PP-COSE: A P2P Community Search Scheme

Hsing Mei and Steven Chang
Computer Science and Information Engineering Department
Fu Jen Catholic University
mei@csie.fju.edu.tw

Abstract

P2P systems allow peers communicate with each other directly. Most P2P systems aimed at one specific application only, such as file sharing, or instant messaging. Each P2P system requires its own search scheme to find the sharing resources. We propose to integrate P2P systems with community support. Thus, a peer joining one P2P system could access multiple applications. This paper introduces a P2P Community Search (PP-COSE) scheme support efficient peer search in generalized P2P systems. To make efficient searching, the P2P network is formed by small clusters. Queries are processed by search router in each cluster. A community filtering mechanism is designed to reduce the management overhead. Less important or unpopular communities are filtered out, and the number of advertised communities in each search router is controlled. A test bed is implemented to measure the performance. The results show that the PP-COSE with routed forwarding may reduce the query cost up to 80% comparing with flooding forwarding.

Keywords—P2P system, community management

1. INTRODUCTION

The users of Peer-to-Peer (P2P) system have grown significantly in recent years. In P2P systems, each node is called a "peer". It means that a node has equal capabilities to exchange information directly with other nodes. In traditional Client-Server systems, data flows between servers and clients, and resources are often provided by servers. In P2P systems, each peer plays the role of both client and server, and peers provide resources to each other. Peers are autonomous. Each peer may decide what resources it wants to share, and when it joins and leaves a P2P system. Peers are not always connected, and might not have permanent address.

P2P processing model provides a novel thinking to help designing and applying to real world systems. At present, P2P applies to applications such as file-sharing (e.g. Napster, Gnutella, KaZaA); instant messaging (e.g. MSN Messenger, ICQ); and group collaboration (e.g. Groove Workspace). Although P2P systems are adopted widely, they still face some obstacles need to be overcome. Some researches deal with P2P security and privacy issues. Other researches try to design efficient schemes to save the network bandwidth. One fundamental problem yet to be solved is how to efficiently discover peers and locate shared resources.

Peer searching could be classified to three types. The first is centralized search (e.g. Napster). A central server is responsible for indexing peers and sharing resources. In this approach, it is easy to discover peers and locate resources, but it lacks scalability and has the single point of failure. The second is distributed searching (e.g. Gnutella, FreeNet). There is no server or coordinator involved in searching process. Each peer has to discover other peers by itself. In distributed searching, a peer may use broadcast [5], random [2], routing [6] or hash [7] algorithms to forward the query searching request. Without single point of failure, it may have good reliability and fault-tolerance. However, the discovery and searching process are usually more complex. The last one is hybrid search (e.g. CAP [4]). It divides the network into clusters. Each cluster has one or more coordinators performing management and inter-cluster communication. Hybrid searching has the advantages of centralized and distributed searching. Dividing network into clusters not only reduce the traffic over the network but also balance the workload of coordinators.

On the other hand, community is a common technique used to organize the big group. In file sharing applications, there may exist communities such as game, photo, art and music etc. People join a community because they are interested in the particular topic or they belong to particular group. In generalized P2P systems, peers with the same interests or groups may form a community. A peer is allowed to join a community or leave one during system operation. A peer can be a member of more than one community. Consequently, mechanisms are needed for managing communities.

This paper is target on providing an efficient scheme to find peers of specific community, and this search scheme can also be applied to generalized P2P systems. Taking computer game for example, people want to play on-line games with others. Under the traditional Client-Server environment, players have to connect to the game server and wait for enough players to start the game. With our proposed scheme, players can easily send a request for inviting other members to join the game with low overhead. As mention earlier, we need an approach to perform searching and a mechanism to manage communities. To guarantee the scalability, the PP-COSE scheme proposed adopted the hybrid searching to perform searching, and manage communities in the distributed manner.

In PP-COSE environment, P2P system is formed by clusters. There is a coordinator called *Search Router* in each cluster. The architecture of PP-COSE is shown in Figure 1. Peers have to register to a Search Router; and they can create, join, and leave a community by themselves. A Search Router first indexes the profiles of local peers, and then constructs a *Search Routing Table*. According to the Search Routing Table, the Search Router constructs a *Forwarding Table*. The Forwarding Table is exchanged with neighboring Search Routers. Avoiding wasting bandwidth on exchanging those useless communities, PP-COSE provide a *filter function* to filter out less important communities. Thus, a query can be routed following the route in the *Search Routing Table*.

Under PP-COSE scheme, search requests are forwarded following the routes in *Search Routing Table*. The requests could also be forwarded with flooding mechanism, or just randomly. Following the PP-COSE routed forwarding search, bandwidth overhead is lower. The obvious performance improvements are shown from our experiments results. Comparing with random forwarding, the routed forwarding saves up to 65% of query cost. The saving is up to 80% comparing with flooding forwarding.

