Distributed Content-Based Visual Information Retrieval System on Peer-to-Peer Networks

IRWIN KING, CHEUK HANG NG, and KA CHEUNG SIA
The Chinese University of Hong Kong

With the recent advances of distributed computing, the limitation of information retrieval from a centralized image collection can be removed by allowing distributed image data sources to interact with each other for data storage sharing and information retrieval. In this article, we present our design and implementation of DISCOVIR: DIStributed COntent-based Visual Information Retrieval system using the Peer-to-Peer (P2P) Network. We describe the system architecture and detail the interactions among various system modules. Specifically, we propose a Firework Query Model for distributed information retrieval, which aims to reduce the network traffic of query passing in the network. We carry out experiments to show the distributed image retrieval system and the Firework information retrieval algorithm. The results show that the algorithm reduces network traffic while increases searching performance.

Categories and Subject Descriptors: H.3.4 [**Information Storage and Retrieval**]: Systems and Software—*Distributed systems*

General Terms: Design, Performance

Additional Key Words and Phrases: Peer-to-peer (P2P) network, information retrieval, peer clustering, intelligent query routing, content-based image retrieval (CBIR)

1. INTRODUCTION

In recent years, Peer-to-Peer (P2P) applications such as Gnutella [Gnutella], Napster [Napster], and Freenet [Freenet] have demonstrated the significance of distributed information sharing systems. These applications accomplish tasks that are difficult for the traditional centralized computing models to achieve. For example, by distributing data storage over networked computers, one can have virtual data storage that is possibly many magnitudes greater than what can be stored in a local computer. In addition, one may also envision data security by distributing pieces of an encrypted file over many computers. By doing so, one imposes a difficult barrier for an intruder to overcome because one

The work described in this article is supported in part by grants from the Research Grants Council of the Hong Kong Special Administration Region, China.
Authors' address: Department of Computer Science and Engineering, The Chinese University of Hong Kong, Shatin, New Territories, Hong Kong; email: {king,chng,kcsia}@cse.cuhk.edu.hk.

needs to break into several computers before getting the file [Kubiatowicz and Anderson 2002]. Likewise, one may also distribute the computation among different computers to achieve a high performance throughput [SETI]. Although all the above can be achieved through a centralized coordinator in the existing network, Peer-to-Peer (P2P) Network offers a completely decentralized and distributed paradigm on top of the physical network, which avoids the coordinator bottleneck problem.

Currently, most content-based image retrieval (CBIR) systems are based on the centralized computing model. Some are stand-alone applications while others are web-based systems. We foresee the advantages of using the P2P network for CBIR in several ways. First, with an increased number of users joining the P2P network, the image collection will grow drastically due to individual contributions. This gives diversity and variety. Second, it overcomes the scalability problem of image retrieval by using a decentralized retrieval algorithm. Although this idea sounds interesting, many difficulties remain unresolved. For example, there is no centralized storage of feature vector index of the images. Hence, there must be some standardized feature extraction methods agreed upon among the peers before applying CBIR to P2P networks.

In this article, we present the design and implementation of building the DISCOVIR (DIStributed COntent-based Visual Information Retrieval) [MIPLAB] system on the P2P network for users to share and retrieve images. In particular, we tackle the scalability problem using the proposed Firework Query Model by routing query messages selectively to reduce the network traffic.

In the following, we first review current issues in CBIR and P2P in Section 2. We then present the architecture of DISCOVIR in Sections 3.1 and 3.2 and the detail algorithm of the Firework Query Model of our proposed system in Sections 3.3 and 3.4. We report and analyze our experimental results in Section 4 and give our final remarks and conclusion in Section 5.

2. BACKGROUND

Since the mid 1990s, many CBIR systems have been proposed and developed, including QBIC [Faloutsos et al. 1994], WebSEEK [Smith and Chang 1997], WBIIS [Wang et al. 1998], SIMPLIcity [Wang et al. 2001], MARS [Mehrotra et al. 1997], NeTra [Ma and Manjunath 1997], Photobook [Pentland et al. 1994], AMORE [Mukherjea et al. 1999], Virage [Gupta and Jain 1997], and other systems for domain-specific applications [Lau and King 1998; King and Jin 2001]. The images are preprocessed and are represented in the form of a feature vector with their similarity being based on the distance between the feature vectors. 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 querying are all done in a centralized fashion, which can be computationally intensive, and is difficult to scale up. As indicated by several researchers [Rui et al. 1999; Smeulders et al. 2000], one of the promising future trends in CBIR includes distributed computing on 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

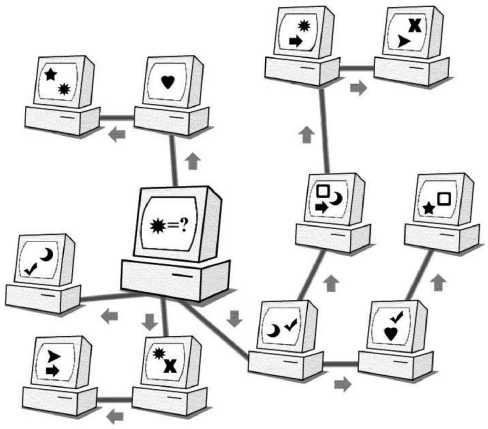

Fig. 1. Illustration of information retrieval in a P2P network.

bottleneck problem by distributed processing of image information and image retrieval.

Peep-to-Peer (P2P) Network is a recently evolved paradigm for distributed computing. Emerging P2P networks or their implementations such as Gnutella [Gnutella], Napster [Napster], Freenet [Freenet], LimeWire [LimeWire], and eDonkey [eDonkey] offer the following advantages:

(1) **Distributed Resource**—The storage, information and computational cost can be distributed among the peers, allowing many individual computers to achieve a higher throughput [SETI].
(2) **Increased Reliability**—The P2P network increases reliability by eliminating reliance on centralized coordinators. In other words, the P2P network can still be operational even after a certain portion of peers are down [Cooper and Gracia-Molina 2002].
(3) **Comprehensiveness of Information**—The P2P network has the potential to reach every computer on the Internet.

Figure 1 shows an example of a P2P network. In this example, different files are shared by different peers. When a peer initiates a search for a file, it broadcasts a query request to its connecting peers. Its peers then propagate the request to their own peers and this process continues. Unlike the client-server architecture of the web, the P2P network aims at allowing individual computers that join and leave the network frequently to share information directly with each other without the help of dedicated servers. Each peer acts as a server and as a client simultaneously. 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 while each peer looks up its locally shared collection and responds to queries. Plainly, this model is wasteful because peers are forced to handle irrelevant query messages. This type of search is called a Breadth-First Search (BFS).

There are several solutions proposed to solve the query broadcasting problem.[1] Chord [Stoica et al. 2001], CAN [Ratnasamy et al. 2001], Pastry [Rowstron and Druschel 2001] and Tapestry [Zhao et al. 2001] tackle it by distributing the index storage into different peers, thus sharing the workload of a centralized index server. Distributed infrastructure of both CAN and Chord use Distributed Hash Table (DHT) technique to map a filename to a key; each peer is responsible for storing a certain range of (key, value)[2] pairs. When a peer looks for a file, it hashes the filename to a key and asks the peers responsible for this key for the actual storage location of that file. Chord models the key as an m-bit identifier and arranges the peers into a logical ring topology to determine which peer is responsible for storing which pair (key, value). CAN models the key as a point on a d-dimension Cartesian coordinate space, while each peer is responsible for the pairs (key, value) 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). Some extensions of DHTs to perform content-based retrieval and textual similarity matches are proposed in Tang et al. [2002] and Harren et al. [2002]. Although DHTs are elegant and scalable, their performance under the dynamic conditions of prevalent P2P systems is still unknown due to the penalty incurred in joining and leaving [Ratnasamy et al. 2002].

