Efficient Content-Based Image Retrieval In a Distributed Environment

Sia Ka Cheung
Department of Computer Science and Engineering
The Chinese University of Hong Kong
Shatin, N.T., Hong Kong SAR

kcsia@cse.cuhk.edu.hk

Abstract

With the recent advances of distributed computing, those limitations of information retrieval from a centralized image collection can be removed by allowing distributed image data sources to interact with each other for both data storage sharing and query computation sharing. In this paper, we present our initial study of the design of a distributed image database system using the current Peer-to-Peer (P2P) Network. We propose a Firework Query Model for distributed information retrieval which aims to reduce the network traffic. We carry out experiments to show the distributed image retrieval system and the Firework information retrieval algorithm. The results show that the algorithm reduces the network traffic while increases the recall performance.

Keywords

Peer-to-Peer (P2P) Network, Information Retrieval, Peer Clustering, Intelligent Query Routing, Content-Based Image Retrieval (CBIR), Distributed System

List of Figures

List of Tables

1 Introduction

In recent years, the distributed computing models have accomplished tasks that are difficult for the traditional centralized computing models to achieve. For example, by distributing data storage over networked computers, one can have a virtual data storage that is possibly many magnitudes larger than what can be stored in a local computer. In addition, such distributed file system with data redundancy would provide zero down time and a powerful fault tolerance mechanism [16, 2]. One may also envision data security by distributing pieces of an encrypted file over many computers. Doing so, one imposes a difficult barrier for intruder to overcome because he needs to break into several computers before getting the file [22]. With a suitable data segmentation technique, we are able to deliver high-bandwidth data, e.g., streaming video, using a collection of computers with slower connection speed [9]. Likewise, one may also distribute the computation among different computers to achieve a high throughput such as the SETI (Search for Extraterrestrial Intelligence) [23]. 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 at least two ways. First, with the increasing users joining the P2P network, the image collection will become more huge and diverse due to individual contribution. Second, with the task of image processing and retrieval distributed over all the peers, it overcomes the scalability problem of image retrieval using the traditional centralized retrieval approach. These ideas sound interest and great, however, many difficulties remain unsolved. For example, images are distributed across different computers, there is no centralized indexing method to locate them efficiently in contrast to the original approach. On the implementation side, we need to standardize a set of image features that all peers compromise to use in describing their shared images. Last but not least, the issue of charge for using and down-loading resource need to be addressed before a system can be applied in real life, or at least be applied commercially.

In this paper, we present the design of building a CBIR system on the P2P network for users to share and retrieve images. In particular, we overcome the scalability problem using the proposed Firework Query Model.

In the following, we first review current issues in CBIR and P2P in Section 2. Then, we proceed to present the architecture of a P2P information retrieval and the detail algorithm of the Firework Query Model of our proposed system in Section 3. We then report and analyze our experimental results in Section 4. We give our final remarks and conclusion in Section 5.

2 Background and Related Works

2.1 Current Trends in Content Based Image Retrieval

In the past few years, due to emergence of large scale digital image collections, the difficulties faced by the manual annotation in classifying images become more and more acute, CBIR was proposed. Instead of annotating images by text-based keywords entered manually, images would be indexed by their own visual contents, such as the color, texture and shape. Several researchers [20, 25] had done comprehensive surveys of current techniques in CBIR and addressed on some issues as the future direction.

Since early 1990's, many CBIR systems have been proposed and developed, some of them are QBIC [5], WebSEEK [26], SIMPLIcity [28], MARS [12], NeTra [11], Blobworld [1], Photobook [17], and some other systems. 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 querying are all done in a centralized fashion which can be computationally intensive and it is difficult to scale up. As indicated by several researchers [20, 25], one of the promising future trends in CBIR includes the distributed computing on data collection, data processing, and information retrieval. By reforming the centralized system model, we not only can increase the size of image collections, but we also overcome the scalability bottleneck problem by distributing the processing of the image information retrieval.

Fig. 1 shows a typical web-based content-based image retrieval system, The image at the top is the example query images retrieved by user, while images below are images extracted from the database based on their content similarity, in this example, co-occurance based texture feature is used.

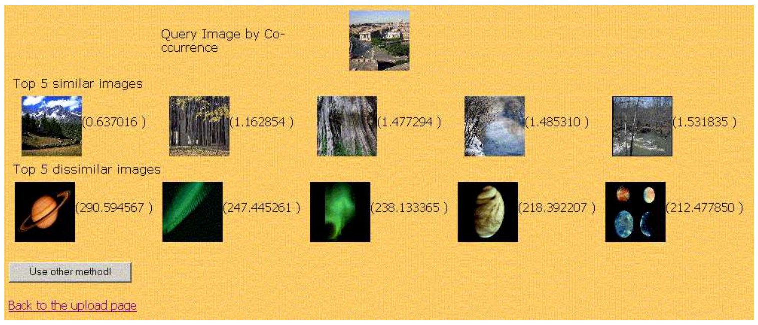

Figure 1: Screen-shot of a web-based CBIR system.

2.2 Content Based Image Retrieval Model Selection

Similar to traditional text retrieval, current implementations of CBIR mainly fall into two categories, vector model and probabilistic model. As the performance of one shot image retrieval approach is not so accurate because extracted image feature cannot fully describe the semantic meaning of an images. Based on these two models, researchers formulated different relevance feedback strategies to help improving the accuracy of search result. In the following, we will describe a relevance feedback strategy in each of the model and explain the choice of our model in implementing distributed CBIR.

2.2.1 Rui's Weight Updating Method

Base on the vector model, Rui et. al. [21] formulated a weight updating method to capture user's preference on different features, such as color or texture. In [21], objects in image database are modeled as

$$O = O(D, F, R), \tag{1}$$

where D is the raw image data, $F = \{f_i\}$ is a set of low level visual features, such as color, texture, and shape, and $R = \{r_{ij}\}$ is the set of representations for f_i, which is defined as

$$r_{ij} = [r_{ij1}, ..., r_{ijk}, ..., r_{ijK}]. \tag{2}$$

Moreover, the feature vector is organized in a hierarchical manner. The overall similarity of two images O^a and O^b is defined as

$$S(O^a, O^b) = \sum_i W_i S(f_i^a, f_i^b), \tag{3}$$

$$S(f_i^a, f_i^b) = \sum_j W_{ij} S(r_{ij}^a, r_{ij}^b), \tag{4}$$

$$S(r_{ij}^a, r_{ij}^b) \quad = \quad m(r_{ij}^a, r_{ij}^b, W_{ijk}), \tag{5}$$

where m is the distance measure function, while W_i, W_{ij} and W_{ijk} are the weights associated with each features, its representation and each dimension respectively. In each feedback, they will follow two procedures, namely inter-weight updating and intra-weight updating to update the weight in order to capture user's interest in different features.

