An Ultra-Peer-Based Message Routing Architecture for Heterogeneous P2P File Sharing Networks*

Chung-Ming Huang
Department of Computer Science and Information Engineering
National Cheng Kung University
Taiwan,R.O.C.
huangcm@locust.csie.ncku.edu.tw

Tz-Heng Hsu
Department of Computer Science and Information Engineering
National Cheng Kung University
Taiwan, R.O.C.
hsuth@locust.csie.ncku.edu.tw

ABSTRACT

A P2P file sharing network provides a resource sharing platform for Internet users. To increase higher-degree resource sharing, heterogeneous P2P file sharing networks need a way to collaborate and communicate with each other. Based on the approach of interconnecting heterogeneous P2P file sharing networks, users on one P2P file sharing network can share resources and search data with other P2P file sharing networks. The main objective of our work is to enable connectivity and universal access among heterogeneous P2P file sharing networks. A novel architecture for P2P file sharing networks interconnection called Shoran is proposed to serve peers' requests and route peers' requests to other P2P file sharing networks. Shoran provides (i) a message routing mechanism to route query messages among different P2P file sharing networks and (ii) an uniform message format that can ease the message exchange among heterogeneous P2P file sharing networks.

Keywords

File Sharing Network, Ultra-Peer, Peer-to-Peer (P2P), XML, Resource Description Framework (RDF), Resource Sharing.

1. INTRODUCTION

Peer-to-peer (P2P) overlay networks have recently been realized through file sharing applications such as Napster [1], Gnutella [2], and Freenet [3]. In P2P overlay networks, computers can act as both clients and servers. P2P overlay networks have many interesting characteristics such as self-organization, distributed computing, scalability, and tolerating to network failures. Additionally, P2P overlay networks make resources, e.g., storages, CPU cycles, and media contents, available at the edges of the Internet.

The initial designs of previous P2P systems have several serious problems. For example, Napster uses a centralized directory approach, which makes it hard to scale and has vulnerable of service failure. Gnutella employs a flooding-based query mechanism. However, Gnutella's flooding-based approach limits the searching scope at some points. In order to retrieve decentralized resources, several peer-to-peer query protocols have been proposed [3, 4, 5]. However, few of them can efficiently service requests under the environment of unstable connectivity and unpredictable network congestion. The need for efficiently locating data and routing query mechanism makes several research groups to design a new generation of scalable P2P overlay networks. Most of them use the distributed hash table (DHT) techniques, including Pastry [6], Tapestry [7], Chord [8], and Content Addressable Networks (CAN) [9]. The distributed hash table (DHT) technique associates resources with a key (produced by hashing the resource name), and each peer in the P2P overlay networks is responsible for storing data and routing queries based on the given key.

P2P overlay networks can survive on a limited number of users, but P2P overlay networks that are associated with a limited number of users make them difficult to find and share resources. Without a resource exchange mechanism, users can only retrieve resources within one P2P overlay network with a limited number of users. For example, a user in the Gnutella network looks for songs containing the word "holiday" may get no response from the query. However, the song may exist on the other P2P file-sharing applications such as Morpheus [10], KaZaA [11], and iMesh [12]. As both the number and the variety of P2P overlay networks continue to increase, resources sharing between heterogeneous P2P overlay networks is becoming an important issue. That is, since file-sharing applications are increasing rapidly, how to collaborate and communicate in different P2P file-sharing applications becomes an important issue. Therefore, an architecture for resource sharing among heterogeneous P2P overlay networks is required.

The main objective of our work is to enable connectivity and universal access among heterogeneous P2P file-sharing networks. In order to achieve the resource sharing in heterogeneous P2P file-sharing networks, we propose a novel architecture named "Shoran". Shoran is a peer-to-peer network, and it provides (i) a message routing mechanism to route query messages among different P2P file sharing networks and (ii) a message exchanging mechanism that can ease the communication among heterogeneous file sharing applications.

The remaining part of this paper is organized as follows:

*This research is supported by the National Science Council of the Republic of China under the grant NSC 92-2213-E-006-025.

.

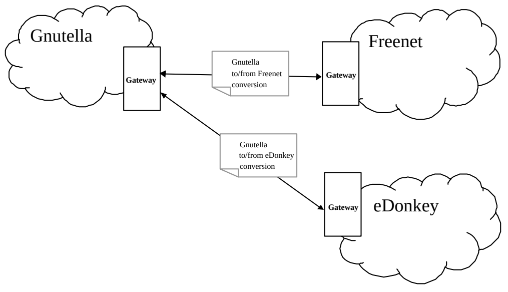
Gnutella
Gateway
Gnutella to/from Freenet conversion
Gateway
Freenet
Gnutella to/from eDonkey conversion
Gateway
eDonkey

Figure 1: Traditional gateway approach.