As DHTs mandate a specific network structure and incur a certain penalty on joining and leaving the network, some researchers propose methods that operate under the prevalent dynamic P2P environment, for example, Gnutella. Crespo [Crespo and Gracia-Molina 2002] proposed a routing indices approach for retrieving text documents in P2P systems. Under this scheme, each peer maintains a routing index, which is used to assist in forwarding queries to peers that are supposed to contain more documents of the same category as the queries. This method requires all peers to agree upon a set of document categories. Sripanidkulchai et al. [2003] proposed the use of shortcuts to connect a peer to another one from which it has previously downloaded documents. Evaluations are done based on simulation using text document retrieval, and promising results are shown. Our proposed method targets content-based image retrieval in a P2P network. It makes use of strategies similar to routing indices and shortcuts to reduce network traffic and computation time, a simplier version of the algorithm and system architecture was proposed in Ng and Sia [2002], Ng et al. [2003], and Sia et al. [2003]. Similar problems of CBIR in distributed databases have been studied in Chang et al. [1998]. In essence, more studies of P2P systems are needed.

3. CONTENT-BASED IMAGE RETRIEVAL OVER PEER-TO-PEER NETWORK

In Sections 3.1 and 3.2 we detail the architecture of our system (DISCOVIR) and the modification made on the current Gnutella network in order to perform

[1]All peers are required to handle numerous query messages and most of them are irrelevant, it is annoying for low-bandwidth peers.

[2]The key is the hash value of filename, and the value includs the filename and location of a file.

CBIR in P2P network. As every query is broadcast to every peer under the current Gnutella network protocol, each peer has to waste resources in handling irrelevant queries. This query message flooding within the network increases the traffic. In Section 3.3, we tackle this problem by clustering peers that share similar image data, which helps one to search more efficiently. Based on this network topology, in Section 3.4, we present the Firework Query Model to reduce network traffic and enhance query performance.

3.1 How Does DISCOVIR Perform Content-Based Image Retrieval?

We briefly outline the process of sharing and retrieving images in DISCOVIR. First, each peer is responsible to perform feature extraction, for example, color, texture, shape, and so on, on its shared images in each peer using the DISCOVIR client program. With this program, each peer maintains its local index of feature vectors of its image collection. When a peer, the requester, initiates a query by giving an example image and a particular feature extraction method, it performs feature extraction on the example image and sends the feature vector, contained in a query message, to all its connecting peers. Consequently, other peers compare this query to their feature vector index. Based on a distance measure, they find a set of similar images and return results back to the requester. Likewise, these peers will propagate the query to their connecting peers and this process continues to query other peers in a large portion of the network.

DISCOVIR uses a plug-in architecture to support different feature extraction methods. User may choose to download a plug-in in the format of compiled Java bytecode if they want to perform CBIR based on that particular feature extraction method. For a screen capture of the DISCOVIR client program, see Figure 2, which can be downloaded from the site [MIPLAB]. In the following, we describe the overall architecture of DISCOVIR, which operates on the current Gnutella network, and the modification of query messages.

3.1.1 *Gnutella Message Modification.* The DISCOVIR system is compatible with the Gnutella (v0.4) protocol [Gnutella]. In order to support the image query functionalities mentioned, two types of messages are added. They are:

—**ImageQuery**—Special type of Query message. It carries the name of the feature extraction method and the feature vector of the query image; see Figure 3.

—**ImageQueryHit**—Special type of QueryHit message. It responds to the ImageQuery message. It contains the location, filename and size of similar images retrieved, and their similarity measure to the query. In addition, the location information of corresponding thumbnails is added for the purpose of previewing the result set in a faster speed; see Figure 4.

3.2 Architecture of DISCOVIR

In this section, we describe the architecture of a DISCOVIR client and the interaction within modules in order to perform CBIR over a P2P network.

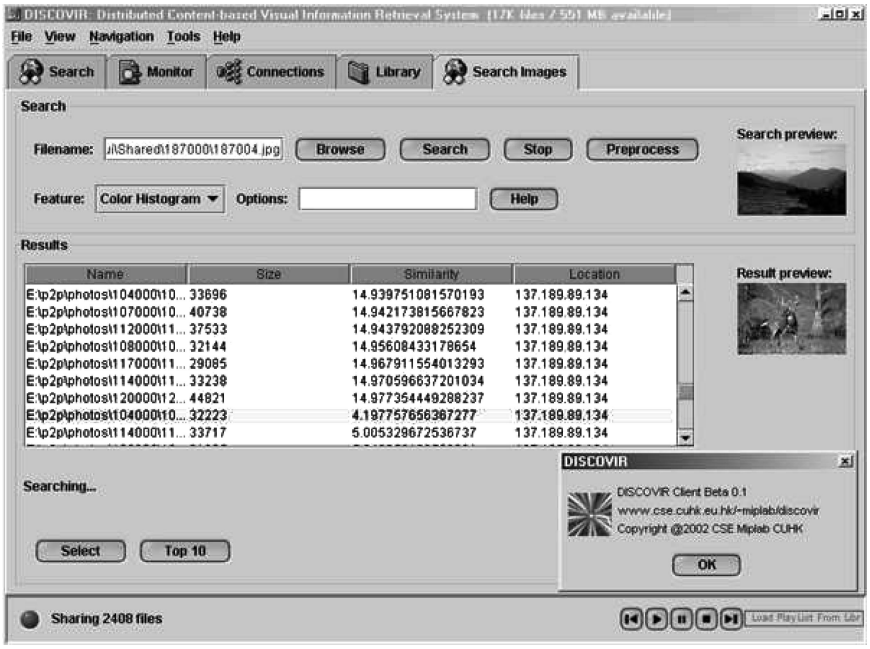

Fig. 2. Screen-shot of DISCOVIR.

Fig. 3. ImageQuery message format.

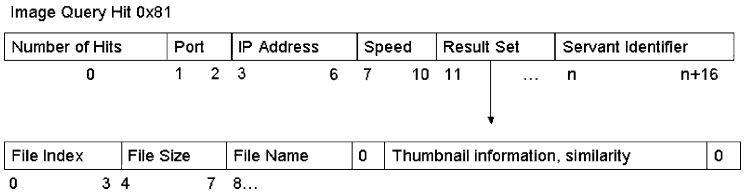

Fig. 4. ImageQueryHit message format.

Figure 5 depicts the key components and their interaction in the architecture of a DISCOVIR client. As DISCOVIR is built based on the LimeWire open source project [LimeWire], the operations of Connection Manager, Packet Router, and HTTP Agent remain more or less the same with some additional functionalities to improve the query mechanism used in the original Gnutella network. The 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 2 shows a screen capture of DISCOVIR in the image

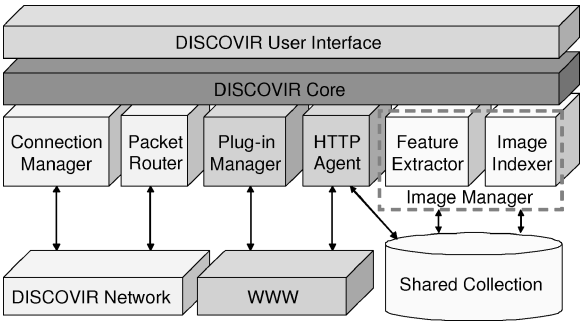

Fig. 5. Architecture of DISCOVIR.

search page. Here are brief descriptions of the six major components:

—**Connection Manager**—is responsible for setting up and managing the TCP connection between DISCOVIR clients.
—**Packet Router**—controls the routing, assemble and disassemble messages between the DISCOVIR network and different components of the DISCOVIR client program.
—**Plug-in Manager**—coordinates the download and storage of different feature extraction plug-ins and their interaction with Feature Extractor and Image Indexer.
—**HTTP Agent**—is a tiny web-server that handles file download requests from other DISCOVIR peers using the HTTP protocol.
—**Feature Extractor**—collaborates with the Plug-in Manager to perform feature extraction and thumbnail generation from the shared image collection. It involves two main functions:
  —**Preprocessing**—extracts the feature vector of shared images in order to make the collection searchable in the network.
  —**Real Time Extraction**—extracts the feature vector of the query image on-the-fly and passes the query to Packet Router.
—**Image Indexer**—indexes the image collection by content feature and carries out clustering to speed up the retrieval of images.

