

Peer Clustering and Firework Query Model in Peer-to-Peer Networks

Ng, Cheuk Hang

A Thesis Submitted in Partial Fulfillment
of the Requirements for the Degree of
Master of Philosophy
in
Department of Computer Science & Engineering

©The Chinese University of Hong Kong

June, 2003

The Chinese University of Hong Kong holds the copyright of this thesis. Any person(s) intending to use a part or the whole of the materials in this thesis in a proposed publication must seek copyright release from the Dean of the Graduate School.

Peer Clustering and Firework Query Model in Peer-to-Peer Networks

submitted by

Ng, Cheuk Hang

for the degree of Master of Philosophy
at the Chinese University of Hong Kong

Abstract

Clustering technique is used in database and information retrieval system for organizing data and improving retrieval efficiency. We surmise such functionality is valuable in a Peer-to-Peer (P2P) distributed environment. In this thesis, we introduce the concept of Peer Clustering at the level of overlaying network topology, thus, data inside the P2P network are organized in a fashion similar to a Yellow Pages. Moreover, the usability of these systems depends on effective techniques to retrieve information; however, the current strategies used in existing P2P systems are inefficient. To avoid query messages flooding and saving resources in handling irrelevant queries, we propose a content-based query routing strategy, the Firework Query Model, to improve existing retrieval methods. In contrast to broadcasting the query message, our query message is routed intelligently according to its content. Once it reaches the target cluster, the query message is broadcasted to all peers inside the cluster much like an exploding firework. We design and implement a DIStributed COntent-based Visual Information Retrieval (DISCOVIR) system with content-based query functionality and improved query efficiency. We demonstrate its scalability and efficiency through simulation.

Contents

Abstract	ii
1 Introduction	1
1.1 Main Contributions	3
1.2 Report Organization	4
2 Background	5
2.1 Background of Peer-to-Peer	5
2.2 Background of Content-Based Image Retrieval System	8
2.3 Literature Review of Peer-to-Peer Application	9
2.4 Literature Review of Discovery Mechanisms for Peer-to-Peer Applications	12
2.4.1 Centralized Search	12
2.4.2 Distributed Search - Flooding	14
2.4.3 Distributed Search - Distributed Hash Table	20
3 Peer Clustering and Firework Query Model	24
3.1 Peer Clustering	25
3.1.1 Peer Clustering - Simplified Version	26
3.1.2 Peer Clustering - Single Cluster Version	28
3.1.3 Peer Clustering - Single Cluster, Multiple Layers of Con- nection Version	34

3.1.4	Peer Clustering - Multiple Clusters Version	34
3.2	Firework Query Model Over Clustered Network	36
4	DIStributed COntent-based Visual Information Retrieval (DIS-	
	COVIR)	42
4.1	Architecture of DISCOVER and Functionality of DISCOVER	
	Components	43
4.2	Flow of Operations	47
4.2.1	Preprocessing (1)	48
4.2.2	Connection Establishment (2)	50
4.2.3	Query Message Routing (3,4,5)	51
4.2.4	Query Result Display (6,7)	52
4.3	Gnutella Message Modification	53
4.4	DISCOVER EVERYWHERE	55
4.4.1	Design Goal of DISCOVER Everywhere	57
4.4.2	Architecture and System Components of DISCOVER Ev-	
	erywhere	57
4.4.3	Flow of Operations	59
4.4.4	Advantages of DISCOVER Everywhere over Prevalent	
	Web-based Search Engine	60
5	Experiments and Results	62
5.1	Simulation Model of Peer-to-Peer Network	62
5.2	Performance Metrics	64
5.3	Experiment Results	66
5.3.1	Performances in different Number of Peers in P2P Network	66
5.3.2	Performances in different TTL value of query packet in	
	P2P Network	70
5.3.3	Performances in different different data sets, synthetic	
	data and real data	73

5.3.4	Performances in different number of local clusters of each peer in P2P Network	76
5.4	Evaluation of different clustering algorithms	82
6	Conclusion	85
	Bibliography	86

List of Tables

2.1	Summary of Discovery Mechanisms for Peer-to-Peer Applications	23
3.1	Definition of Terms using in Peer Clustering	29
3.2	Distance measure between peers in Fig. 3.6	33
4.1	ImageQuery Payload	54
4.2	ImageQueryHit Payload	55
5.1	Characteristic of Simulation Model	63
5.2	Parameters using in experiment 5.3.1	66
5.3	Parameters using in experiment 5.3.2	71
5.4	Parameters using in experiment 5.3.3	74
5.5	Parameters using in experiment 5.3.3	75
5.6	Parameters using in experiment 5.3.4	77
5.7	Parameters using in experiment 5.3.4	79
5.8	Result of confusion matrix of competitive learning.	83
5.9	Average clustering accuracy in estimated cluster number equal to the actual cluster number	84
5.10	Average clustering accuracy in under estimation of cluster number	84
5.11	Average clustering accuracy in over estimation of cluster number	84

List of Figures

2.1	Illustration of information retrieval in a P2P network.	6
2.2	Napster Architecture	13
2.3	Pure P2P Topology	14
2.4	Supernode Model	16
2.5	Illustration of Chord	20
2.6	Illustration of CAN	21
3.1	Illustration of two types of connections in DISCOVIR.	25
3.2	Illustration of Peer Clustering	25
3.3	Basic Peer Clustering.	26
3.4	Illustration of Peer Clustering Step 2	32
3.5	Illustration of Peer Clustering Step 3	32
3.6	Peer Clustering - Single Cluster Version	33
3.7	Peer Clustering - Single Cluster, Multiple Layers of Connection Version	35
3.8	Multiple clusters in a peer.	35
3.9	Illustration of Peer Clustering - Multiple Clusters Version, Step 2 and 3	37
3.10	Illustration of Firework query.	38
3.11	Illustration of Firework Query Model, Step 2	40
4.1	Screen Capture of DISCOVIR	43
4.2	Architecture of DISCOVIR	44

4.3	Interaction between DISCOVIR components	44
4.4	Screen Capture of Connection Manager	45
4.5	Screen Capture of Image Indexer	46
4.6	Screen Capture of Plug-in Manager	47
4.7	Screen Capture of Preprocess Procedure	49
4.8	Illustration of Attractive Connection Re-Establishment	50
4.9	Screen Capture of DISCOVIR draw-pad	51
4.10	Screen Capture of DISCOVIR Image Query	52
4.11	Screen Capture of DISCOVIR Query Result Display	53
4.12	Screen Capture of DISCOVIR Query Result Display 2	53
4.13	ImageQuery message format	54
4.14	ImageQueryHit message format	54
4.15	Screen Capture of DISCOVIR Everywhere	56
4.16	Architecture of DISCOVIR Everywhere	58
4.17	Query Procedure of DISCOVIR Everywhere	61
5.1	Power law distribution simulation model.	63
5.2	Recall against Number of Peers	67
5.3	Query Scope against Number of Peers	68
5.4	Query Efficiency against Number of Peers	69
5.5	Generated Network Traffic against Number of Peers	69
5.6	Minimum Reply Path Length against Number of Peers	70
5.7	Recall against TTL value of query packet	71
5.8	Query Scope against TTL value of query packet	72
5.9	Query Efficiency against TTL value of query packet	73
5.10	Query Efficiency against Number of Peers under different data sets	75
5.11	Query Efficiency against TTL value of query packet under dif- ferent data sets	76

5.12 Recall against Number of Peers	78
5.13 Query Scope against Number of Peers	78
5.14 Query Efficiency against Number of Peers	79
5.15 Generated Network Traffic against Number of Peers	79
5.16 Recall against TTL value of query packet	80
5.17 Query Scope against TTL value of query packet	80
5.18 Query Efficiency against TTL value of query packet	81
5.19 Generated Network Traffic against TTL value of query packet	81
5.20 Screen caption of competitive learning experiment	83

Chapter 1

Introduction

The appearance of Peer-to-Peer (P2P) applications such as Gnutella [14], Napster [34], Morpheus [32] and Freenet [13], have demonstrated the significance of distributed information sharing systems. These applications offer advantages of decentralization by distributing the storage, information and computation cost among the peers. 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 [38, 5]. 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 intruder to overcome because one needs to break into several computers before getting the file [47]. 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 [26]. Likewise, one may also distribute the computation among different computers to achieve a high throughput. Because of these desirable qualities, many research projects have been focused on designing different P2P systems and improving their performance. Regarding to current content-based image retrieval (CBIR) systems, we envisage the potential use of P2P network in both scattering data storage and distributing workload of feature extraction

and indexing.

In this thesis, we propose a strategy for clustering peers that share similar properties together, thus, data inside the P2P network will be organized in a fashion similar to that of the Yellow Pages. In order to make use of our clustered P2P network efficiently, we also propose a new content-based query routing strategy, the Firework Query Model (FQM) [36, 37], which aims to route the query intelligently according to the content of query to reduce the network traffic of query passing in the network. Our proposed routing and searching algorithm makes use of deliberately formed connection between peers and routing of queries intelligently to increase query performance without strict requirements on network topology and location of data placement, while adaptable to current P2P networks. Multimedia features are taken into consideration when building the network and routing queries.

To demonstrate our algorithm and show the desirabilities of P2P application, we design and implement the DIStributed COntent-based Visual Information Retrieval (DISCOVER) [9] and *DISCOVER Everywhere* [10] system, which are compatible to Gnutella¹ network, for users to share and retrieve images [50]. The motivation of proposing DISCOVER is to migrate traditional CBIR to a P2P network as a step to introduce content-based search in P2P. Peer Clustering and Firework Query Model is also implemented on DISCOVER. With the advantages of P2P networks, we utilize not only the distributed data storage, but also the computation power of each peer for the preprocessing and indexing of images. In order to improve the accessibility of P2P network, we further elaborate on current web-based P2P services and propose *DISCOVER Everywhere* to provide web interface for users to carry out

¹Gnutella is a famous protocol designed for sharing files in a distributed network. A protocol is a standard format that allows two pieces of software to communicate, like a language that two people both know. The Gnutella protocol was designed by AOL's Nullsoft division to surpass the file sharing capabilities of Napster, but was soon released to the public domain. It allows a user to share any type of file from his computer and make it available to anyone using Gnutella clients.

CBIR in P2P network with the following characteristics:

1. DISCOVIR increases the query efficiency by routing the queries intelligently according to the content of queries by using our proposed Peer Clustering and Firework Query Model. We reduce the network traffic generated, avoid irrelevant peers to handle the query to reduce the workload of computers and increase the performance of information retrieval.
2. DISCOVIR extends current centralized content-based retrieval systems into P2P fashion and achieve the utilization of both data storage and computation resource at the same time.
3. Queries in DISCOVIR are no longer based on simple texts but on the content of images. The need for annotating shared files is waived, thus, query accuracy does not depend on subjective perception of keywords by users.
4. *DISCOVIR Everywhere* provides an interface for web and mobile users to access the DISCOVIR network. This architecture gets rid of problems existing in current web-based P2P service by tightly integrating the web and P2P.

1.1 Main Contributions

Compared with other researchers' previous works, our main contributions to P2P are:

1. **Efficient Data Location**—efficiently locate the data under an environment with no index storage in centralized server. We formulate a query model that improves query efficiency under the content-based search architecture and reveal the research need for improving query efficiency.

2. **Rich Query in Multimedia Files**—perform query based on content of information rather than simple filename or meta data in P2P networks. The queries in current P2P applications are mostly based on text, entered by users to describe shared files. The accuracy of retrieval depends mainly on whether users can come up with a common description on a file. CBIR raise another aspect of content-based search in P2P other than filename-based search.
3. **DISCOVIR and DICOVIR Everywhere systems Implementation**— extend current CBIR system into P2P fashion and achieve the utilization of both data storage and computation resource. We also implemented clustering algorithm and content based routing strategy on DISCOVIR to demonstrate Peer Clustering and Firework Query Model on P2P applications. Current P2P applications require installing special purpose software and proprietary protocols for information retrieval, which limit the number of audience. To make use of the WWW to increase popularity of P2P, we propose *DISCOVIR Everywhere* to act as an interface to for web and mobile users to access the DISCOVIR network.

1.2 Report Organization

In the following, we first review current issues of P2P and CBIR in Chapter 2. We present the algorithm of Peer Clustering and Firework Query Model in Chapter 3. In Chapter 4, we introduce the architecture of DISCOVIR and the functionality of its components. Then, we proceed to report and analyze our experimental results in Chapter 5. We give our final remarks and conclusion in Chapter 6.

Chapter 2

Background

2.1 Background of Peer-to-Peer

Both Napster and Gnutella have demonstrated the possibility of distributing storage over computers in the Internet. Such kind of P2P networks offer the following advantages:

1. **Resource Utilization**—The storage, information and computational cost can be distributed among the peers, allowing many individual computers to achieve a higher throughput [48].
2. **Increased Reliability**—The P2P network increases reliability by eliminating reliance on centralized coordinators that are potential critical points of failure [7, 6].
3. **Comprehensiveness of Information**—The P2P network has the potential to reach every computers on the Internet, while even the most comprehensive search engine can only cover 20% of web-site available as stated in some statistics [28].

Figure 2.1 shows an example of a P2P network that different information is shared by different peers. When a peer initiates a search, 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 computer, which joins and leaves 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 responds to queries for information it shares locally.

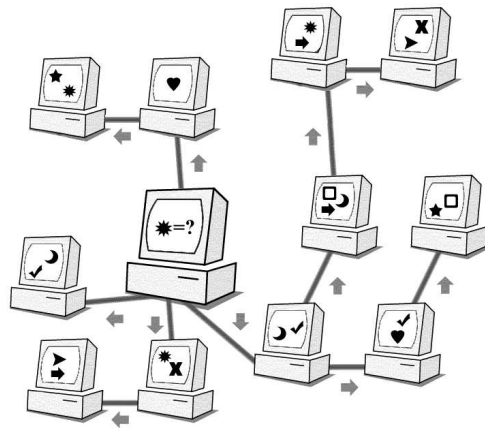


Figure 2.1: Illustration of information retrieval in a P2P network.

Current strategy still needs a lot of improvement to solve the scalability problems:

1. The bottle-neck occurs at the centralized server storing the index, like Napster.
2. The flooding of query messages occurs when data location process is decentralized, like Gnutella.

To address the data location problem, Chord [54], CAN [42], Pastry [44] and Tapestry [61] 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) to map the filename to a key, and each peer is responsible for storing a certain range of pairs (key, value). 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 -bits 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 point on a d -dimension Cartesian coordinate space, while each peer is responsible for pairs (key, value) inside its specific region. 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 [55, 19]. Although DHTs are elegant and scalable, their performance under the dynamic conditions for P2P systems is unknown[43]. Moreover, such kind of schemes rely on the trustworthiness of peers participating in the network. The malicious peers are supposed to be responsible for answering queries but the problem become serious when they deny to respond under the condition of no duplicate index storage in other peers. As DHTs mandate a specific network structure and queries are based on document identifiers, researchers proposed methods that operate under the prevalent P2P environment, like Gnutella, and queries are based on content of documents. Crepsio [8] proposed a routing indices approach for retrieving text documents in P2P systems. Under this scheme, each peer maintains a routing index to assist in forwarding queries to peers that contains more documents of the same category. This method requires all peers to agree a set of document categories. Sripanidkulchai et. al. [53] proposed the use of short-cuts to connect a peer to another one which it has downloaded documents from. Evaluations are done based on text document retrieval and promising results are shown. Our proposed method targets on content-based retrieval in P2P network and aims at reducing the network traffic and workload of computers. Similar problems of Content-Based Image Retrieval (CBIR) in

distributed databases have been described in [4]; in essence, more studies are needed of the P2P systems.

2.2 Background of Content-Based Image Retrieval System

In early image retrieval system, it requires human annotation and classification on the image collection, the query is thus performed using text-based information retrieval method. However, there are several limitations for such implementation, they are:

1. **Human Intervention** - Human intervention is required to describe and annotate the content of images, which is tedious and potentially error-prone.
2. **Non-Standard Description** - As the size of image database grows, limited keywords results in inadequacy for describing the image content. Moreover, the keywords used are subjective and not unique. Different users may use different keywords to annotate the same image.
3. **Linguistic Barriers** - If the image database is to be shared globally around the world, the retrieval of images will be ineffective when different languages are used in the description. It is difficult to map semantically equivalent words across different languages.

In order to solve these problems, Content-Based Image Retrieval (CBIR) is proposed to pass such tedious task to computer. Since the mid 1990's, many CBIR systems have been proposed and developed, some of them are QBIC [12], WebSEEK [52], SIMPLicity [57], WBIIS [56], MARS [31], Photobook [39], NeTra [30], AMORE [33], Virage [18] WALRUS [35] and other

systems for domain specific applications [25, 23]. These systems are not designed to be distributed across different computers in a network. One of the shortcomings is that the feature extraction, indexing, clustering and also the query processing are all done in a centralized fashion which is computationally intensive, and it is difficult to scale up. As indicated by several researchers [45, 51], one of the promising future trends in CBIR includes the distribution of data collection, data processing and information retrieval. By extending the centralized system model, we not only can increase the size of image collections easily, but we also overcome the scalability bottleneck problem by distributing the computationally intensive processes among peers.

2.3 Literature Review of Peer-to-Peer Application

The application of Peer-to-Peer can be divided into three categories, namely, distributed file sharing, person-to-person messaging systems and distributed computing systems [22].

1. Distributed File Sharing

Distributed File Sharing deals with the strategies and technologies for effective ways to retrieve contents existence, its location, and to get it delivered. It involved the information and knowledge management. The power of direct exchanges and discovery searches between peers can be used to enhance the effectiveness management. Distributed file sharing applications, include Napster [34], Gnutella [14], Freenet [13], eDonkey [11] and Morpheua [32], allow peers to share files with every other peer in an application-based network. P2P file sharing extends traditional LAN-based file sharing to the Internet. This is powerful technology for enabling cooperation across organizations and between companies.