2.2.2 Cox's Bayesian Formulation Method

Cox et. al. [4] formulated a Bayesian Learning approach to learn which image is more likely to be user's target based on the feedback. Each image is associated with a probability of being the user's target. The retrieval process consists of two steps. In each pass, the system selects a set of images and presents to user. Through the feedback, the system updates the likelihood measure to the query of each image accordingly. The probability is updated using the Bayes' rule as follows,

$$P(T = T_i|H_t) = \frac{P(A_t|T = T_i, D_t, S_{t-1})P(T = T_i|H_{t-1})}{\sum_{j=1}^{n} P(A_t|T_j, D_t, S_{t-1})P(T_j|H_{t-1})}. \tag{6}$$

The meaning of Eq. (6) is that the probability of T_i being the target image at iteration t is equal to product of the probability of T_i being the target at iteration $t-1$ and the probability of user give such feedback at iteration t provided that T_i is the target, over the summation of probability of other images. Moreover, as each image is associated with a probability of being the target, they also proposed a maximum entropy display strategy to select image presenting to user. As a result, the system is expected to get most information gain from user's feedback. Besides, [8, 24] also details a procedure to apply this strategy.

Since the probabilistic model requires a global update of the probability of each image after a feedback iteration, it is not possible to apply in a distributed environment. We choose to use the vector model instead as the weight for each feature can be stored locally during the retrieval process.

2.3 Peer-to-Peer Network

Peep-to-Peer (P2P) Network is a recently evolved paradigm for distributed computing. With the emerging P2P networks or their variants such as Gnutella [6] and Napster [13], they

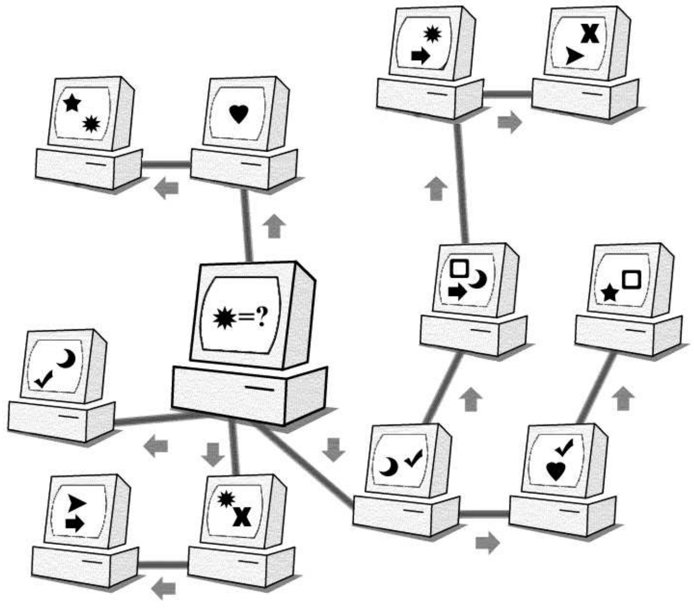

Figure 2: Illustration of information retrieval in a P2P network.

offer the advantages of distributed resource, increased reliability and comprehensiveness of information [23, 3, 10].

Figure 2 shows an example of a typical P2P network. In the example, different files are shared by different peers. 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 connecting peers and this process continues. Each peer will process the query request to and look up their own shared collection. 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 running 24 hours on the Internet. Each peer acts as a server and as a client at the same time. In these networks, a peer can become a member of the network by establishing a connection to one or more peers in the current network. This model is wasteful because peers are forced to handle every query message passing through it and mostly are irrelevant query messages. This type of search is called a Brute-Force Search (BFS) [18], as a query is being broadcasted to all connected computers even they do not contain the relevant information.

Although the BFS technique in a P2P network is able to retrieve information from the connected peers, this technique is inefficient [18] because a query needs to be broadcasted to all neighboring peers before the information is retrieved. Current research of searching techniques in P2P network, such as Chord [27], and Pastry [19] focus on reducing the amount of network traffic generated. Our proposed CBIR over P2P architecture makes use of deliberately formed connection between peers to form a cluster and routing of queries selectively without restricting to have a specific network topology, thus adaptable to current gnutella network. We also incorporate image features in consideration when building the network and routing queries.

3 Content-Based Image Retrieval Over Peer-to-Peer Network

In Section 3.1 we detail the procedure to perform CBIR in the current Gnutella network. Since every query is broadcasted to every peer in the network, each peer has to waste resources in handling irrelevant queries being given. This query message flooding within the network increases the traffic. In Section 3.2, we tackle this problem by clustering similar image data to build a special network topology that helps to search more efficiently. Based on this network topology, we propose the Firework Query Model which seeks to reduce network traffic and enhance query performance in Section 3.3. In Section 3.4, we propose a mechanism to restricting the number of results responding to a query by filtering out unlikely data to further reduce network traffic.

3.1 Proposed Procedure to Perform Distributed Content-Based Image Retrieval

We outline the overall process of sharing and retrieving image in our system. First we perform feature extraction, e.g., color, texture, shape, etc. on shared images in each peer. Each peer maintains his own localized index of feature vectors of his image collection. When a peer, the requester, initiates a query by giving an example image, it performs feature extraction on the example image and sends the feature vector to all its connecting peers. Consequently, other peers compare this query to their feature vector index based on a distance measure to calculate similar images and return result back to the requester. Likewise, the peers will propagate this query to their connecting peers and the search is forwarded to more and more

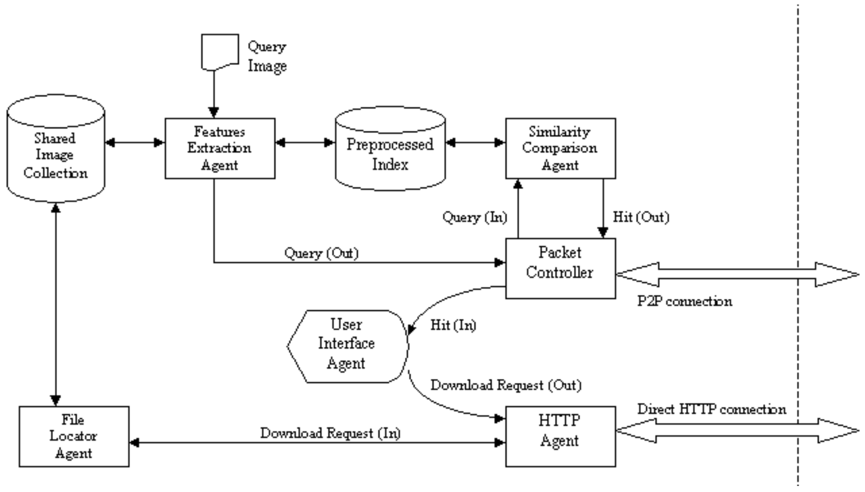

Figure 3: Architecture of CBIRP2P system.

peers throughout the whole network.