3.2.1 *Flow of Operations.* The four main steps of performing CBIR in DISCOVIR are listed in detail as follows:

(1) **Preprocessing**
The *Plug-in Manager* module is responsible for contacting the DISCOVIR control website to query the list of available feature extraction modules. It will download and install selected modules upon user's request. Currently, DISCOVIR supports various feature extraction methods in color and texture categories such as AverageRGB, GlobalColorHistogram, ColorMoment, Co-occurrence matrix, and so on. All feature extraction modules strictly follow a predefined API in order to realize the polymorphic properties of switching between different plug-ins dynamically.

The *Feature Extractor* module extracts features and generates thumbnails for all shared images using a particular feature extraction method chosen by user. The *Image Indexer* module then indexes the image collection using the extracted multi-dimensional feature vectors. Compared with the centralized web-based CBIR approach, sharing the workload of this computationally costly task among peers by allowing them to store and index their own image collection helps to solve the bottle-neck problem by utilizing distributed computing resources.

(2) **Connection Establishment**
For a peer to join the DISCOVIR network, the *Connection Manager* module asks the Bootstrap Server, which is program storing and answering a list of peers currently in the network, for peers available for accepting incoming connections. Once the IP address is known, the peer hooks up to the DISCOVIR network by connecting to currently available peers.

(3) **Query Message Routing**
When a peer initiates a query for similar images, the *Feature Extractor* module processes the query image instantly and assembles an *ImageQuery* message (as shown in Figure 3) to be sent out through the *Packet Router* module. 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**—When propagating the query message, the *Packet Router* module of DISCOVIR employs two checking rules adopted from Gnutella network. In order to prevent messages from looping forever in the DISCOVIR network, they are (1) Gnutella replicated message checking rule and (2) Time-To-Live (TTL) of messages respectively. The replicated message checking rule prevents a peer from propagating the same query message again. The TTL mechanism constrains the reachable horizon of a query message.

—**Local Index Look Up**—The peer searches its local index of shared files for similar images using the *Image Indexer* module and information in the *ImageQuery* message. Once similar images are retrieved, the peer delivers an *ImageQueryHit* message (see Figure 4), back to the requester through *Packet Router* module.

(4) **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* module downloads thumbnails, generated in the preprocessing stage, or a full size image from the peer using HTTP protocol. On the other hand, *HTTP Agent* module in other peers serves as a web server to deliver the requested images.

3.3 Peer Clustering Based on Image Similarity

One of the problems of information retrieval in the P2P network is that since it does not have a centralized coordinator to maintain the feature vector index, the searching mechanism is required to perform a brute force search, which

Table I. Definition of Terms

$link_{random}(p)$	Random link—The original connection in P2P network which a peer p makes randomly to another peer in the network. It is chosen by the user.
$link_{attractive}(p)$	Attractive link—The connection which a peer p makes explicitly to another peer, with which they share similar images.
$Cat(p)$	A signature value representing the characteristic of a peer p.
$Sim(p, q)$	The distance measure between two peers p, q, which is a function of $Cat(p)$ and $Cat(q)$.
$Sim(p, \vec{Q})$	The distance measure between a peer p and an image query $\vec{Q}$.
$Peer(p, t)$	The set of peers that a peer p can reach within t hops.
$Collection(p)$	The set of images that a peer p shares.
$Match(Collection(p), q)$	The distance measure from each image in $Collection(p)$ to the query q.

is decentralized and inefficient. To solve this problem, we propose to cluster peers with similar image features together, making the network organized in a systematic way like the Yellow Pages in order to improve query efficiency. On top of the original P2P network, our peer clustering strategy makes use of an extra layer of connections, called *attractive links*, to group similar peers together based on two peers' image feature similarity within their neighborhood. With these added network topology constraints, we propose the Firework Query Model, which can perform searching efficiently by directing queries to their target cluster.

Let's consider a P2P network that consists of peers sharing images of different categories. Each peer shares a set of images and is responsible for extracting the content-based feature of its shared images. This collection of feature vectors is used to describe the characteristic of the shared images. We use the set of feature vectors as signature value of a peer to determine the similarity between two peers. We begin by defining several key terms. A summary of these terms is listed in Table I.

Definition 1. Let $Collection(p) = \{I_i^p\}_{i=1}^n$ be the set of n images a peer p shares. For each image in the collection, we perform low level feature extraction to map it to a multi-dimensional vector by function f, which extracts a real-valued d-dimensional vector as,

$$f : I \rightarrow R^d, \tag{1}$$

where f means a specific feature extraction function, for example, the color histogram, the co-occurrence matrix-based texture feature or the Fourier descriptor. After the extraction, each peer contains a set of feature vectors $\{\vec{R}_i^p\}_{i=1}^n$, where $\vec{R}_i^p = [R_{i1}^p, R_{i2}^p, \ldots, R_{id}^p]$, d is the number of the dimension, which will serve as its local index and used in computing the signature value and comparing the similarity to a query.

Definition 2. $Cat(p)$ is defined as $(\vec{\mu^p}, \vec{\delta^p})$, where $\vec{\mu^p}$ and $\vec{\delta^p}$ are the mean and variance of the image feature vectors collection $\{\vec{R}_i^p\}_{i=1}^n$ that peer

p shares. The j-th mean and variance is defined as,

$$\mu_j^p = \frac{1}{n}\sum_{i=1}^{n} R_{ij}^p, \quad j = 1 \cdots d, \tag{2}$$

$$\delta_j^p = \frac{1}{n}\sum_{i=1}^{n} \left(R_{ij}^p - \mu_j^p\right)^2, \quad j = 1 \cdots d. \tag{3}$$

Definition 3. $Sim(p,q)$ is defined as the distance measure between two peers' signature values, $Cat(p)$ and $Cat(q)$. Let them be $(\vec{\mu^p}, \vec{\delta^p})$ and $(\vec{\mu^q}, \vec{\delta^q})$ respectively. The following formula is used:

$$Sim(p,q) = \|\vec{\mu^p} - \vec{\mu^q}\|_2 \times \left(\sum_{i=1}^{d} \delta_i^p \times \delta_i^q\right)^{1/2}. \tag{4}$$

We assume that each peer often shares images related to a certain topic. For example, a peer may share a collection of tree pictures, then, the feature vectors of its shared images will form a sub-cluster in the high dimensional space, thus, $\vec{\mu}$ and $\vec{\delta}$ can describe the characteristic of this collection. Our target is to group these sub-clusters to form a cluster of peers that share similar images. Connecting peers with similar $\vec{\mu}$ and small $\vec{\delta}$ is analogous to achieving this. The more similar two peers p and q are, the smaller the value $Sim(p,q)$ is. The $Sim(p,q)$ measure is small when $\vec{\mu_p}$ and $\vec{\mu_q}$ are close and both $\vec{\delta_p}$ and $\vec{\delta_q}$ are small. When the means $\vec{\mu}$ are close, it means that the two subclusters are close in the high dimensional space. If both variances $\vec{\delta}$ measures are small, it means the image feature vectors in the two subclusters are closely clustered, that is, the shared images are highly related to a common topic.

Based on the above definition, we introduce a clustering algorithm to assign the attractive link in order to group similar peers as illustrated in Algorithm 1. There are three steps in our peer clustering strategy:

(1) **Signature Value Calculation**—In the beginning, every peer calculates its signature value, $Cat(p)$, based on the characteristic of images shared by that peer p (see Definition 2). Whenever the shared data collection, $Collection(p)$, of a peer changes, the signature value is updated accordingly.

