

# **ANONYMOUS FILE-SHARING IN LOCAL P2P NETWORKS**

Submitted in Partial Fulfilment of the Requirements

for the Award of the Degree of

**Bachelor of Technology**

by

**D.Nitish Rao - UG107114**

**C.Tejaswini - UG107113**

**Vandana Rani - UG107155**

**B.Rajesh Nayak- UG107111**

Under the guidance of

**Dr. K.Ramesh**

Associate Professor



Computer Science and Engineering

**NATIONAL INSTITUTE OF TECHNOLOGY**

**WARANGAL**

**2013 – 2014**

## **PROJECT APPROVAL**

This is to certify that the project work entitled “**ANONYMOUS FILE SHARING IN LOCAL P2P NETWORKS**” is a bonafide work under research by D.Nitish Rao (UG107114), C.Tejaswini (UG107113), Vandana Rani (UG107155) and B.Rajesh Nayak(UG107111) in partial fulfillment of the requirements for the award of the degree of **BACHELOR OF TECHNOLOGY** in **COMPUTER SCIENCE AND ENGINEERING** from **NATIONAL INSTITUTE OF TECHNOLOGY, WARANGAL** during the year **2013-2014**.

Examiners

**Dr.S.G.Sanjeevi**

**Sri T.Ramakrishnudu**

**Dr. Rashmi Ranjan Rout**

Supervisor

**Dr. K. Ramesh**

Head of the Department

**Dr. K. Ramesh**

Date:

## **DECLARATION**

We declare that this written submission represents our ideas in our own words and where others ideas or words have been included, we have adequately cited and referenced the original sources. We also declare that we have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in our submission. We understand that any violation of the above will be cause for disciplinary action of the institute of and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

D.Nitish Rao

UG107114

C.Tejaswini

UG107113

Vandana Rani

UG107155

B.Rajesh Nayak

UG107111

Date: \_\_\_\_\_

## **ABSTRACT**

This project presents an anonymous peer-to-peer (P2P) file sharing system. A peer-to-peer (P2P) network is a type of decentralized and distributed network architecture in which individual nodes in the network act as both suppliers and consumers of resources. If such networks are used for file-sharing, then it is P2P File-Sharing Network. A key weakness of most existing systems is the lack of anonymity. Without anonymity, it is possible to identify the participants involved. First, anonymous peer to peer system should guarantee that only the content receiver, knows the content. Second, anonymous peer to peer system should allow the content publisher to reasonably deny that the content originated from him or her.

Here, we present our design of such Anonymous P2P Networks, Protocols used and the implementation details.

## **ACKNOWLEDGEMENT**

We wish to express our gratitude to our beloved guide Dr. K. Ramesh, Associate Professor, Department of Computer Science and Engineering, National Institute of Technology, Warangal for his valuable guidance in the Project. We are indebted to him for helping us at every stage for progress of the project. His knowledge and eye for detail have helped us to learn and produce more in the given time duration.

Also, we wish to thank all those who have been directly or indirectly involved in the guidance/help for our project work.

D.Nitish Rao - UG107114

C.Tejaswini - UG107113

Vandana Rani - UG107155

B. Rajesh Nayak - UG107111

## **TABLE OF CONTENTS**

1	Introduction . . . . .	9
1.1	P2P Network Overview . . . . .	10
1.2	Architecture of Existing P2P File Sharing Systems . . . . .	10
1.3	The Centralized Model of P2P File-Sharing . . . . .	11
1.4	The Decentralized Model of P2P File-Sharing . . . . .	11
1.5	Comparison of the two Models . . . . .	12
1.6	File Sharing in P2P Networks . . . . .	12
1.7	Anonymity Issues in P2P File Sharing. . . . .	13
1.8	RSA Public-Key CryptoSystem . . . . .	14
1.9	AES Symmetric Key Encryption . . . . .	14
2.	Problem statement and solution . . . . .	15
2.1	Problem statement . . . . .	16
2.2	How do decentralised networks work? . . . . .	16
2.3	How the RIAA finds people to sue? . . . . .	18
2.4	The key privacy weakness. . . . .	19
2.5	Solution . . . . .	20
2.6	Why maintain anonymity? . . . . .	21