3.1.1 Gnutella Message Modification

Our system, CBIRP2P extends the current Gnutella network (v0.4 protocol) [6] by adding two more types of messages ImageQuery and ImageQueryHit listed in the Appendix.

3.1.2 Scenario Walk-through and Functions of Each Component

Referring to Fig. 3, we illustrate how the components inside our CBIRP2P client collaborate to perform CBIR. In the architecture of CBIRP2P, two components are used to interface with other peers: (1) the HTTP Agent, which is used for direct transfer of image data outside the CBIRP2P network, and (2) the Packet Controller, which is used for exchanging messages within the CBIRP2P network. The Packet Controller component is responsible for delivering and receiving CBIRP2P messages, identifies their types and routes them to the corresponding component.

For instance, an incoming ImageQuery message received from other peers, is routed to the Similarity Comparison Agent. When the Similarity Comparison Agent receives the ImageQuery message, it searches the local index, which is preprocessed by Features Extraction Agent, to find similar images and delivers the ImageQueryHit message back to requester through the Packet Controller. Likewise, when a user initiates a query, the Features Extraction Agent extracts the features of the query image, packs the features to the ImageQuery message, and delivers it to other peers through the Packet Controller.

After initiating a query, the user waits for the replies from his peers. The User Interface Agent is notified when an ImageQueryHit message is received from his peers. Those messages are processed in order to show a preview of the query result. In order to download a shared image from other peers after previewing the result, the HTTP Agent makes a HTTP connection to the source provider and downloads the target image. On the other hand, when the source provider receives a download request, the File Locator Agent verifies the location of the requested file in the Shared Image Collection and completes the file transfer using the HTTP Agent.

3.2 Peer Clustering Based on Image Similarity

One of the problems in the P2P network for searching 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 is decentralized and inefficient. To solve the above problem, we propose to cluster peers that share similar image together, making the network organized in a systematic way like the Yellow Pages in order to improve query efficiency. Our peer clustering strategy makes use of forming deliberate *attractive links* between peers. These links are formed based on the two peers' shared image feature similarity within a neighborhood. With this added network topology as constraints, we propose the Firework Query Model that can perform searching efficiently by directing queries to their target cluster.

Similar to the concept proposed in BIRCH [29] for merging sub-clusters incrementally based on the clustering feature, we derive a strategy to group similar peers [14]. Inside the P2P network, each peer shares a set of images, it is responsible for extracting the content-based feature of 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 and use it to determine similarity between two peers. We begin by defining several key terms. A summary of these terms is listed in Table 1.

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-*

Table 1: Definition of Terms

$link_{random}(p)$	Random link–The connection 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, 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 which a peer p can reach within t hops.
$Collection(p)$	The set of images which a peer p shares.
$Match(Collection(p), Q)$	The collection of distance measure between each image in $Collection(p)$ to the query q.
$Number(Q)$	The number of results requested by user
$Threshold(Q), \theta$	The similarity threshold of a query

dimensional vector by function f, which extracts a real-valued d-dimensional vector as,

$$f : I \to R^d, \tag{7}$$

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 dimension, which will be served 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,*

$$\mu_j^p = \frac{1}{n}\sum_{i=1}^{n} R_{ij}^p \quad j = 1 \ldots d \tag{8}$$

$$\delta_j^p = \frac{1}{n}\sum_{i=1}^{n} (R_{ij}^p - \mu_j^p)^2 \quad j = 1 \ldots d \tag{9}$$

The following example shows a peer p shares seven images $[I_1^p, \ldots, I_7^p]$, where we extract the three-dimensional feature vectors as $[\vec{R_1^p}, \ldots, \vec{R_7^p}]$ respectively. $\vec{\mu^p}$ and $\vec{\delta^p}$ are computed as the signature value for this peer p as,

$$\begin{aligned}
\vec{R_1^p} &= (5.141, 5.735, 8.620) \\
\vec{R_2^p} &= (5.182, 6.121, 7.970) \\
\vec{R_3^p} &= (5.663, 5.911, 8.695) \\
\vec{R_4^p} &= (4.505, 4.656, 8.603) \\
\vec{R_5^p} &= (5.041, 6.499, 7.990) \\
\vec{R_6^p} &= (4.975, 5.820, 8.735) \\
\vec{R_7^p} &= (4.762, 5.677, 9.106) \\
\vec{\mu^p} &= (5.038, 5.774, 8.531) \\
\vec{\delta^p} &= (0.112, 0.274, 0.146)
\end{aligned}$$

Definition 3 *$Sim(p,q)$ is defined as the distance measure between $Cat(p)$ 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.*

$$Sim(p,q) = ||\vec{\mu^p} - \vec{\mu^q}||_2 + (\sum_{i=1}^{d} \delta_i^p \times \delta_i^q)^{1/2} \tag{10}$$

We assume each peer often shares images related to a certain topic. For example, a peer may share 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 each sub-cluster to form a cluster of peers which share similar images. Connecting peers with similar $\vec{\mu}$ and small $\vec{\delta}$ is analogous to achieving this. In (10), the more similar two peers p and q are, the smaller the value $Sim(p,q)$ is. $Sim(p,q)$ 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 sub-clusters are close in the high dimensional space. If both variances $\vec{\delta}$ measure are small, it means the image feature vectors in the two sub-cluster 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 calculated its a signature value, $Cat(p)$ based on the characteristic of images shared by that peer p, see Definition 2.