(2) **Neighborhood Discovery**—After a new peer joins the 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 [Gnutella], to ask for the signature values of peers within its neighborhood, $(Peer(p,t))$. This task is not only done when a peer first joins the network; it repeats at a regular interval in order to maintain the latest information about other peers.

(3) **Similarity Calculation and Attractive Link Establishment**—After acquiring the signature values of other peers, one can find the peer with signature value closest to its signature value according to Definition 3, and make an attractive connection to link them up. This attractive connection

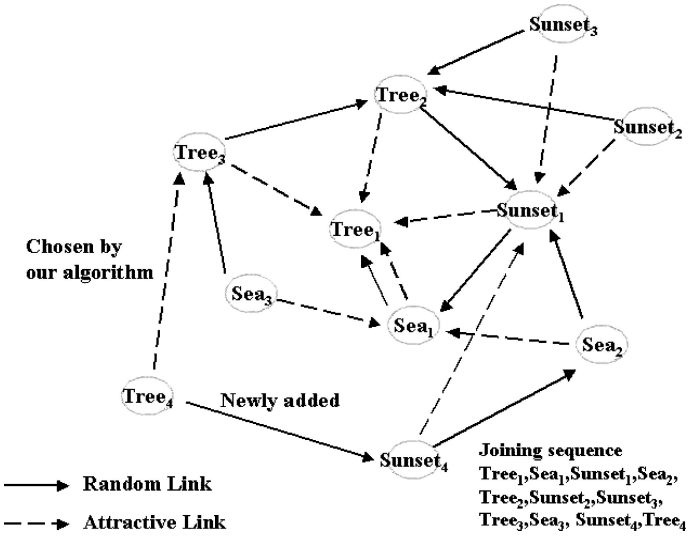

Fig. 6. Illustration of peer clustering.

Table II. Distance Measure of Each Peer in Figure 6

Sim	$Tree_1$	Sea_1	$Sunset_1$	Sea_2	$Tree_2$	$Sunset_2$	$Sunset_3$	$Tree_3$	Sea_3	$Sunset_4$	$Tree_4$
$Tree_1$	0.0	**7.0**	**2.9**	2.2	**0.4**	3.7	1.9	**0.5**	6.5	2.0	0.9
Sea_1		0.0	3.0	**0.6**	6.0	1.0	4.0	2.3	**0.3**	2.5	4.5
$Sunset_1$			0.0	4.0	6.6	**0.3**	**0.5**	4.5	7.3	**0.4**	5.7
Sea_2				0.0	0.9	2.2	3.5	9.8	0.5	1.7	2.6
$Tree_2$					0.0	7.8	7.8	0.7	7.1	7.7	0.8
$Sunset_2$						0.0	0.7	5.5	3.6	0.6	6.1
$Sunset_3$							0.0	6.3	2.8	1.0	5.1
$Tree_3$								0.0	8.2	4.3	***0.2***
Sea_3									0.0	9.0	3.3
$Sunset_4$										0.0	6.2
$Tree_4$											0.0

will be re-established to the second closest one whenever the current connection breaks.

As shown in Figure 6, there are three main classes of image being shared over the network, namely *Tree*, *Sunset*, and *Sea*. A peer named as $Tree_1$ means the majority of images it shares are related to tree. When a new peer $Tree_4$ joins the network, by connecting to a randomly selected peer $Sunset_4$, it sends out a signature query to learn the location and signature value of other peers. After collecting the replies from other peers, peer $Tree_4$ makes an attractive link to $Tree_3$ to perform peer clustering because $Sim(Tree_4, Tree_3)$ is the smallest, as shown in Table II. As peers continue to join the network using this algorithm, a clustered P2P network is formed. Peers of similar characteristic will gradually be connected by an attractive link to form a cluster. A selective query routing

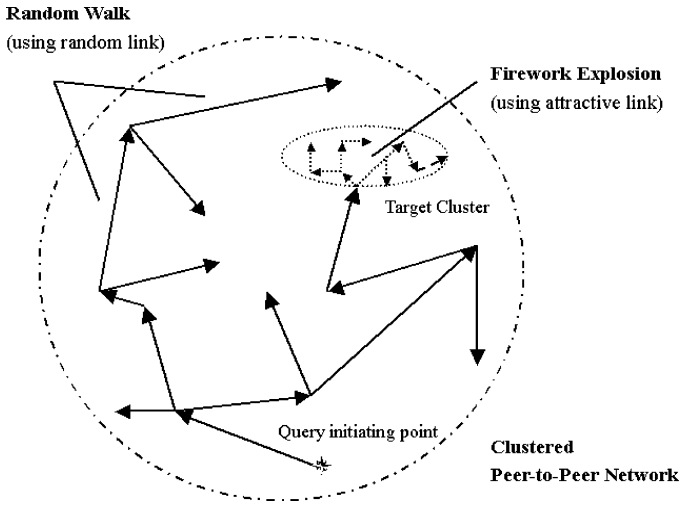

Fig. 7. Illustration of firework query.

scheme thus can be applied, which makes information retrieval much more systematic and efficient.

Algorithm 1 Algorithm for choosing attractive link
Attractive-Link-Selection(peer p, integer ttl)
for all q in $Peer(p, ttl)$ **do**
Compute $Sim(p, q)$
end for
assign $link_{attractive}(p)$ to q with $min_{argq}(Sim(p, q))$

3.4 Firework Query Model Over Clustered Network

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 broadcast by peers through the attractive connections inside the cluster much like an exploding firework as shown in Figure 7.

Here we introduce the algorithm to determine when and how a query message is propagated like a firework in Algorithm 2 using the example in Figure 6. Assume the peer $Tree_4$, whose shared images are mostly under the topic of *tree*, initiates a search to find similar images to its query image, which is an image of *sea*. First, the features of this query image are extracted and used to calculate the similarity between the query and its own signature value $Cat(p)$. Since the similarity measure between the query and its signature value is smaller than a preset threshold, θ, according to Algorithm 2, $Tree_4$ sends the query to $Sunset_4$ through the random link because the target of query is not likely to appear in the cluster connected by an attractive link. When $Sunset_4$ receives this query,

it needs to carry out two steps:

(1) **Shared File Look Up**—The peer looks up its shared image collection for those matching the query. Let $\vec{Q}$ be the query feature vector. $Sim(\vec{R_i^p}, \vec{Q})$ is the distance measure between the query and an image i shared by peer p, it is an L_2 norm defined as,

$$Sim\left(\vec{R_i^p}, \vec{Q}\right) = \left[\sum_j^d \left(R_{ij}^p - Q_j\right)^2\right]^{\frac{1}{2}}. \tag{5}$$

If any shared image matches within the matching criteria of the query in a peer, it will reply to the requester.

(2) **Route Selection**—The peer calculates the similarity between the query and its signature value, which is represented as,

$$Sim(p, \vec{Q}) = \prod_j^d \frac{1}{\sqrt{2\pi}\delta_j^p} e^{-\frac{(Q_j - \mu_j^p)^2}{2(\delta_j^p)^2}}. \tag{6}$$

For the same reason, since the similarity measure $Sim(p, \vec{Q})$ between $Sunset_4$ and query is smaller than θ, it then forwards the query to Sea_2, again through the random link. After the query traverses the network randomly, the query message reaches its target cluster and starts to expand the search much like a firework. During the explosion, Sea_2 also looks up its shared image collection for those matching the query. Obviously, the number of shared images in Sea_2 matching the query will be much larger than the number in $Sunset_3$. After it replies to the requester, it will forward the query to Sea_1 through the attractive link. Likewise, Sea_1 will carry out the same checking, forwarding the query to Sea_3, $Tree_1$ and so on.

