec{c_v}$. In the notion of image processing, c_v is the raw image data, f is a specific feature extraction method, $\vec{c_v}$ is the extracted feature vector characterizing the image. Likewise, f is also used to map a query q to a query vector $\vec{q}$, to be sent out when user makes a query.

Table 1: Definition of Terms

Symbol	Description
$G\{V, E\}$	The P2P network, with V denoting the set of peers and E denoting the set of connection
$E = \{E_r, E_a\}$	The set of connections, composed of random connections, E_r and attractive connections, E_a.
$e_a = (v, w, sig_v, sig_w)$, $v, w \in V, e_a \in E_a$	The attractive connection between peers v, w based on sig_v and sig_w
$\|V\|$	Total number of peers.
$\|E\|$	Total number of connections.
$Horizon(v, t) \subseteq V$	Set of peers reachable from v within t hops
$SIG_v, v \in V$	Set of signature values characterizing the data shared by peer v
$D(sig_v, sig_v)$, $v, w \in V$	Distance measure between specific signature values of two peers v and w.
$D_q(sig_v, q)$ $sig_v \in SIG_v$	Distance measure between a query q and peer v based on sig_v.
$C = \{C_v : v \in V\}$	The collection of data shared in the DISCOVIR network.
C_v	The collection of data shared by peer v, which is a subset of C.
$REL(c_v, q)$, $c_v \in C_v$	A function determining relevance of data c_v to a query q. 1-relevant, 0-non-relevant

Definition 2 *SIG_v is the set of signature values representing data characteristic of peer v, with each sig_v representing each specific cluster of data. We define*

$$sig_v = (\vec{\mu}, \vec{\sigma}), \tag{4}$$

where $\vec{\mu}$ and $\vec{\sigma}$ are the statistical mean and standard deviation of the collection of data belonging to a subcluster, $C_v^{'}, C_v^{'} \subseteq C_v$. Thereafter, sig_v characterizes certain portion of data shared by peer p.

Definition 3 *$D(sig_v, sig_w)$ is defined as the distance measure between sig_v and sig_w, in other sense, the similarity between particular sub-cluster belonging to two different peers v and w. It is defined as,*

$$D(sig_v, sig_w) = ||\vec{\mu_v} - \vec{\mu_w}||. \tag{5}$$

$||\vec{\mu_v} - \vec{\mu_w}||$ is the Euclidean distance between centroid of two sub-cluster symbolized by sig_v, sig_w. With this formula, we define the data affinity of two peers, we will later use this to help organizing the network. The actual implementation of this function can be varied, like the five distance measures described in BIRCH [31], we choose D0 as the distance measure between two clusters.

Based on the above definitions, we introduce a peer clustering algorithm, to be used in the network setup stage, in order to help building the DISCOVIR as a self-organized network oriented in content affinity. It consists of three steps:

1. **Signature Value Calculation**–Every peer preprocess its data collection and calculates a set of signature values SIG_v to characterize its data properties. Whenever the shared data collection, C_v, of a peer changes, the signature value should be updated accordingly. The whole data collection of the peer will be divided into sub-clusters automatically by a clustering algorithm, e.g. k-means, competitive learning, and expectation maximization. The number of signature values is variable and is a trade-off between data characteristic resolution and computational cost.

2. **Neighborhood Discovery**–After a peer joins the DISCOVIR network by connecting to a random peer in the network, it broadcasts a signature query message, similar to that of ping-pong messages in Gnutella, to reveal the location and data characteristic of its neighborhood, $Horizon(v, t)$. This task is not only done when a peer first joins the network, it repeats every certain interval in order to maintain the latest information of other peers.

3. **Attractive Connection Establishment**–By acquiring the signature values of other peers, one can reveal the peer with highest data affinity (similarity) to itself, and make an attractive connection to link them up. When an existing attractive connection breaks, a peer should check its host cache, which contains signature values of other peers found in the neighborhood discovery stage, and reestablish the attractive connection using peer clustering algorithm again.

Having all peers joining the DISCOVIR network perform the three tasks described above, you can envision a P2P network with **self-organizing** ability to be constructed. Peers sharing similar content will be grouped together like a Yellow Pages. Based on this content similarity based clustering, we will delineate a more complex query strategy in the next section. The detail steps of peer clustering is illustrated in Algorithm 1, and Fig. 7 depicts the peer clustering.