- (a) **Napster** is an excellent application for distributed MP3 search engine. Its architecture describes the server-mediated file-sharing model popularized by this client and discuss some of the reasoning behind this compromise away from "pure" P2P. Since the information in traditional web search engine can get easily out of date, existing MP3 search engines are really difficult to maintain listings, because many of the sites posting MP3 files may not exist for long, especially if they post illegally copied songs. In contrast, Napster gets its listings from those running its software. If you are running Napster, it will tell the server what information you have. That allows someone to search your listings and be connected to where the song can be downloaded. Due to violations of the copyright laws, Napster is forced to shutdown servers. Besides the copyright infringements, centralized character of server is another problem of Napster. Under this structure, Napster fails to provide quality-of-service because the services are easily interrupted if the centralized servers crash. In order to overcome the problem, decentralized peer-to-peer model are proposed. Gnutella, Freenet and Morpheus are some examples of such model.
- (b) **Gnutella** is one of the well-known protocols for file sharing and searching in a decentralized peer-to-peer environment. In the release of beta version, almost everyone saw a competitor to Napster designed to overcome its restrictions and limitations for swapping data files. Under the Gnutella network, users can find information from peers based on the filename and some simple Meta data. Gnutella application that builds on its protocol is a peer-to-peer system with client software that also acts as a server, called servant. Gnutella servant provide client-side interfaces through which users can issue queries and view search results, while at the same

time they also accept queries from other servers, check for matches against their local data set, and respond with applicable results.

- (c) **Freenet** is decentralized and automatically adapts when hosts leave and join. It is a peer-to-peer network application that permits the publication, replication, and retrieval of data while protecting the anonymity of both authors and readers. Unlike Napster and Gnutella, no brute force search or centralized location index is employed. Files are referred to in location-independent manner, and are dynamically replicated in locations near requesters and deleted from locations where there is no interest. Freenet uses an interesting approach to indexing. Each node builds an index with the location of recently requested documents, so if they are requested again, the document can be retrieved at a low cost.
- (d) **Morpheus** is a decentralized peer-to-peer file sharing application that allows users connect directly and share information. Under Morpheus, users are able to search for all types of digital media by providing Meta data such as media type, performer and product name, and not only limited to filename.

2. Person-to-Person Messaging

Instant messaging systems such as Jabber [20] or Yahoo! Messenger [59] allow peers to exchange text as well as white-board type of messages. Jabber is an XML-based peer architecture, not any specific application. Current Jabber clients focus on instant messaging and allow P2P file transfer between users, however, its protocol and architecture is open to any number of other uses. The stated focus of Jabber is not only for person to person communication, but also person to application and especially application to application.

3. Distributed Computing

In a peer-to-peer distributed computing system, a node gathers results from computations on raw information that was scattered across several tightly or loosely coupled processors. Companies and organizations can utilize available computational cycles to solve complex scientific and engineering problems.

2.4 Literature Review of Discovery Mechanisms for Peer-to-Peer Applications

Peer-to-peer applications allow peers to connect or disconnect from a network at any time and are based on a loosely coupled resource distribution model. Therefore, robust and efficient discovery mechanisms are central to the efficient functioning of distributed file sharing applications. Under a pure peer-to-peer network, searches are spread among many nodes because there is no centralized database. Different distributed file sharing model represents a trade-off between the open-ended approach on one hand and constraint issues on the other, such as requirements on efficient search, good content availability, secure, redundant storage and possibly anonymity of source.

2.4.1 Centralized Search

Napster

Figure 2.2 shows the server-mediated architecture of Napster. Under the Napster network, clients connect automatically to a connection manager. This connection manager assigns an available and lightly connected server to the incoming client. This client then registers with the assigned server, providing its shared information to the server side database. Once this process is finished, other clients is possible to request his registered shared information. In

turn, the client can also receive information from other connected users, and exchange files directly.

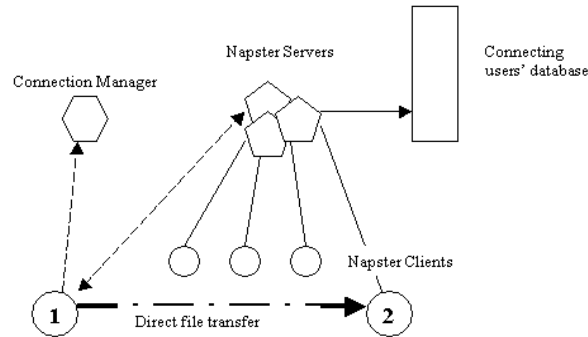


Figure 2.2: Napster Architecture

Under this model, users are almost anonymous to each other because the users are never queried directly. When the client search for a file, the query will be sent to the centralized user database to search for the requested content and determine a neighbor which to download. The centralized user database therefore serves as a background translation services, from the host identity associated with particular content, to the currently registered IP needed for a download connection to this client.

The network performance of Napster is excellent comparing to other P2P discovery mechanisms. The query is only sent to connecting centralized server. It is not necessary to broadcast the query to ask everyone in the network. The response time of searching is also good in Napster because query is not necessary to past several peers in the network, however, its performance is limited to the performance of server which is affected by the number of connecting clients, number of query per client and the number of shared files of each client.

Besides the copyright infringements, centralized character of server is another problem of Napster. Under this structure, Napster fails to provide

quality-of-service because the services are easily interrupted if the centralized servers crash. Too much connections to a single server will also degrade the services greatly. In order to overcome the problems, decentralized peer-to-peer model are proposed. Gnutella, Freenet and Morpheus are some examples of such model which aim to solve the limitations of Napster.

2.4.2 Distributed Search - Flooding

Gnutella

Gnutella is a file sharing and exchange protocol that support arbitrary types. Under Gnutella network, there are no central servers, thus, no central shutdown point. Since central servers do not exist, therefore, queries are required to broadcast to all his neighbors within a certain range. This kind of searching is called Brute Force Search(BFS).

In Gnutella, when the user initializes a query, he needs to broadcast the query to his direct connecting neighbors. His neighbor will look up his shared collection and answer the query. Then, he will help to propagate the query to his connecting neighbors. Likewise, his neighbors continue to answer and propagate the query. In practice, flooding is limited to peers' neighborhoods by limiting query propagation to a fixed number of hops, called Time-To-Live(TTL). Typically, the TTL is set between 5 to 7, and the value is decremented by each neighbors as it relays the message.

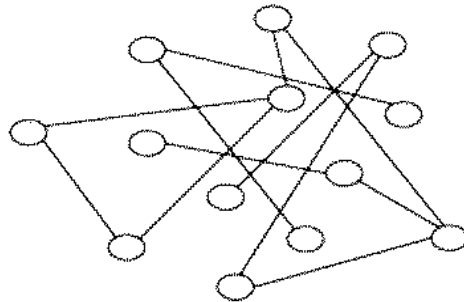


Figure 2.3: Pure P2P Topology

Although this protocol eliminates the problems of central server failure and gives optimal results in a network with a small number of peers, however, it does not scale well. Since every query is broadcasted to every peer in the network, each peer has to waste resources in handling irrelevant query. Moreover, broadcasting query messages across the network also increases network traffic. Portmann et. al. [40] investigated the problem of scalability in P2P network due to network traffic cost. Furthermore, accurate discovery of peers is not guaranteed and the response time is usually slow because queries are required to propagate from peer to peer.

Gnutella Varieties

Following strategies try to improve the limitation and weakness of Gnutella network by routing the query intelligently, reducing network traffic and solving the scalability problems.

1. Gnutella Supernode Model

The current network of Gnutella makes query messages fly everywhere. In the Gnutella supernodes model proposed by Limewire [28] as shown in Figure 2.4, each node comes to the system as a super node, and establishes the configured number of supernode connection. The nodes remains as supernode on probation during the probation time, if it receives required number of Client connection requests so as to achieve at least min clients, it continues to remain as Supernodes. At any instant if the number of client connections fall below min clients, it again goes on probation. In the probation time, if it never achieved min clients connection, it becomes a client node and establishes a client connection to a supernode. When the client node wants to query an item, he sends the message to his supernode and his supernode broadcast the query message to other supernodes. Each supernode maintains the index of

his client nodes. Therefore, the client nodes only receive the queries if they actually have the request files. Otherwise, the supernode does not forward the search to them. In their preliminary results, client nodes receive almost no traffic over their single, supernode connection, while supernodes are not adversely affected by the increase in connections. The supernodes and clients also use query routing. They allow the network to function for more efficiently, with far less messaging required across the network to search for files and to connect. With this technology, Gnutella may significantly increase the number and quality of search results, and increase the scalability of Gnutella network.

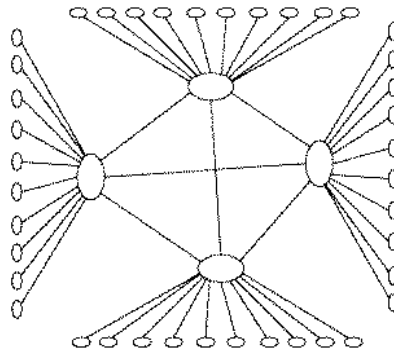


Figure 2.4: Supernode Model

In my opinion, supernode can be elected by a set of nodes. Several nodes are divided into a cluster and each node inside the cluster elects a representative to be the supernode of this cluster. The cluster may be formed based on the shared content or geographical location. The electing criteria of the supernode may depend on their bandwidth, client alive time or other possible factors.

This model forms a more reliable and persistent backbone. This also gives the option of some permanent nodelist server to make it easier for new users to connect to the network for the first time. This approach makes the network far less sensitive to the transient connectivity and

limited bandwidth caused by the large number of dial-up users. However, this architecture is more sensitive to the failure of the supernodes and faces the same problems of central servers as Napster.

2. Prinkey Scheme

In Prinkey's Scheme [41], the metadata keywords are indexed by a hash function. Therefore, this information, called bitmask signature, can be used to determine if a node possibly contains information relevant to our query. Then, this signature is passed up to their host node. The node remembers each of the bitmask signature for its hosted nodes. Then it takes the bitmasks from its own index and logically ORs them with all of the bitmasks from its hosted nodes. This aggregate bitmask provides a signature of all of the information in the entire branch. Likewise, this aggregate bitmask is passed up its host continuously until it reaches the root host.

Under this model, queries are only need to be routed to the nodes that possibly contain information relevant to the queries. It guarantees that the query will always sent to the hosts possibly contains information relevant to our query. However, it does not guarantee that the hosts received the query must contain information relevant to our query because different keywords may map to the same value in a hash function.

Moreover, this scheme has another limitations. First of all, all leaves nodes are needed to propagate their bitmask to their root nodes; therefore, the network is needed to be a tree topology with a designated root. Another limitation is that if the leave node wants to query the network, it must send the query up to the root. This causes the root nodes which handles extremely high amount of traffic and very slow response time because queries are required to propagate from leave node to root node.

3. Iterative Deepening Approach

Over the iterations of the iterative deepening technique [60], multiple breadth-first searches are initiated with successively larger depth limits, until either the query is satisfied, or the maximum depth limit D has been reached. Because the number of nodes at each depth grows exponentially, the cost of processing the query multiple times at small depths is small, compared to processing query once at a large depth. In addition, if it can satisfy the query at a depth less than D , then it can use much fewer resources than a single BFS of depth D .

The iterative deepening technique is implemented as follows: first, a system-wide policy is needed, that specifies at which depths the iterations are to occur. For example, say it want to have three iterations: the first iteration searches to a depth a , the second to depth b , and the third at depth c . Its policy is therefore $P = \{a; b; c\}$. Note that in order for iterative deepening to have the same performance as a BFS of depth D , in terms of satisfaction, the last depth in the policy must be set to D . In addition to a policy, a waiting period W must also be specified. W is the time between successive iterations in the policy. After waiting for a period W , if the requester finds that the query already has been satisfied, then it does nothing. Otherwise, the requester will start the next iteration, continue to repeat the procedures until the query is satisfied or the maximum depth limit D has been reached.

This approach tries to reduce unnecessary traffic if the query can be satisfied in a small deep. However, the response time of this algorithm is even more slowly than the basic Gnutella network because the time taken by multiple iterations is always long. If minimizing response time is important to a particular application, then the iterative deepening technique may not be applicable.

4. Directed Brute Force Search

The Directed BFS technique [60] implements this strategy by having a query source which sends the query messages to just a subset of its neighbors, but selecting neighbors through which nodes with many quality results may be reached. For example, one may select a neighbor that has produced or forwarded many quality results in the past, on the premise that past performance is a good indication of future performance. The neighbors that receive the query then continue forwarding the message to all neighbors as with BFS.

In order to intelligently select neighbors, a node will maintain statistics on its neighbors. These statistics can be very simple, such as the number of results that were received through the neighbor for past queries, or the latency of the connection with that neighbor. From these statistics, it helps to select the best neighbor to send the query. Similar idea is also proposed in [21].

Under this model, the number of nodes that receives the query is greatly decrease. The query will only be forwarded intelligently to the selected neighbors who are believed to produce many results.

Conclusion of Distributed Search - Flooding

Gnutella and its varieties are described as loose systems. They do not tightly control the data placement and topology within the network. They do not guarantee location of content if it exists. These kinds of network are suitable to be used in a wide range of non-cooperating and non-trusted organizations. However, broadcasting query messages across the network makes peer-to-peer network impossible to scale up. Therefore, many varieties of Gnutella tried to improve the Gnutella network by routing queries intelligently.

Some systems with strong guarantees on availability include Chord [54], CAN [42], Pastry [44], and Tapestry [61]. These four techniques are quite similar in concept, but differ slightly in algorithmic and implementation details. In Chord, Pastry and Tapestry, nodes are assigned a numerical identifier, while in CAN, nodes are assigned regions in a d -dimensional identifier space. A node is then responsible for owning objects, or pointers to objects, whose identifiers map to the node's identifier or region. Nodes also form connections based on the properties. More details of these system will be discussed in the following section.

2.4.3 Distributed Search - Distributed Hash Table

Chord

The Chord project is the part of an ongoing large distributed secure file system project that looks at the key location and routing in a overlay network represented as a one dimensional circular identifier space as shown in Figure 2.5. They guarantee location of content if it exists.

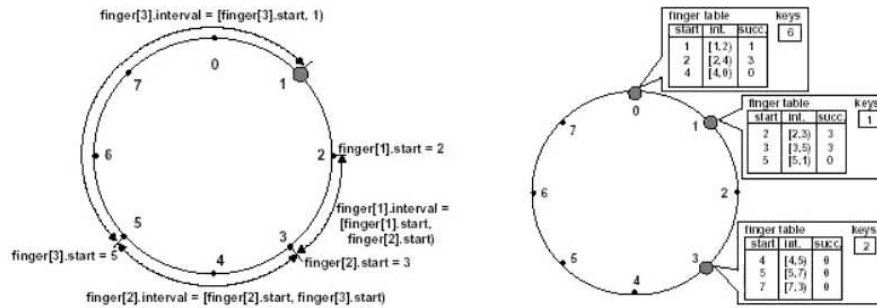


Figure 2.5: Illustration of Chord

In the Chord network, keys are assigned to peers using consistent hashing algorithm, enabling a node to locate a key in $O(\log N)$ hops when N is the total number of peers in the system. For each peer, it maintain a finger table pointing to the successor of $\log N$ identifier peers along the ring. Therefore,

each peer only stores information about only a small number of other peers, and knows more about peers closely following it on identifier circle than about peers farther away.

Pastry

The concept of Pastry is similar to Chord, but differ slightly in algorithm and implementation. In Pastry, peers are also assigned a numerical identifier. Pastry is a peer-to-peer routing substrate that is efficient, scalable, resilient and self-organizing. Given a field ID, Pastry routes an associated message towards the peer whose node ID is numerically closest to the 128 bits circular index space among all live peers. Each peers maintains a routing table of $O(\log N)$, where N is the number of active Pastry peers. Pastry attempts to minimize the distance traveled by the message by taking network locality into account.

Content-Addressable Network(CAN)

The concept of CAN is a little bit different to Chord and Pastry. In Chord and Pastry, peers are assigned a numerical identifier, while in CAN, peers are assigned regions in a d -dimensional identifier or region.

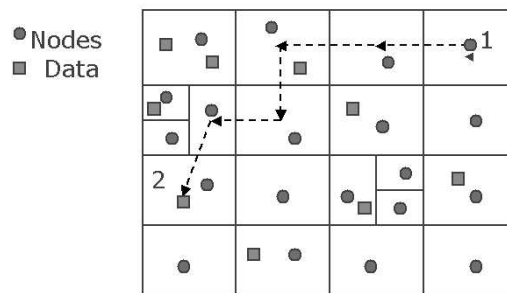


Figure 2.6: Illustration of CAN

A Content-Addressable Network(CAN) is a mesh of n peers in a virtual d -dimensional coordinate space. This virtual coordinate space is dynamically

partitioned among all the peers such that every node owns its distinct zone within this space. The coordinate space is used to store (key, value) pairs such that every key k is deterministically mapped onto a point P in the coordinate space using a uniform hash function. The corresponding key-value pair is stored at the node in whose zone point P lies. To retrieve a value corresponding to key k , any node can apply the same deterministic hash function to map k onto point P and then retrieve the value from point P either directly or via neighboring peers.

Conclusion of Distributed Search - Distributed Hash Table

Chord, CAN and Pastry are systems with strong guarantees on availability. Their techniques are quite similar in concept, but differ slightly in algorithmic and implementation details. In Chord, Pastry and Tapestry, nodes are assigned a numerical identifier, while in CAN, nodes are assigned regions in a d -dimensional identifier space. A node is then responsible for owning objects, or pointers to objects, whose identifiers map to the node's identifier or region. Nodes also form connections based on the properties.

In these systems, they guarantee location of content if it exists. However, the topology of these networks are needed to be fixed. Some of them tightly control the data placement. In order to lookup information correctly and quickly, they need top-reserve invariants. For example, each peer's identifier is needed to correctly maintain, the network topology is fixed, it is also desirable for the routing tables to be correct in every peers. These kinds of network are only suitable to be used in some cooperating computers and trusted organizations.