In our model, we retain two existing mechanisms in the Gnutella network for preventing query messages from looping forever in the distributed network, namely the Gnutella replicated message checking rule and the Time-To-Live (TTL) of messages. When a new query appears to a peer, it is checked against a local cache for duplication. If it is found that the same message has passed through before, the message will not be propagated. The second mechanism is to use the Time-To-Live value to indicate how long a message can survive. Similar to IP packets, every Gnutella message is associated with a TTL. Each time the message passes through a peer, the TTL value is decreased by one. Once the TTL reaches zero, the message will be dropped and no longer forwarded. There is a modification, however, 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 the TTL value is 1. For attractive connections, the probability of decreasing the 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.

Algorithm 2 Algorithm for the Firework Query Model

```
Firework-query-routing (peer p, query Q)
if Sim(p, Q) > θ (threshold) then
  reply the query Q
  if rand() > CTS then
    TTL_new(Q) = TTL_old(Q) − 1
  end if
  forward Q to all link_attractive(p)
else
  TTL_new(Q) = TTL_old(Q) − 1
  if TTL_new(Q) > 0 then
    forward Q to all link_random(p)
  end if
end if
```

4. EXPERIMENTS AND RESULTS

In this section, we discuss the design of experiments and evaluate the performance of our proposed Firework Query Model. First, we present the Peer-to-Peer network model used in our simulation. We then introduce the performance metrics used to evaluate different search mechanisms. Simulations were conducted to study the performance of different mechanisms using different parameters. Last, we show how our strategy performs and behaves with various network sizes.

4.1 Simulation Model of Peer-to-Peer Network

The goal of our experiment is to model a typical Peer-to-Peer network where each node contains a set of images. We generated different sizes of Peer-to-Peer networks to evaluate the performance of different search mechanisms. The number of peers in each network varies from 2,000 to 20,000. The diameters[3] of the network vary from 9 to 11, and the average distances between two peers vary from 5.36 to 6.58.

To evaluate the search mechanism accurately, the simulations were performed over different network topologies against the TTL and the number of peers. For each set of TTL and number of peer parameters, we ran five trials. Each trial would have a different network topology with ten iterations (or queries). For each iteration, we initiated a query starting from a randomly selected peer. Hence, for each set of parameters we would collect statistical information for 50 iterations (or queries). For each set of parameters, we would collect the average recall, the average number of visited peers, and the average of number of query messages. In our experiment, we use two sets of data, synthetic data and real data:

(1) **Synthetic data**—We generated 100 sets with random mean and variance. For each set, 100 points are generated according to a Gaussian distribution using the parameters, which is to model feature vectors of images belonging to the same class.

[3]The longest shortest path between any two peers in the network.

(2) **Real data**—We used 10,000 images (from 100 categories) in the Corel-Draw's Image Collection CD and use the Color Moment feature as test data.

We randomly assigned different classes of images to each peer; therefore, the data location of every build of network was different. Competitive Learning clustering algorithm [Rumelhart and Zipser 1985] was used to cluster data inside each peer. The simulation was done on Sun Blade 1000 (2GB RAM) running Solaris v.8 using C. For a simulation of 20,000 peers and TTL 7, the running time was approximately 3 hours for 10 iterations. For the DISCOVIR system, we built our client program based on Gnutella v0.4 protocol. For image related operations, we used Java image manipulating routines to assist in extracting visual features. During our field test, we used different port numbers to simulate different peers.

4.2 Performance Metrics

The metrics we use to evaluate the performance are:

(1) **Recall**—The success rate of the desired result retrieved. It is the fraction of the relevant documents that have been retrieved,

$$Recall = R_a/R, \tag{7}$$

where R_a is the number of retrieved relevant documents, and R is the total number of relevant documents in the Peer-to-Peer network. If Recall is high, it means more relevant documents can be retrieved, so the performance is better.

(2) **Query scope**—The fraction of peers being visited by each query,

$$Visited = V_{peer}/T_{peer}, \tag{8}$$

where V_{peer} is the number of peers that received and handled the query and T_{peer} is the total number of peers in the Peer-to-Peer network. For each query, if the fraction of involved peers in the network is lower, the system will be more scalable.

(3) **Query efficiency**—The ratio between the Recall and Query Scope,

$$Efficiency = Recall/Visited. \tag{9}$$

In general, the performance of the P2P network is more desirable if we can retrieve more relevant documents (high recall) but visit fewer peers (small query scope). Observing either Recall or Query Scope only, is not enough to determine the quality of the algorithm. Therefore, we defined the query efficiency as the ratio between Recall and Query Scope. If the algorithm can retrieve more relevant documents while visiting fewer peers, the query efficiency will be a large value. If the data locations are evenly distributed, the Query Efficiency will be equal to 1 under the BFS algorithm—if we visited 50% of peers in the network, it is expected that we can retrieve 50% of the relevant documents in the network.

Table III. Experiment Parameters

Number of Peer	2,000–20,000
Diameter of the P2P network	9–11 hops
Average distance between 2 peers	5.4–6.6 hops
Number of documents assigned to each peer	100 images (1 class)
Dimension of extracted feature vector to represent the image	9
TTL value of the query packet	(FQM-5) (BFS-7)

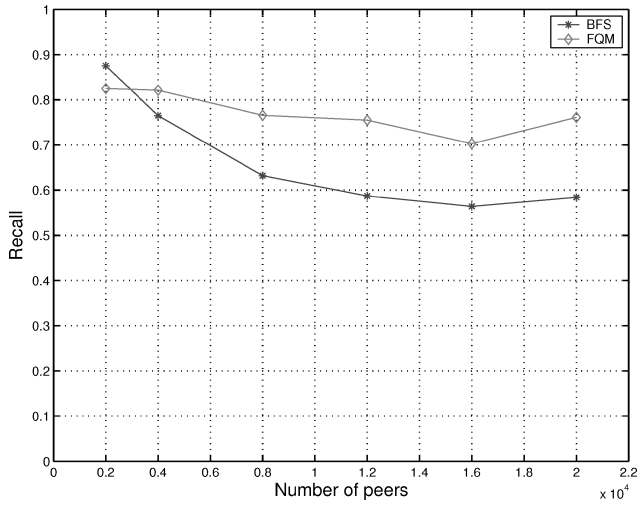

Fig. 8. Recall versus number of peers.

4.3 Experiment Results

In this section, we experimentally compare our proposed Firework Query Model against the Brute Force Search algorithm. We explore how the performances are affected by different numbers of peers and different Time-To-Live (TTL) values of query packets.

Performance with different Number of Peers in the P2P Network. This experiment tests the scalability of search mechanisms. The number of peers in each network varies from 2,000 to 20,000. The experiment parameters are listed in Table III.

4.3.1 *Recall.* Figure 8 depicts recall versus number of peers in two search mechanisms for the real data set. When the size of the network increases, the recall of Firework Query Model continues to remain at a higher range, while the recall for BFS drops when the size of the network grows. We conclude that our algorithm is insensitive to the change of network size. Our mechanism is more scalable. Even when the size of the network grows, FQM still can reach the portion of the network containing the query target.

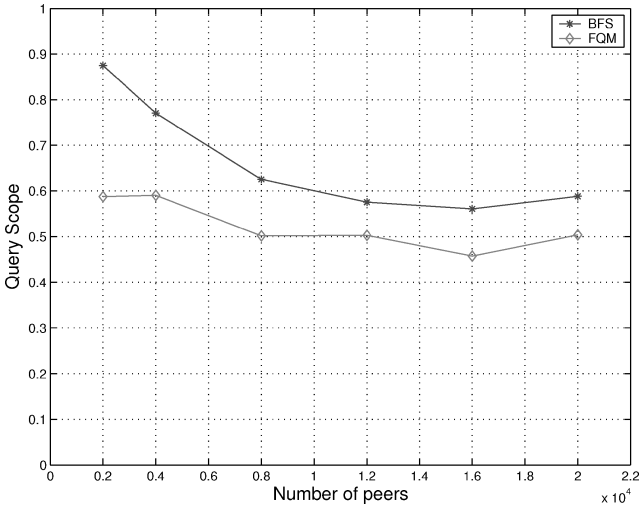

Fig. 9. Query scope versus number of peers.