The remained paper is organized as following. Section 2 provides the background related to this research. Section 3 presents the PP-COSE architecture and mechanism of community management. Section 4 shows the experiment environment and results. Finally, the conclusion is presented.

2.BACKGROUND

In this section, we present the related researches on P2P search and also on community management.

2.1. P2P Searching

P2P peer searching can be classified into centralized, distributed, and hybrid searching Napster is the typical P2P system using centralized searching. Napster uses a set of central hub servers to avoid the single point of failure problem and enhance its scalability. All peers must register to a server, and then the server can index the sharing resources by registrations. When a peer wants to find a file, it sends a query to the registered server, and waits for a response. The server searches it local index by key-word from the query. After search is finished, the server sends a response to the requesting peer. The response includes the information of matching peers. With the information, the peer can download the file directly from other peers [1].

In distributed searching, each peer has to discover other peers by itself, without help of coordinators (or servers). All peers must know at least one neighbor (peer) to discovery other peers or forward the query. In distributed searching, a peer may use broadcast, random, routing or hash algorithms to forward the searching queries. Without single point of failure, it may have good reliability and fault-tolerance. However, the discovery and searching process are usually either more costly or complex.

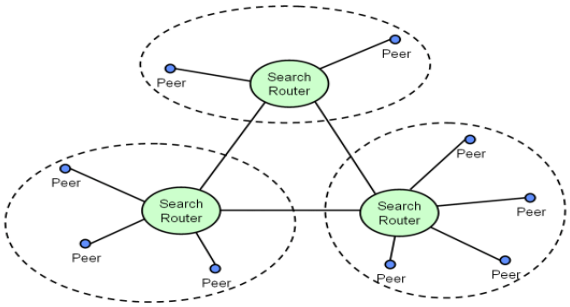

Figure 1. PP-COSE architecture.

Gnutella is a widely studied P2P file-sharing system, and it uses broadcast searching algorithm (i.e. flooding). When a peer likes to find a file, it sends a query to all known peers. Those peers, who receive the query, search theirs local sharing indices for matching files, and then forward the query to their known peers until the TTL is equal to 0. Gnutella's search scheme is costly and lacks of scalability. Yang and Garcia-Molina proposed a scheme to improve search. Queries are just forwarded to a subset of neighbors. They argue that while their techniques maintain the same quality of results, they consume up to 5 times less resources [5].

Portmann and Seneviratne suggested that the peer randomly chooses a neighbor to forward the query. They named the method as Rumor Mongering [2]. Crespo and Garcia-Molina proposed a concept of Routing Indices, which allow nodes to forward queries to neighbors that are more likely to have answers. The query is forwarded like a message routing according to the Routing Index in each peer [6]. Some of these search schemes use hash algorithm to locate objects (files). Although a peer just requires a few hops to locate an object by using hash algorithm, the operation is usually more complex, and it also required peers to provide private storage to keep objects or indices.

In hybrid searching, network is divided into clusters. Each cluster has one or more coordinators performing management and inter-cluster communication. When one peer needs to find a resource, it sends a query to the coordinator it has registered. The coordinator will complete the search with other coordinators and return a response to the requesting peer. Hybrid approach has the advantages of centralized and distributed approach. Dividing network into clusters not only reduce the traffic over the network but also balance the workload of coordinators.

2.2. Community

Community can be managed with a community server which responsible for creating and deleting a community. The community server maintains a complete data base of the communities. This centralized approach is straightforward, efficient, and easy to implement. Unfortunately, it has the major disadvantages as other centralized systems: single point of failure, and performance bottleneck on server.

The alternative is the distributed community management. A peer may multicast a message to all community members announcing its wish to join the community. To leave a community, a member peer just sends a goodbye message to everyone. The problem is: there is no polite announcement that a peer crash. Other members have to discover this experimentally by noticing that the crashed member no longer responds to any query. Once the crashed member is really down and not responding slowly, it can be removed from the community [4].

Our goal is to find "enough" members of a specific community not for "all". We do not need to waste so much effort to keep a global view of communities. Hence, we proposed a filter function for community management, and the filter function is presented in section 3.2.

3. PP-COSE COMMUNITY MANAGEMENT

This section describes the details of proposed PP-COSE Search scheme. Search queries may be forwarded by flooding, random, or routing algorithm. Flooding algorithm wastes bandwidth, and random algorithm to forward the query cannot guarantee the performance. PP_COSE adopts better the routing algorithm approach. Comparing with the other two algorithms, routing algorithm is more efficient. This section also discusses the community management with filter functions. Then, the search routing algorithm is presented.