3	Protocol Design. . . . .	23
3.1	Virtual topology formation. . . . .	24
3.2	Why to select two nodes? . . . . .	24
3.3	Link Check . . . . .	25
3.4	Hop count Query . . . . .	25
3.5	Avoiding Broadcast storm . . . . .	26
3.6	Nodes leaving the Network . . . . .	27
3.7	Optimization . . . . .	28
4	File-sharing Protocol . . . . .	29
4.1	Search Query execution . . . . .	30
4.2	Nodes leaving during search query. . . . .	32
4.3	Download Query execution . . . . .	33
4.4	Relay node leaving during data transfer . . . . .	34
4.5	Advantages and disadvantages . . . . .	35
4.6	Summary of the Protocol . . . . .	37
5	Results. . . . .	38
6	Conclusion . . . . .	40
7	References. . . . .	42

## **LIST OF FIGURES:**

1.	A decentralized network	fig 2.1.....17
2.	Sending a file in Decentralized Network	fig 2.2.....19
3.	RIAA in the Network	fig 2.3.....20
4.	Topology Formation	fig 3.1.....26
5.	Hop Count	fig 3.2.....28
6.	Avoiding Broadcast storm	fig 3.3.....29
7.	Disconnection and New link formation	fig 3.4.....31
8.	Search query execution	fig 4.1.....34
9.	How relay node works	fig 4.2.....35
10.	Download query execution	fig 4.3.....36



# **CHAPTER-1**

## **INTRODUCTION**

## **1. INTRODUCTION:**

This section provides a brief overview of P2P Network. It gives an explanation of the necessary background information that is required to understand the project. The aim of the project is to develop an Anonymous Peer-to-Peer File sharing system.

### **1.1 P2P Network Overview:**

Peer-to-peer file sharing applications have been in existence for some time now. This is much different from traditional client/server systems, where files would lie on one central machine, and all transfers would occur only between the individual clients and that machine.

### **1.2 Architecture of Existing Peer-to-Peer File Sharing Systems**

Peer-to-Peer (P2P) systems are distributed systems which do not have any centralized control or hierarchical organization and each node is equivalent in functionality. Few of the familiar peer-to-peer file-sharing network applications are Napster, Gnutella, Freenet, Morpheus. Peer-to-peer file sharing applications can have two main models:

- Peer-to-Peer system with centralized servers such as that used by Napster

Network functionality depends largely on a central server and each peer accesses the central server to upload and download information. The central server coordinates the communication among peers

- Pure decentralized peer-to-peer system such as Gnutella and Freenet.

There is no central server and peers are normally organized in an unstructured fashion. Each peer can only communicate directly with a few neighbor peers; However a peer can communicate with the rest of network through hop-by-hop message propagation

One of the advantages of decentralized networks is that they do not have any single point of failure. Another benefit is that these networks are highly scalable.

### **1.3 The Centralized Model of P2P File-Sharing:**

In this model, a central server or a cluster of central servers directs the traffic between individual registered peers. The file transfer is pure P2P while the file search is client-server. The central servers maintain directories of the shared files stored at each registered peer of the current network. Every time a user logs on or off the network, the directories at the central servers are updated to include or remove the files shared by the user. A user logs on the network by connecting with one of the central servers. Each time a user wants a particular file, it sends a request to the central server to which it is connected. The central server will search its database of files shared by peers who are currently connected to the network, and creates a list of files matching the search criteria. The resulted list will be sent to the user. The user can then select the desired file from the list. The file is directly transferred from one peer to another peer. The central server only holds the information of the shared file but not the file itself. There is no need to query individual users to discover a file. Every user has to register with the central server to be on the network. Thus the central index will include all files shared in the system, and a search request at the central server will be matched with all files shared by all logged-on users.