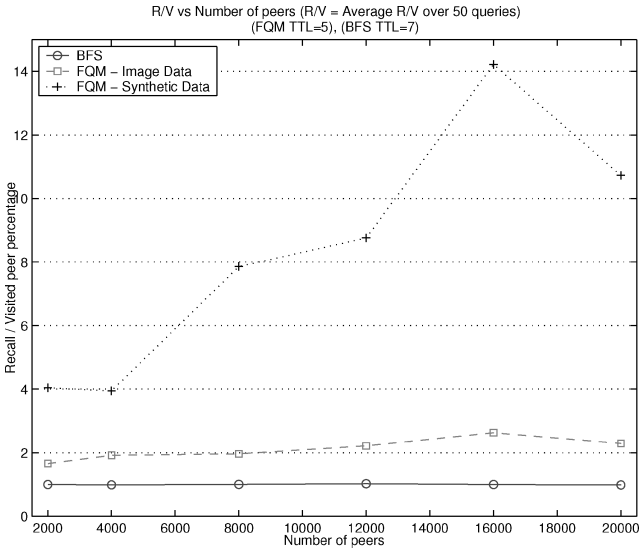

Fig. 10. Query efficiency versus number of peers.

4.3.2 *Query Scope.* As seen in Figure 9, we achieve load reduction by using FQM. Fewer peers are exposed to each query.

4.3.3 *Query Efficiency.* As seen in Figure 10, FQM outperforms BFS in both synthetic data and real data set. The efficiency is improved by 60% in real data. The efficiency is improved by 13 times in synthetic data. Since the variance of synthetic data we generated is much smaller than the variance of real data, the synthetic data can be clustered more appropriately, resulting in a better performance.

Table IV. Experiment Parameters

Number of Peers	10,000
Diameter of the P2P network	10 hops
Average distance between 2 peers	6.2 hops
Number of documents assigned to each peer	100 images (1 class)
Dimension of extracted feature vector to represent the image	9
TTL value of the query packet	4–9

The curve of both FQM also follows a small bell shape. Query efficiency increases at first due to two reasons:

(1) The network can be clustered more appropriately when the network size increases. When the number of peers increases, new peers can have more choices and make their attractive link to a more similar peer.
(2) The percentage of peers visited is inversely proportional to the network size when the TTL is fixed. FQM advances the recall percentage when the query message reaches the target cluster.

When the network size increases further, a query might not reach its target cluster for low TTL value (The TTL is fixed to 5 in this experiment and the diameter of network increases from 9 to 11), so query efficiency starts to drop. Therefore, choosing a good TTL value is important in our algorithm and this will be discussed in the next section.

Performance with different TTL value of query packet in P2P Network. In this section, we explore how the performances are affected by different Time-To-Live (TTL) values of query packet. The number of peers in each network is fixed to 10,000. The TTL value varies from 4 to 9. The experiment parameters are listed in Table IV.

4.3.4 *Recall.* Figure 11 shows recall versus TTL of query message. When the value of TTL increases, both the recall of Firework Query Model and the BFS increase, while our proposed strategy reaches the maximum value at a much faster rate. When the TTL is larger than 8, the recall graph tails down in the Firework Query Model because the recall is nearly saturated and cannot be further improved.

4.3.5 *Query Scope.* Figure 12 shows the number of visited peers in both strategies. We vary the TTL of the query message to observe the changes in the query scope when a peer initiates a search. The Firework Query Model shows a promising sublinear increase in the query scope subject to increasing TTL of the query message, while the BFS increases at a much faster rate. The query scope of the Firework Query Model is larger than the BFS when the TTL value is small because a Chance-To-Survive (CTS) value is introduced in the Firework Query Model. This strategy lets the query message have a higher chance to survive when forwarding through attractive connections; therefore the query scope is larger. Specifically, we choose $CTS = 1$ in all the simulations.

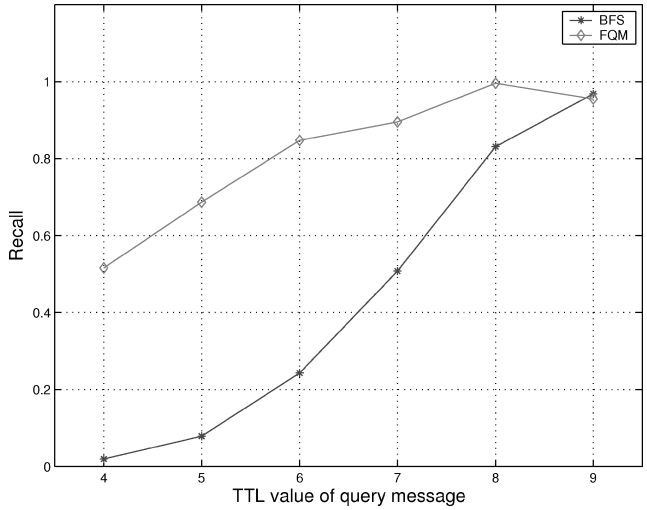

Fig. 11. Recall versus TTL value of query packet.

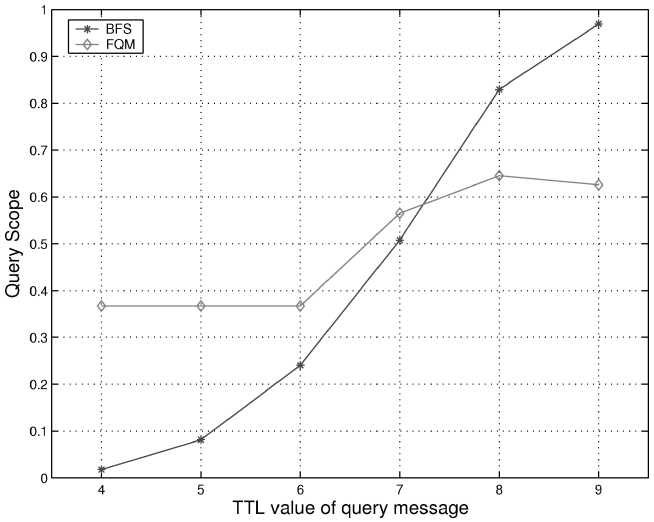

Fig. 12. Query scope versus TTL value of query packet.

4.3.6 *Query Efficiency.* As seen in Figure 13, FQM outperforms BFS for different TTL values of query packet. In synthetic data, we find that the optimal TTL value is 8 in a network size of 10,000 peers in the Firework Query Model. The Query Efficiency is low at the beginning because the TTL value is not enough for the query packet to reach its target cluster. When the TTL value increases, the query has a larger chance to reach its target cluster; therefore, the Query Efficiency increases. When the TTL value is 8, the Query Efficiency is optimal because the query packet can just reach its target cluster. However, further increasing the TTL value will only generate unnecessary traffic; therefore, the Query Efficiency starts to level off beyond TTL 8.

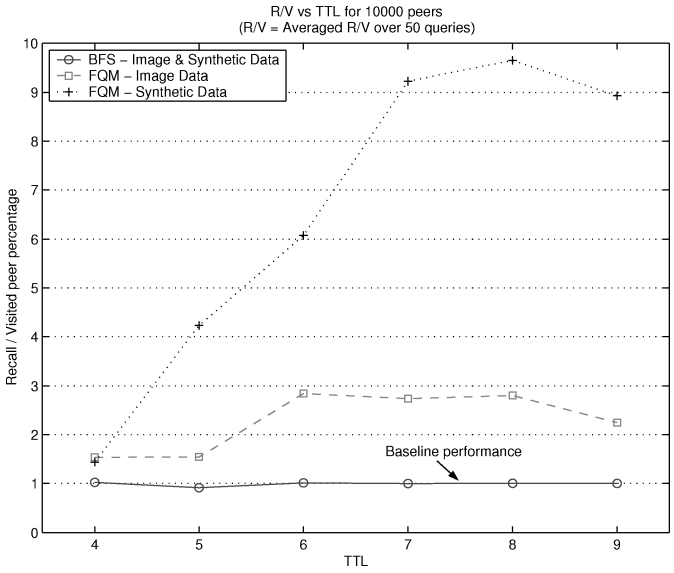

Fig. 13. Query efficiency against TTL value of query packet.