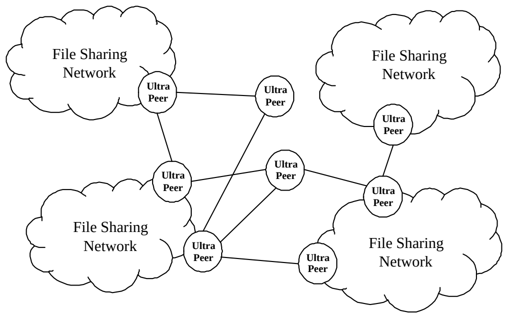

Figure 2: Shoran's abstract architecture.

Section 2 introduces some existing works of P2P overlay networks. Section 3 introduces the Shoran architecture. Section 4 discusses the design issues of the proposed architecture. Section 5 depicts the uniform message format (UMF). Section 6 presents experiment results and analysis of the proposed Shoran architecture. Section 7 has the conclusion remarks.

2. PRELIMINARY

The basic operation in existing DHT-based P2P overlay networks is $lookup(key)$, which returns an identifier of the peer that stores the resource with the given key. The DHT technique has proved to be an efficient mechanism for locating data and routing query on large distributed systems. Peers that use the DHT technique construct a P2P overlay network. In the P2P overlay network based on the DHT technique, each peer has several message routing information of other neighbor peers. When a $lookup(key)$ is issued, the lookup query is routed through the overlay network to the peer that is responsible for the given key. There are several different routing algorithms based on the DHT techniques [6, 7, 8, 9]. Based on the DHT technique, several projects were proposed to build distributed storage systems [13], application-layer multicast services [14], and event notification services [15].

Both flooding-based and DHT-based mechanisms provide solutions to solve the problems of locating data and routing query in P2P overlay networks. It is hard to figure out which approach is the best solution for locating data and routing query in P2P overlay networks. Each approach has its own benefits and drawbacks. In addition to the issues of locating data and routing query, there are several limitations in current P2P overlay networks.

Resource sharing among heterogeneous P2P overlay networks is an important issue. Morpheus, KaZaA, Limewire, and eDonkey are the currently most popular P2P file sharing applications. They are built on top of current IP network. These P2P file sharing applications provide file lookup services and resource retrieval services. Each application has its special design to satisfy users' requirements. However, these applications cannot share resource among each other. For example, a Limewire's user cannot access the resource that is hold by a eDonkey's user. Therefore, a new communication architecture is needed to overcome this problem.

One solution to solve this problem is to design a gateway for the transmission of both query messages and data over the P2P file sharing networks. In [16], Lui and Kwok proposed a framework to integrate various P2P file sharing protocols using a P2P gateway. For a given P2P overlay network, the gateway approach needs to provide multiple communication interfaces in order to connect with other P2P overlay networks. For example, a Gnutella gateway that wants to connect with Freenet and eDonkey needs to provide two communication interfaces for transferring query messages and resource data. Figure 1 shows the traditional architecture for communicating different P2P file sharing networks.

By connecting directly to different P2P file sharing networks, the gateway approach can seamlessly integrates into existing P2P file sharing applications. However, the lack of a standardized exchange agreement makes the complexity of message converting among heterogeneous resource sharing protocols. Developers need to write many programs to provide multiple communication interfaces for message converting among heterogeneous P2P file sharing networks. Besides, the gateway architecture embeds a centralized component into the P2P systems, which makes it be the bottleneck. Peers that use different P2P file sharing networks need to communicate through the gateways which make them unscalable.

3. SYSTEM ARCHITECTURE

The proposed Shoran method provides a convenient solution to simplify the design of message converting among heterogeneous P2P systems and applications. The goals of Shoran are to resolve the following concerns:

- **Simplicity:** Shoran provides a uniform message format to ease the communication among heterogeneous P2P file sharing networks. Developers can be free from hardship in converting messages among different message formats.
- **Availability:** Shoran makes resources (storages, CPU cycles, and media contents) available among heterogeneous P2P file sharing networks. The resources can be easily found among heterogeneous P2P file sharing networks.

In order to achieve the above goals, Shoran uses an ultra-peer topology for routing message among heterogeneous P2P file sharing applications. An ultra-peer provides routing service and data exchanging service for heterogeneous P2P file sharing applications. Several ultra-peers form an ultra-peer network for resource discovery among heterogeneous P2P file sharing networks. The ultra-peer is able to provide (i) a message routing mechanism to route the

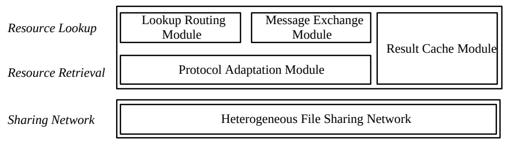

Figure 3: Shoran's ultra-peer components.

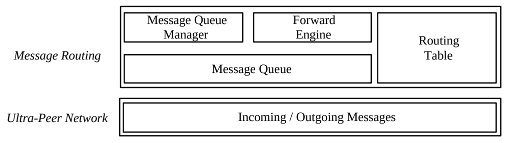

Figure 4: The abstract diagram of the lookup routing module.