Table 2.1: Summary of Discovery Mechanisms for Peer-to-Peer Applications

Centralized Search	
Applications	Napster
Characteristics	File indexing, searching are all done in the centralized server.
Strength	<ol style="list-style-type: none"> 1) Less network traffic, broadcasting is not needed. 2) Fast response time if the server is not overloaded.
Weakness	<ol style="list-style-type: none"> 1) Single logical point of failure. 2) Potential for congestion. 3) Fail to provide quality-of-service as too much connections to the server will degrade the services greatly. 4) Easy interrupted if the centralized servers crash.
Distributed Search - Flooding	
Applications	Gnutella, Kazaa
Characteristics	<ol style="list-style-type: none"> 1) No centralized servers. 2) Query is needed to broadcast within a certain range.
Strength	<ol style="list-style-type: none"> 1) No fixed network topology. 2) No fixed data and index placement.
Weakness	<ol style="list-style-type: none"> 1) Broadcasting query generated heavy network traffic. 2) Irrelevant computers are forced to handle the query. 3) The network is not scalable. 4) Slow in response time because of query propagation.
Distributed Search - DHT	
Applications	CAN, Chord, Pastry
Characteristics	<ol style="list-style-type: none"> 1) Use Distributed Hash Table(DHT). 2) Map and lookup the file by hash key. 3) Speed up and reduce message passing for the process of key lookup.
Strength	<ol style="list-style-type: none"> 1) Efficient at locating information. 2) Scalable.
Weakness	<ol style="list-style-type: none"> 1) Impossible to perform a fuzzy search. 2) Susceptible to malicious activity. 3) Performance under dynamic conditions is unknown.

Chapter 3

Peer Clustering and Firework Query Model

The design goal of our strategy, Peer Clustering and Firework Query Model, is to improve data lookup efficiency in a completely distributed P2P network. We aim to maximize the retrieval performance, minimize the number of message passing through the network, and retain the simple, robust and completely decentralized nature of Gnutella network.

We purpose Peer Clustering at the level of overlaying network topology to make the network organized in a systematic way like the Yellow Pages. In our proposed network, there are two types of connections, namely *random* and *attractive* as shown in Fig. 3.1. Random links are the original connections used in Gnutella network. Attractive links are the newly introduced connections made by our algorithm. On top of the original Gnutella network, our strategy makes use of an extra layer of connections, attractive links, to group similar peers together based on two peers' similarity within their neighborhood as shown in Fig. 3.2. With these added network topology constraints, we propose a content-based query routing strategy, the Firework Query Model, which can perform searching efficiently by directing queries to their target cluster according to the query content. Therefore, our algorithm manages to be scalable when network grows.

We have implemented a prototype version of Distributed Content-based Visual Information Retrieval System (DISCOVER), built on top of LimeWire open source project [28] with content-based image searching capability, to demonstrate how Peer Clustering and Firework Query Model work.

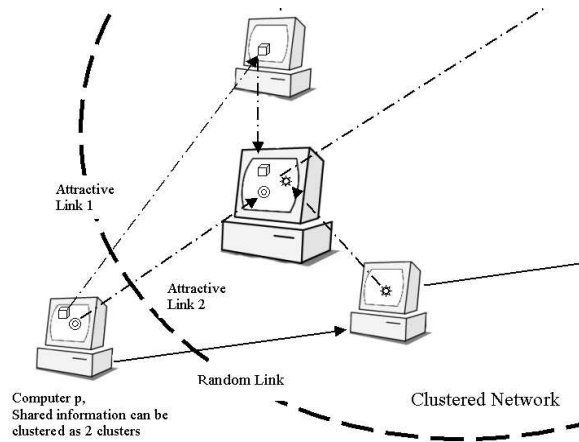


Figure 3.1: Illustration of two types of connections in DISCOVER.

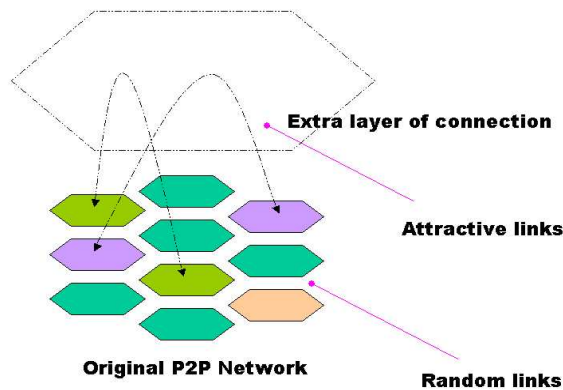


Figure 3.2: Illustration of Peer Clustering

3.1 Peer Clustering

In this section, we introduce four versions of Peer Clustering algorithms from the simplest version to the most complicated and practical version. Four versions are: *Simplified Version*, *Single Cluster Version*, *Single Cluster-Multiple*

Layers of Connection Version and Multiple Clusters Version.

3.1.1 Peer Clustering - Simplified Version

We start by describing a more restricted version of our proposed algorithm to illustrate the simplest idea of peer clustering. Consider a P2P network consisting of peers sharing data of different interests as shown in Fig. 3.3. In this network, shared data are classified into 3 different categories, which are (C)omputer, (H)istory, and (S)cience respectively. Each peer is represented by one of the three letters above to indicate the majority of documents one shares, and we call this the signature value.

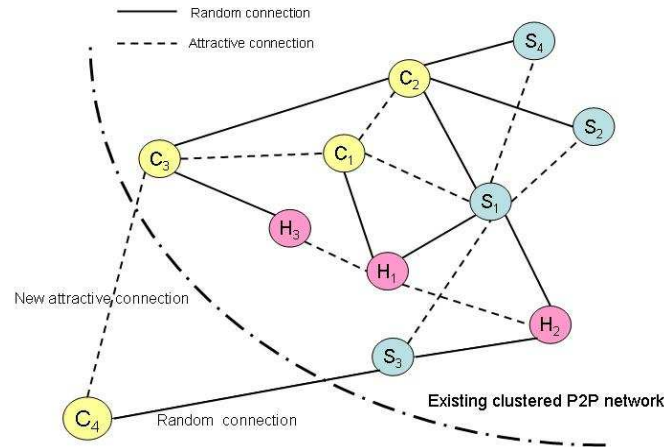


Figure 3.3: Basic Peer Clustering.

As shown in the Fig. 3.3, a peer named as C_1 means the majority of documents it shares is related to Computer. The existing clustered P2P network in the figure is formed by the joining sequence: $C_1, H_1, S_1, H_2, C_2, S_2, S_4, C_3, H_3, S_3$. When a new peer C_4 joins the network, by connecting a random connection to a randomly selected peer S_3 (chosen by the peer or assigned by

the bootstrap server¹), it sends out a signature query² to learn the location and signature value of other peers. The signature query is first sent to S_3 and then propagates to S_1 and H_2 in the first hop, then, to C_1 , C_2 , H_1 , S_2 and S_4 in the second hop and onwards. After collecting the signature replies from other peers, peer C_4 knows that C_1 , C_2 and C_3 share the same type of documents (Computer) with it, thus, peer C_4 makes an attractive connection to either one of them, in this example C_3 . As peers continue to join the network using this algorithm, a clustered P2P network is formed.

Referring to Fig. 3.3, all peers with the same signature value are interconnected by attractive connections to form several clusters. For example, C_1 , C_2 , C_3 and C_4 are interconnected by attractive connections to form the (C)omputer cluster. C_1 and S_1 are also interconnected by attractive connection because when S_1 joins the network, there are only 2 peers (C_1 and H_1) in the existing clustered network. It cannot find any peer share the same type of documents (Science) with it, thus peer S_1 makes an attractive connection to either one of them randomly. This problem is only happened when the network size is small. When the network continues to grow, the new peer has more chances to find peers which share the same type of documents with it. After a clustered P2P network is formed, a selective query routing scheme thus can be applied which makes information retrieval much more systematic and efficient.

Let assume the new user, peer C_4 , wants to find some (H)istory documents. It initiates a query and checks against its own signature value. It finds that the query and its signature value are mis-matched (not belonged to same category), thus the query is forwarded through random connection to S_3 . S_3 receives the query and performs the same checking. It finds that the query and its

¹Gnutella host cache server that provides a high-availability network bootstrap point for Gnutella servants

²Similar to that of ping-pong messages in Gnutella, to ask for signature value of peers within its neighborhood

signature value are still mis-matched, it continues to forward the query through random connection to H_2 . Once H_2 receives this query, it checks against its signature value and finds the query reached the target cluster (belonged to same category). Therefore, it broadcasts the query inside the cluster through attractive connection to H_1 . Likewise, when H_1 receives the query, it broadcasts the query to H_3 , and so on. Under this selective query routing scheme, we avoid the query to pass through some unrelated peers. The number of query messages used is reduced while query is still able to reach peers containing the answer.

The detailed description and analysis of this algorithm was proposed in [36], which shows promising result against the conventional Gnutella query mechanism. As this version of Peer Clustering requires users to assign signature value to a peer and compromises a set of categories in the distributed P2P environment, which is not practical enough in an open environment, we propose two enhanced versions based on this foundation to address these problems in the next two sections.

3.1.2 Peer Clustering - Single Cluster Version

Since clustering based on the category of shared documents stated by the user explicitly is not practical enough as aforementioned, we move on to clustering based on content feature of shared documents. In the information retrieval literature, text documents, images and multimedia data are processed to use vectors as their representation, while similarity between two documents are distance measure in the vector space. Clustering in the vector space had been vastly studied to improve retrieval performance by serving as an indexing structure in the centralized approach.

With the inherent nature of DISCOVIR network, we apply notation in graph theory to model it (see Table. 3.1). For the sake of generality, we try

to keep this in high level of abstraction. In this version of peer clustering algorithm, we choose the mean and standard deviation of cluster as signature value of peers, while Euclidean distance is used for similarity measure. In the actual realization, we choose image feature vector as the underlying data structure for representing images, which give birth to the name of our system DISCOVER (DIStributed COntent-based Visual Information Retrieval). Here are some definitions:

Table 3.1: Definition of Terms using in Peer Clustering

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

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

$$f : c_v \rightarrow \vec{c}_v, \quad (3.1)$$

$$f : q \rightarrow \vec{q}, \quad (3.2)$$

f is the mapping function from data c_v to a vector \vec{c}_v . In the notion of image processing, c_v is the raw image data, f is a specific feature extraction method (e.g. color histogram, co-occurrence matrix), \vec{c}_v is the extracted feature vector characterizing the image. Likewise, f is also used to map a query q to a query vector \vec{q} , to be sent out when user makes a query.

Definition 2 sig_v is the signature values used to represent the characteristic of data shared by peer v . We define

$$sig_v = (\vec{\mu}, \vec{\delta}), \quad (3.3)$$

where $\vec{\mu}$ and $\vec{\delta}$ are the statistical mean and standard deviation of the collection of shared data, in the corresponding vector space, belonging to peer v . From now on, sig_v characterizes the data shared by peer p .

Definition 3 $D(sig_v, sig_w)$ is defined as any distance measure between 2 signature values, sig_v and sig_w , in other sense, the similarity between two different peers v and w . One of the possible and simplest definition is,

$$D(sig_v, sig_w) = \|\vec{\mu}_v - \vec{\mu}_w\|, \quad (3.4)$$

where $\|\vec{\mu}_v - \vec{\mu}_w\|$ is the Euclidean distance between two centroids symbolized by sig_v, sig_w . Although this definition is simple and easy to calculate, it may not be accurate enough. A more complicated definition is,

$$D(sig_v, sig_w) = \|\vec{\mu}_v - \vec{\mu}_w\|_2 \times \left(\sum_{i=1}^d \delta_{pi} \times \delta_{qi} \right)^{1/2}. \quad (3.5)$$

This definition aims to connect peers with similar $\vec{\mu}$ and small $\vec{\delta}$. In (3.5), the more similar two peers p and q are, the smaller the value $D(sig_v, sig_w)$ is.

$D(sig_v, sig_w)$ 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 feature vectors in the two sub-cluster are closely clustered, that is, the shared data are highly related to the same area. We define the data similarity of two peers by this formula and use it to help organizing the network.

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

1. **Signature Value Calculation**—Every peer preprocesses its data collection and calculates signature values sig_v to characterize its data properties by assuming all feature vectors as a single cluster. Whenever the shared data collection, C_v , of a peer changes, the signature value is updated accordingly.
2. **Neighborhood Discovery**—After a peer joins the DISCOVIR network by connecting to a random peer in the network, it broadcasts a signature query message, similar to that of ping-pong messages in Gnutella, to ask for signature value of peers within its neighborhood, $sig_w, w \in Horizon(v, t)$, as shown in Fig. 3.4. This task is not only done when a peer first joins the network, it repeats every certain interval in order to maintain the latest information of other peers.
3. **Attractive Connection Establishment**—After acquiring the signature values of other peers, one can reveal the peer with signature value closest to its according to definition 3, and make an attractive connection to link them up, as shown in Fig. 3.5. This attractive connection is reestablished to the second closest one in the host cache whenever current connection breaks.

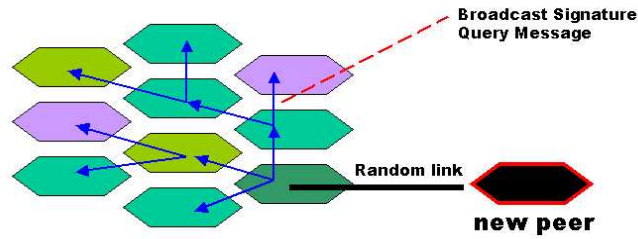


Figure 3.4: Illustration of Peer Clustering Step 2

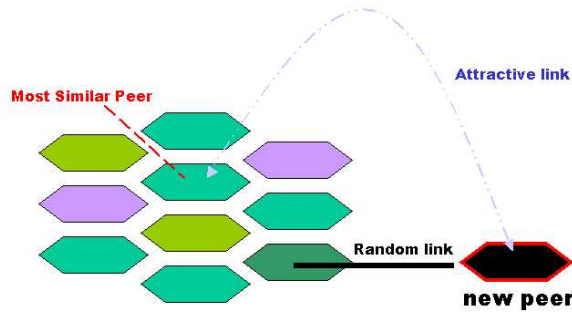


Figure 3.5: Illustration of Peer Clustering Step 3

Fig 3.6 illustrate the revised peer clustering model. The signature value is the mean and the standard deviation of all feature vectors of documents shared by a peer, while the attractive connection is established based on distance of Equation 3.4.

For example, when a new peer C_4 joins the network, by connecting to a randomly selected peer S_3 , it sends out a signature query to learn the location and signature value of other peers. The signature query is first sent to S_3 and then propagates to S_1 and H_2 in the first hop, then, to C_1 , C_2 , H_1 , S_2 and S_4 in the second hop and onwards. After collecting the replies from other peers, peer C_4 knows that C_3 is the most similar peer (smallest value in distance measure with other peers) as shown in Table 3.2. Thus, peer C_4 makes an attractive connection C_3 . Having all peers joining the DISCOVER network perform the three tasks described above, you can envision a P2P network with

Table 3.2: Distance measure between peers in Fig. 3.6

<i>Sim</i>	C_1	H_1	S_1	H_2	C_2	S_2	S_3	C_3	H_3	S_4	C_4
C_1	0.0	1.1	2.6	1.0	0.2	2.1	2.4	0.1	1.3	3.0	0.3
H_1		0.0	3.7	0.1	0.9	3.2	3.5	1.2	0.2	4.1	1.4
S_1			0.0	3.6	2.8	0.5	0.2	2.5	3.9	0.4	2.3
H_2				0.0	0.8	3.1	3.4	1.1	0.3	4.0	1.3
C_2					0.0	2.3	2.6	0.3	1.1	3.2	0.5
S_2						0.0	0.3	2.0	3.4	0.9	1.8
S_3							0.0	2.3	3.7	0.6	1.1
C_3								0.0	1.4	2.9	0.2
H_3									0.0	4.3	1.6
S_4										0.0	2.7
C_4											0.0

self-organizing ability to be constructed. Peers sharing similar content will be grouped together like a Yellow Pages. The detail steps of peer cluster is illustrated in Algorithm 1.

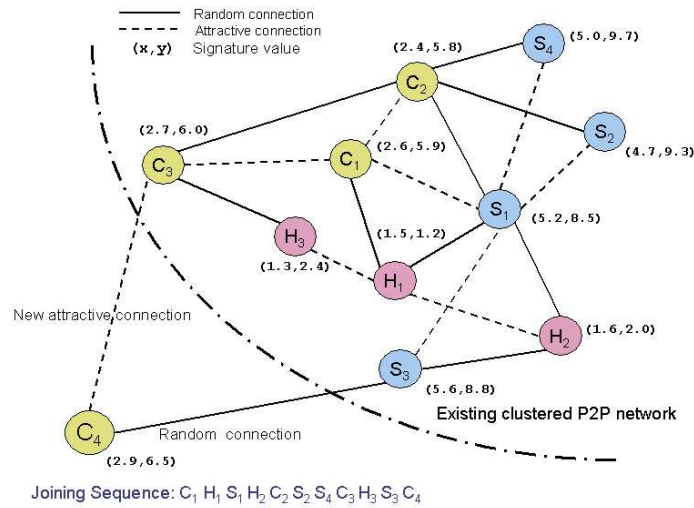


Figure 3.6: Peer Clustering - Single Cluster Version

Algorithm 1 Algorithm for peer clustering

```

Peer-Clustering(peer v, integer ttl)
for all  $w \in Horizon(v, t)$  do
  Compute  $D(sig_v, sig_w)$ 
end for
 $E_a = E_a \cup (v, w, sig_v, sig_w)$  having  $min(D(sig_v, sig_w))$ 

```

3.1.3 Peer Clustering - Single Cluster, Multiple Layers of Connection Version

The previous version of peer clustering just uses one feature extraction function (one layer of attractive links) to map the data to feature vectors. However, users may have different interests in information retrieval. Using image processing as the example, some user may be interested to find images based on color, some user may be interested based on shape or texture. Using a single feature extraction function to represent the data is not enough.

We propose to use multiple layers of attractive links to cluster peers using different extraction functions. As shown in Fig. 3.7, two layers of attractive links are make. The first layer of attractive links are make according to the signature value, sig_1 , which is extracted by color extraction function f_1 . The second layer of attractive links are make according to the signature value, sig_2 , which is extracted by color extraction function f_2 . If user wants to find images based on color, the query will be forwarded through the first layer of attractive links. If user wants to find images based on shape, the query will be forwarded through the second layer of connections. A set of signature values are kept in each peer. More details of multiple feature extraction modules will be discussed in Chapter 4.