4.4 Discussion

There are a few more interesting points worthy of mentioning here. They are the local indexing method used, the effects of data distribution on performance, and the impact of the number of attractive connections and TTL on performance.

On the issue of local indexing, there exist many high-dimensional spatial indexing methods used in content-based image retrieval systems, for example, R-tree and VP-tree, that can make the local search more efficient. However, the further elaboration of indexing methods is outside of the scope of this article. In addition, most peers only share a limited number of images in the P2P network; a linear search shows acceptable response time in retrieving similar images using the DISCOVIR client program. More sophisticated methods can be added to the system later for improved performance.

If some peers contain many images with different types of semantic meanings, this might be a concern for the query broadcasting problem since such peers will probably maintain many attractive connections. To solve this problem, we restrict the maximum number of attractive connections made for a node in order to avoid the broadcasting condition. Specifically, we choose 3 as the maximum number used in the simulation experiments and implementation of DISCOVIR. Moreover, as the network grows, many clusters exist and there might be difficulties in reaching the cluster in a short number of hops. However, according to recent research, the Gnutella network follows a power-law distribution, which has the special property that the diameter of the network is small [Lv et al. 2002]. A query message is guaranteed to reach a large portion of the network with a TTL of 7. With reference to the query efficiency versus TTL experiment as shown in Figure 13, a TTL of 4 and 5 shows relatively low performance improvement, which illustrates the problem of not being able to

Table V. Data Distribution Characteristics of 4 Datasets

		Synthetic data	Corel image data	Real-world data	Synthetic data 2[a]
ID[b]	max	1.5297	1.4467	0.5718	0.8697
	min	0.3074	0.0272	0.0351	0.0318
	avg	0.9385	0.3298	0.2097	0.2138
VAR[c]	max	0.0388	0.0153	0.0504	0.0396
	min	0.0161	0.0006	0.0174	0.0141
	avg	0.0268	0.0113	0.0298	0.0263

[a]The synthetic data is generated to simulate real-world data distribution.
[b]ID—inter-cluster distance between each class.
[c]VAR—the mean of variance of each class.

reach the target cluster in a small number of hops. On the other hand, it shows that with a TTL of 6 and 7, the system is able to attain good performance in a network size of 10,000 peers.

Apart from those aforementioned, the point of utmost concern is whether FQM is applicable in real world P2P networks, and how its performance is affected with different data distributions. In addition to the Corel image dataset and synthetic dataset used in Section 4, we have added one more real world dataset to further evaluate FQM. We downloaded images from 18 different peers in the Gnutella network in July 2003. For each set of images shared by peers, we assume that they fall in the same category and form one cluster in the feature vector space. Using the statistics of inter-cluster distance, means, and variance, collected from real-world data, we generated another synthetic dataset to model the image distribution in a real world P2P network. The characteristics of the four datasets used are listed in Table V and Figure 14. Indeed, even after modeling the real world data distribution, the FQM still shows an improvement.

Figure 15 shows the query efficiency of FQM under three different data distributions, with synthetic dataset being the most improved, Corel image dataset being the second, and simulated real world dataset being the least improved. Nonetheless, FQM still obtains a gain of 20–30% in query efficiency when compared to BFS. Based on this observation, we conclude that the query efficiency of FQM is sensitive to data distribution. The best performance can be obtained if each peer shares one or a small number of image categories, and images in each category are closely clustered in the feature vector space. On the other hand, if peers share many different types of images, it is unlikely to form a community or cluster in the network, thus the performance would be the same as not having FQM at all. In conclusion, the query efficiency can be related to the data distribution by the following relationship:

$$\textbf{Query Efficiency} \propto ID \times \frac{1}{VAR}, \tag{10}$$

where ID is the inter-cluster distance and VAR is the mean of variance of each class.

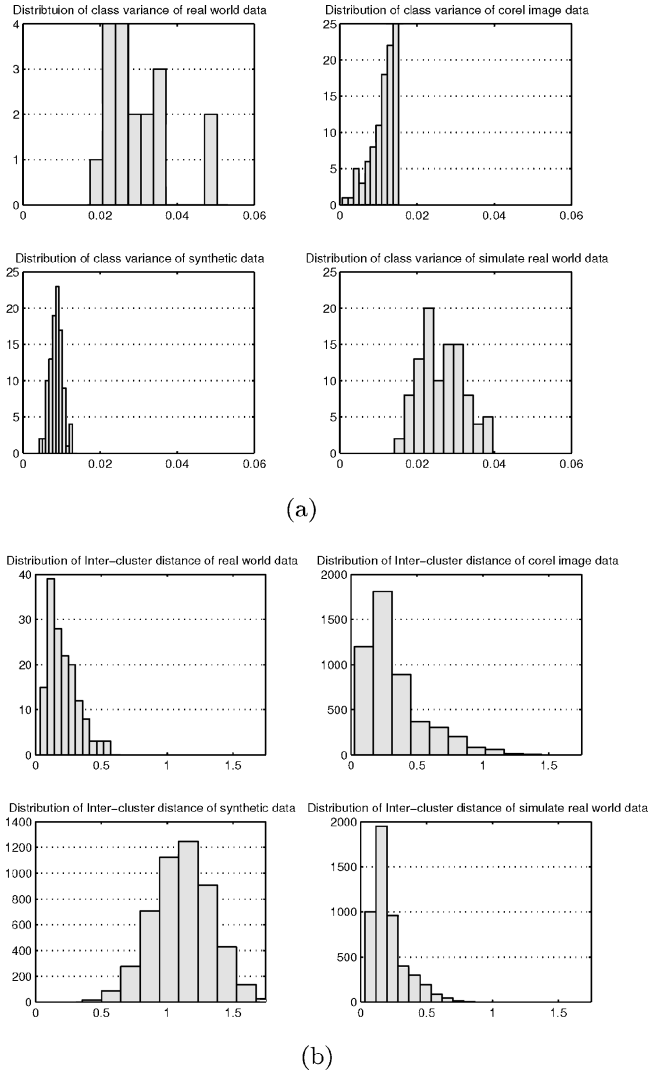

Fig. 14. (a) VAR distribution, (b) ID distribution of the four datasets.

5. CONCLUSION

In this paper, we demonstrate how to implement a CBIR system over the current Gnutella network. When users need to search for an image, all peers inside the network will look up their own collection of images and respond to the requesters. Such architecture fully utilizes the storage and computation capability of computers in the Internet. However, the lack of a centralized index requires a query to be broadcast throughout the network in order to achieve a satisfactory result. To solve this query broadcasting problem, we propose a peer clustering and intelligent query routing strategy to search images efficiently over the P2P network. We verify our proposed strategy by simulations

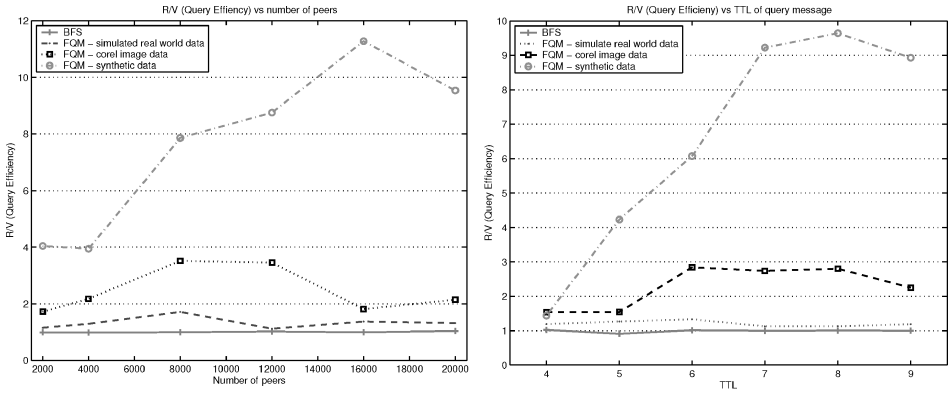

Fig. 15. Query efficiency versus (a) number of peers, (b) TTL of query message.

with different parameters to investigate the performance changes subject to different network size and TTL of query messages. We show that our Firework Query Model outperforms the BFS method in both network traffic cost and query efficiency measure.