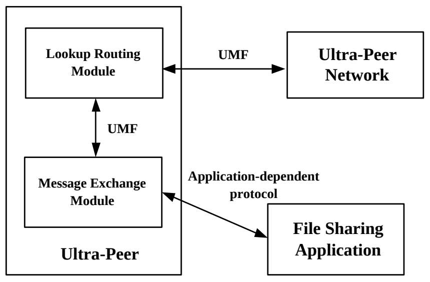

Figure 5: The abstract diagram of message exchange module.

query messages among heterogeneous P2P file sharing networks and (ii) a message exchanging mechanism that can ease the communication among heterogeneous P2P file sharing networks. Figure 2 depicts the abstract architecture of Shoran. Each ultra-peer consists of five components: (1) a lookup routing module that provides message routing service over the ultra-peer network, (2) a message exchange module that provides the message converting service for heterogeneous P2P file sharing networks, (3) a protocol adaptation module that provides resource retrieval service for heterogeneous P2P file sharing networks, (4) a result cache module that provides the cache service for query/ response message, and (5) a XML-based uniform message format (UMF) defines the communication protocol for heterogeneous P2P file sharing networks. Figure 3 depicts the abstract diagram of Shoran's ultra-peer components.

The lookup routing module is responsible for routing the messages that are based on the uniform message format (UMF) to/from different ultra-peers. The message exchange module is responsible for converting network dependent query messages to/from the UMF-formatted messages that the ultra-peer network can understand. The uniform message format (UMF) is used to define the network independent messages for finding resources that are stored in heterogeneous P2P file sharing networks. A standardized message format can simplify the complexity of managing multiple communication message format. Using the XML technique, each ultra-peer can convert network dependent query messages to/from UMF-formatted messages. In other words, it will be much easier for developers to add new P2P file sharing protocols using UMF.

In the remaining part of this Section, we introduce the modules of Shoran's ultra-peer in detail. The details of UMF are introduce in Section 5.

3.1 Lookup Routing Module

The lookup routing module consists of (i) a message queue that stores incoming messages, (ii) a message queue manager that controls and maintains state information of the message queue, (iii) a forward engine that performs message forwarding according to the message routing table, and (iv) a message routing table that holds ultra-peers' addresses. The ultra-peers' addresses are referenced by a set of indexes, which determine the corresponding address mapping from an incoming message to an outgoing message. The message routing table can be configured statically or dynamically. Figure 4 depicts the abstract architecture of the proposed lookup routing module.

The lookup routing module is responsible for routing messages in the ultra-peer network. Two main functions of the lookup routing module are (i) managing incoming messages and (ii) forwarding messages to the corresponding ultra-peer. When one ultra-peer sends a UMF-formatted query/response message to another ultra-peer, the lookup routing module scans the routing table for an address of other ultra-peer and forwards the message to the selected ultra-peer. When a selected ultra-peer receives a UMF-formatted message, it then converts the UMF-formatted messages and sends the converted messages to its underlying P2P file sharing network by using the message exchange module.

In order to avoid message flooding problem, the message routing module uses similar techniques of the Gnutella protocol to prevent loops and indefinite propagation in the ultra-peer network. If the lookup routing modules receives a query message, the forward engine forwards the incoming message to all ultra-peers that are connected with it except the ultra-peer from which it receives the message. By using the Time To Live (TTL) counter to limit the range of message forwarding, the indefinite propagation can be avoided. By using a Globally Unique Identifiers (GUIDs), an ultra-peer will not receive same message twice.

3.2 Message Exchange Module

The message exchange module consists (i) a message converter that performs the message conversion, (ii) a message state table that records the state information of the query message, and (iii) a message translation table that stores transition information for converting messages. These tables are used for processing ultra-peers' queries and responses.

Two main functions of the message exchange module are (i) converting each underlying P2P file sharing network's message format to/from the uniform message format (UMF) and (ii) providing protocol interoperability among various P2P file-sharing applications. The message exchange module handles all messages of its underlying P2P file sharing network. When the message exchange module receives query messages from its underlying P2P file sharing network, it converts these messages to the uniform message format (UMF). These UMF-formatted messages are routed through the lookup routing module, and then reach the destination ultra-peer. The destination ultra-peer then converts these UMF-formatted messages into the

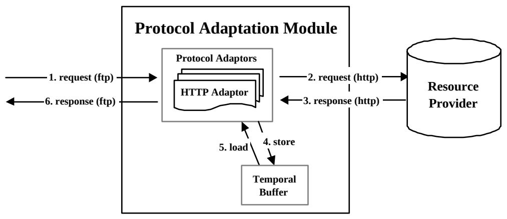

Figure 6: Steps involved in protocol adaptation among heterogeneous P2P file sharing networks.

message format that is understood by the destined P2P file sharing network. Figure 5 shows the abstract diagram of the message exchange module.

3.3 Protocol Adaptation Module

The protocol adaptation module consists of a set of protocol adaptors which offer data transfer services for heterogeneous P2P file sharing applications. The protocol adaptor provides the adaptation of one resource retrieval protocol to another resource retrieval protocol so that resources can be retrieved among different resource-sharing applications. In current P2P file sharing networks, there is a need to integrate various protocols for different P2P resource-sharing applications. For example, Gnutella uses HTTP as its data retrieval protocol, but Napster uses its own application dependent protocol as the data retrieval protocol. Without protocol adaptation, file-sharing applications cannot share data with each other.