2. **Neighborhood Discovery**–When this peer joins the network, it will connect to another peer randomly chosen by the user. Through the ping-pong messages [6], it learns

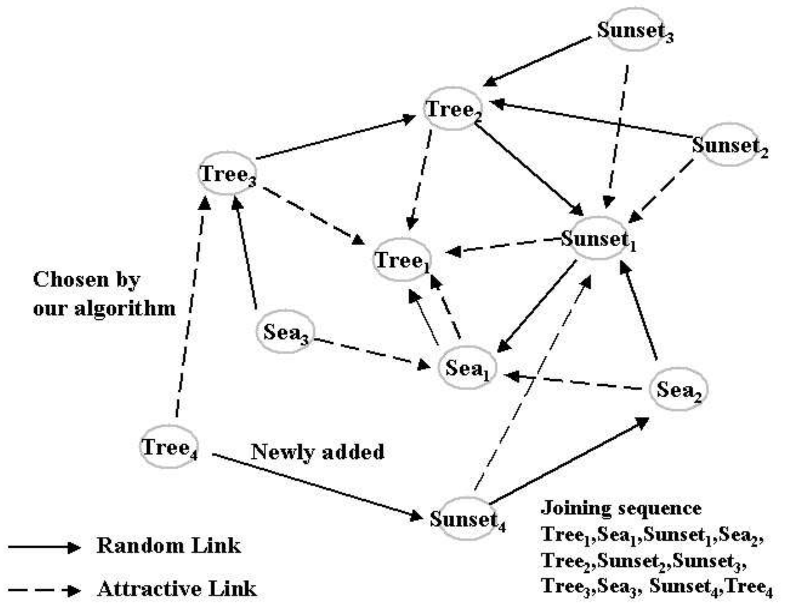

Figure 4: Illustration of peer clustering.

the characteristic of the set of peers within a certain number of hops away from it ($Peer(p,t)$).

3. **Similarity Calculation and Attractive Link Establishment**–Using the attractive link algorithm as shown in Algorithm 1, it will connect to another peer q having the lowest $Sim(p,q)$ value through attractive links.

Referring to Fig. 4, 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 is related to tree. When the peer $Tree_4$ joins the network, it makes a random link to $Sunset_4$, and by ping-pong messages, it learns the location and signature value of other peers, then it makes an attractive link to $Tree_3$ to perform peer clustering because $Sim(Tree_4, Tree_3)$ is the smallest, as shown in Table 2. As peers continue to join the network using this algorithm, peers of similar characteristic will be connected by attractive link gradually to form a cluster, which makes information retrieval much more systematic and efficient.

Algorithm 1 Algorithm for choosing attractive link

```
Attractive-Link-Selection(peer p, integer t)
for all $q$ in $Peer(p,t)$ do
    Compute $Sim(p,q)$
end for
assign $link_{attractive}(p)$ to $q$ with $min_{argq}(Sim(p,q))$
```

Table 2: Distance measure of each peer in Fig. 4 **A-Sea B-Sunset C-Tree**

Sim	C_1	A_1	B_1	A_2	C_2	B_2	B_3	C_3	A_3	B_4	C_4
C_1	0.0	**7.0**	**2.9**	2.2	**0.4**	3.7	1.9	**0.5**	6.5	2.0	0.9
A_1		0.0	3.0	**0.6**	6.0	1.0	4.0	2.3	**0.3**	2.5	4.5
B_1			0.0	4.0	6.6	**0.3**	**0.5**	4.5	7.3	**0.4**	5.7
A_2				0.0	0.9	2.2	3.5	9.8	0.5	1.7	2.6
C_2					0.0	7.8	7.8	0.7	7.1	7.7	0.8
B_2						0.0	0.7	5.5	3.6	0.6	6.1
B_3							0.0	6.3	2.8	1.0	5.1
C_3								0.0	8.2	4.3	**0.2**
A_3									0.0	9.0	3.3
B_4										0.0	6.2
C_4											0.0

3.3 Firework Query Model Over Clustered Network

To make use of our clustered P2P network, we propose the Firework Query Model, which aims to reduce the query message traffic. In this model, a query message first walks around the network from peer to peer through random links, by doing this, the message is routed selective towards its target cluster and avoids from passing through peers containing irrelevant data. Once it reaches the designated cluster, the query message will be broadcasted by peers through attractive link insides the cluster as shown in Fig. 5.

Here we introduce the algorithm to determine when and how a query message is propagated like a firework in Algorithm 2. Referring to Fig. 4 again, we illustrate with an example of the Firework Query Model. 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 distance between the query and its signature value $Cat(p)$. Since the distance measure between the query and its signature value is larger than a preset threshold, θ, according to Algorithm 2, $Tree_4$ sends the query to $Sunset_4$ through the random link because the target

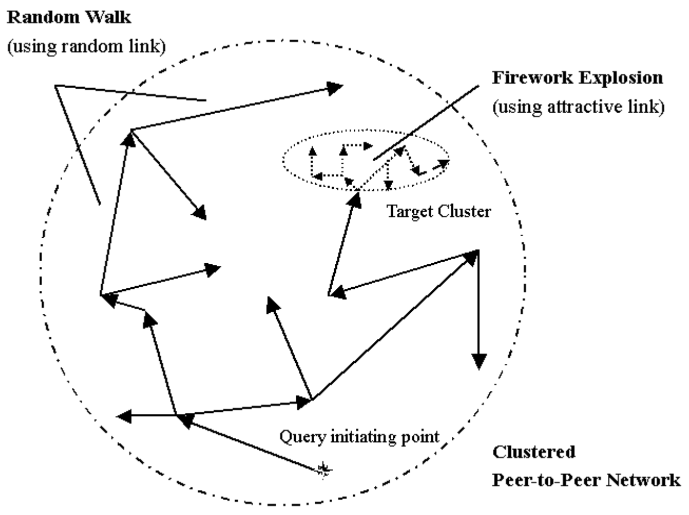

Figure 5: Illustration of firework query.

of query is not likely to appear in the cluster connected by 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 a L_2 norm defined as,

$$Sim(\vec{R_i^p}, \vec{Q}) = [\sum_j^d (R_{ij}^p - Q_j)^2]^{\frac{1}{2}}. \tag{11}$$

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

2. **Route Selection**–The peer calculates the distance between the query and his signature value, which is represented as,

$$Sim(p, \vec{Q}) = [\sum_j^d (\mu_j^p - Q_j)^2)]^{\frac{1}{2}}. \tag{12}$$

For the same reason, since the distance measure $Sim(p, \vec{Q})$ between $Sunset_4$ and query is larger than θ, it then forwards the query to Sea_2 through the random link again.

After walking around 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 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.

There are two mechanisms to prevent a query message from looping forever in the distributed network. One of the mechanisms is the inherent Gnutella replicated message checking rule. 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 forwarded. The second mechanism is to use the Time-To-Live (TTL) value to indicate how long a message can survive. Similar to IP packets, every Gnutella messages are associated with a TTL. Each time when the message passes through a peer, the TTL value is decreased by one. Once the TTL is zero, the message will be dropped and no longer forwarded.

The difference between our query messages and the Gnutella messages is that the TTL is not decreased when the messages are sent through attractive link. The reason is that, when a query reaches its target cluster, all peers inside the cluster are sharing highly related images, in order to get as much hits as possible, there is no reason to decrease the TTL and prevent further searching inside the highly related cluster. Hence, the only way to stop the infinite looping problem in a clustered network is by using the default message checking rule.