ACKNOWLEDGMENTS

The authors would like to thank the reviewers for their valuable comments and suggestions.

REFERENCES

CHANG, W., SHEIKHOLESLAMI, G., WANG, J., AND ZHANG, A. 1998. Data Resource Selection in Distributed Visual Information Systems. *IEEE Trans. Knowl. Data Eng. 10*, 6 (November/December), 926–946.

COOPER, B. F. AND GRACIA-MOLINA, H. 2002. Peer-to-peer data trading to preserve information. *ACM Trans. Info. Syst. 20*, 2 (April), 133–170.

CRESPO, A. AND GRACIA-MOLINA, H. 2002. Routing Indices For Peer-to-Peer Systems. In *Proceedings of the Internation Conference on Distributed Computing Systems (ICDCS)*.

eDonkey. The eDonkey2000 homepage. http://www.edonkey2000.com.

FALOUTSOS, C., BARBER, R., FLICKNER, M., HAFNER, J. W. N., PETKOVIC, D., AND EQUITZ, W. 1994. Efficient and Effective Querying by Image Content. *Journal of Intelligent Information Systems: Integrating Artificial Intelligence and Database Technologies 3*, 3-4, 231–262.

Freenet. The Freenet homepage. http://freenet.sourceforge.net.

Gnutella. The Gnutella homepage. http://www.gnutella.com.

GUPTA, A. AND JAIN, R. 1997. Visual Information Retrieval. In *Comm. ACM*. Vol. 40. 70–79.

HARREN, M., HELLERSTEIN, J. M., HUEBSCH, R., LOO, B. T., SHENKER, S., AND STOICA, I. 2002. Complex Queries in DHT-based Peer-to-Peer networks. In *Proceedings of the International Peer-to-Peer Workshop*.

KING, I. AND JIN, Z. 2001. 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)*, L. Zhang and F. Gu, Eds. Vol. 2. Fudan University, Fudan University Press, Shanghai, China, 699–704.

KUBIATOWICZ, J. AND ANDERSON, D. P. 2002. The Worldwide Computer. *Scientific American* (March).

LAU, T. K. AND KING, I. 1998. 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, vol. 1351. Berlin, Germany: Springer Verlag, I 410–417.

LimeWire. The Limewire homepage. http://www.limewire.org.

LV, Q., CAO, P., COHEN, E., LI, K., AND SHENKER, S. 2002. Search and Replication in Unstructured Peer-to-Peer Network. In *Proceedings of the 16th ACM International Conference on Supercomputing (ICS'02)*. New York.

MA, W. Y. AND MANJUNATH, B. 1997. Natra: A toolbox for navigating large image databases. In *Proceedings of the IEEE International Conference on Image Processing*. 568–571.

MEHROTRA, S., RUI, Y., ORTEGA, M., AND HUANG, T. 1997. Supporting Content-based Queries over Images in MARS. In *Proceedings of the IEEE International Conference on Multimedia Computing and Systems*. 632–633.

MIPLAB. The homepage of Multimedia Information Processing Lab, cse, cuhk. http://www.cse.cuhk.edu.hk/~miplab.

MUKHERJEA, S., HIRATA, K., AND HARA, Y. 1999. AMORE: a World Wide Web Image Retrieval Engine. In *World Wide Web*. Vol. 2. 115–132.

Napster. The Napster homepage. http://www.napster.com.

NG, C. H. AND SIA, K. C. 2002. Peer Cluster and Firework Query Model. In *Poster Proceedings of The Eleventh International World Wide Web Conference, Poster ID 195*. Hawaii.

NG, C. H., SIA, K. C., AND CHAN, C. H. 2003. Advanced Peer Clustering and Firework Query Model. In *Poster Proceedings of The Twelfth International World Wide Web Conference, Poster ID S130*. Hungary.

PENTLAND, A., PICARD, R. W., AND SCLAROFF, S. 1994. Photobook: Tools for Content-based Manipulation of Image Databases. In *Proceedings of SPIE*. Vol. 2185. 34–47.

RATNASAMY, S., FRANCIS, P., HANDLEY, M., AND KARP R. SHENKER, S. 2001. A Scalable Content-Addressable Network. In *Proceedings of ACM SIGCOMM*. 161–172.

RATNASAMY, S., SHENKER, S., AND STOICA, I. 2002. Routing Algorithms for DHTs: Some Open Questions. In *Proceedings of the First International Peer-to-Peer Workshop*. 45–52.

ROWSTRON, A. AND DRUSCHEL, P. 2001. Pastry: Scalable, Decentralized Object Location and Routing for Large-Scale Peer-to-Peer Systems. In *Proceedings of the 18th IFIP/ACM International Conference on Distributed Systems Platforms*. 329–350.

RUI, Y., HUANG, T. S., AND CHANG, S.-F. 1999. Image Retrieval: Current Techniques, Promising Directions and Open Issues. *Journal of Visual Communication and Image Representation 10*, 39–62.

RUMELHART, D. AND ZIPSER, D. 1985. Feature Discovery by Competitive Learning. *Cognitive Science 9*, 75–112.

SETI. The search for extraterrestrial intelligence homepage. http://www.setiathome.ssl.berkeley.edu/.

SIA, K. C., NG, C. H., CHAN, C. H., CHAN, S. K., AND HO, L. Y. 2003. Bridging the P2P and WWW Divide with DISCOVIR—DIStributed COntent-based Visual Information Retrieval. In *Poster Proceedings of The Twelfth International World Wide Web Conference, Poster ID S172*. Hungary.

SMEULDERS, A. W., WORRING, M., SANTINI, S., GUPTA, A., AND JAI, R. 2000. Content-Based Image Retrieval at the End of the Early Years. *IEEE Trans. Pattern Anal. Mach. Intel. 22*, 12 (December), 1349–1380.

SMITH, J. R. AND CHANG, S. F. 1997. An Image and Video Search Engine for the World-Wide Web. In *Proceedings of SPIE*. Vol. 3022. 84–95.

SRIPANIDKULCHAI, K., MAGGS, B., AND ZHANG, H. 2003. Efficient Content Location Using Interest-Based Locality in Peer-to-Peer Systems. In *Proceedings of IEEE INFOCOM 2003*.

STOICA, I., MORRIS, R., KARGER, D., KAASHOEK, M. F., AND BALAKRISHNAN, H. 2001. Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications. In *Proceedings of ACM SIGCOMM*. 149–160.

TANG, C., XU, Z., AND MAHALINGAM, M. 2002. PeerSearch: Efficient Information Retrieval in Peer-to-Peer Networks. Tech. rep., HP Labs. July.

WANG, J. Z., G., W., FIRSCHEIN, O., AND SHA, X. 1998. Content-based Image Indexing and Searching using Daubechies' wavelets. *Int. J. Dig. Libraries 1*, 4, 311–328.

WANG, J. Z., LI, G., AND WIEDERHOLD, G. 2001. SIMPLIcity: Semantics-sensitive Integrated Matching for Picture LIbraries. In *IEEE Trans. Pattern Anal. Mach. Intel*. Vol. 23. 947–963.

ZHAO, B., KUBIATOWICZ, J., AND JOSEPH, A. 2001. Tapestry: An Infrastructure for Fault-tolerant Wide-area Location and Routing. Tech. rep., Computer Science Division, U.C. Berkeley. April.

Received July 2002; revised October 2003; accepted December 2003