Shoran supports the protocol adaptation among heterogeneous P2P file sharing networks. The protocol adaptor acts as a proxy module, which includes a temporal buffer for storing the resource data. Upon receiving a request from a peer, the message exchange module passes it to a selected protocol adaptor. If the resource provider (peer) is ready, the protocol adaptor sends a request to retrieve the data and then stores the data into a temporal buffer. After finishing the data retrieval, the protocol adaptor sends the data back to the peer that sends the request. The developers can write their own adaptors to integrate various protocols such as HTTP, FTP, TCP and application dependent protocols. Shoran provides access interfaces for exposing protocol adaptors' parameters, which enables the message exchange module to choose a suitable protocol adaptor for transforming data. Through the use of the protocol adaptor, different P2P resource-sharing applications are able to share resources among each other. Figure 6 depicts steps involved in protocol adaptation among heterogeneous P2P file sharing networks.

3.4 Result Cache Module

In Shoran, two data structures that each ultra-peer maintains are (i) a query result cache that contains recently query/response information and (ii) a data cache for processing data retrieval request. The query result cache is a list containing an entry for every unique query request received and processed. Each entry contains the following fields: a string stores the original query message, a query message identifier, a source identifier (requester's id), a destination identifier (resource provider's id), the time of a query message that was lastly used for searching resource, and the number of times that a query message is found in a result cache. A query result entry which has not been used for more than QueryLifeTime seconds is deleted.

The data cache is a list containing an entry for every unique data request. Each entry contains the following fields: a source identifier (requester's id), a destination identifier (resource provider's id), a session number of the data request, a source protocol description, a destination protocol description, a set of ultra-peers that cooperate to deliver the data, and a record stores status of the data request.

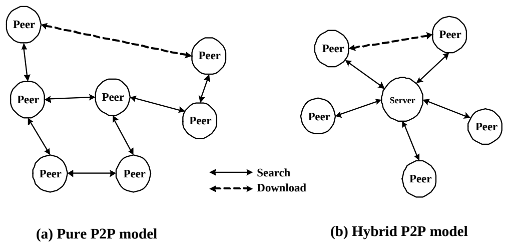

Figure 7: The relationships of ultra-peer network and heterogeneous file sharing networks.

4. DISCOVERY STRATEGIES

The architectures of current file sharing applications can be classified into two models: pure and hybrid. In the pure model, peers are responsible for resource discovery and no centralized server exists for resource lookup. Gnutella and Freenet are examples of the pure P2P model. In the hybrid model, one or more servers are responsible for resource discovery. Peers uses these servers to query resources and then retrieve resources using the peer-to-peer retrieval protocol. Napster, eDonkey, and iMesh are examples of the hybrid P2P model. Figure 7 shows the architectures of current file sharing applications.

In order to achieve resource sharing among heterogeneous P2P file sharing applications, Shoran uses various approaches to integrate ultra-peers with heterogeneous P2P file sharing networks. In the pure P2P model, an ultra-peer acts as a normal peer that cooperates with other peers in a file sharing network. When an ultra-peer receives a query from other peers, it forwards the query to both the ultra-peer network and the original file sharing network. From a normal peer's view, the ultra-peer is nothing different with other normal peers despites it forwarding message to/from ultra-peer network. In the hybrid P2P model, an ultra-peer acts as a server that provides resource lookup service for a file sharing network. When an ultra-peer receives a query message from peers, it forwards the query message to the ultra-peer network if it cannot find any result from its lookup service. Figure 8 shows the relationship of the ultra-network and file sharing networks.

For file sharing applications that uses the pure P2P model, the resource discovery mechanisms can be classified into two methods: flooding and distributed hash table (DHT). In flooding-based discovery methods, each peer receives query messages from other peers and then floods (broadcast) these messages to connected peers until the query is responded or a maximum number of flooding hops is reached. In DHT-based discovery methods, each peer is assigned a random ID and knows routing information of

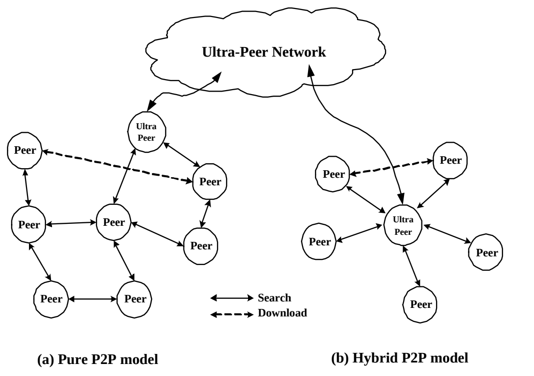

Figure 8: The architectures of current file sharing applications.