3.1. PP-COSE Architecture

As discussed in section 1, PP-COSE adopts hybrid searching approach. The architecture is shown in Figure 1. The network is divided into small clusters. The cluster may be divided by geographic or by organization structure. The cluster may be formed by peers naturally, or assigned by authorities. As to the details of how to form a cluster is out of scope of this paper. In each cluster, there is a coordinator called Search Router. Search Routers are responsible for three tasks. The first one is to manage communities, the second task is to maintain tables (Search Routing Table and Forwarding Table), and the last task is to route queries. The peers within a cluster must register to the Search Router with a profile. A profile contains the communities that a peer joined. We designed a filter function for community management. We will describe filter function in detail next section. In this section, we focus on the role of a Search Router.

Firstly, a Search Router indexes the profiles that register to it, and uses the index to lookup the matching peers. According to the index, a Search Router constructs a Search Routing Table first, then constructs a Forwarding Table based on the Search Routing Table. Forwarding Tables are exchanged by adjacent Search Routers periodically. Each Search Router updates its Search Routing Table according to the content of Forwarding Tables it received. When a Search Router receives a query, the Search Router firstly uses its index to lookup matching peers. If the number of matching peers that has been found is not enough to satisfy the query, the Search Router forwards the query to a suitable neighbor according to its Search Routing Table. The operational details of the Search Router are depicted in section 3.3.

3.2. Community Filtering

This subsection discusses the community management. An example for community filtering is presented. In PP-COSE, we use a three level notation to represent a community. A filter function is designed to reduce the overhead of community management.

In current PP-COSE version, we use a three level notation to form a community. The community is notated like "x.y.z", where x is a subset of y, and y is a subset of z. For example, a peer belongs to a community "csie.fju.edu". It means that the peer is a member of csie community, and csie community is a sub-community of fju, and fju community is a sub-community of edu. On the other hand, the peer is a member of fju, and also a member of edu. By using this three level structure of community notation, a Search Router can forward a query to a suitable neighbor.

A peer can create a community by itself. The peer just adds the community it wants to create into its profile, and then registers to a Search Router. Then, a community is created. The peer that creates a community can announce the existence of the community through the third-party (e.g. BBS, web, or forum). At current PP-COSE, it is not necessary to delete a community. A filter function is used to simulate the effect of community deleting. When a community becomes unpopular, the number of members that join the community or the number of queried to that community will be small. The filter function will filter out these communities. Communities that are filtered out will not be shown in Search Routing Table. Therefore, filter function can simulate the effect of community deleting. Joining and leaving a community are operated similarly. A peer who wants to join or leave a community just updates its own profile, and then registers to the corresponding Search Router in the cluster. A peer may get the community information from Search Router.

Because communities are created by peers autonomously, the size of Search Routing Tables may grow quickly. As time goes on, some community may become unpopular. If there is no way to deal with this situation, it may affect the performance and waste resource. A filter function is proposed:

$$F = G + \frac{N_m}{\theta_m} + \frac{N_q}{\theta_q}, \quad where$$

G: God's hand
N_m: number of members
Θ_m: member threshold
N_q: number of query times

Each community in Search Router is calculated by the filter function to get a value called F. If the value of F is smaller than 1, the community will be filter out, and will not be added into Search Routing Table. G stand for God's hand. The value of G is either 1 or 0. If someone wants a community is always shown on Search Routing Table, G can be set to the value 1. The default value of G is 0. N_m represents the number of members that joined the community. N_q represents the number of times the community has been queried. Θ_m is the threshold

of members in this Search Router. Θ_q is the threshold of query times in this Search Router.

For simplicity, a two level community notation is used for illustration. Assume the membership within a Search Router is shown in Table 1, and Θ_m = 10, Θ_q = 5. Thus, we can calculate the value of F for each community.

Table 1: The membership within a Search Router.

Community	G	N_m	N_q
a.A	0	5	5
b.A	0	3	2
A	0	8	7
a.B	0	3	10
b.B	0	1	30
c.B	1	1	0
B	1	5	40
a.C	0	1	0
C	0	1	0

F(a.A) stands for the value of F for community "a.A". The value of F for each community is listed as follow: F(a.A) = 1.5; F(b.A) = 0.7; F(A) = 2.2; F(a.B) = 2.3; F(b.B) = 6.1; F(c.B) = 1.1; F(B) = 9.5; F(a.C) = 0.1; and F(C) = 0.1. By the value of F, the filter function filters out community "b.A", "a.C", and "C". Because G is set to 1 for the community "c.B", even only one peer join the community, it will not be not filtered out. The communities that are not filtered out will be added into the Search Routing Table.

3.3. Search Routing (Routing-based Forwarding)

In PP-COSE, each query contains five elements: community name, result threshold, found results, TTL (Time-To-Live), and visited list. The TTL is a system parameter, and it means how many hops this query can be forwarded before discarded by Search Routers. If a peer intends to find at least 500 peers from community "csie.fju.edu", and result threshold is set to be 500. The number of peer found is accumulative. The visited list is used to record Search Routers that query has visited before.