Algorithm 1 Algorithm for peer clustering

```
Peer-Clustering(peer v, integer ttl)
for all sig_v ∈ SIG_v do
  for all w ∈ Horizon(v, t) do
    for all sig_w ∈ SIG_w do
      Compute D(sig_v, sig_w)
    end for
  end for
  E_a = E_a ∪ (v, w, sig_v, sig_w) having min(D(sig_v, sig_w))
end for
```

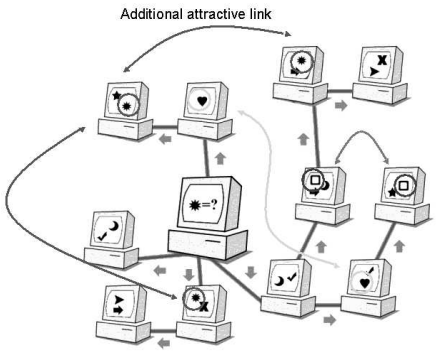

Figure 7: Illustration of peer clustering.

3.4 Firework Query Model

To make use of our clustered P2P network, we propose a content-based query routing strategy called Firework Query Model. In this model, a query message is routed selectively according to the content of the query. Once it reaches its designated cluster, the query message is broadcasted by peers through the attractive connections inside the cluster much like an exploding firework as shown in Fig. 8. Our strategy aims to minimize the number of messages passing through the network, reduce the workload of each computer and maximize the ability of retrieving relevant data from the peer-to-peer network.

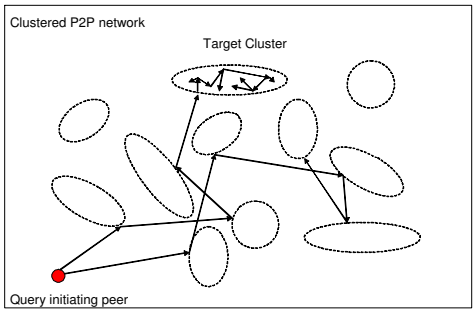

Figure 8: Illustration of firework query.

Here, we introduce the algorithm to determine when and how a query message is propagated like a firework in Algorithm 2. When a peer receives the query, it needs to carry out two steps:

1. **Shared File Look Up**–The peer looks up its shared information for those matched with the query. Let q be the query, and $\vec{q}$ be its vector representation, $REL(c_v, q)$ is the relevance measure between the query and the information c_v shared by peer v, it depends on a L_2 norm defined as,

$$REL(c_v, q) = \begin{cases} 1 & ||\vec{c_v} - \vec{q}|| \leq T \\ 0 & ||\vec{c_v} - \vec{q}|| > T, \end{cases}$$

where T is a threshold defining the degree of result similarity a user wants. If any shared information within the matching criteria of the query, the peer will reply the requester. In addition, we can reduce the number of $REL(c_v, q)$ computations by performing local clustering in a peer, thus speeding up the process of query response.

2. **Route Selection**–The peer calculates the distance between the query and each signature value of its local clusters, sig_v , which is represented as,

$$D_q(sig_v, q) = \sum_i \frac{q_i - \mu_i}{\sigma_i}, \quad sig_v = (\mu, \sigma). \qquad (6)$$

If none of the distance measure between its local clusters' signature value and the query, $D_q(sig_v, q)$, is smaller than a preset threshold, θ, the peer will propagate the query to its neighbors through random connections. Otherwise, if one or more $D_q(sig_v, q)$ is within the threshold, it implies the query has reached its target cluster. Therefore, the query will be propagated through corresponding attractive connections much like an exploding firework.

In our model, we retain two existing mechanisms in Gnutella network for preventing query messages from looping forever in the distributed network, namely, the Gnutella replicated message checking rule and Time-To-Live (TTL) of messages [7]. There is a modification on DISCOVIR query messages from the original Gnutella messages. In our model, the TTL value is decremented by one with a different probability when the message is forwarded through different types of connection. For random connections, the probability of decreasing TTL value is 1. For attractive connections, the probability of decreasing TTL value is an arbitrary value in [0, 1] called Chance-To-Survive (CTS). This strategy can reduce the number of messages passing outside the target cluster, while more relevant information can be retrieved inside the target cluster because the query message has a greater chance to survive depending on the CTS value. In the experiment described thereafter, we choose CTS to be 0, i. e. TTL of query message is not decremented when forwarded through attractive link.