3.1.4 Peer Clustering - Multiple Clusters Version

The previous version of peer clustering assumes most of the data shared by a peer fall in the same category, e. g. , a peer shares collection of sunset images

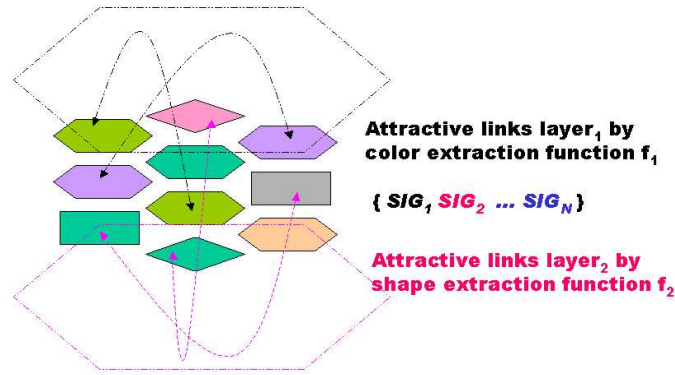


Figure 3.7: Peer Clustering - Single Cluster, Multiple Layers of Connection Version

or collection of computer science paper, thus the extracted feature vectors will clustered together and signature value is able to describe the data characteristic reasonably. However, most users share documents of various topics in real-world situation, as shown in Fig. 3.8, there should be an attractive connection between sub-cluster A_2 of peer A and sub-cluster B_3 of peer B although their cluster centroid are far apart.

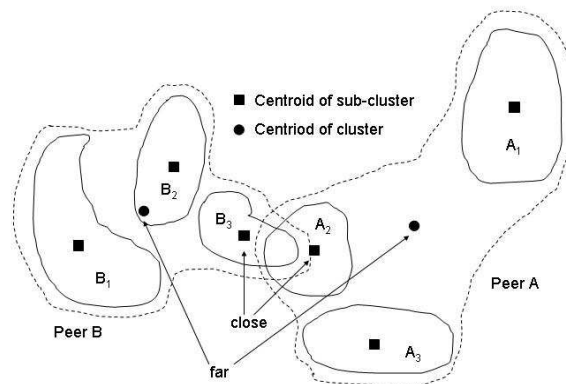


Figure 3.8: Multiple clusters in a peer.

We propose to use multiple signature values $sig_v, sig_v \in SIG_v$ to represent a peer because the document collection is likely to fall in several categories and their extracted feature vectors form several clusters as well. With these

changes, peers may form several attractive connections depending on the number of local sub-clusters. The revised algorithm is illustrated in Algorithm 2 and two changes in the main steps are updated as follows:

1. **Signature Value Acquisition**—In the preprocessing stage, a set of signature values SIG_v is calculated, which are the statistical mean and standard deviation of sub-clusters found using some clustering algorithm like k -means [1], competitive learning [46], and expectation maximization [23]. The number of signature values is variable and is a trade-off between resolution of cluster and computational cost. In the DISCOVIR, we choose competitive learning and 3 as the number of signature values in a peer for the sake of low computation cost.
2. **Attractive Connection Establishment**—This process is the same as that in previous section except we make attractive connection for every signature values a peer possess, as shown in Fig. 3.9. The standard deviation of a signature value is reserved to control the quality of attractive connections, the smaller the standard deviation, the better to make attractive connection because it implies a dense cluster.

We have introduced the final and most practical version of peer clustering algorithm in DISCOVIR, having all peers follow this procedure to build the network, a self-organizing network is formed, we delineate a selective query routing strategy in the next section to improve retrieval performance.

3.2 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

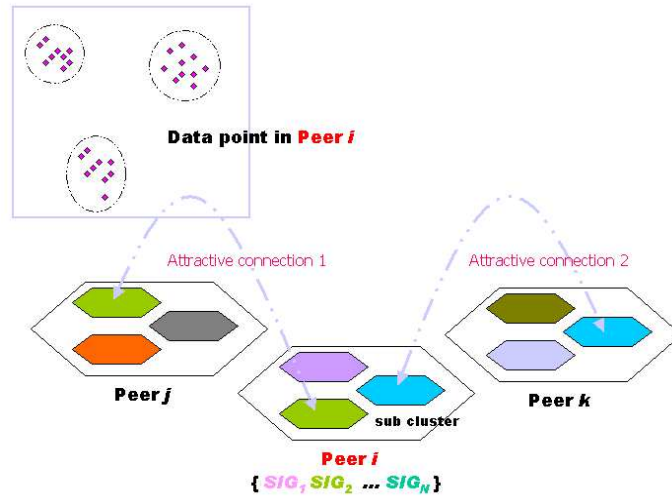


Figure 3.9: Illustration of Peer Clustering - Multiple Clusters Version, Step 2 and 3

Algorithm 2 Algorithm for peer clustering - multiple clusters

```

Peer-Clustering(peer v, integer ttl)
  for all  $sig_v \in SIG_v$  do
    for all  $w \in Horizon(v, t)$  do
      for all  $sig_w \in SIG_w$  do
        Compute  $D(sig_v, sig_w)$ 
      end for
    end for
     $E_a = E_a \cup (v, w, sig_v, sig_w)$  having  $min(D(sig_v, sig_w))$ 
  end for

```

is routed selectively according to the content of the query. The query message first walks around the network through random connections. Once it reaches its designated cluster, the query message is broadcasted through the attractive connections inside the cluster much like an exploding firework as shown in Fig. 3.10. Our strategy aims to:

1. minimize the number of messages passing through the network,
2. avoid irrelevant computers to handle the query and reduce the workload of each computer,
3. maximize the ability of retrieving relevant data from the peer-to-peer network.

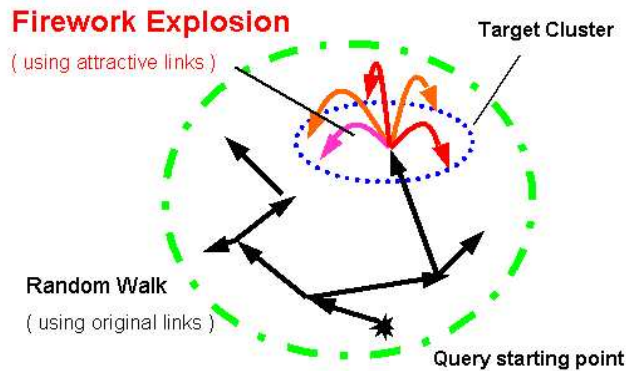


Figure 3.10: Illustration of Firework query.

Here, we introduce the algorithm to determine when and how a query message is propagated like a firework in Algorithm 3. When a peer receives the query, it needs to carry out two steps:

1. **Shared File Look Up**—The peer looks up its shared information for those matched with the query. Let q be the query, and \vec{q} be its vector representation, $REL(c_v, q)$ is the relevance measure between the query

and the information c_v shared by peer v , it depends on a L_2 norm defined as,

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

where T is a threshold defining the degree of result similarity a user wants. If any shared information matches the criteria of query³, the peer will reply the requester. In addition, we can reduce the number of $REL(c_v, q)$ computations inside the peer by building an index of shared data using local clustering algorithm, thus speeding up the process of query response.

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

$$D_q(sig_v, q) = \prod_i \frac{1}{\sqrt{2\pi}\delta_i} e^{-\frac{(x_i - \mu_i)^2}{2\delta_i^2}}. \quad (3.6)$$

If none of the distance measure between its local clusters' signature value and the query, $D_q(sig_v, q)$, is larger than a preset threshold, θ , the peer will propagate the query to its neighbors through random connections. Otherwise, if one or more $D_q(sig_v, q)$ is larger the threshold, it implies the query has reached its target cluster. Therefore, the query will be propagated through corresponding attractive connections much like an exploding firework. Figure 3.11 shows an example of route selection using in Firework Query Model. The query message is first propagated through random connections. Once it reaches its target cluster(peer of purple polygon), the message is broadcasted through attractive connections.

In our Firework Query Model, we retain two existing mechanisms in Gnutella network for preventing query messages from looping forever in the distributed

³The threshold value set by user to determine the degree of result which a user wants. If the threshold is set to a large value, less results but more relevant results are expected to be received.

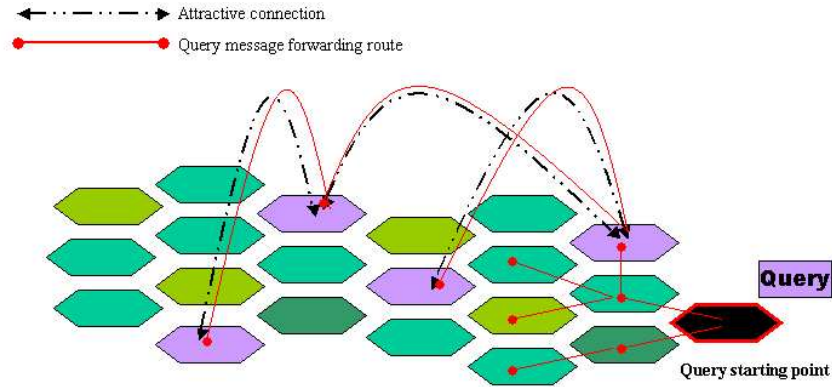


Figure 3.11: Illustration of Firework Query Model, Step 2

network:

1. Gnutella replicated message checking rule,
2. Time-To-Live (TTL) value 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 be dropped and 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 when the message passes through a peer, the TTL value is decreased by one. Once the TTL value reaches zero, the message will be dropped and no longer forwarded.

There is a modification on DISCOVER query messages from the original Gnutella messages. In our model, the TTL value is decremented by one with a different probability when the message is forwarded through different types of connection. For random connections, the probability of decreasing TTL value is 1. For attractive connections, the probability of decreasing TTL value is an arbitrary value in $[0, 1]$ called Chance-To-Survive (CTS). This strategy can reduce the number of messages passing outside the target cluster, while

more relevant information can be retrieved inside the target cluster because the query message has a greater chance to survive inside the cluster depending on the CTS value.

Algorithm 3 Algorithm for the Firework Query Model

```

Firework-query-routing (peer  $v$ , query  $q$ )
for all  $sig_v \in SIG_v$  do
  if  $D_q(sig_v, q) > \theta$  (threshold) then
    if  $rand() > CTS$  then
       $q_{ttl} = q_{ttl} - 1$ 
    end if
    if  $q_{ttl} > 0$  then
      propagate  $q$  to all  $e_a(a, b, c, d)$  where  $a = v, c = sig_v$  or  $b = v, d = sig_v$ 
      (attractive link)
    end if
  end if
end for
if Not forwarding to attractive link then
   $q_{ttl} = q_{ttl} - 1$ 
  if  $q_{TTL} > 0$  then
    forward  $q$  to all  $e_r(a, b)$  where  $a = v$  or  $b = v$  (random link)
  end if
end if

```

Chapter 4

DIStributed COntent-based Visual Information Retrieval (DISCOVER)

DIStributed COntent-based Visual Information Retrieval (DISCOVER) is a software package implemented by us which enables individuals to search for and share multimedia files based on their contents with anyone on the Internet. A product of DISCOVER is built on top of the Limewire open source project which is compatible with the current Gnutella file-sharing protocol and can connect with anyone else running Gnutella-compatible software. DISCOVER is written in Java, and can run on Windows, Macintosh, Linux, Sun, and other computing platforms.

The motivation of building DISCOVER is to migrate traditional Content Based Image Retrieval (CBIR) to a P2P network as a step to introduce content-based search in P2P. With the advantages of P2P network, we utilize not only the distributed data storage, but also the computation power of each peer for the preprocessing and indexing of multimedia data. Queries in DISCOVER are no longer based on simple texts but on the content of multimedia data. The need for annotating shared files is waived, thus, query accuracy does not depend on subjective perception of keywords. Peer Clustering and Firework

Query Model are also implemented on DISCOVER to demonstrate how our algorithm works on the P2P network. We reduce the network traffic generated, avoid irrelevant peers to handle the query to reduce the workload of computers and increase the information retrieval performance as well. Figure 4.1 shows a screen capture of the DISCOVER client program, which can be downloaded from <http://www.cse.cuhk.edu.hk/~miplab/discovir>.

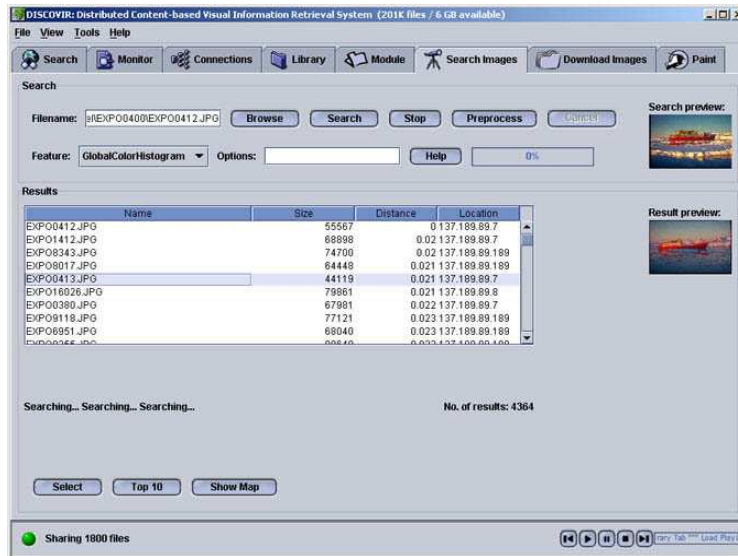


Figure 4.1: Screen Capture of DISCOVER

4.1 Architecture of DISCOVER and Functionality of DISCOVER Components

In this section, we will describe the architecture of a DISCOVER client, functionality of DISCOVER Components and the communication protocol in order to perform clustering, content-based query routing and CBIR over the P2P network. Through the DISCOVER program, users can share images among peers around the world. Each peer is responsible for extracting and indexing the feature of the shared images, by doing so, every peers can search for similar

images based on image content, like color, texture and shape among images shared by other DISCOVER peers in the network.

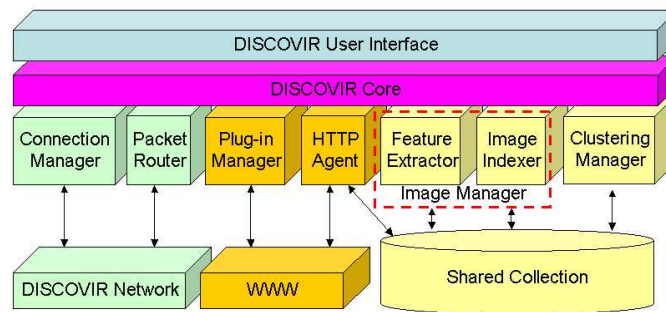


Figure 4.2: Architecture of DISCOVER

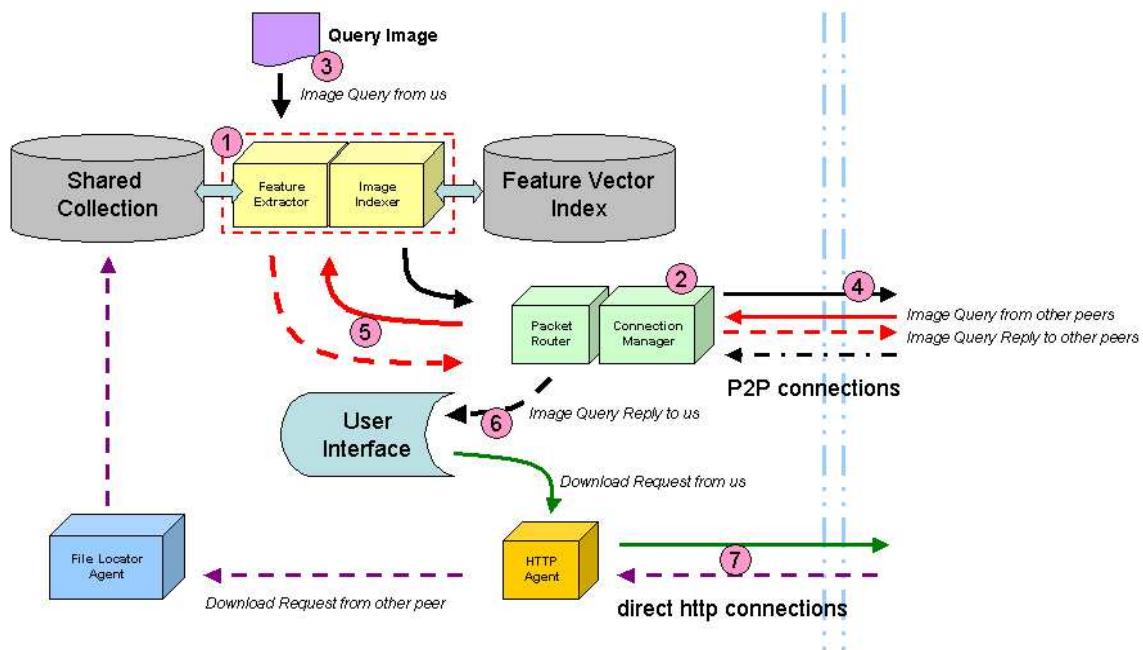


Figure 4.3: Interaction between DISCOVER components

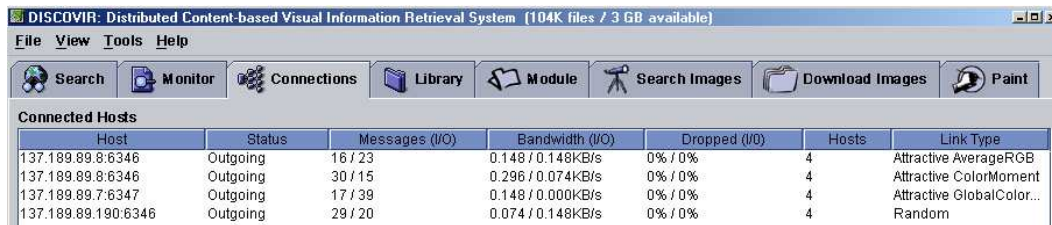
Figure 4.2 and 4.3 depicts the key components and their interaction in the architecture of a DISCOVER client. As DISCOVER is derived from LimeWire [28] open source project, the operations of Connection Manager, Packet Router and HTTP Agent remain more or less the same with additional functionality to improve the query mechanism used in original Gnutella network. Connection

manager is modified to be able to handle different types of connections (Random connections and Attractive connections). Packet Router is modified to be able to co-operate with other modules to perform content-based query routing. Plug-in Manager, Feature Extractor and Image Indexer are introduced to support the CBIR tasks. Clustering Manager is introduced to support peer clustering and shared data clustering tasks. The User Interface is modified to incorporate the image search panel.