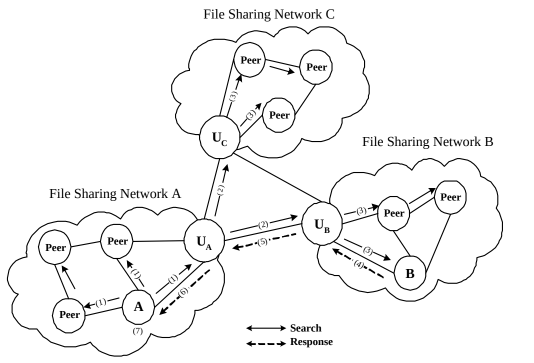

Figure 9: The query message flow in Shoran.

neighbor peers. The basic operation in DHT-based file sharing networks is $lookup(key)$, which returns an identifier of the peer that stores the resource with the given key. When a $lookup(key)$ is issued, the lookup query is routed through the file sharing network to the peer that is responsible for the given key.

Shoran is capable of sending requests from flooding-based P2P networks to networks of the same type or DHT-based P2P networks. However, Shoran does not allow requests from DHT-based networks to other P2P file sharing networks. It is due to the way DHT works: in a DHT network, an ultra-peer cannot see all requests despites the request that routes through it. Therefore, an ultra-peer is not able to forward all requests to the ultra-peer network. In order to ensure that the ultra-peer can catch the requests, the ultra-peer needs to know keys of all the outside files. From a practical view, it is hard to do it unless all peers in DHT networks are ultra-peers.

In Shoran, a message received by an ultra-peer is forwarded to all its neighbors, except the one from which message was received. Each message is in the UMF format and has a Time To Live (TTL) field. The TTL field is decremented by one at each visited ultra-peer. When the value of TTL reaches zero, the message is stop forwarding. Each message has a 32 byte identifier to uniquely identify it on the ultra-network. When a message is passed through an ultra-peer, the ultra-peer keeps the identifiers of messages. If an ultra-peer receives a message with the same identifier which it has received before, the message is ignored and is not forwarded. By checking the identifiers of messages, Shoran can prevent loops and indefinite propagation in ultra-peer network. Figure 9 depicts the query flow diagram of the proposed architecture. The query flow is explained as follows:

1. Peer A issues a searching request for finding a desired resource.
2. When the request is received by ultra-peer U_A, ultra-peer U_A translates the A-formatted message into the UMF format. ultra-peer U_A then broadcasts the UMF-formatted message to all connected ultra-peers.
3. When ultra-peer U_B and ultra-peer U_C receive the UMF-formatted messages, they convert the UMF-formatted messages into their own query message format. After converting the UMF-formatted messages, ultra-peer U_B and ultra-peer U_C forwards the B-formatted and C-formatted messages to peers within its internal file sharing network.
4. When peer B receives the B-formatted query message, it checks the resource list to find the desired resource. Peer B responses a resource-found message to ultra-peer U_B if it has the resource that Peer A is required.
5. When the ultra-peer U_B receives the B-formatted response message, it converts the message to the UMF format and then sends the UMF-formatted message to ultra-peer U_A.
6. When ultra-peer U_A receives the UMF-formatted response message, it converts the UMF-formatted message to its own A-formatted message and then sends the message to the original peer A.
7. Peer A receives the A-formatted response message and is ready for resource retrieving.

In some cases, queries may not be answered by ultra-peer U_B due to network congestion or system failure. In order to solve such problems, ultra-peer U_A has a timeout policy. When the timeout has elapsed and no response is sent from ultra-peer U_B or ultra-peer U_C, ultra-peer U_A sends a response message that indicates no resource found for peer A.

5. UNIFORM MESSAGE FORMAT

Most of currently existing P2P file sharing applications use application dependent discovery format for resource searching. Some of them just support filename matching using a simple query string. For example, Gnutella and Napster share resources by matching user specified query string. For P2P file-sharing applications, searching files by matching string may be enough. However, it restricts these applications to extend their systems to share other types of resources. The lack of a standard way for describing resources makes resource discovery more and more difficult among heterogeneous P2P file sharing applications.

In order to solve the problem of resource discovery in different P2P applications, we proposed an XML-based Uniform Message Format (UMF) for the resource discovery in P2P file sharing networks. The UMF consists of a query request and a query response. We adopt the Resource Description Framework (RDF) [17] as the resource description part of UMF. In this Section, we introduce the Resource Description Framework (RDF) at first and then introduce the components of UMF.

5.1 Resource Description Framework

Resource Description Framework (RDF) is developed by the W3C for Web-based metadata [17]. RDF uses XML as an interchange syntax for resource description. RDF's essential aim is to make work easier for finding resources precisely in the WWW space [18]. In RDF, resources are represented by Uniform Resource Identifiers (URIs). The advantage of RDF is that it can extend the resource description to the format that machines are readable. A search engine can use standard cataloging metadata to find the resource precisely. For example, the Dublin Core specification for library-like metadata is useful in describing the web resources [19]. Figure 10 depicts an example of a book description that uses the Dublin Core specification.

```
<?xml version="1.0" encoding="UTF-8" ?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:dc="http://purl.org/dc/elements/1.1/">
  <rdf:Description>
    <dc:title>The Lord of the Rings</dc:title>
    <dc:creator>J. R. R. Tolkien</dc:creator>
    <dc:format>Book</dc:format>
    <dc:identifier>ISBN 0618260587</dc:identifier>
  </rdf:Description>
</rdf:RDF>
```

Figure 10: An example of a book description that uses the Dublin Core specification.