Algorithm 2 Algorithm for the Firework Query Model

```
Firework-query-routing (peer p, query Q)
if $Sim(p, Q) < \theta$ (threshold) then
  reply the query $Q$
  $TTL_{new}(Q) = TTL_{old}(Q)$
  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
```

3.4 Real-time Query Adjustment

So far, we have been focusing on how to reduce the network traffic by limiting the amount of query messages generated. On the other hand, we can also reduce the traffic by limiting the amount of query-hit messages. As the query-hit messages are usually long, they greatly increase the network traffic. When a user initiates a query, he may only be interested in a certain number of most similar matches. Any further matches does not improve the goodness of search result. On the other hand, the extra query-hit messages introduces unnecessary traffic in the network. Therefore, we might need to refine the matching criteria adaptively to restrict irrelevant images from being retrieved as the query message travels along the P2P network. We propose a strategy here to re-adjust the query message on-the-fly such that the amount of query-hit message generated will be reduced. Consider a user want to search for images that is of 80% similar to an example images provided by him. In a retrieval process, he might only want to get the top 20 results, it is wasteful for every peers to reply this query by returning all result that is 80% similar to the query. The idea comes like if a peer can return 20 results of 90% similar to the query, there is no need to further ask other peer for results with similarity 80% or more, instead, we have to restrict the similarity to 90% or more.

Referring to Fig. 6, suppose peers B, C, D and E contain 20 images of similarity within 1.5 to the query initiated by peer A. Without query adjustment, peers B, C, D and E will return the top 15 results in their collection back to peer A, generating a total of 60 results. (this result is not the actual image data, but the information about the location of images and the similarity value, please refer to Appendix) With the help of real-time query adjustment, peer B and C refine the query to be searching for images with in 0.8 and 0.9 similarity respectively. This is because the similarity of the 20^{th} retrieved images from B and C to the query is 0.8 and 0.9 respectively. Thus the number of images matches this criteria in peer D and E are reduced, so they only return 6 and 7 results respectively. The total number of results generated is 43 in this case.

Referring to the definitions in Table. 1, $Number(Q)$ is the maximum number of results to be retrieved in a query, $Threshold(Q)$ or θ is a degree of similarity chosen by the user to filiter out non-relevant result. By this algorithm, The query Q is refined as Q_{new} and pass along to other peers, the threshold is iteratively restricted, os non-relevant results are filtered out. Due to time limitation, the simulation of this part is not yet done, but we are sure the algorithm is able to reduce the amount of query-hit message generated.

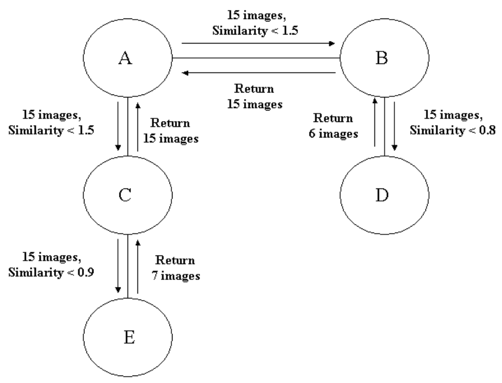
15 images, Similarity < 1.5
A
B
Return 15 images
15 images, Similarity < 1.5
Return 15 images
Return 6 images
15 images, Similarity < 0.8
C
D
15 images, Similarity < 0.9
Return 7 images
E

Figure 6: Real-time query adjustment.

Algorithm 3 Algorithm for the Real-time Query Adjustment

```
Real-time-Query-Adjustment (peer p, query Q)
$Result = Match(Collection(p), Q)$
Sort $Result$
$Q_{new} = Q$
**if** $|Result| > Number(Q)$ **then**
    $Threshold(Q_{new}) = Sim(Result[Number(Q)], Q)$
**end if**
Apply Firework-query-routing($p$, $Q_{new}$)
```

4 Experiments and Results

In this section, we present our experiment results and analyses. We investigate the effect on query efficiency subject to increasing number of peers and increasing of query message's TTL. In each of these two experiments, we further take the average measure of (1) number of query messages generated (network traffic cost), (2) recall (comprehensiveness of search), and (3) recall per query message (query efficiency) in twenty iterations giving us a total of six cases. In each of the six cases, we compare the differences in performance of the Random BFS, one of the de facto methods in the Gnutella network, and our proposed Firework Query Model.

Let P be the set of peers that receive query message, p_r be the number of query messages a peer p received. The number of query messages generated M, the recall RE and the query efficiency E are defined as follows:

$$\begin{aligned} M &= \sum_{p \in P} p_r, &(13)\\ RE &= \frac{|\{\forall_{I_i^p}, Sim(\vec{R_i^p}, \vec{Q}) < \theta, p \in P\}|}{|\{\forall_{I_i^p}, Sim(\vec{R_i^p}, \vec{Q}) < \theta\}|}, &(14)\\ E &= \frac{RE}{M}. &(15)\end{aligned}$$

In our simulation environment, we generate a certain number of nodes, assign each of them to a specific class respectively. Then, each node randomly picks a value of mean and variance according to the association to the class. Using this set of parameters, each node generates a set of data points following the Gaussian distribution, which is to model as the feature vectors set of the shared images it possesses. With each node having its own signature value as in Eq. 8, 9, we connect nodes together based on our proposed peer clustering strategy to form a special network topology. After building the network, we perform twenty runs of query evenly distributed within the high dimensional feature vector space and repeat this four times for different network topology. We take the average for number of query messages generated and recall accordingly.

The simulation is done on Sun Enterprise E4500 (12 400MHz Ultra IIi) running Solaris v.7 using C. For a simulation of 30,000 peers, the running time is approximately around 15 minutes. For the CBIR system, we build our client program based on Gnutella v0.4 protocol. For image related operations, we use ImageMagik library [7] to assist in extracting visual feature. When testing our client program, we use different port numbers to simulate different peers.

4.1 Increasing The Number of Peers

The first experiment tests the scalability of our system and algorithm by measuring the model's efficiency against the network size and the size of the image collection. The experiment parameters are listed in Table 3.

4.1.1 Number of Query Messages Against The Number of Peers

Figure 7 shows the number of query messages generated in two cases. We vary the number of peers in the network to observe the changes in number of query messages generated when a

Table 3: Parameters Used in the Experiments

Parameters	**Value and Description**
TTL of query message	6
Number of classes	30, evenly distributed within [0, 10]
Range of mean within a class	[-1,+1]
Range of variance	[0.08,0.6]
Number of dimension	6
Query range threshold, θ	0.9
Number of data points in one peer	15-30

peer initiates a search. From the observation, the Firework Query Model shows a promising sub-linear increase in the number of query messages subject to increase in number of peers in the network, while the BFS increases in a much faster rate. We conclude our model generate mush less traffic.