As mention earlier, there are three forwarding approaches can be used with PP-COSE: flooding, random, and routing approach. When a query is received by a Search Router, the Search Router uses its index to lookup matching peers. If thenumber of peers found is smaller than the value of result threshold, the query will be forwarded by the Search Router. With the flooding algorithm, a Search Router forwards the query to all neighbors excluding the previous one. With the random algorithm, a Search Router randomly chooses a neighbor excluding the previous Search Router to forward the query. With the routing algorithm, a Search Router selects a suitable neighbor excluding the previous Search Router to forward the query. We expect routing algorithm will be more efficient and has better performance.

Search Routers exchange theirs Forwarding Tables with neighbors periodically. When a Search Router receives a neighbor's Forwarding Table, it updates its Search Routing Table immediately. Initially, each Search Router constructs its own table according to the profiles form peers. A record in the Search Routing Table is composed of community name, number of members, hops, and path, where hops mean how many hops the community is located. The path field shows the information resource (i.e. a Search Router) of this community.

After Search Routing Table is constructed, each Search Router then constructs a Forwarding Table. A record in a Forwarding Table is the same as the record in a Search Routing Table. Records in Forwarding Table are selected based on an *evaluation function* for distinct communities in the Search Routing Table. The *evaluation function* is defined as follows:

$$w_1 = \frac{N_m}{h+1} \qquad w_2 = \frac{N_m}{h+2}, \text{where}$$

$w_{1,2}$: weight

N_m: Number of hops

To compare the value of w_1 for each record of the same community, we pick up the record with the highest w_1. If more than one record are picked up, compares the value of w_2. Search Router collects the best records for distinct communities by evaluation function. Then, the Search Router adds the selected records into its Forwarding Table. Whenever a record is added into a Forwarding Table, the value of hops must add one to represent the additional distance.

After Forwarding Table is constructed, Search Router sends Forwarding Table to its neighbors. To avoid Count-to-Infinity Problem, when a Search Router sends its Forwarding Table to a neighbor, all records in the Forwarding Table excluding the records that are provided by the neighbor will be sent. Whenever a Search Router receives a Forwarding Table from a neighbor, it will update its Search Routing Table (if necessary). There are two rules for updating Search Routing Table. The first rule is that if there is a record in the Forwarding Table whose community name, hops, and path are the same as a record in the Search Routing Table, the Search Router updates the corresponding information in the Search Routing Table. The other one rule is that for each record in the Forwarding Table, if there exists a record with the same community name and path (but the value of hops is not the same) in the Search Routing Table, makes an evaluation for the two records by applying evaluation function. If the record from Forwarding Table has the larger weight value, the Search Router updates its Search Routing Table by the record. When the procedure of updating Search Routing Table is finished, the Search Router will then reconstruct its Forwarding Table (if necessary).

Forwarding Tables are exchanged by Search Routers periodically, and the updating of Search Routing Table and Forwarding Table are done in the background repeatedly. When a Search Router receives a query, it first subtracts one from the TTL value in the query. If the query arrive this Search Router for the first time, the Search Router uses its

index to lookup the matching peers and updates the value of found results in the query. If the same query has been handled by this Search Router, the query will be discarded. If the number of found peers is not enough to satisfy the query and the TTL value is still grater than 0, the Search Router then selects a suitable neighbor to forward the query.

The procedure to judge which one is the suitable neighbor consists of two steps. The first step is to select the neighbors that have not been visited. The second step is to search the Search Routing Table by the community name, and pick out the matching records as candidates. If there does not exist any record whose community name matches the query one exactly, choose the record whose community name matches the query one partially. If more than one record are picked out, evaluates each record by evaluation function, and selects the record with the largest value. If there still have more than one record with the same value, select one in a random manner. If there is no candidate, randomly select one neighbor excluding the previous Search Router. After a decision is made, the Search Router adds itself into visited list from the query, and then forwards the query to the neighbor that has been chosen before. The pseudo code of search routing algorithm is shown in Figure 2.

```
TTL = TTL - 1;

if (IsHandled == true)
   discard the query :
else
{
   if (match peer > 0)
      found results += match peer;

   if (found results <  required && TTL > 0)
   {
      if (Exist match neighbors that are not in visited list)
         path = the suitable neighbor;
      else
         path = random choose one neighbor;

      attach found results and update visited list then forward the query to path;
   }
}
```

Figure 2. The forwarding algorithm.

Assume a network is composed as in Figure 3. S{a(30)} means that a Search Router named "S", and in Search Routing Table of S there is a community "a" with 30 members. Similarly, T{b(10)} means that a Search Router named "T", and in its Search Routing Table there is a community "b" with 10 members.