In the following, we describe the functionality of DISCOVER Components:

- **Connection Manager** - It is responsible for setting up and managing the TCP connection between DISCOVER clients. In the design of DISCOVER, there are 2 types on connection:
 1. **Random connections** - Connected at startup or chosen by user
 2. **Attractive connections** - Automatically chosen and connected by our algorithm, used to enhance the searching performance

By default, the Connection Manager will connect to the bootstrap server automatically. Figure 4.4 shows the screen capture of connection manager.



The screenshot shows the DISCOVER Connection Manager window. The title bar reads "DISCOVER: Distributed Content-based Visual Information Retrieval System (104K files / 3 GB available)". The menu bar includes "File", "View", "Tools", and "Help". The toolbar contains icons for "Search", "Monitor", "Connections", "Library", "Module", "Search Images", "Download Images", and "Paint". Below the toolbar is a table titled "Connected Hosts" with the following data:

Host	Status	Messages (I/O)	Bandwidth (I/O)	Dropped (I/O)	Hosts	Link Type
137.189.89.8:6346	Outgoing	16 / 23	0.148 / 0.148KB/s	0% / 0%	4	Attractive AverageRGB
137.189.89.8:6346	Outgoing	30 / 15	0.296 / 0.074KB/s	0% / 0%	4	Attractive ColorMoment
137.189.89.7:6347	Outgoing	17 / 39	0.148 / 0.000KB/s	0% / 0%	4	Attractive GlobalColor...
137.189.89.190:6346	Outgoing	29 / 20	0.074 / 0.148KB/s	0% / 0%	4	Random

Figure 4.4: Screen Capture of Connection Manager

- **Packet Router** - It controls the routing of message in DISCOVER network between components and peers. It is modified to be able to co-operate with Connection Manager, Clustering Manager and Plug-in Manager to perform content-based query routing (Firework Query

Model).

- **HTTP Agent** - It is a tiny web-server that handles file download request from other DISCOVER peers using HTTP protocol.
- **Feature Extractor** - It collaborates with the Plug-in Manager to perform various feature extraction and thumbnail generation of the shared image collection.
- **Image Indexer** - It indexes the image collection by content feature and carry out clustering to speed up the retrieval of images.

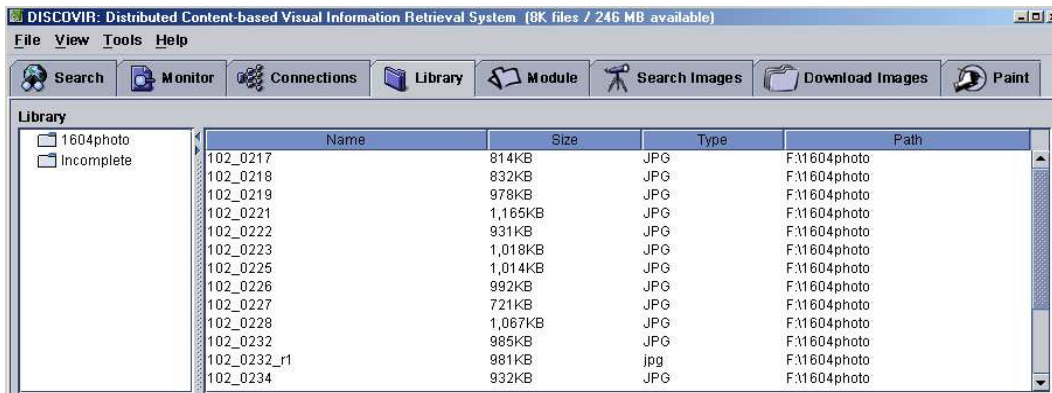


Figure 4.5: Screen Capture of Image Indexer

- **Clustering Manager** - It clusters the extracted feature vectors from Image Indexer to form local sub-clusters used for Peer Clustering algorithm. (used in Peer Clustering - Multiple Clusters Version)
- **Plug-in Manager** - It coordinates the storage of different feature extraction modules and their interaction with Feature Extractor and Image Indexer. DISCOVER uses a plug-in architecture to support different feature extraction method, as shown in Figure 4.6. User may select 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.

It also helps the Connection Manager to monitor different feature extraction modules used in formation of different layers of attractive connection. (used in Peer Clustering - Single Cluster, Multiple Layers of Connection Version).

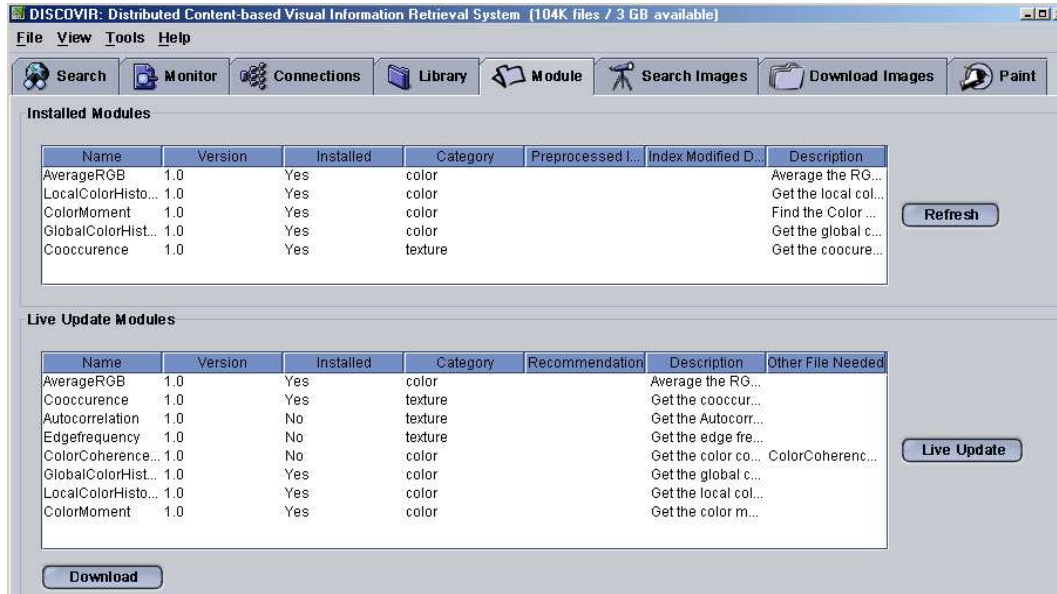


Figure 4.6: Screen Capture of Plug-in Manager

4.2 Flow of Operations

The following is a scenario walk-through to demonstrate the interaction between components in Fig.4.3.

When a user wants to share his data in DISCOVER network, some pre-processing works are needed to be done. Unlike other text-based P2P sharing applications, content-based P2P sharing application requires the program to extract the feature vectors of shared data first because feature extraction is a time consuming procedure. Real time processing will seriously downgrade the retrieval performance. The *Feature Extractor* collaborates with *Plug-in Manager* to extract content features and generate thumbnail from data in the *Shared Collection*. The extracted features are then passed to *Image Indexer*

and *Clustering Manager* for indexing and clustering purposes, to build the *Feature Vector Index* (1).

In case when a new DISCOVIR client wants to join the P2P network, the *Connection Manager* asks the dedicated *Bootstrap Server* for a list of currently available DISCOVIR clients in order to hook up to the network. *Bootstrap Server* is a host cache server that provides a high-availability network bootstrap point for DISCOVIR clients. Then, the *Connection Manager* will broadcast a signature query to learn the location and signature value of other peers. After collecting the signature replies, the *Connection Manager* will make and handle the attraction connections according to the Peer Clustering algorithm (2).

Once a user initiates a content-based query, the *Feature Extractor* extracts the feature vector from the provided sample image or drawn pictures instantly (3), *Packet Router* is responsible for assembling an ImageQuery message and sending out to the DISCOVIR network (4). For instance, when an ImageQuery message is received from other peers, the *Packet Router* checks for any duplication and propagates to other peers through DISCOVIR network. Meanwhile, it passes the message to *Image Indexer* for searching similar images (5). Upon similar images are found, an ImageQueryHit message is assembled and passed to *Packet Router* for replying the initiating peer. When ImageQueryHit messages return to the initiating peer (6), its *HTTP Agent* downloads thumbnail or full size images from other peers upon receiving user request from the user interface (7). We will talk about each detailed steps in the next section.

4.2.1 Preprocessing (1)

Plug-in Manager is responsible for contacting web-site of DISCOVIR to inquire the list of available feature extraction modules. It will download and install selected modules upon user's request. Currently, DISCOVIR supports

various feature extraction methods in color and texture categories such as AverageRGB, GlobalColorHistogram, ColorMoment, Co-occurrence matrix, etc [17]. All feature extraction modules strictly follow a predefined interface in order to realize the polymorphic properties of switching between different plugins dynamically, see Fig. 4.6.



Figure 4.7: Screen Capture of Preprocess Procedure

Feature Extractor will extract feature and generate thumbnail for all images in the shared collection by using a particular feature extraction module when the button is pressed, as shown in Fig. 4.7. A progress bar is used to show the percentage of files which has been preprocessed. Let I represent a raw image data, f be the feature extraction method, the *Feature Extractor* performs the task illustrated in Eq. 4.1,

$$f : I \times \theta \rightarrow \vec{I} \quad (4.1)$$

where θ is the feature extraction parameter and \vec{I} is the extracted feature vector. *Image Indexer* will then index the image collection using the multi-dimensional feature vectors \vec{I} in order to answer an incoming query. *Feature Extractor* will keep checking on the time stamp of shared files and file-size. If new files are found or the old files are modified, the processing procedure will only re-do on the new files and modified files. Then, *Clustering Manager* clusters the set of feature vector for the sake of improving query efficiency by acquiring statistical distribution information of the local image collection.

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

4.2.2 Connection Establishment (2)

For a peer to join the DISCOVIR network, it connects to the bootstrap server using the *Connection Manager*. The bootstrap server is responsible for storing a finite list of IP address of peers currently in the DISCOVIR network and randomly picks an IP address to return to the peer. Once the IP address is received, the peer is able to hook up to the DISCOVIR network by connecting to the selected peer. Then, the *Connection Manager* broadcasts a signature query to the P2P network to learn the location and signature value of other peers. It stores and sorts the signature replies in a signature cache. After a certain period (10 second is set in DISCOVIR), the *Connection Manager* picks the most similar peer and make an attractive connection to it. This signature broadcasting task is not only done when a peer first joins the network, it repeats every certain interval (10 minutes is set in DISCOVIR) in order to maintain the latest information in signature cache. If the current attractive connection is broken, the *Connection Manager* will try re-connect or pick the second most similar peer and make an new attractive connection to it.

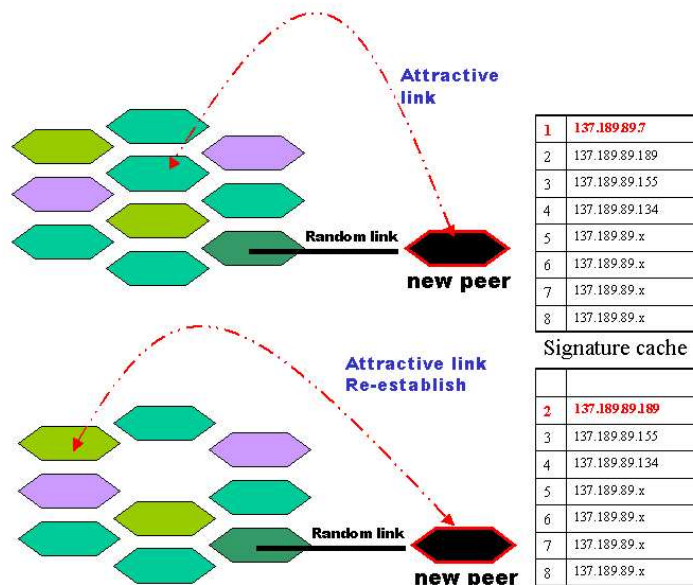


Figure 4.8: Illustration of Attractive Connection Re-Establishment

4.2.3 Query Message Routing (3,4,5)

After a peer joins the DISCOVER network, it may initiate a content-based query by providing a sample image or drawn pictures. The *Feature Extractor* extracts the feature vector from the provided sample image or the drawn pictures from the DISCOVER draw-pad (see Figure 4.10). The *Packet Router* is responsible for assembling an ImageQuery message, checking its signature values and determine how to send the query out to the DISCOVER network (by attractive connections or random connections). For instance, when an ImageQuery message is received from other peers, they need to perform two operations, *Local Index Look Up* and *Query Message Propagation*.

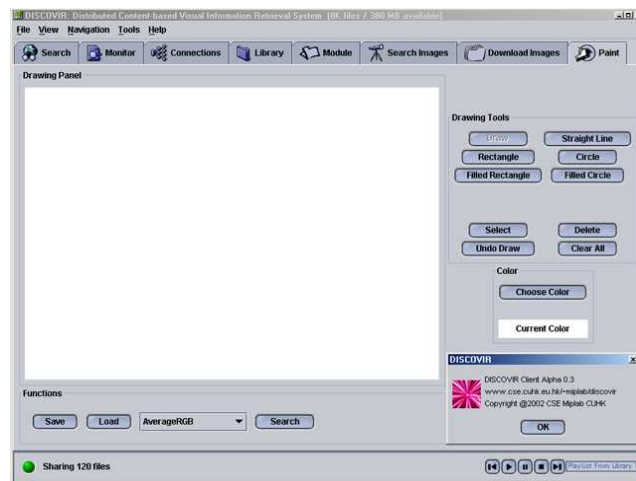


Figure 4.9: Screen Capture of DISCOVER draw-pad

- **Local Index Look Up** - The *Image Indexer* looks up local index of shared files for similar images. Once similar images are found, the *Image Indexer* assembles and delivers an ImageQueryHit message back to the requester through *Packet Router*. Since images indexing and clustering are performed on the peer in preprocessing stage, the retrieval time can be speeded up.
- **Query Message Propagation** - In order to prevent query messages

from looping forever in the DISCOVER network, two mechanisms are inherited from Gnutella, namely, the Gnutella replicated message checking rule and Time-To-Live (TTL) value of messages. After these two checkings, the *Packet Router* co-operates with *Clustering Manager* to calculate the distance between the query and each signature value of its local clusters. If none of the distance measure between its local clusters' signature value and the query is larger than the preset threshold, the *Packet Router* will propagate the query to its neighbors through random connections. Otherwise, the query will be propagated to corresponding attractive connections through *Connection Manager*.

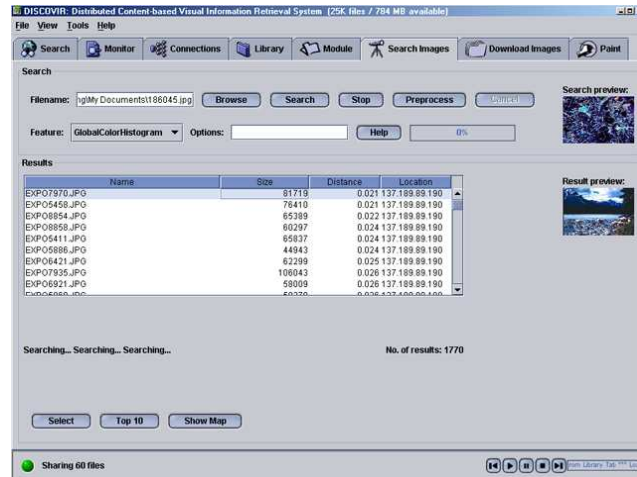


Figure 4.10: Screen Capture of DISCOVER Image Query

4.2.4 Query Result Display (6,7)

When an ImageQueryHit message returns to the requester, user will obtain a list detailing the location and size of matched images. In order to retrieve the query result, the *HTTP Agent* will download thumbnail or full size image from the peer using HTTP protocol. On the other hand, *HTTP Agent* in other peers will serve as a web server to deliver the requested images. This *HTTP*

Agent is essential for integration to WWW which will be described later in detail.

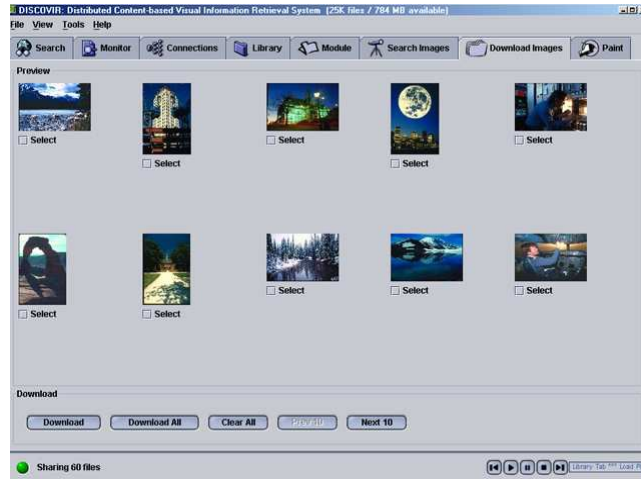


Figure 4.11: Screen Capture of DISCOVER Query Result Display

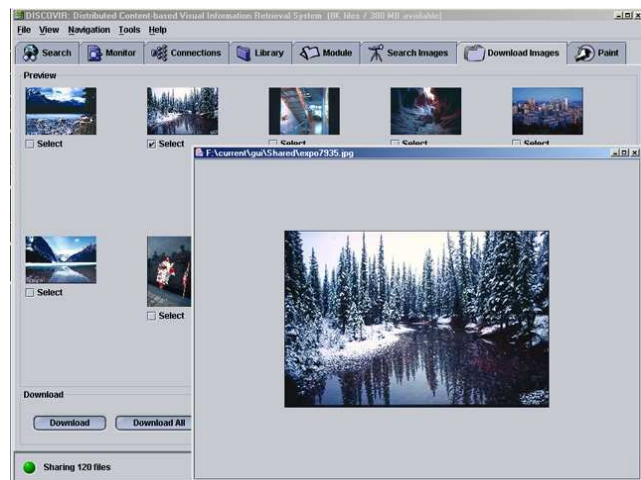


Figure 4.12: Screen Capture of DISCOVER Query Result Display 2

4.3 Gnutella Message Modification

The DISCOVER system is compatible to the Gnutella (v0.4) protocol [15]. In order to support the image query functionalities mentioned, two types of messages are added. They are:

- **ImageQuery** - Special type of the Query message. It is to carry name of feature extraction method and feature vector of query image, see Fig. 4.13.
- **ImageQueryHit** - Special type of the QueryHit message. It is to respond to the ImageQuery message, it contains the location, filename and size of similar images retrieved, and their similarity measure to the query. Besides, the location information of corresponding thumbnail are added for the purpose of previewing result set in a faster speed, see Fig. 4.14.