4.1.2 Recall Against the Number of Peers

Figure 8 shows the recall against number of peers in two cases. When the size of network increases, the recall of Firework Query Model continues to remain at a higher range, while the recall for BFS drops when size of network grows. We conclude that even the size of network grows, our model still can reach a large portion of the network containing the query target.

4.1.3 Query Efficiency Against the Number of Peers

Figure 9 shows the query efficiency against number of peers in two cases. When the size of network increases, both the efficiency of Firework Query Model and the BFS decrease. However, our proposed strategy always outperforms the BFS and the ratio of Firework Query Model to BSF keeps increasing as network size grows.

4.2 Increasing The Query Message's TTL

This experiment investigates what parameters setting can perform optimally in Firework Query Model. We look into the relationship of number of query messages generated and recall measure to figure out how to use the minimal number of query messages to cover the maximal portion of network containing relevant data. The experiment parameters are listed in Table 4.

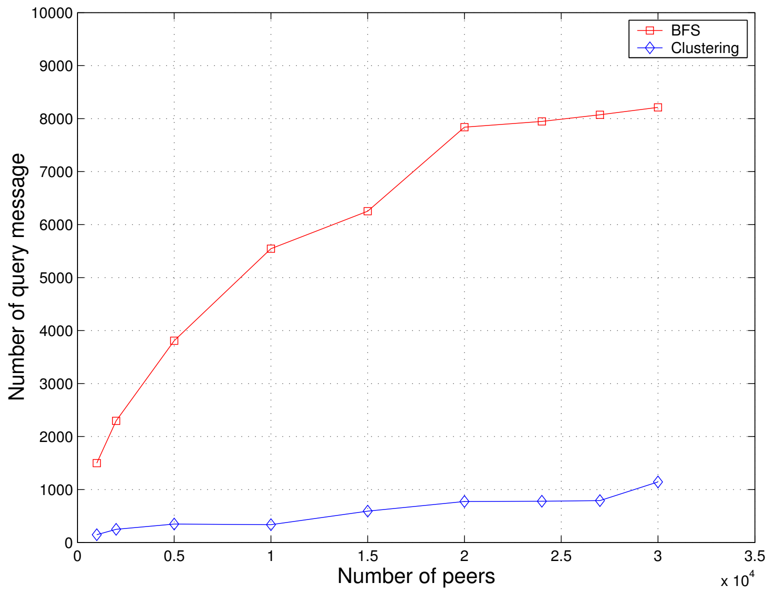

Figure 7: Number of query messages against the number of peers.

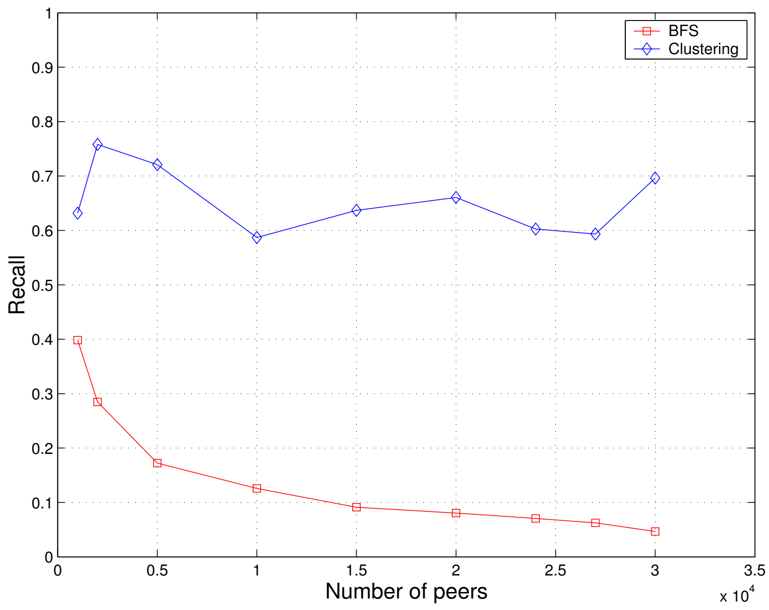

Figure 8: Recall against the number of peers.

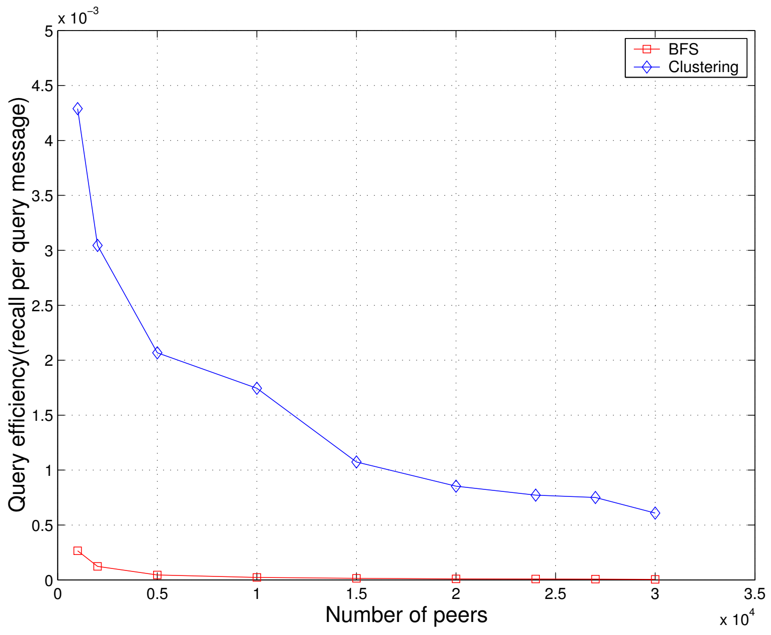

Figure 9: Query efficiency against the number of peers.

Table 4: experiment parameters used in investigating performance subject to change of TTL

Parameter	**Value and Description**
number of peers	10000
number of classes	30, evenly distributed within [0, 10]
range of mean within a class	[-1,+1]
range of variance	[0.08,0.6]
number of dimension	2
query range threshold, θ	0.3
number of data points in one peer	15-30

4.2.1 Number of Query Messages Against TTL of Query Message

Figure 10 shows the number of query messages generated in two cases. We vary the TTL of query message to observe the changes in the number of query messages generated when a peer initiates a search. Similar to the result in Section 4.1.1, the Firework Query Model shows a promising sub-linear increase in number of query messages subject to increasing TTL of query message, while the BFS increases in a much faster rate.