Algorithm 2 Algorithm for the Firework Query Model

```
Firework-query-routing (peer v, query q)
for all sig_v ∈ SIG_v do
  if D_q(sig_v, q) < θ (threshold) then
    if rand() > CTS then
      q_ttl = q_ttl − 1
    end if
    if q_ttl > 0 then
      propagate q to all e_a(a, b, c, d) where a = v, c = sig_v or b =
      v, d = sig_v (attractive link)
    end if
  end if
end for
if Not forwarding to attractive link then
  q_ttl = q_ttl − 1
  if q_TTL > 0 then
    forward q to all e_r(a, b) where a = v or b = v (random link)
  end if
end if
```

4 Implementations and Experimental Results

We have built a beta release of DISCOVIR. The client is implemented in Java on top of the LimeWire open source project. LimeWire is a Java implementation of Gnutella Network and its key components are under GNU Public License. We have released DISCOVIR as a free software and intend to release the source code in near future. Figure 9 illustrates the top ten results panel, it makes use of the generated thumbnails in the preprocessing stage to avoid heavy traffic caused by downloading full size image directly, while users can preview lower quality images, Moreover, drawpad functionality is incorporated in DISCOVIR. Besides supplying an example image for query, users may draw their own sketch and search. Figure 4 shows the plug-in module download page of DISCOVIR. Currently, DISCOVIR supports AverageRGB, GlobalColorHistogram, LocalColorHistogram, ColorMoment and ColorCoherenceVector on color-based feature, Co-occurrence matrix, Auto correlation, Edge frequency and Primitive length on texture-based feature. We are working on shape-based feature and encourage contributions from the community by implementing plug-ins.

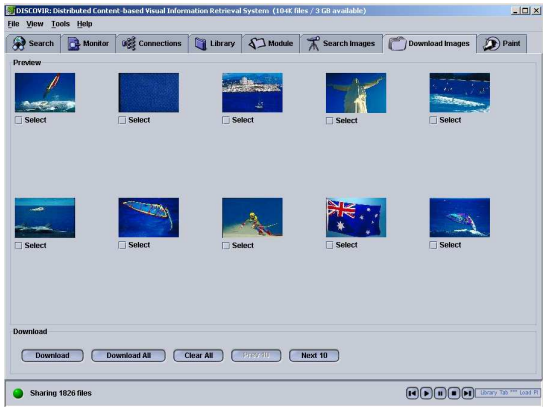

Figure 9: Top 10 preview of images

4.1 Simulation

As it is difficult or even impossible to carry out performance evaluation by running several thousands of DISCOVIR client simultaneously, we choose to predict the performance and scalability by simulation. The two main goals of our proposed routing algorithm in P2P network are to: 1) increase the percentage of desired result retrieved (Recall - R) and 2) decrease the percentage of peers visited (Visited - V) for every query. Therefore, we define the query efficiency as R/V, the higher this value, the higher the efficiency, and we investigate how this quantity varies with different number of total peers. In the experiments, we generate a certain number of peers and randomly assign two to four classes of data points to each of them. Then, we simulate the setting up of DISCOVIR network using the peer clustering algorithm, Later, We initiate a query starting from a randomly selected peer and the retrieved data point is treated as relevant result if it belongs to the same class as the query data point. We simulate the environment using both synthetic data and real data. For the synthetic data, each class of data points follows a Gaussian distribution and we generate 200 classes of data points totally. For the real data, each class of data points is extracted from a category in CorelDraw's Image Collection using the Color Moment plug-in module in DISCOVIR. Details of experiment parameters are listed in Table 2. Fig. 10 shows the in-degree distribution of peers under the simulated network. It is similar to the real world Gnutella graph as indicated in [15] with a two-segment power-law distribution.