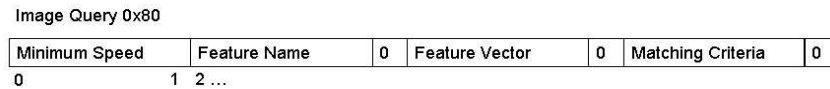


Figure 4.13: ImageQuery message format

Table 4.1: ImageQuery Payload

Minimum Speed	The minimum speed (in kb/second) of servants that should respond to this message. A servant receiving a Query descriptor with a Minimum Speed field of n kb/s should only respond with a QueryHit if it is able to communicate at a speed larger than n kb/s
Feature Name	The feature extraction module used to extract the query
Feature Vector	The extracted vector used to represent the query
Matching Criteria	The threshold value set by user to determine the degree of result which a user wants. If the threshold is set to a large value, less results but more relevant results are expected to be received.

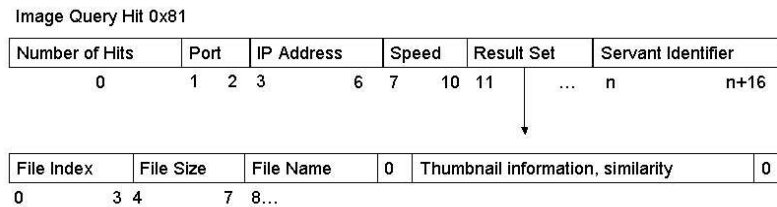


Figure 4.14: ImageQueryHit message format

Table 4.2: ImageQueryHit Payload

Number of Hits	The number of query hits in the result set
Port	The port number on which the responding host can accept incoming connections
IP Address	The IP address of the responding host
Speed	The speed (in kb/second) of the responding host
Result Set	A set of responses to the corresponding Query
Servant Identifier	A 16-byte string uniquely identifying the responding servant on the network. This is typically some function of the servants network address. The Servant Identifier is instrumental in the operation of the Push Descriptor
File Index	A number, assigned by the responding host, which is used to uniquely identify the file matching the corresponding query.
File Size	The size (in bytes) of the file whose index is File Index
File Name	The double-nul (i.e. 0x0000) terminated name of the file whose index is File Index
Thumbnail Information	The thumbnail of the file
Similarity	The distance measure between the query and the file

4.4 DISCOVER EVERYWHERE

Although migrating CBIR on P2P network has many advantages as aforementioned, this system still encounters limitations like the requirement of installing client software and the low accessibility compared to WWW. These limitations always introduce inconveniences for the users when they are not using their own computers. For example, when the users are in libraries or cyber-cafes, installing personal software is usually prohibited, therefore, they cannot use the services. For this reason, some web-based P2P applications have been developed. With these services, user can access the P2P network through a web browser while the web server serves as one of the peer in the P2P network. Some examples of web-based P2P applications are **AsiaYeah** [2], **Gnutellait** [16], **LinkGrinder** [29] and **AudioFind** [3].

When users submit queries through the web page, the server helps distributing the query and collecting the results from the P2P network. Such

kind of search engine for file retrieval through P2P network provides an alternative source of files in addition to the documents on WWW. This provides a more comprehensive and larger file database when the number of users in the P2P network is large enough. However, there are drawbacks concerning these models:

1. **Centralization** - The web server is public to everyone who is able to access the web page. When many users use this service, the server has to handle huge amount of queries and collection of results. The problem remains the same as prevalent search engine. Moreover, the web server will generate lots of traffic to its neighboring peers, which skews the workload in P2P network.
2. **CBIR functionality** - All the web-based P2P applications mentioned above are based on text search. When adapting to CBIR approach, it will incur lots of penalty when feature extraction of images is done by the server.

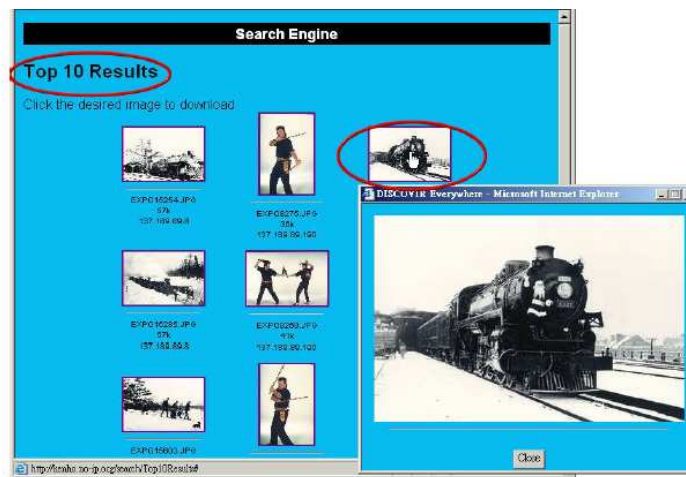


Figure 4.15: Screen Capture of DISCOVER Everywhere

4.4.1 Design Goal of DISCOVER Everywhere

To account for the two problems raised above, *DISCOVER Everywhere* is designed and implemented. *DISCOVER Everywhere* is a group project among several people. Part of the system is designed by me and the system is mainly implemented by a FYP group. *DISCOVER Everywhere* aims to overcome the two problems by distributing the heavy workload to peers evenly while keeping its accessibility through web. Unlike other web-based P2P applications, the web server does not act like a peer in the P2P network. Instead, it acts as the match-maker to coordinate the forwarding of queries and returning of result between web clients and peers. Preprocessing of query image, initiation of query and collection of result are all done in a DISCOVER peer assigned by the web server. Even there are huge number of users, this architecture is scalable because the web server is only responsible for distributing workload to DISCOVER client. It allows users to perform CBIR in P2P network through a web browser or other mobile devices, like J2ME phone.

4.4.2 Architecture and System Components of DISCOVER Everywhere

Referring to Fig. 4.16, we identify the four main components in the *DISCOVER Everywhere* design. They are:

- **Web Browsers and Mobile Devices** - It is a device running a web browser with network access to the WWW. The mobile devices can be J2ME phone or PDAs with wireless access to the *DISCOVER Everywhere* Gateway, they can access the web page either by WML or XML.
- **DISCOVER Peers** - They are interconnected computers running the

DISCOVER client program in the Internet. In addition to P2P query service, the HTTP Agent of each peer will accept GET and POST HTTP requests to provide seamless integration with the WWW. Moreover, every DISCOVER peer is required to send 'heart-beat' message to the Bootstrap Server periodically to indicate their availability in DISCOVER network.

- **DISCOVER Everywhere Bootstrap Server** - It is originally the bootstrap server of DISCOVER. In *DISCOVER Everywhere*, the bootstrap server is responsible for maintaining an updated list of accessible DISCOVER peers and their availability for providing HTTP access, if it cannot receive the 'heart-beat' message from a peer for a certain period of time, its record will be removed and considered off-line.
- **DISCOVER Everywhere Gateway** - It is a server program providing users with web-based searching interface. It contacts the Bootstrap Server for the list of IP address of available DISCOVER Peers and coordinates the redirection of users' query request to different peers.

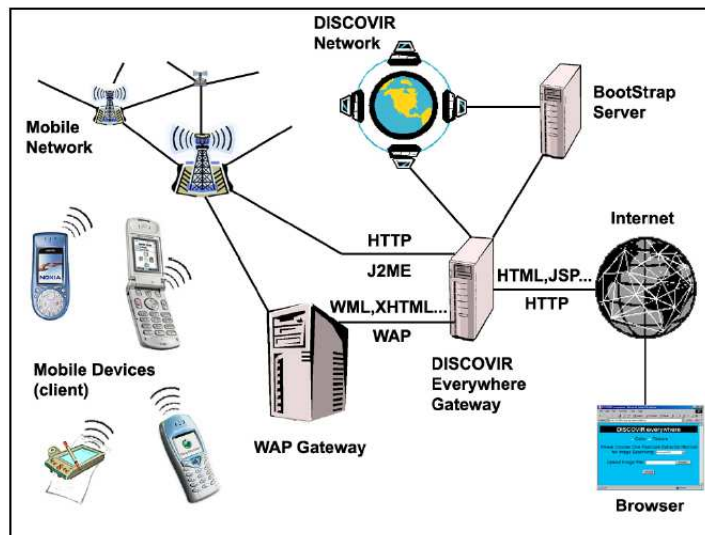


Figure 4.16: Architecture of DISCOVER Everywhere

4.4.3 Flow of Operations

Referring to Fig. 4.17, the process of query in *DISCOVIR Everywhere* consists of six steps. Details of each step are shown in the following:

1. A user initiates a query request through the web interface provided by *DISCOVIR Everywhere* Gateway. If the user access the web interface by a mobile phone, he should access it through the mobile Gateway and the medium of communication is WML instead of HTML.
2. The *DISCOVIR Everywhere* Gateway receives the request and inquire the bootstrap server about the IP address and port number of a available DISCOVIR peer capable of handling this query.
3. Upon receiving the inquiry from *DISCOVIR Everywhere* Gateway, the DISCOVIR Bootstrap Server picks one of the available peers from the list in a round robin manner in order to distribute the workload evenly. This is similar to the techniques used by DNS servers to distribute workload among web servers.
4. Once knowing the IP address of DISCOVIR peer capable of handling the query, the gateway generates a HTML page instantly for users to upload his query image and intended feature extraction method to the selected peer using HTML form submission procedure.
5. User uploads the query image to selected DISCOVIR peer through a HTTP POST request. Meanwhile, the *HTTP Agent* of that selected peer stores the image and requests its *Feature Extractor* to extract feature and assemble an ImageQuery message to be sent out through *Packet Router*, which is analogous to the processing of initiating query using the DISCOVIR client program. The web browser keeps this HTTP connection open until results return from the DISCOVIR peer.

6. Once the selected DISCOVER peer accumulates up to a certain number of results or reaches a time limit, it packages the result in HTML format and sends back to user through *HTTP Agent*. With the inherent support of HTTP defined in Gnutella protocol, web users are able to download thumbnail or full size images directly from DISCOVER peers without the help of gateway.

4.4.4 Advantages of DISCOVER Everywhere over Prevalent Web-based Search Engine

Compared to existing search engines and web-based P2P services, *DISCOVER Everywhere* exhibits the following advantages:

1. **Comprehensiveness** - By utilizing the storage capacity and individual contribution of peers in the network, we increase the comprehensiveness of data archive for searching. Besides, the web-based interface provide a handy access compared to using pre-installed P2P client programs.
2. **Query Richness** - *DISCOVER Everywhere* possess CBIR functionality beyond existing text based retrieval, while eliminates the need for preprocessing, storage and indexing in existing CBIR search engines by delegating them to peers in DISCOVER network.
3. **Scalability** - Compared to existing web-based P2P service, the *DISCOVER Everywhere Gateway* is much more light weighted. Instead of serving as a centralize server for web users to access the P2P network, it takes the role of coordinator between web users and DISCOVER peers. Apart from reducing the workload for initiation of query and collection of results, this also avoids perverted usage of P2P network by distributing query requests among peers evenly.

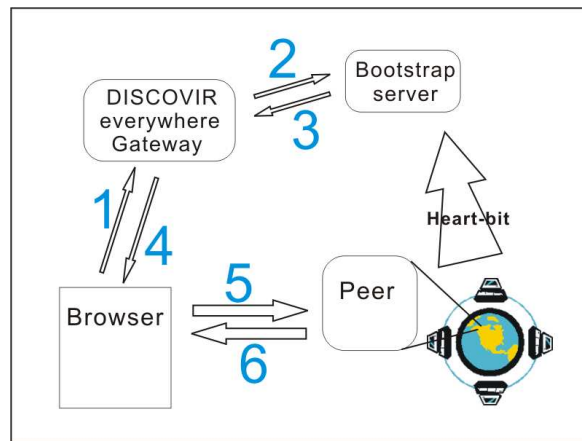


Figure 4.17: Query Procedure of DISCOVIR Everywhere

Detailed information of DISCOVIR Everywhere can be found in the homepage of DISCOVIR Everywhere [10] and our technical report written in December of 2002 [49].

Chapter 5

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 our model of Peer-to-Peer network used in our simulation. We then introduce the performance metrics used to evaluate different search mechanisms. We simulate the experiments to study the performance of different mechanisms using different parameters. We show how our strategy performs and behaves at scale.

5.1 Simulation Model of Peer-to-Peer Network

Our goal of the experiment is to model a typical Peer-to-Peer network where each node contains a set of documents. We built different size of Peer-to-Peer networks to evaluate the performance of different search mechanisms. As shown in figure 3.10, the simulation models have a power law distribution with an average degree of 3.97. The number of peers in each network varies from 2,000 to 20,000. The diameters of the network¹ vary from 9 to 11, and the average distances between two peers vary from 5.36 to 6.58. Table. 5.1 lists the detailed information of each model.

To evaluate search mechanisms accurately, the simulations are done over different network formations and data locations. We run 100 iterations and

¹The longest shortest path between any two peers in the network.

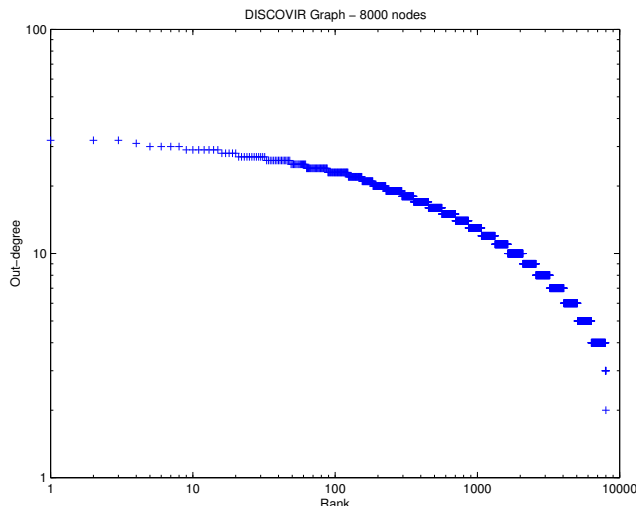


Figure 5.1: Power law distribution simulation model.

Table 5.1: Characteristic of Simulation Model

Number of Peer	2000	4000	8000	12000	16000	20000
Diameter	9	9	10	10	11	11
Average distance between two peers	5.36	5.77	6.12	6.33	6.47	6.58

average the results for each set of parameters. For each iteration, we initiate a query starting from a randomly selected peer and collect the statistical information listed in next section. We rebuild the network every 10 iterations. In our experiment, we use two set of test data, synthetic data and real data:

1. **Synthetic data**—We generated 100 sets of random mean and variance. For each set, 100 data points are generated according to the Gaussian distribution, which is to model feature vectors of data belonging to the same class.
2. **Real data**—We use 10,000 images (from 100 categories) in the Corel-Draw’s Image Collection CD and use the Color Moment feature as test data. The images in the CD are grouped based on their semantic meaning and under the same semantic meaning, images may not necessary have closely clustered feature vector. In our experiments, we show that

there is still 60% improvements in the query efficiency, which means that images having same semantic meaning are more or less clustered in low level feature vector space. But we can expect the real data cannot be clustered as well as our generated synthetic data.

In the experiment, we randomly assign different classes of images to each different peer; therefore, the data location of every build of network is different. Competitive Learning clustering algorithm [46] is used to cluster data insides each peer. The simulation is done on Sun Enterprise E4500 (12 400MHz Ultra Iii) running Solaris v.7 using C. In our C simulation program, each peer is represented by a data structure and the message passing between peers in the logical network is simulated by simple parameter passing between data structures. For a simulation of 20,000 peers, the running time is approximately 15 minutes. For the DISCOVIR system, we build our client program based on Gnutella v0.4 protocol. For image related operations, we use Java's image manipulation routines to assist in extracting visual feature.

5.2 Performance Metrics

The metrics we use to evaluate the performance are:

1. **Recall**– The success rate of desired result retrieved. It is the fraction of the relevant documents which has been retrieved, i.e.,

$$Recall = R_a/R, \quad (5.1)$$

where R_a is the number of retrieved relevant documents, 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, i.e.,

$$Visited = V_{peer}/T_{peer}, \quad (5.2)$$

where V_{peer} is the number of peers received and handled the query, 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, i.e.,

$$Efficiency = Recall/Visited. \quad (5.3)$$

In general, the performance of P2P network is more desirable if we can retrieve more relevant documents (high recall) but only visited fewer peers (small query scope). Observing either Recall or Query Scope only is not enough to determine the goodness of algorithm. Therefore, we defined the query efficiency as the ratio between Recall and Query Scope. If the algorithm can retrieval more relevant documents but only visited 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 BFS algorithm, i.e., if we visited 50% of peers in the network, it is expected that we can retrieve 50% of relevant documents in the network also.

4. **Generated network traffic**– The number of packets generated for each query. Reducing the load at individual peers and network traffic are desirable for scalability.
5. **Minimum reply path length**– The number of hops for the reply to come back. In our experiments, it is defined as the average number of hops for first ten replies. It is more desirable if the minimum reply path length is shorter. The requester can receive the reply in a shorter time and the system is also more scalable because fewer packets are generated.

5.3 Experiment Results

In this section, we experimentally compare our proposed Firework Query Model against Brute Force Search algorithm. We explore how the performances are affected by:

1. different number of peers in the P2P network
2. different Time-To-Live (TTL) value of query message
3. different data sets, synthetic data and real data
4. different number of local clusters of each peer

5.3.1 Performances in different Number of Peers in 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. 5.2.

Table 5.2: Parameters using in experiment 5.3.1

Number of Peer	2,000 - 20,000
Test Data Set	Real Image Data Set 1 from CorelDraw's Image Collection CD
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 documents
Dimension of extracted feature vector to represent the image	9
TTL value of the query packet	Fixed to 5
Number of local cluster per peer	1 (No local clustering)

1. Recall

Figure 5.2 depicts the recall against number of peers in two search mechanisms. 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 our algorithm is insensitive to the change of network size. Our mechanism is more scalable. Even the size of network grows, FQM still can reach a large portion of the network containing the query target.