4.2.2 Recall Against TTL of Query Message

Figure 11 shows the recall against TTL of query message in two cases. When the value of TTL increases, both the recall of Firework Query Model and the BFS increase, while our proposed strategy increases in a much faster rate. When the TTL is larger than 7, the recall graph tails down in the Firework Query Model because the recall is nearly saturated and cannot be improved anymore.

4.2.3 Query Efficiency Against TTL of Query Message

Figure 12 shows the query efficiency against TTL of query message in two cases. As the recall remains nearly unchanged after TTL greater than six, increasing the TTL of query message successively introduce more unnecessary query messages, so we experience an increase in query efficiency when TTL less than six, while query efficiency drops when TTL is greater than six. We found that the optimal TTL value is six in a network size of ten thousands peers.

5 Future Work and Conclusion

Among the possible ways to extend the current work, the most challenging one is to perform normalization of feature vectors in a distributed environment. The need for normalizing image feature vectors is a must in order to avoid one particular dimension overshadowing the others. We need to have an algorithm to normalize image feature vector by just knowing a partial picture of the network in order to achieve better performance in CBIR.

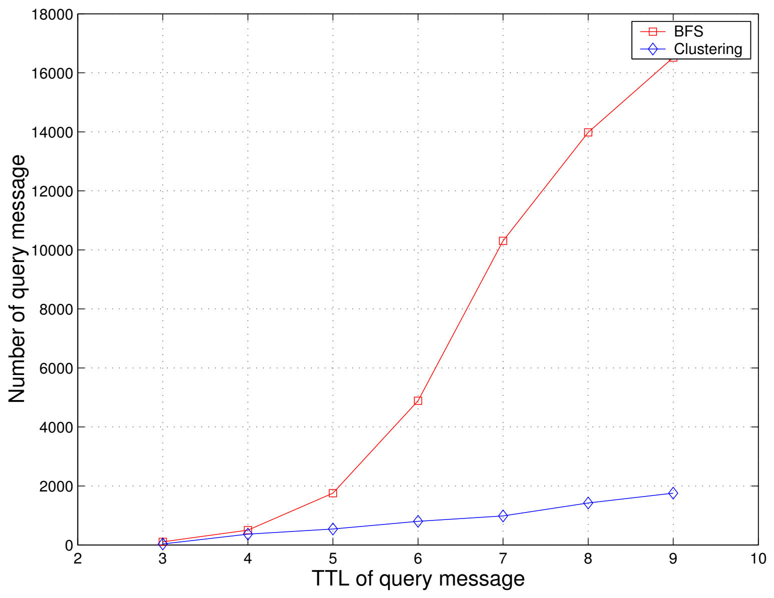

Figure 10: Number of query messages against TTL.

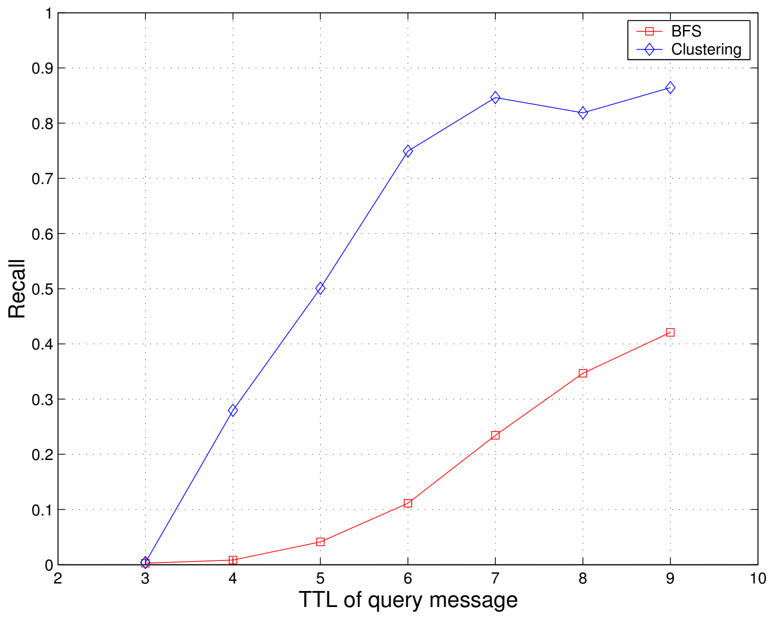

Figure 11: Recall against TTL.

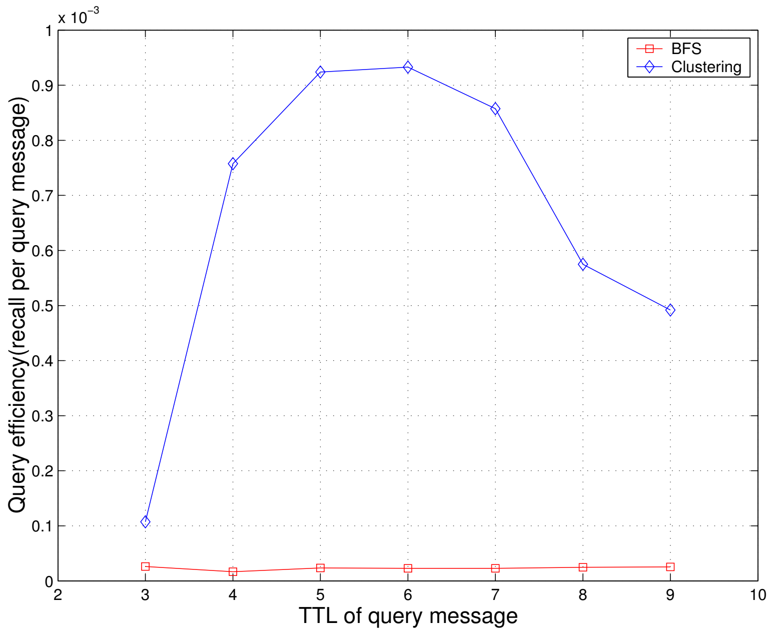

Figure 12: Query efficiency against TTL.