Table 2: Parameters of simulation

properties	synthetic and image
cardinality of dataset	20000
number of classes	200
number of peers	{500,1000,2000,4000, 8000,10000,140000, 20000}
dimensionality	9
number of classes per peer	[2..4]
number of local clusters	{1,3}
clustering method	Expectation Maximization
avg. # of random links	1
avg. # of attractive links	depends
max. in-degree of a node	40

4.2 R/V Against Network Size

We measure the query efficiency against the number of peers with four different routing methods or data sets: (1) Brute Force Search (BFS), (2) FQM with 1 local cluster per peer using synthetic data ,(3) FQM with 3 local clusters per peer using synthetic data and (4) FQM with 1 local cluster using real data. We did not simulate BFS because the performance of flooding algorithm is equivalent to random probing [**?**]. As seen in Fig. 11, FQM outperforms BFS algorithm and it can perform even better if an appropriate number of signature values per peer is used. As expected, recall and visited peers percentage in BFS are more or less equal because data classes are evenly distributed among peers, the more peers visited, the more desired data retrieved. The curve of FQM follows a bell shape with a long tail. Query efficiency increases at first due to two reasons: 1) the percentage of peers visited is inversely proportional to the number of peers when TTL is fixed and 2) FQM advances the recall percentage when the query message reaches the target cluster. When the number of peers increases further, a query might not reach its target cluster, so query efficiency starts to drop. The result shows that the improvement in FQM is reduced when real data is used, yet, it has on average about 20% improvement compared to BFS. This is when the real

data cannot be clustered appropriately, see Fig. 13 and explanation thereafter.

4.3 R/V Against TTL

Fig. 12 shows the measurement of query efficiency against TTL of query message. Refer to the previous experiment, 3 cases are tested, namely, (1), (2) and (4). The curve of FQM more or less follows bell shape because 1) When TTL value is low, the chance for first k walk to hit the target cluster is low and 2)further increasing the TTL value increases the chance to hit target cluster, thus R/V reaches a maximum. It tails down because large TTL floods the whole network with query message, while V increases, R/V in turn drops. From this experiment, we observe that our proposed FQM depends on the choice of TTL in order to achieve best performance. In this simulation, we conclude that choosing 6 to be the TTL perform the best when the network consists of 8000 peers.

Fig. 13 might account for the inefficiency of Peer Clustering and FQM when using real data. From the Self-organizing map visualization of both synthetic data and real data, we see that clusters in synthetic dataset are more sparsely distributed, while clusters in real dataset are closely situated. Referring to Section 3.2, the distance measure between two peers is the distance between the centroids of their clusters. As a result, global clustering for real dataset might be worse than synthetic dataset, thus, query efficiency is generally lower.