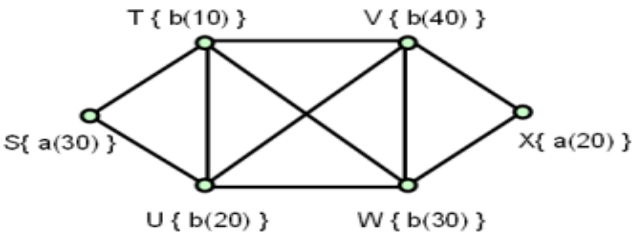

Figure 3. An example network

The initial Search Routing Tables and Forwarding Tables for each router are shown in Figure 4. After the first exchanging of Forwarding Tables, S received two Forwarding Tables form T and U, and then S updates its Search Routing Table and Forwarding Table.

Search Routing Table of Search Router S			
community name	# of member	hops	path
a	30	0	-
Forwarding Table of Search Router S			
a	30	1	S

Search Routing Table of Search Router V			
community name	# of member	hops	path
b	40	0	-
Forwarding Table of Search Router V			
b	40	1	V

Search Routing Table of Search Router T			
community name	# of member	hops	path
b	10	0	-
Forwarding Table of Search Router T			
b	10	1	T

Search Routing Table of Search Router W			
community name	# of member	hops	path
b	30	0	-
Forwarding Table of Search Router W			
b	30	1	W

Search Routing Table of Search Router U			
community name	# of member	hops	path
b	20	0	-
Forwarding Table of Search Router U			
b	20	1	U

Search Routing Table of Search Router X			
community name	# of member	hops	path
a	20	0	-
Forwarding Table of Search Router X			
a	20	1	X

Figure 4. Initial Search Routing Table and Forwarding Table and Forwarding Table.

The Search Routing Table after updating is shown in Figure 5. Because of the record from U has the larger value than T, the record form U is added to S's Forwarding Table. Other Search Routers update their Search Routing Table and Forwarding Table respectively.

Search Routing Table of Search Router S			
community name	# of member	hops	path
a	30	0	-
b	10	1	T
b	20	1	U
Forwarding Table of Search Router S			
a	30	1	S
b	20	2	S

Search Routing Table of Search Router U			
community name	# of member	hops	path
b	20	0	-
a	30	1	S
b	10	1	T
b	40	1	V
b	30	1	W
Forwarding Table of Search Router U			
a	30	2	U
b	40	2	U

Search Routing Table of Search Router W			
community name	# of member	hops	path
b	30	0	-
b	10	1	T
b	20	1	U
b	40	1	V
a	20	1	X
Forwarding Table of Search Router W			
a	20	2	W
b	30	1	W

Search Routing Table of Search Router T			
community name	# of member	hops	path
b	10	0	-
a	30	1	S
b	20	1	U
b	40	1	V
b	30	1	W
Forwarding Table of Search Router T			
a	30	2	T
b	40	2	T

Search Routing Table of Search Router V			
community name	# of member	hops	path
b	40	0	-
b	10	1	T
b	20	1	U
b	30	1	W
a	20	1	X
Forwarding Table of Search Router V			
a	20	2	V
b	40	1	V

Search Routing Table of Search Router X			
community name	# of member	hops	path
a	20	0	-
b	40	1	V
b	30	1	W
Forwarding Table of Search Router X			
a	20	1	X
b	40	2	X

Figure 5. Tables after first exchanging

After the second exchanging, Search Router's Search Routing Tables and Forwarding Tables are shown in Figure 6. Looking at Search Router S's table, S received T's Forwarding Table with a record (b, 40, 2, T), and received another record (b, 40, 2, U) form U. Following the updating rule of Search Routing Table, the record (b, 40, 2, T) has the same community name and path as the record (b, 10, 1, T) in the Search Routing Table, and the record (b, 40, 2, T) is a better one. Hence, the record (b, 10, 1, T) is replaced by (b, 40, 2, T), and the record (b, 20, 1, U) is also replaced by (b, 40, 2, U).

Since the Search Routing Table is modified, the Forwarding Table should be reconstructed again.

Search Routing Table of Search Router S

community name	# of member	hops	path
a	30	0	-
b	40	2	T
b	40	2	U
Forwarding Table of Search Router S			
A	30	1	S
b	40	3	S

Search Routing Table of Search Router U

community name	# of member	hops	path
b	20	0	-
a	30	1	S
a	30	2	T
b	40	2	T
a	20	2	V
b	40	1	V
a	20	2	W
b	30	1	W
Forwarding Table of Search Router U			
a	30	2	U
b	40	2	U

Search Routing Table of Search Router W

community name	# of member	hops	path
b	30	0	-
a	30	2	T
b	40	2	T
a	30	2	U
b	40	2	U
a	20	2	V
b	40	1	V
a	20	1	X
b	40	2	X
Forwarding Table of Search Router W			
a	20	2	W
b	30	1	W