Normalization

In the definition of $Sim(p, q)$ described in section 3.2, we have assumed that the value of each vector component in the signature value, $\vec{\mu^p}$ and $\vec{\delta^p}$, are of the same dynamic range. Otherwise, the distance measure between two peers becomes less meaningful because one dimension may overshadow the others if its dynamic range is too large compared with others. For this reason, the vector components should be normalized before applying the similarity measure $Sim(p, q)$. Assume that there are N peers in the network and let p be the peer id index, then

$$\begin{aligned}\vec{\mu^p} &= [\mu_1^p, \mu_2^p, \mu_3^p, \ldots, \mu_k^p] \\ \vec{\delta^p} &= [\delta_1^p, \delta_2^p, \delta_3^p, \ldots, \delta_k^p]\end{aligned}$$

is the signature value for peer p, where k is the k-th component in vector $\vec{\mu^p}$ and $\vec{\delta^p}$. A good approach to normalize the above vector components have been proposed in [15],[12] , used the Gaussian normalization as follow :

$$\vec{k_{new}} = \frac{\vec{k} - \vec{\mu}}{\vec{\delta}} \tag{16}$$

However, the P2P network does not have centralized server to keep track of what images are being shared in what peer, and stores the signature value of each peer in the network, therefore, we cannot perform normalization without a collection of all the images features. In order to solve this, we may try to normalize the signature value as follow. First, a peer will send ping-pong messages to ask other's signature value of the set of peers within a certain number of hops away form it ($Peer(p,t)$). After collecting the replies from other peers, our strategy will perform a normalization of his personal signature value based on other's signature values using eq. 16. Likewise, the extracted features vectors of images are also need to perform normalization in order to get a satisfactory result. In our proposed strategy, it has one limitation, the size of ($Peer(p,t)$) must be large enough, so that the set of collected signature value of peers is able to represent the whole collection of images in the network. In order to show our strategy work, we need to carry out empirical experiments to verify in the future.

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 lookup 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 broadcasted throughout the network in order to achieve a satisfactory result. To solve this 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 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

CBIRP2P is a collaborative work with Mr. Ng Cheuk Hang. The author would like to thank him and the guidance of Prof. Irwin King in finishing this paper.

References

[1] C. Carson, M. Thomas, S. Belongie, J. M. Hellerstein, and J. Malik. Blobworld: a system for region-based image indexing and retrieval. In *Proc. Int. Conf. on Visual Information Systems*, 1999.

[2] P. Chen, E. Lee, G. Gibson, R. Katx, and D. Patterson. Raid: High-performance, reliable secondary storage. In *ACM Computing Surveys*, volume 26, pages 145–188, June 1994.

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

[4] I. J. Cox, M. L. Miller, T. P. Minka, T. V. Papathomas, and P. N. Yianilos. The bayesian image retrieval system, pichunter, theory, implementation, and psychophysical experiments. *IEEE Transactions on Image Processing*, 9(20-37), January 2000.

[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 gnutella homepage. http://www.gnutella.com.

[7] The imagemagik homepage. http://www.imagemagik.com.

[8] I. King and Z. Jin. Relevance feedback content-based image retrieval using query distribution estimation based on maximum entropy principle. In L. Zhang and F. Gu, editors, *Proceedings to the International Conference on Neural Information Processing (ICONIP2001)*, volume 2, pages 699–704, Shanghai, China, November 14-18 2001. Fudan University, Fudan University Press.

[9] Y. B. Lee and P. C. Wong. A server array approach for video-on-demand service on local area networks. In *Proc. IEEE INFOCOM' 96*, pages 27–34, 1996.

[10] Modern peer-to-peer file-sharing over the internet. http://www.limewire.com/index.jsp/p2p.

[11] W. Y. Ma and B. Manjunath. Natra: A toolbox for navigating large image databases. In *Proc. IEEE Int. Conf. Image Processing*, pages 568–571, 1997.

[12] 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.

[13] The napster homepage. http://www.napster.com.

[14] C. H. Ng and K. C. Sia. Peer clustering and firework query model. In *Proceedings of 11th World Wide Web Conference*, May 2002.

[15] M. Ortega, Y. Rui, K. Chakrabarti, S. Mehrotra, and T. S. Huang. Supporting similarity queries in mars. In *Prof. of ACM Conf. on Multimedia*, 1997.

[16] D. Patterson, G. Gibson, and R. Katz. A case for redundant arrays of interactive disks. In *Proc. of ACM International Conf. on Management of Data (SIGMOD)*, pages 109–116, May 1988.

[17] 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.

[18] M. Portmann, P. Sookavatana, S. Ardon, and A. Seneviratne. The cost of peer discovery and searching in the gnutella peer-to-peer file sharing protocol. In *Proceedings to the International Conference on Networks*, volume 1, 2001.

[19] A. Rowstron and P. Druschel. 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*, November 2001.

[20] 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.

[21] Y. Rui, T. S. Huang, M. Ortega, and S. Mehrota. Relevance feedback: A power tool for interactive content-based image retrieval. *IEEE Transactions on Circuits and Video Technology*, 8(644-655), September 1998.

[22] The worldwide computer. http://www.scientificamerican.com/2002/0302issue/0302anderson.html.

[23] The search for extraterrestrial intelligence homepage. http://www.setiathome.ssl.berkeley.edu.

[24] K. C. Sia and I. King. Relevance feedback based on parameter estimation of target distribution. In *Proceedings of International Joint Conference on Neural Networks*, May 2002.

[25] 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.

[26] 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.

[27] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *Proceedings of ACM SIGCOMM*, pages 149–160, August 2001.

[28] 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.

[29] 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.

Appendix: Modification of Gnutella Message

In order to implement a CBIR system on the current Gnutella network, we propose to extend the Gnutella protocol v0.4 [6] by adding two new descriptors as following:

- **ImageQuery**–Extension of the Query descriptor. This is the primary mechanism for searching images based on their servent content in the distributed network. A servent receiving an ImageQuery descriptor will respond with an ImageQueryHit if a match is found against its local data set. Four extract bytes are added to store the id of the feature extraction method. Another four extract bytes are added to store the matching criteria. Fig. 13

- **ImageQueryHit**–Extension of the QueryHit descriptor. It is the response to an ImageQuery. This descriptor provides the recipient with enough information to acquire the images matching the corresponding ImageQuery. 4 extra bytes of data are added in each result of the Result Set to store the similarity between each result to the corresponding query. Fig. 14

ImageQuery (0x90)

- **Minimum Speed**–The minimum speed of servents that should respond to this message.

- **Feature Extraction Method**–The identification number of the feature extraction method used in this query message.

- **Matching criteria**–The maximum distance measure of the query image to the shared images that peer required to reply the requester.

- **Search criteria**– A vector that is extracted by the method stated in **Feature Extraction Method** field.

ImageQuery Hit (0x91)

The descriptor of ImageQueryHit is the same as the original QueryHit, only that 4 bytes of data are added in each result of the Result Set to store the similarity between each result to the corresponding query.

	Minimum Speed	Feature Extraction Method	Matching criteria	Search criteria
Byte offset	0 1	2 5	6 9	10 …

Figure 13: ImageQuery message.

Result Set:

	File Index	File Size	Similarity	File Name
Byte offset	0 3	4 7	8 11	12 …

Figure 14: ImageQueryHit message.