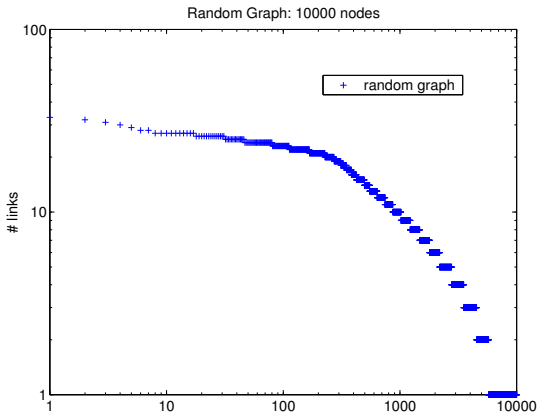

Figure 10: Distribution of in-degree of nodes in our simulated network

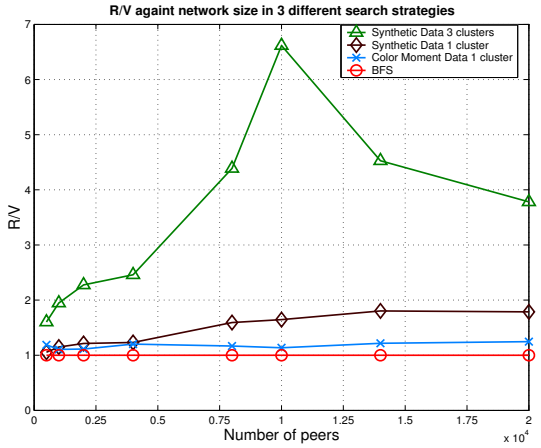

Figure 11: R/V against number of peers

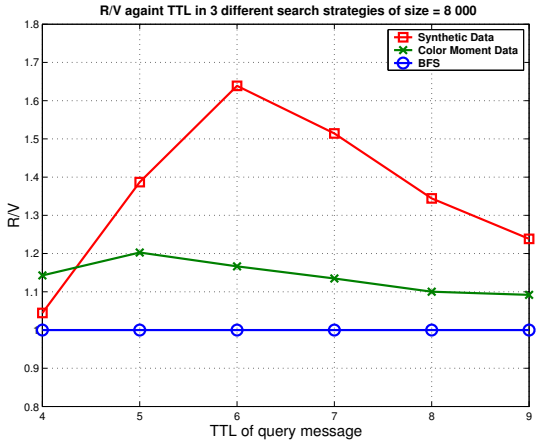

Figure 12: R/V against TTL of query message

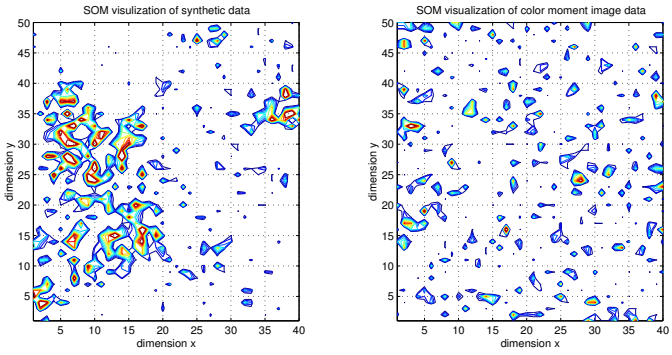

Figure 13: SOM visualization of data used in simulation

5 Future Works

So far, we have built CBIR on P2P and proposed a basic framework for facilitating content-based search in P2P. The proposed method is not only applicable to image, but also other media, such as text and audio. Moreover, Gnutella 2 [8] will be coming out soon, we ought to figure out whether our proposed algorithm is still compatible or not. Comparing with prevalent researches, our proposed algorithm need more in depth mathematical analysis to model the observations from simulation, the resemblance between P2P and Neural Networks might be an opportunity for us to conduct a formal analysis on P2P.

Relevance feedback is commonly used in information retrieval system, especially for CBIR, while current retrieval methods in P2P focus on the one-shot approach. We envision the potential of adding relevance feedback

in P2P system using the resemblance between P2P and Neural Networks. Competitive Hebbian Learning in the literature offers a way to describe organization of data given a certain number of input signal (observation of data). It shines light on our proposed P2P Decentralized Quasi-structured architecture for organizing the network. Users' relevance feedback might be considered as input signal, while we use Competitive Hebbian Learning rule to train the network.

6 Conclusion

In this paper, we outline the design and implementation of DISCOVIR, a CBIR software running on P2P environment. Moreover, we propose a peer clustering and content-based query routing strategy to retrieve information based on their content efficiently over the P2P network. Our algorithm is resilient to nodes entering and leaving the network frequently, thus, applicable in an environment of transient population of nodes like the Internet. We verify our proposed strategy by simulations with different parameters to investigate the performance changes subject to different network size and TTL of query message. We argue that current researches in Decentralized Unstructured P2P cannot be applied in the notion of CBIR, thus, we show that our FQM outperforms the BFS method, the only existing algorithm available for comparison, in both network traffic cost and query efficiency measure.

References

[1] R. J. J. Bayardo, R. Agrawal, D. Gruhl, and A. Somani. Youserv: A web-hosting and content sharing tool for the masses. In *Proceedings of 11th World Wide Web Conference*, May 2002.

[2] E. Cohen and S. Shenker. Replication Strategies in Unstructured Peer-to-Peer Networks. In *Proceedings of SIGCOMM'02*, Pittsburgh, Pennsylvania, USA, August 19-23 2002.

[3] G. Coulouris, J. Dollimore, and T. Kindberg. *Distributed Systems Concepts and Design*. Addison-Wesley, third edition, 2001.

[4] DIStirbuted COntent-based Visual Information Retrieval. http://www.cse.cuhk.edu.hk/∼miplab/discovir.

[5] C. Faloutsos, R. Barber, M. Flickner, J. Hafner, N. W., D. Petkovic, and W. Equitz. Efficient and Effective Querying by Image Content. *Journal of Intelligent Information Systems: Integrating Artificial Intelligence and Database Technologies*, 3(3-4):231–262, 1994.

[6] The Freenet homepage. http://freenet.sourceforge.net.