Search Routing Table of Search Router T

community name	# of member	hops	path
b	10	0	-
a	30	1	S
b	20	2	S
a	30	2	U
b	20	1	U
a	20	2	V
b	40	1	V
a	20	2	W
b	30	1	W
Forwarding Table of Search Router T			
a	30	2	T
b	40	2	T

Search Routing Table of Search Router V

community name	# of member	hops	path
b	40	0	-
a	30	2	T
b	10	1	T
a	30	2	U
b	20	1	U
a	20	2	W
b	30	1	W
a	20	1	X
Forwarding Table of Search Router V			
a	20	2	V
b	40	1	V

Search Routing Table of Search Router X

community name	# of member	hops	path
a	20	0	-
b	40	1	V
b	30	1	W
Forwarding Table of Search Router X			
a	20	1	X
b	40	2	X

Figure 6. Table after the second exchanging

Assume a peer sends a query to Search Router T after Forwarding Tables are exchanged for two times, and the peer wants to find 50 members belong to community "a". The query will be routed, and the procedure is shown in Figure 7.

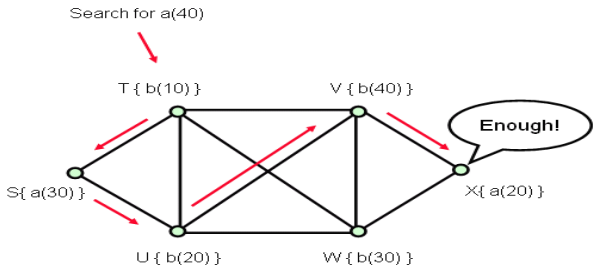

Figure 7. Search routing process.

When Search Router T received the query, it search is local registrations, and gets 0 results. Because the number of found results < 50 (result threshold), T has to forward the query. By the Search Routing Table of T, T forwards the query to S. Each Search Router who received the query does the same action as Search Router T and so on. Follow the search routing algorithm we proposed, the query will be forwarded form T to S, S to U, U to V, and V to X. Finally, found results are enough, so Search Router X stops to forward the query.

So far we have introduced the architecture and routed search forwarding algorithm. In PP-COSE, the system has two sources of overheads. One is that a Search Router must provide storages to keep Search Routing Table and Forwarding Table. The other is that the costs of Forwarding Table exchanging. The storage of Search Routing Table and Forwarding Table required is relatively small. Focus on costs (number of message that produced) of operation, we conclude that as long as the average degree is grater than or equal to 2, the routing forward algorithm has better performance than flooding algorithm. The details of algebraic analysis of three P2P forwarding algorithms are depicted in [8].

4. EXPERIMENTS

In this section, we present the experiments for the Community based P2P Search scheme. In order to verify the performance of design, we designed a testbed environment for experiments. Through the results of experiments, we prove that our design is efficient. This section is organized as following. Section 4.1 shows the experiment environment. Section 4.2 introduces the experiment parameters. Finally, we discuss each experiment and the results.

4.1. Experiment Environment

The experiments are conducted on PP-COSE testbed. There are three main components within the system (Environment Manager, Search Router, and Event Trigger). The most important one is "Environment Manager", which controls the whole system. There are four elements within the Environment Manager: topology manager, membership manager, result collector, and static global information. The topology manager is responsible for producing the topology of network (i.e. the connection state of Search Routers). The membership manager is responsible for generating the membership within each Search Router. The result collector is responsible for collecting experiment results. The static global information element integrates the information provided by topology manager and membership manager, and keeps the whole information of the system during operation.

The Search Router is a virtual device in our experiment environment, and it simulates the functions of a physical Search Router. There are two elements within the Search Router component. They are routing handler and query handler. The routing handler handles the actions of Forwarding Table exchange. We have presented the procedure of Forwarding Table exchanging in section 3.3. The query handler handles the queries sent by peers or forwarded by other Search Routers. In our design, the queries sent by peers are simulated by the query trigger within Event Trigger.

4.2. Experiment Parameters

In PP-COSE testbed environment, parameters are rough classified into environment parameters and query parameters. The environment parameters describe either for all Search Routers, or for single Search Router.

The Environment Parameters are summarized in Table 2 and are explained as follow:

Table 2: Environment Parameters.