The Dublin Core elements are started by the namespace name dc. The namespace gives the semantics required by particular types of resources. In the example depicted in Figure 10, a book description of ”The Lord of the Rings” is given. A variety of namespaces can be depicted using RDF. In the book space, an additional definition may be the ISBN number; in the movie space, it may be the digital video identifier.

The uniform message format (UMF) can be a uniform query exchange mechanism for P2P file sharing networks. The UMF uses the RDF as the standardized query format for resource description. An XML schema defines the terms that will be used in UMF statements and gives specific meanings to them. A variety of schema forms can be adopted using UMF. An example is given in Figure 11.

```
<?xml version="1.0" encoding="UTF-8" ?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:dc="http://purl.org/dc/elements/1.1/"
         xmlns:mv="http://p2p.example.com/movies/0.1">
  <rdf:Description about="http://video.example.com/
                          two_towers.mov">
     <dc:title>The Lord of the Rings: The Two Towers<
     /dc:title>
     <dc:creator>J. R. R. Tolkien</dc:creator>
     <dc:format>Movie</dc:format>
     <mv:director>Peter Jackson</mv:director>
     <mv:producer>New Line Cinema </mv:producer>
  </rdf:Description>
</rdf:RDF>
```

Figure 11: An example of a resource description in P2P file sharing networks.

In Figure 11, the namespace ”http://p2p.example.com// movies/0.1” provides the movie information schema in P2P file sharing networks. The $Description$ element indicates the subject of the enclosed statements. Attribute $about$ in the $Description$ element points to an external resource ”http://video.example.com/two_tower s.mov”. The remaining statements give the detailed description of the movie ”The Lord of the Rings: The Two Towers”.

When a peer sends a query to find the movie ”The Lord of the Rings: The Two Towers”, the peer receives a RDF description about the movie and then can retrieve the movie according to the $about$ attribute in the $Description$ element.

5.2 Query Message

In UMF, the query message can be considered as a resource that is described using the RDF syntax. The query message is contained within the envelope $< query >$ ... $< /query >$. Query messages are uniquely identified by UUID [20]. The original query protocol is specified in $< protocol >$ ... $< /protocol >$ tags. The query time is specified in $< date >$ ... $< /date >$ tags. The detail information of query data is embedded in the the envelope $< desc >$ ... $< /desc >$, and is described using the RDF description. When an ultra-peer receives a query message from its own P2P file sharing network, it converts its own query message into UMF and fill the information into the corresponding fields. An example is depicted in Figure 12.

```
<?xml version="1.0" encoding="UTF-8" ?>
<query id="5a389ad2-22dd-11d1-aa77-002035b29192">
  <date>Tue, 17 Dec 2002 17:40:11 +0800</date>
  <protocol>Gnutella</protocol>
  <desc>
    <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:dc="http://purl.org/dc/elements/1.1/">
      <rdf:Description>
         <dc:title>The Two Towers</dc:title>
      </rdf:Description>
    </rdf:RDF>
  </desc>
</query>
```

Figure 12: An example of the query message specified in UMF.

In Figure 12, a peer wants to query the movie ”The Two Towers”. The peer uses a file-sharing application based on the Gnutella protocol to send the query string. When the ultra-peer receives the query string, it then converts this query string into the corresponding fields in the UMF format. After converting the query string, the ultra-peer sends the UMF-formatted message to other ultra-peers. When an ultra-peer receives the UMF-formatted message, it then forwards the UMF-formatted message to other ultra-peers for resource discovery. When the destined ultra-peer receives the UMF-formatted message, it converts the UMF-formatted message to its own query format. The destined ultra-peer then sends this query message to find the resource.

5.3 Response Message

The response message is contained within the envelope $< res-ponse >$ ...$< /response >$. The response messages are also uniquely identified by UUID [20]. The response protocol is specified in $< protocol >$ ... $< /protocol >$ tags. The response time is specified in $< date >$ ... $< /date >$ tags. The response data is a RDF description and is embedded in $< desc >$ ... $< /desc >$ tags. When the ultra-peer receives a response message from its own P2P file sharing network, it converts the response message into the UMF format and fills the information into the corresponding fields. An example is depicted in Figure 13.

```
<?xml version="1.0" encoding="UTF-8" ?>
<response id="5a389ad2-22dd-11d1-aa77-002035b29192">
  <date>Tue, 17 Dec 2002 17:41:07 +0800</date>
  <protocol>Jxta</protocol>
  <desc>
    <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
           xmlns:dc="http://purl.org/dc/elements/1.1/"
           xmlns:mv="http://p2p.example.com/movies/0.1">
      <rdf:Description about="http://video.example.com/
                              two_towers.mov">
         <dc:title>The Lord of the Rings: The Two Towers
         </dc:title>
         <dc:creator>J. R. R. Tolkien</dc:creator>
         <dc:format>Movie</dc:format>
         <mv:director>Peter Jackson</mv:director>
         <mv:producer>New Line Cinema </mv:producer>
      </rdf:Description>
    </rdf:RDF>
  </desc>
</response>
```

Figure 13: An example of the response message specified in UMF.