[7] The Gnutella homepage. http://www.gnutella.com.

[8] Gnutella2 Homepage. http://www.gnutella2.com/.

[9] R. C. Gonzalez and R. E. Woods. *Digital Image Processing*. Addison Wesley, first edition, 1992.

[10] KaZaA file sharing network. http://www.kazaa.com/.

[11] I. King and Z. Jin. Relevance feedback content-based image retrieval using query distribution estimation based on maximum entropy principle. In *Proceedings to the International Conference on Neural Information Processing (ICONIP2001)*, pages 699–704, November 14-18 2001.

[12] T. K. Lau and I. King. Montage: An image database for the fashion, clothing, and textile industry in Hong Kong. In *Proceedings of the Third Asian Conference on Computer Vision (ACCV'98)*, Lecture Notes in Computer Science, January 4-7, 1998.

[13] The LimeWire homepage. http://www.limewire.com.

[14] Modern Peer-to-Peer File-Sharing over the Internet. http://www.limewire.com/index.jsp/p2p.

[15] Q. Lv, P. Cao, E. Cohen, K. Li, and S. Shenker. Search and Replication in Unstructured Peer-to-Peer Networks. In *Proceedings of the 16th annual ACM International Conference on supercomputing*, 2002.

[16] S. Mehrotra, Y. Rui, M. Ortega, and T. Huang. Supporting Content-based Queries over Images in MARS. In *Proc. IEEE Int. Conf. Multimedia Computing and Systems*, pages 632–633, 1997.

[17] The Morpheus homepage. http://www.musiccity.com.

[18] The Napster homepage. http://www.napster.com.

[19] A. Natsev, R. Rastogi, and K. Shim. WALRUS: A Similarity Retrieval Algorithm for Image Databases. In *Proc. SIGMOD, Philadelphia, PA*, 1999.

[20] C. H. Ng and K. C. Sia. Peer Clustering and Firework Query Model. In *Poster Proc. of The 11th International World Wide Web Conference*, May 2002.

[21] A. Pentland, R. W. Picard, and S. Sclaroff. Photobook: Tools for Content-based Manipulation of Image Databases. In *Proc. SPIE*, volume 2185, pages 34–47, February 1994.

[22] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. A Scalable Content-Addressable Network. In *In Proc. ACM SIGCOMM*, August 2001.

[23] A. Rowstron and P. Druschel. Pastry: Scalable, decentralized object location and routing for large-scale peer-to-peer systems. In *Proc. of the 18th IFIP/ACM International Conference on Distributed Systems Platforms*, November 2001.

[24] Y. Rui, T. S. Huang, and S.-F. Chang. Image Retrieval: Current Techniques, Promising Directions and Open Issues. *Journal of Visual Communication and Image Representation*, 10:39–62, April 1999.

[25] The Search for Extraterrestrial Intelligence homepage. http://www.setiathome.ssl.berkeley.edu.

[26] K. C. Sia, C. H. Ng, C. H. Chan, and I. King. P2P Content-Based Query Routing Using Firework Query Model. In *The 2nd International Workshop on Peer-to-Peer Systems, submitted*, Feb 2003.

[27] A. W. Smeulders, M. Worring, S. Santini, A. Gupta, and R. Jai. Content-Based Image Retrieval at the End of the Early Years. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 22(12):1349–1380, December 2000.

[28] J. R. Smith and S. F. Chang. An Image and Video Search Engine for the World Wide Web. In *Proc. SPIE*, volume 3022, pages 84–95, 1997.

[29] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications. In *Proc. of ACM SIGCOMM*, pages 149–160, August 2001.

[30] J. Z. Wang, G. Li, and G. Wiederhold. SIMPLIcity: Semantics-sensitive Integrated Matching for Picture LIbraries. In *IEEE Trans. on pattern Analysis and Machine Intelligence*, volume 23, pages 947–963, 2001.

[31] T. Zhang, R. Ramakrishnan, and M. Livny. Birch: An efficient data clustering method for very large databases. In *In Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data*, pages 103–114, 1996.

[32] B. Zhao, J. Kubiatowicz, and A. Joseph. Tapestry: An infrastructure for fault-tolerant wide-area location and routing. Technical report, Computer Science Division, U.C. Berkeley, April 2001.