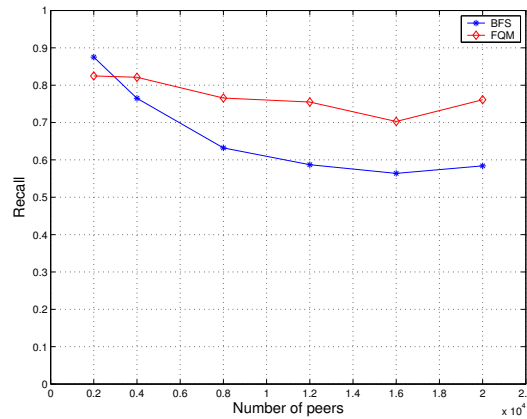


Figure 5.2: Recall against Number of Peers

2. Query Scope

As seen in Figure 5.3, we achieve the load reduction by using FQM. Fewer peers are exposed to each query.

3. Query Efficiency

As seen in Figure 5.4, FQM outperforms BFS because more relevant data are retrieved but fewer peers are visited. The efficiency is improved up 60% - 160%. The curve of FQM follows a small bell shape. Query efficiency increases at first due to two reasons:

- (a) The network can be clustered more appropriately when the network size increases. When the number of peers increases, new peers can

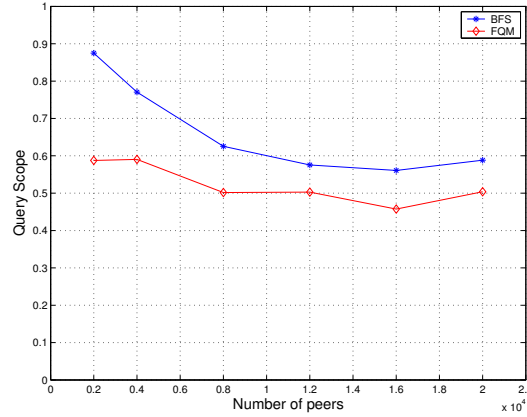


Figure 5.3: Query Scope against Number of Peers

have more choices and make their attractive link to a more similar peer.

- (b) 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 increases from 9 to 11 when the number of peers increases from 2000 to 20000), 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.

4. Generate Network Traffic

Our proposed Firework Query Model reduces the generated traffic by routing the query selectively rather than broadcasting. Figure 5.5 shows the average number of packets generated.

5. Minimum Reply Path Length

Figure 5.6 depicts the minimum reply path length of the average number of hops for first ten replies. On average, the relevant documents are 3-7

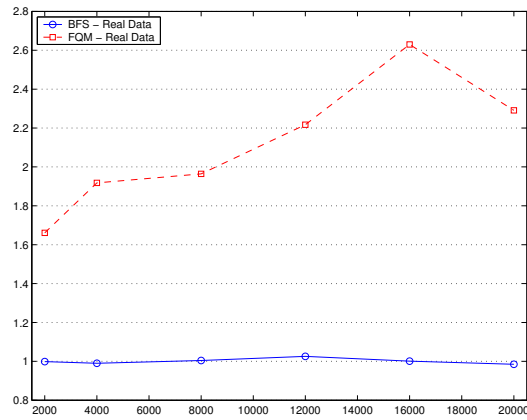


Figure 5.4: Query Efficiency against Number of Peers

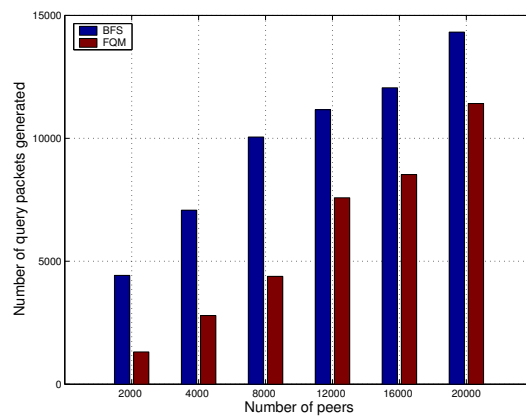


Figure 5.5: Generated Network Traffic against Number of Peers

hops away in Brute Force Search algorithm, however, the path length in Firework Query Model is just 2-5 hops. It is more desirable if the minimum reply path length is shorter. The requester can receive the reply in a shorter time and the system is also more scalable.

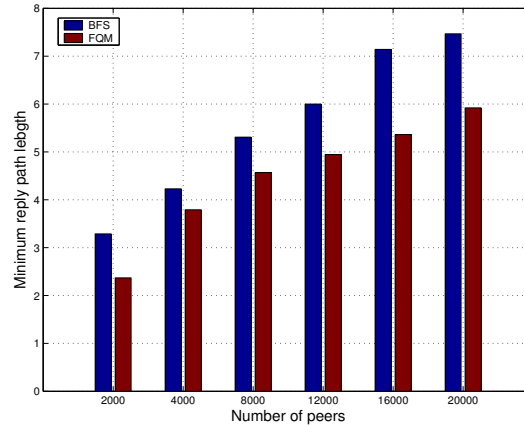


Figure 5.6: Minimum Reply Path Length against Number of Peers

6. Conclusion

In conclude, Firework Query Model outperforms Brute Force Search in all measures. The Query Efficiency of FQM is better because it finds more relevant data while visits fewer peers. Less query messages are generated. Some of the irrelevant peers avoid to receive and handle the query and the requester also receives the reply from other peers in a shorter reply path length.

5.3.2 Performances in 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. 5.3.

Table 5.3: Parameters using in experiment 5.3.2

Number of Peer	10,000
Test Data Set	Real Image Data Set 2 from CorelDraw's Image Collection CD
Diameter of the P2P network	10 hops
Average distance between 2 peers	6.2 hops
Number of documents assigned to each peer	100 documents
Dimension of extracted feature vector to represent the image	9
TTL value of the query packet	4 - 9
Number of local cluster per peer	1 (No local clustering)

1. Recall

Figure 5.7 shows the recall against different TTL values 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 in 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 improved anymore.

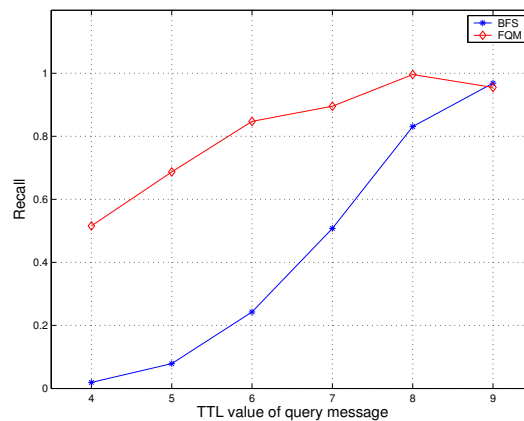


Figure 5.7: Recall against TTL value of query packet

2. Query Scope

Figure 5.8 shows the number of visited peers in both strategies. We vary the TTL of query message to observe the changes in the query

scope when a peer initiates a search. The Firework Query Model shows a promising sub-linear increase in the query scope subject to increasing TTL of query message, while the BFS increases in a much faster rate. The query scope of Firework Query Model is larger than BFS when the TTL value is small because a Chance-To-Survive (CTS)² value is introduced in Firework Query Model. This strategy lets the query message to 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.

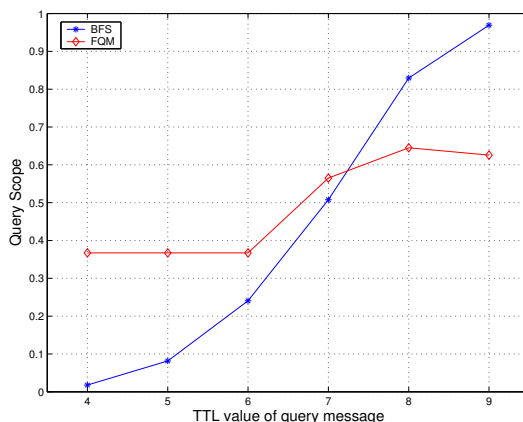


Figure 5.8: Query Scope against TTL value of query packet

3. Query Efficiency

As seen in Figure 5.9, FQM outperforms BFS under different TTL values of query packet. We found that the optimal TTL value is 6-8 in a network size of 10,000 peers under 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 increase, the query has a larger chance to reach its target cluster, therefore, the Query Efficiency increases. When the TTL value is 6-8, the Query Efficiency

²The inverse probability of decreasing TTL value. It is an arbitrary value in $[0, 1]$. If CTS is small, the chance to decrease the TTL value when the query is passing from one peer to another peer is larger.

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.

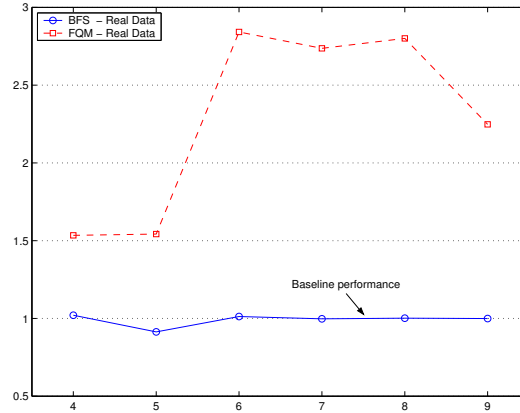


Figure 5.9: Query Efficiency against TTL value of query packet

4. Conclusion

In conclude, we find that FQM is sensitive to the TTL value of query message, so, choosing a suitable TTL value is important. If the chosen TTL is too small, the query cannot reach its target cluster. If the chosen TTL is too large, unnecessary traffic is generated. The performance of FQM is somehow related to the distance between the requester and the target cluster. However, FQM still outperforms BFS in general.

5.3.3 Performances in different different data sets, synthetic data and real data

In this section, we explore how the performances are affected in different data sets, synthetic data and real data. We assign different data set to each peer. In each experiment, we carry two different parts. In the first part, the number of peers in each network varies from 2,000 to 20,000 while the TTL value is

fixed to 5. In the second part, the number of peers in each network is fixed to 10,000 while the TTL value varies from 4 to 9. The experiment parameters are listed in Table 5.4 and 5.5.

1. Query Efficiency against different number of peers

Table 5.4: Parameters using in experiment 5.3.3

Number of Peer	2,000 - 20,000
Test Data Set	1) Real Image Data Set 3 from CorelDraw's Image Collection CD 2) Synthetic data generated following Gaussian distribution
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 documents
Dimension of extracted feature vector to represent the image	9
TTL value of the query packet	Fixed to 5
Number of local cluster per peer	1 (No local clustering)

As seen in Figure 5.10, FQM outperforms BFS in both synthetic data and real data set. The efficiency is improved up 60% - 160% 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. Therefore, the synthetic data can be clustered more appropriately insides the network, thus, the performance is much better.

2. Query Efficiency against different TTL values of query packet

As seen in Figure 5.11, FQM outperforms BFS under different TTL values of query packet. In synthetic data, we found that the optimal TTL value is 8 in a network size of 10,000 peers under Firework Query Model. Since the variance of synthetic data we generated is much smaller than the variance of real data. Therefore, the synthetic data can be clustered

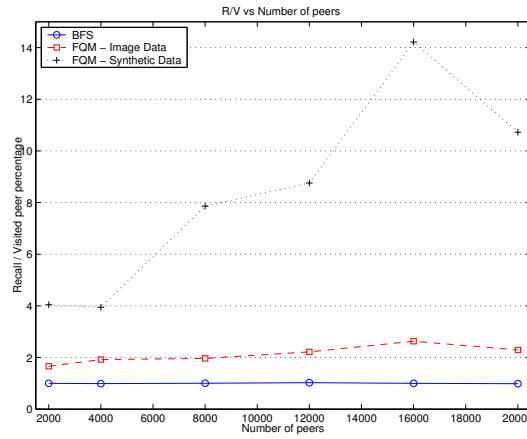


Figure 5.10: Query Efficiency against Number of Peers under different data sets

Table 5.5: Parameters using in experiment 5.3.3

Number of Peer	10,000
Test Data Set	1) Real Image Data Set 3 from CorelDraw's Image Collection CD 2) Synthetic data generated following Gaussian distribution
Diameter of the P2P network	10 hops
Average distance between 2 peers	6.2 hops
Number of documents assigned to each peer	100 documents
Dimension of extracted feature vector to represent the image	9
TTL value of the query packet	4 - 9
Number of local cluster per peer	1 (No local clustering)

more appropriately inside the network, thus, the performance is also much better.

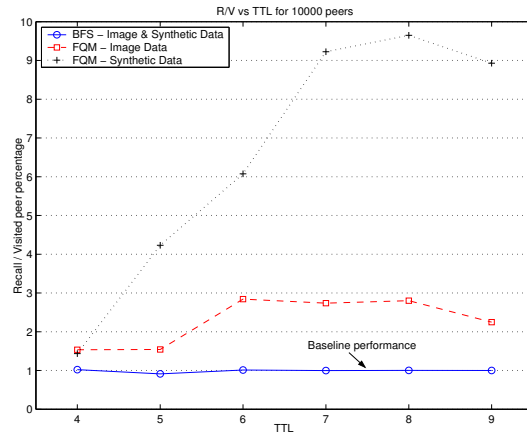


Figure 5.11: Query Efficiency against TTL value of query packet under different data sets

3. Conclusion

From the results of the experiments, we found that our algorithm work well in both real data and synthetic data. Using both data sets have at least 60% improvement. In our experiment, we use the feature vector extracted by Color Moment algorithm. However, since the images in the CorelDraw Image CD are grouped based on their semantic meaning and under the same semantic meaning, images may not necessary have closely clustered feature vector in color space, therefore, the real image data cannot be clustered as well as our generated synthetic data, thus, the performance is worse as expectation.

5.3.4 Performances in different number of local clusters of each peer in P2P Network

In this section, we explore how the performances are affected in different number of local clusters of each peer in P2P network. Two settings are used in

the experiments: 1 single cluster to represent a peer (Peer Clustering - Single Cluster Version: no local clustering is performed) and 3 local sub-clusters to represent a peer (Peer Clustering - Multiple Clusters Version: local clustering is performed) In each experiment, we carry two different parts. In the first part, the number of peers in each network varies from 2,000 to 20,000 while the TTL value is fixed to 5. In the second part, the number of peers in each network is fixed to 10,000 while the TTL value varies from 4 to 9. The experiment parameters are listed in Table 5.6 and 5.7.

Table 5.6: Parameters using in experiment 5.3.4

Number of Peer	2,000 - 20,000
Test Data Set	Real Image Data Set 4 from CorelDraw's Image Collection CD
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 documents
Dimension of extracted feature vector to represent the image	9
TTL value of the query packet	Fixed to 5
Number of local cluster per peer	1 , 3

1. Recall, Query Scope and Query Efficiency against different number of peers

Figure 5.12, 5.13 and 5.14 depict the recall, query scope and query efficiency against different number of peers under different local clusters of each peer in P2P Network. In general, FQM with 3 clusters outperforms FQM with 1 cluster. As shown in Figure 5.12, both algorithm have the same recall while FQM with 3 clusters visits less peers than FQM with 1 cluster, thus, the query efficiency is better in FQM with 3 clusters.

The curve of query efficiency follows a small bell shape because 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. However, when the network further increases, a query might not reach its target cluster for low TTL value (The TTL is fixed to 5 in this experiment), therefore, the efficiency drops.

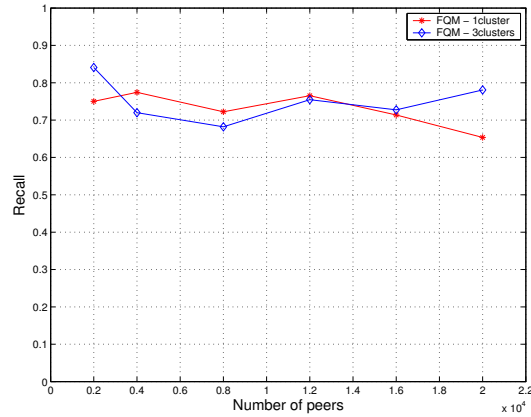


Figure 5.12: Recall against Number of Peers

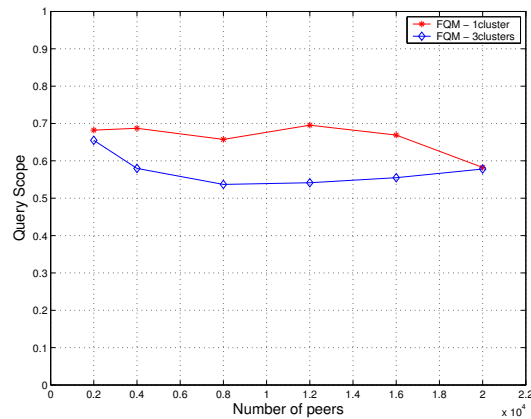


Figure 5.13: Query Scope against Number of Peers

2. Generate Network Traffic against different number of peers

As shown in Figure 5.15, our proposed Firework Query Model with 3 local sub-clusters can represent the peer more accurately and routes the query more efficiently, therefore, less unnecessary traffic is generated.

3. Recall, Query Scope and Query Efficiency against different TTL

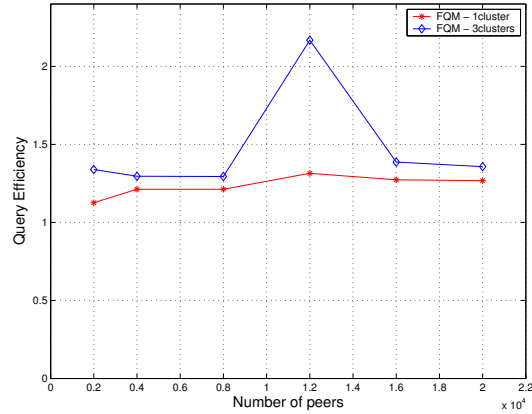


Figure 5.14: Query Efficiency against Number of Peers

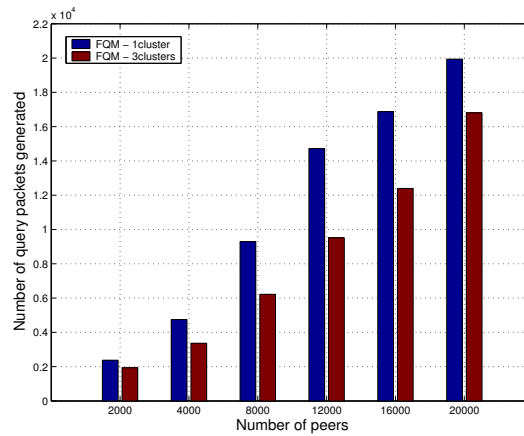


Figure 5.15: Generated Network Traffic against Number of Peers

Table 5.7: Parameters using in experiment 5.3.4

Number of Peer	10,000
Test Data Set	Real Image Data Set 4 from CorelDraw's Image Collection CD
Diameter of the P2P network	10 hops
Average distance between 2 peers	6.2 hops
Number of documents assigned to each peer	100 documents
Dimension of extracted feature vector to represent the image	9
TTL value of the query packet	4 - 9
Number of local cluster per peer	1 , 3

values of query packet

Figure 5.16, 5.17 and 5.18 depict the recall, query scope and query efficiency against different TTL values of query packet under different local clusters of each peer in P2P Network. The results are the nearly same as the last experiment. FQM with 3 clusters still outperforms FQM with 1 cluster in general.