In Figure 13, a peer that uses a file-sharing application based on the Jxta protocol has the movie ”The Two Towers”. The file-sharing application can give the detail information about the movie ”The Two Towers”. This peer gives the response message to the ultra-peer. When the ultra-peer receives the response message, it then converts the movie information into the corresponding fields in the UMF format. After converting the response message, the ultra-peer sends the UMF-formatted response message to other ultra-peers. Then an ultra-peer forwards the UMF-formatted message to the original ultra-peer. The original ultra-peer converts the UMF-formatted message to the response message format of Gnutella and then sends the Gnutella-formatted message to the original peer. When the original peer receives the response message, it sends a request to the peer for retrieving the movie ”The Two Towers”.

6. PERFORMANCE EVALUATION

In order to evaluate Shoran, we have an evaluation test on a pre-configured network environment. We used two P2P file-sharing applications to evaluate the performance of Shoran. The first one is an open source P2P file-sharing application called Limewire [21]. LimeWire is running on the Gnutella protocol. Limewire uses a cache schema to increase the search performance. The second one is a P2P file-sharing application called eDonkey [22]. eDonkey is running on its own protocol. The most important feature of eDonkey is the possibility to download the same file from several peers at the same time. These two file-sharing systems are popular in the current P2P file-sharing community. There are several open source projects that have developed tools for these two applications.

We performed experiments on computers with 800MHz Pentium III processors, 256MB RAM, and 30GB disk running on Windows2000. Figure 14 depicts the network configuration in the file sharing network's view. A peer P_1 is located in the network domain of Taiwan Academic Network (TANET). Peer P_1 is running on the Gnutella protocol and is used to evaluate the performance of the proposed architecture. Three ultra-peers are located in the same LAN with 100Mbits/s Ethernet connection. Each ultra-peer connects to a different P2P file sharing network.

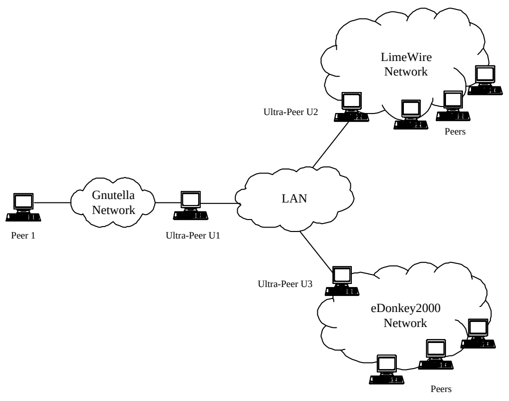

Figure 14: The network configuration of the evaluation test.

Ultra-peer U_1 connects with the test peer P_1, ultra-peer U_2 connects with Limewire network, and ultra-peer U_3 connects with the eDonkey2000 network. In order to simplify the testing and get a precise performance result, Peer P_1 does not join any Gnutella network and only connects to ultra-peer U_1 by using the Gnutella protocol. The experiments are designed to measure the following performance factors:

- *Query Hit Ratio (QHR)*: It is the response ratio that a peer can find the desired resource.

$$QHR = \frac{Response_f}{Query_s}$$

  where $Query_s$ denotes the number of queries that are sent to find desired resources, $Response_f$ refers to the number of responses that the resources are found. QHR is a good indicator of the possibility of finding a resource among heterogeneous P2P file sharing networks.

- *Average Response Time*: It is defined as the interval between the times that a peer sends a query to find the desired resource and then receives the response from P2P file sharing networks. The average response time to query resource measures the time spent in finding the desired resource.

In order to evaluate the performance of the proposed architecture, we generate queries by picking a query string from a dictionary file. The test dictionary file contains 5000 frequent used words in a free on-line lexicon and encyclopedia database [23]. Three traces were collected independently at ultra-peer S_1, S_2, and S_3. Each entry in the trace contains the following fields: (1) Protocol Identifier, (2) Message ID, (3) Query String, (4) Response Time, (5) Number of Results Found, and (6) File Names of Query Results. The trace was collected from March 01, 2003 through March 03, 2003, approximately three days. Table 1 depicts the network status of the Limewire and eDonkey2000 networks when the experiments were triggered.

We observed that Limewire has more participant peers than eDonkey2000 has. Because the Limewire network uses super-peer and caching concept to enhance the Gnutella

File sharing Network	Protocol	#Participant Peers	#Sharing Files
Limewire	Gnutella	23,121	402,743
eDonkey2000	eDonaky	11,272	918,293

Table 1: Status of the file sharing network.