Environment Parameters		
Parameter Name	**Base Value**	**EXP.**
Number of Search Routers (R)	50	-
Max. value of AVG_m (MAX_{avg})	15	-
Forwarding Table exchange period	10 (s)	EXP 2
Average degree (D)	3	EXP 3
% of Search Router with MAX_{avg}	20 (%)	EXP 4
For each Search Router		
Number of peers (N)	10000	-
Membership similarity	10-80-80 (%)	EXP 1 EXP 6
Member threshold (θ_m)	20 (%)	EXP 1
Query threshold (θ_q)	10 (%)	-

- In experiments environment, following values are assumed:
  - Number of Search Routers (R): 50.
  - Max. value of AVG_m (MAX_{avg}): 15 for all Search Routers.
  - Forwarding Table exchange period: 10 seconds.
  - Average degree (D): 3.
  - Percentage of Search Router with MAX_{avg}: This parameter means how many Search Routers whose average amount of membership is the maximum value (15). We set 20% to be the base value (i.e. there are 20% of Search Routers with the maximum value of average amount of membership).
- For each Search Router, following values are assumed:
  - Number of peers (N): 10000.
  - Membership similarity: For example, the membership similarity of the 3 level community "war3.game.tw" is 20-40-80. Assume the total amount of membership in the Search Router is M. The community "tw" has 0.8*M members, "game.tw" has 0.4*0.8*M = 0.32M members, and "war3.game.tw" has 0.2*0.32M = 0.064M members. Our experiments use 10-80-80 as the base value.
  - Member threshold (θ_m): 20%.
  - Query threshold (θ_q): 10%.

Query parameters are as follow:

- Number of queries: 1000.
- Query period: 200 ms.
- Result threshold: 500.
- Query similarity: If the number of queries is 1000 and the query similarity is 10%, there are 100 queries requesting the same community. 10% is set to be the base value.
- TTL: It controls the maximum number of times a query can be forward. 50 is set to be the base value.

4.3. Experiment Results

In this paper, we present selected results from 4 experiments.

◆ Experiment 1

As memtion in Section 3, the size of Search Routig Table can be controlled by filter function, and member threshold (θ_m) is the main player. In this experimnent, we want to obersver the influence of member threshold on diferent membership similarities.

Figure 8 shows the result by Search Routing Table size, we can see as member threshold increses, the size of Search Routing Table decreases. With different membership similarities the trends of decrease are not the same. When member threshold is 10%, 10-10-80 has the largest size. However, when member threshold is 15%, the size of 10-10-80 becomes smaller than 10-80-80 and 80-80-80.

◆Experimnet 2

In this experiment, we want to observe the influence of average degree on each forwarding algorithm. Figure 9 shows the result of this experiment. Because the flooding forwarding algorithm forwards each query to d_j-1 neighbors, query cost will increase as the degree increases. As the degree increases, the query cost of routing forwarding drops. This is because the larger degrees a Search Router has, more information from other Search Routers, and the most suitable neighbor to forward the query can be identified.

Random forwarding at average degree equals to 2 is more

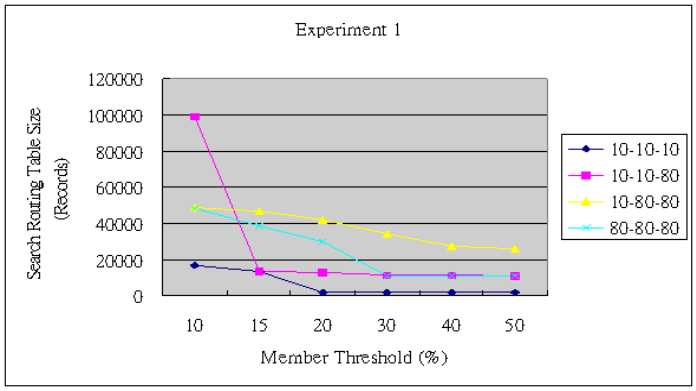

Figure 8. Member Threshold vs. Search Routing Table Size

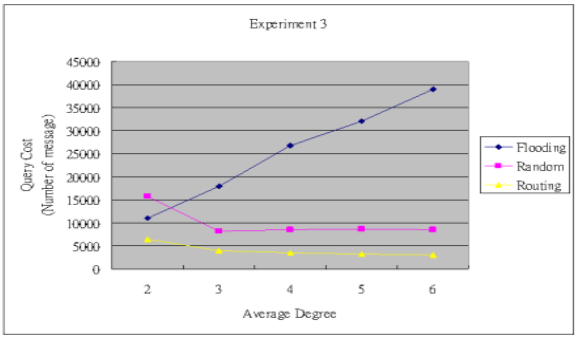

Figure 9. The influence of degree.

costly than flooding forwarding, but goes down at average degree equels to 3, and becomes smooth when average degree is larger than 3. When average degree is 2, some Search Router may has just one neighbor. A query which is forwarded by random forwarding will back to previous Search Router when the Search Router does not has another neighbor.

◆Experimnet 3

In this experiment, we want to see the performance for each forwarding alogrithm in different percentage of Search Router with MAXavg. Figure 10 shows the result. We can see that routing forwarding has the best performance in all situations.