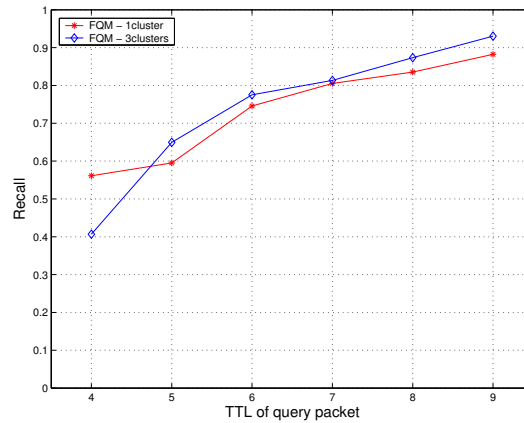


Figure 5.16: Recall against TTL value of query packet

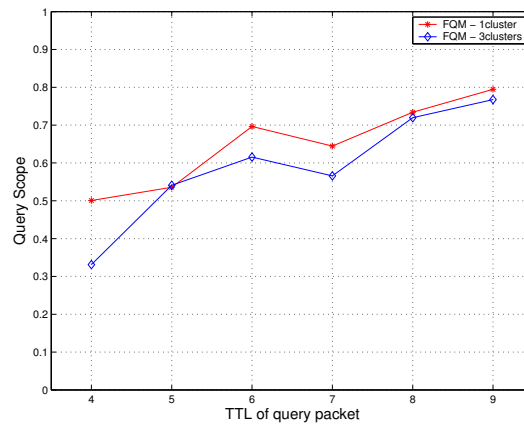


Figure 5.17: Query Scope against TTL value of query packet

4. Generate Network Traffic against different TTL values of query packet

As shown in Figure 5.19, our proposed Firework Query Model with 3

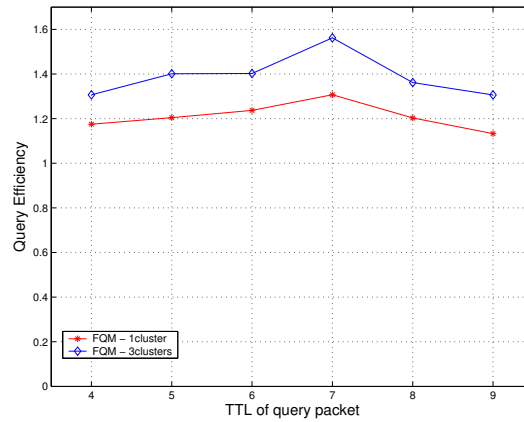


Figure 5.18: Query Efficiency against TTL value of query packet

local sub-clusters can represent the peer more accurately and routes the query more efficiently, therefore, less unnecessary traffic is generated.

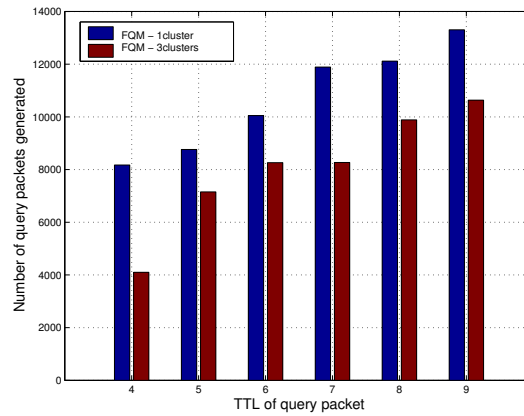


Figure 5.19: Generated Network Traffic against TTL value of query packet

5. Conclusion

In conclude, 3 local sub-clusters to represent a peer (Peer Clustering - Multiple Clusters Version: local clustering is performed) outperforms 1 single cluster to represent a peer (Peer Clustering - Single Cluster Version: no local clustering is performed). Since our proposed Firework Query Model with 3 local sub-clusters can represent the peer more accurately and routes the query more efficiently, therefore, the query efficiency is better and less unnecessary traffic is generated. We expect the

performance will be even better if more local sub-clusters are used to represent a peer. However, it is a trade-off between resolution of cluster and computational cost.

5.4 Evaluation of different clustering algorithms

To choose a good clustering algorithm to be used in the simulation experiments and the development of DISCOVIR, we tested the efficiency and correctness of different clustering algorithms in three sets of experiments. The first set of experiments examine the performance of clustering algorithms when the estimated cluster number (k) is equal to the actual cluster number (c) we generated. The second set of experiments examine the performance of under estimation of cluster number ($k < c$). The last set of experiments examine the performance of over estimation of cluster number ($k > c$). In each experiment, we compare the performance of following clustering algorithms: k -means (KM) [1], competitive learning (CL) [46], shift mean (SM), expectation maximization (EM) [23], branching competitive learning (BCL) [58] and adaptive rival penalized competitive learning (RPCL) [27, 24].

In our experiments, we use a 2-dimensional generated data sets to evaluate the clustering performance of each clustering algorithms. We generate 5 Gaussian distribution with $\sigma = 5$ and centered at $(0, 500)$ and $(0, 500)$ respectively. Each cluster has 1000 generated data points. In the experiments, we measure the accuracy of clustering, which is measured by the average percentage of correct classification data over 10 consecutive runs.

Figure 5.20 shows one of the results of competitive learning (CL). The black points are the data points generated by 5 Gaussian distribution. The color paths are the moving paths of our estimated cluster centers. We construct the confusion matrix of each algorithm and calculate the accuracy of it. The accuracy is defined as the proportion of the total number of predictions that

were correct. It is determined using the equation:

$$Accuracy = T_c/T. \quad (5.4)$$

where T_c is the total number of predictions that were correct and T is total number of data. Table 5.8 shows one of the sample result of confusion matrix. The accuracy is 88.82% $((977+985+909+920+650)/5000)$ in this example.

Table 5.8: Result of confusion matrix of competitive learning.

	Number of data points classified by CL				
	Class A	Class B	Class C	Class D	Class E
Class A	977	5			8
Class B		985			15
Class C		91	909		
Class D		17		920	63
Class E	222	128			650

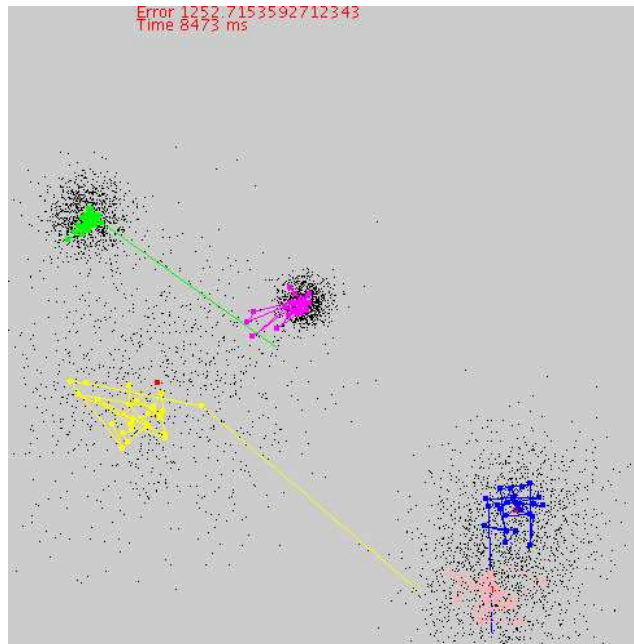


Figure 5.20: Screen caption of competitive learning experiment

Table 5.9 shows the average clustering accuracy when the estimated cluster number (k) is equal to the actual cluster number. Table 5.10 shows the average clustering accuracy of under estimation for cluster number ($k < c$, and $k = 4$).

Table 5.11 shows the average clustering accuracy for over estimation of cluster number ($k > c$, and $k = 7$). We can see that competitive learning (CL) gives more accurate clustering result than others in all three cases. They do not be affected by k easily and give a more stable result. Therefore, we have chosen competitive learning as our clustering algorithm in simulations and DISCOVER development.

Table 5.9: Average clustering accuracy in estimated cluster number equal to the actual cluster number

algorithm	average correctness
KM	74.92%
SM	65.22%
CL	77.92%
RPCL	70.56%
BCL	84.30%
EM	76.52%

Table 5.10: Average clustering accuracy in under estimation of cluster number

algorithm	average correctness
KM	73.22%
SM	66.26%
CL	73.80%
RPCL	70.82%
BCL	73.04%
EM	75.94%

Table 5.11: Average clustering accuracy in over estimation of cluster number

algorithm	average correctness
KM	66.96%
SM	68.92%
CL	70.86%
RPCL	70.24%
BCL	65.02%
EM	73.08%

Chapter 6

Conclusion

In this thesis, we propose a peer clustering and content-based routing strategy to retrieve information based on their content 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 value of query message. We show that our Fire Query Model outperforms the Brute Force Search method in both network traffic cost and query efficiency measure.

Moreover, we migrate the traditional CBIR to the P2P network to distribute storage capacity and workload among peers and provide content-based search in P2P network. We illustrate the design and implementation of DISCOVER, in order to exhibit the key components required in a P2P based CBIR system. We also illustrate how Fire Query Model can be integrated into P2P systems to increase retrieval performance.

Bibliography

- [1] M. R. Anderberg. Cluster analysis for applications. In *Academic Press, New York*, 1973.
- [2] AsiaYeah Gnutella Search. http://www.asiayeah.com/service/searchFile_c.jsp.
- [3] AudioFind. <http://www.audiofind.com/>.
- [4] Wendy Chang, Gholamhosein Sheikholeslami, Jia Wang, and Aidong Zhang. Data Resource Selection in Distributed Visual Information Systems. *IEEE Transactions on Knowledge and Data Engineering*, 10(6):926–946, November/December 1998.
- [5] 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.
- [6] Brain F. Cooper and Hector Gracia-Molina. Peer-to-peer data trading to preserve information. *ACM Transcation of Information Systems*, 20(2):133–170, April 2002.
- [7] George Coulouris, Jean Dollimore, and Tim Kindberg. *Distributed Systems Concepts and Design*. Addison-Wesley, third edition, 2001.
- [8] Arturo Crespo and Hector Gracia-Molina. Routing Indices For Peer-to-Peer Systems. In *Proceedings of the Internation Conference on Distributed Computing Systems (ICDCS)*, July 2002.

- [9] DIStirbuted COntent-based Visual Information Retrieval. <http://www.cse.cuhk.edu.hk/~miplab/discovir>.
- [10] DISCOVIR Everywhere. <http://www.cse.cuhk.edu.hk/~ik0203>.
- [11] The eDonkey2000 homepage. <http://www.edonkey2000.com>.
- [12] C. Faloutsos, R. Barber, M. Flickner, J. Hafner, Niblack 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.
- [13] The freenet homepage. <http://freenet.sourceforge.net>.
- [14] The gnutella homepage. <http://www.gnutella.com>.
- [15] The gnutella protocol. http://www9.limewire.com/developer/gnutella_protocol_0.4.pdf.
- [16] Gnutellait Web Search. <http://gnutella.abctella.it/>.
- [17] Rafael C. Gonzalez and Richard E. Woods. *Digital Image Processing*. Addison Wesley, first edition, 1992.
- [18] A. Gupta and R. Jain. Visual information retrieval. In *Communications of the ACM*, volume 40, pages 70–79, 1997.
- [19] Matthew Harren, Joseph M. Hellerstein, Ryan Huebsch, Boon Thau Loo, Scott Shenker, and Ion Stoica. Complex Queries in DHT-based Peer-to-Peer networks. In *Proceedings of the International Peer-to-Peer Workshop*, 2002.
- [20] The jabber homepage. <http://www.jabber.org>.
- [21] Vana Kalogeraki and Dimitrios Gunopulos. Information retrieval in peer-to-peer networks. 2001.

- [22] Mandar Kelaksar, Vincent Matossian, Pretti Mehra, Dennis Paul, Anand Vaihyathan, and Manish Parashar. A study of discovery mechanisms for peer-to-peer applications. In *IEEE International Symposium on Cluster Computing and the Grid (CCGrid'2002)*, 2001.
- [23] Irwin King and Zhong Jin. Relevance feedback content-based image retrieval using query distribution estimation based on maximum entropy principle. In Liming Zhang and Fanji 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.
- [24] Irwin King and Tak-Kan Lau. Non-hierarchical clustering with rival penalized competitive learning for information retrieval. In *In Petra Perner and Maria Petrou, editors, Proceedings of the First International Workshop on Machine Learning and Data Mining in Pattern Recognition (MLDM'99)*, pages 116–130, 1999.
- [25] T. K. Lau and I. King. Montage: An image database for the fashion, clothing, and textile industry in Hong Kong. In *Proceedings of the Third Asian Conference on Computer Vision (ACCV'98)*, volume 1351 of *Lecture Notes in Computer Science*, pages I 410–417. Berlin, Germany: Springer Verlag, January 4-7, 1998.
- [26] 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.
- [27] Xue Qun Li and Irwin King. Hierarchical rival penalized competitive learning binary tree for multimedia feature-based indexing. In *First International Workshop on Intelligent Multimedia Computing and Networking (IMMCN2000)*, 2000.

- [28] Modern peer-to-peer file-sharing over the internet. <http://www.limewire.com/index.jsp/p2p>.
- [29] LinkGrinder. <http://www.gnutellagrinder.com/>.
- [30] 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.
- [31] S. Mehrotra, Y. Rui, M. Ortega, and T.S. Huang. Supporting content-based queries over images in mars. In *Proc. IEEE Int. Conf. Multimedia Computing and Systems*, pages 632–633, 1997.
- [32] The morpheus homepage. <http://www.musiccity.com>.
- [33] S. Mukherjea, K. Hirata, and Y. Hara. Amore: a world wide web image retrieval engine. In *World Wide Web*, volume 2, pages 115–132, 1999.
- [34] The napster homepage. <http://www.napster.com>.
- [35] A. Natsev, R. Rastogi, and K. Shim. Walrus: A similarity retrieval algorithm for image databases. In *Proc. SIGMOD, Philadelphia, PA*, 1999.
- [36] Cheuk Hang Ng and Ka Cheung Sia. Peer Clustering and Firework Query Model. In *Poster Proc. of The 11th International World Wide Web Conference*, May 2002.
- [37] Cheuk Hang Ng, Ka Cheung Sia, and Chi Hang Chan. Advanced Peer Clustering and Firework Query Model. In *Poster Proc. of The 12th International World Wide Web Conference*, May 2003.
- [38] 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.

- [39] 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.
- [40] Marius Portmann, Pipat Sookavatana, Sebastien Ardon, and Aruna 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.
- [41] Michael T. Prinkey. An efficient scheme for query processing on peer-to-peer networks. 2001.
- [42] S. Ratnasamy, P. Francis, M. Handley, and S. Karp R. Shenker. A scalable content-addressable network. In *In Proc. ACM SIGCOMM*, August 2001.
- [43] Sylvia Ratnasamy, Scott Shenker, and Ion Stoica. Routing algorithms for DHTs: Some open questions. In *Proceedings of the International Peer-to-Peer Workshop*, 2002.
- [44] Antony Rowstron and Peter 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.
- [45] Yong Rui, Thomas S. Huang, and Shih-Fu Chang. Image retrieval: current techniques, promising directions and open issues. *Journal of Visual Communication and Image Representation*, 10:39–62, April 1999.
- [46] D.E. Rumelhart and D. Zipser. Feature discovery by competitive learning. In *Cognitive Science*, 1985.
- [47] The worldwide computer. <http://www.scientificamerican.com/2002/0302issue>.

- [48] The Search for Extraterrestrial Intelligence homepage.
<http://www.setiathome.ssl.berkeley.edu>.
- [49] Ka Cheung Sia, Cheuk Hang Ng, Chi Hang Chan, Siu Kong Chan, and Lai Yin Ho. Bridging the P2P and WWW Divide with DISCOVER - DIStributed COntent-based Visual Information Retrieval. In *Technical Report*, December 2002.
- [50] Ka Cheung Sia, Cheuk Hang Ng, Chi Hang Chan, Siu Kong Chan, and Lai Yin Ho. Bridging the P2P and WWW Divide with DISCOVER - DIStributed COntent-based Visual Information Retrieval. In *Poster Proc. of The 12th International World Wide Web Conference*, May 2003.
- [51] Arnold W.M. Smeulders, Marcel Worring, Simone Santini, Amarnath Gupta, and Ramesh 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.
- [52] 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.
- [53] Kunwadee Sripanidkulchai, Bruce Maggs, and Hui Zhang. Efficient Content Location Using Interest-Based Locality in Peer-to-Peer Systems. In *Proceedings of Infocom'03*, 2003.
- [54] Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek, and Hari Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *Proceedings of ACM SIGCOMM*, pages 149–160, August 2001.
- [55] Chuqiang Tang, Zhichen Xu, and Mallik Mahalingam. PeerSearch: Efficient Information Retrieval in Peer-to-Peer Networks. In *Proceedings of HotNets-I*, October 2002.

- [56] J. Z. Wang, Wiederhold G., O. Firschein, and X.W. Sha. Content-based image indexing and searching using Daubechies' wavelets. *International Journal of Digital Libraries*, 1(4):311–328, 1998.
- [57] 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.
- [58] Huilin Xiong and Irwin King. Branching competitive learning for clustering. In *International Conference on Neural Information Processing (ICONIP2000)*, pages WBP–27, 2000.
- [59] The yahoo! messenger homepage. <http://messenger.yahoo.com>.
- [60] Beverly Yang and Hector Garcia-Molina. Efficient search in peer-to-peer networks. In *Proceedings of ICDCS*, 2002.
- [61] B. Zhao, J. Kubiawicz, and A. Joseph. Tapestry: An infrastructure for fault-tolerant wide-area location and routing. Technical report, Computer Science Division, U.C. Berkeley, April 2001.