	MP3	MPG	JPG	GIF	ZIP	EXE	AVI
Limewire	13,668	1,675	525	519	451	378	227
eDonkey2000	31,063	4,479	985	869	698	547	535

Table 2: The number of shared files in different media formats.

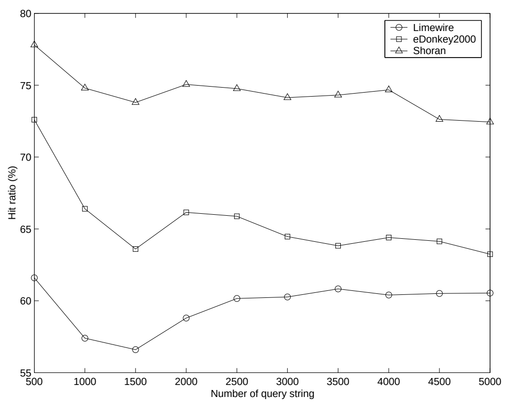

Figure 15: Query hit ratio of the experiment result.

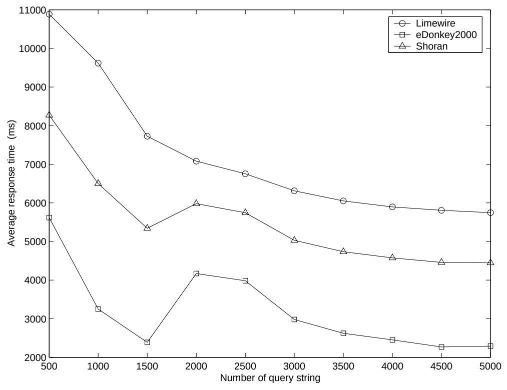

Figure 16: Average response time of the experiment result.

protocol, the flooding-based approach doesn't limit Limewire to increase its scalability. The eDonkey2000 has less participant peers than Limewire has. It is because the eDonkey network uses a server-like approach for quick searching, which limits its scalability. The number of participant peers is limit by the server's computing capacity. However, eDonkey has no official server, servers can be set at any IP address. Each eDonkey's server maintains a list of other working servers. When peers connect to a server, each peer is given a server list that contains other known servers. When a peer finds a connected server is failed, it can connect to another server according to the server list. The eDonkey's approach makes peers be able to always find a candidate server to searching files, which makes it more reliable than napster's approach.

For the lifetime measurements, we receive 22,781 responses from Limewire network and 53,063 responses from eDonkey network. Table 2 shows the media distribution in the trace results. In Table 2, it shows that a significant amount of shared files is in audio and video formats.

Figure 15 shows the query hit ratio in the trace results. The Shoran has better query hit ratio than Limewire and eDonkey. It is because the Shoran can send queries to several different P2P file sharing networks. The Limwire's and eDonkey's users can only search resources within their networks. Therefore, by connecting more and more P2P file sharing networks, the possibility of finding desired resources is improved. Figure 15 also shows that the eDonkey has better query hit ratio than Limewire. The reason is that the eDonkey has more shared files than Limewire has. The scalability of P2P file sharing networks is an important factor that affects the query hit ratio of finding resources.

Figure 16 shows the average response time in the trace results. The eDonkey has a better average response time than Shoran and Limewire. It is because that eDonkey uses a server-like approach, which makes it have a quick searching and response time. Shoran has a better average response time than Limewire. It shows that Shoran can reach a balance in average response time. The Limewire has the worst average response time. Since queries are flooded in Limewire, the forwarding operations increase the response times of queries.

7. CONCLUSION

Comparing with related works proposed in the literature, our work focuses on the resources retrieval in the heterogeneous P2P file sharing networks environment. Different P2P file sharing networks may use different resource retrieval protocols. Therefore, peers need a mechanism to retrieve resources among heterogeneous P2P file sharing networks. A resource retrieval scheme is proposed in our paper. The peers can retrieve resources with the help of ultra-peers. The ultra-peers consists a set of protocol adaptors, which provide the adaptation of one resource retrieval protocol to the other resource retrieval protocol. Our future work includes the in-depth analysis of the relationships of heterogeneous P2P file sharing networks, and the study of other more advanced and useful resource discovery mechanisms to fast up the speed of finding desired resources.

8. REFERENCES

[1] Napster, http://www.napster.com.

[2] Gnutella, http://gnutella.wego.com.

[3] I. Clarke, O. Sandberg, B. Wiley, and T. W. Hongang, "Freenet: A Distributed Anonymous Information Storage and Retrieval System," *Lecture Notes in Computer Science*, VOL. 2009, pp. 46–66, July 2000.

[4] R. Matei, A. Iamnitchi, and P. Foster, "Mapping the Gnutella Network," *IEEE Internet Computing*, VOL. 6, NO. 1, pp. 50–57, January 2001.

[5] L. Gong, "JXTA: A Network Programming Environment," *IEEE Internet Computing*, VOL. 5, NO. 3, pp. 88–95, May 2001.

[6] A. Rowstron and P. Druschel, "Pastry: Scalable, Distributed Object Location and Routing for Large-Scale Peer-to-Peer Systems," *IFIP/ACM*