◆Experimnet 4

In this experiment, we want to observe the influence of TTL on each forwarding algorithm. We call a query which cannot satisfy the result threshold as a fail query. Figure 10 shows the result of this experiment by the percentage of fail query, and Figure 11 shows the result of query cost. We can see that flooding forwarding has the best performance in small degree, but it is the most costly one. When TTL is larger than 15, routing forwarding can satisfy 90% of queries, and routing forwarding is the most efficient algorithm.

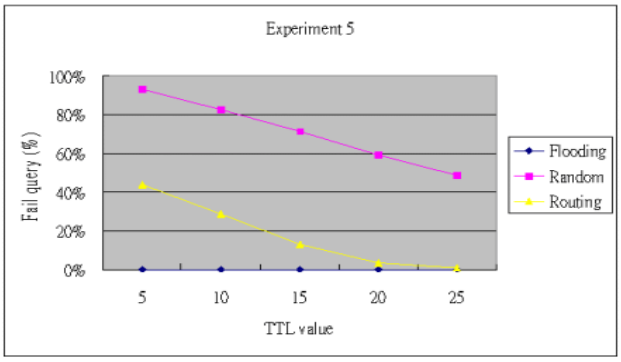

Figure 10. The result of experiment 5 by fail query.

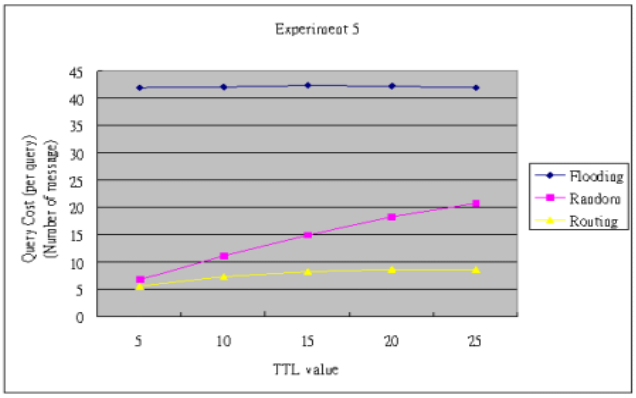

Figure 11. The result of experiment 5 by query cost.

5. CONCLUSION

In this paper, we proposed a PP-COSE scheme Community based P2P Search. This scheme provides an efficient way to find peers of specific community in P2P systems. PP-COSE can also be applied to generalized P2P systems. To manage communities, we use a community filtering function which can filter out the less important or unpopular communities. The filter function keeps the size of Search Routing Tables under control and reduces the resource wasting. By using the search routing algorithm, queries can be routed efficiently.

From experimental results in section 4, we conclude that routing forwarding is much more efficient than flooding forwarding and random forwarding in most cases. We also observe that the more concentration of membership similarity is, and routing forwarding has the better performance. In the best case, the routing forwarding saves up to 80% of query cost comparing with flooding forwarding, and up to 65% comparing with random forwarding.

There are still some works can be done in the future. The three levels representation of community format is limited. We are working on extending it to a generalized community format. Also, the evaluation function we used in section 3.3 is simple and straightforward. We may design a more suitable evaluation function in the future. We believe that PP-COSE scheme can be applied to current and future P2P systems, and makes the community search process more efficient.

6. REFERENCES

[1] Li Gong, "Peer-to-Peer Networks in Action," *IEEE Internet Computing*, Jan. 2002, pp. 37-39.

[2] M. Portmann, and A. Seneviratne, "The Cost of Application-Level Broadcast in a Fully Decentralized Peer-to-Peer Network," *Proc. of 7th International Symposium on Computers and Communications*, July 2002, pp. 941-946.

[3] M. Portmann, P. Sookavatana, S. Ardon, and A. Seneviratne, "The Cost of Peer Discovery and Searching in The Gnutella Peer-to-Peer File Sharing Protocol," *Proceedings of 9th IEEE International Conference on Networks*, Oct. 2001, pp. 263-268

[4] B. Krishnamurthy, J. Wang, and Y. Xie, "Early Measurements of a Cluster-based Architecture for P2P Systems," *ACM SIGCOMM Internet Measurement Workshop*, Nov. 2001, pp. 105-109

[5] B. Yang, and H. Garcia-Molina, "Improving Search in Peer-to-Peer Networks," *Proc. of 22nd International Conferenceon Distributed Computing Systems*, July 2002, pp. 5-14

[6] A. Crespo, and H. Garcia-Molina, "Routing Indices for Peer-to-Peer Systems," *Proc. of 22nd International Conference on Distributed Computing Systems*, July 2002, pp. 23-32

[7] H. Balkrishnan, M.F. Kaashoek, D. Karger, R. Morris, and I. Stoica, "Looking Up Data in P2P Systems," *Communication of The ACM* , Feb. 2003, Vol. 46, No. 2, pp. 43-48

[8]S. Chang, "Peer-to-Peer Search based on Community," *MS Thesis*, Computer Science and Information Engineering Department, Fu Jen Catholic University, June 2003.