### **1.4 The Decentralized Model of P2P File-Sharing:**

In a decentralized P2P file-sharing model, peers have the same capability and responsibility. The communication between peers is symmetric in the sense that each peer acts both as a client and a server and there is no master-slave relationship among peers. At any given point in time, a node can act as a server to the nodes that are downloading files from it and as a client to the nodes that it is downloading files from. The software running at each node includes both the server and client functionality.

Unlike a centralized P2P file sharing system, the decentralized network does not use a central server to keep track of all shared files in the network. Index of shared files is stored locally among all peers. To find a shared file in a decentralized file sharing network, a user asks its friends (nodes to which it is connected), who, in turn, asks their friends for directory information.

## **1.5 Comparison of Centralized and Decentralized P2P Systems:**

In a centralized model, the file indexing information is kept at central server. At the same time, the central server is the only point of entry for peers in the system. The central server itself can become a bottleneck to the system. A central server failure can lead to the collapse of the whole network. Information provided by the central server might also be out of date because the database at the central server is only updated periodically. To store information about all files shared in the system also requires a significant amount of storage space and a powerful processor at the central server to handle all the file search requests in the system. When central servers are overloaded, file search in the system can become very slow and some peers might not be able to connect to the network due to the central server's limited capability.

In a decentralized model, the responsibility for file discovery is shared among peers. If one or more peers go down, search requests can still be passed along other peers. There is no single point failure that can bring the network down. Therefore, a decentralized P2P system is more robust compared with a centralized P2P system. At the same time, since indexing information is kept locally at individual user's computer, file discovery requires search through the network. File discovery in a decentralized P2P system can become inefficient and produce a large amount of network traffic.

## **1.6 File Sharing on P2P Networks:**

Most P2P file sharing applications publish file content along with a key. Usually the key is a hash value of the filename or file description, and it is used by queries to search for the published file. Each file is split into small chunks and the chunks are then distributed to other peers. Dividing a file into chunks makes it possible for content receivers to speed up the retrieval process by retrieving various parts of the file from multiple peers simultaneously.

Frequently, file querying is supported by a keyword search engine. Keywords usually consist of filenames or file descriptions. Queries are hashed by query senders so they are not

known to intermediate peers during transmission. However, hashed queries only provide limited protection against eavesdropping since an eavesdropper is still able to identify the query content by computing a dictionary of hash values, provided he or she has enough computational power.

## **1.7 Anonymity Issues in P2P File Sharing:**

Recently there has been an increasing interest in anonymous P2P file sharing. The reason is due to concerns regarding freedom of speech. Another obvious reason is that anonymity appeals to those who want to share copyrighted files. As a result, more and more P2P file sharing systems are starting to provide anonymity features.

There are three basic roles involved inside file sharing process: publisher, sender, and receiver.

A publisher is a peer that publishes files in the system. In addition to a publisher, an anonymous P2P file sharing system also includes receivers and senders. A receiver is a peer that receives published files from the system while a sender is a peer that can send published files to the receiver. The goal of receiver and sender anonymity is to guarantee that a message cannot be traced back to the real sender or receiver. Many systems achieve receiver and sender anonymity by having messages go through a number of intermediary peers to form a path.

Even if a P2P file sharing system provides anonymity for all the participants, an adversary can still exploit the system via traffic analysis. Therefore, a truly anonymous system must also be able to thwart traffic analysis attacks. The most common traffic analysis attacks include colluding peers, timing analysis, and eavesdropping.

One way to attack the system is to populate a target peer's neighbor list with colluding peers. The target peer is then exposed since messages to or from it can be easily detected by a colluder. However, in reality this kind of attack is not feasible since it requires a huge number of colluding peers to pollute the neighbor list .

An adversary may also attempt to identify a peer as a message initiator by analyzing round trip time (RTT) delays. Peers are exposed by immediately responding to a message since RTT is larger for more distant peers. To deter this kind of timing attacks, some P2P systems delay responses for some random interval to blind them with other messages including bogus messages .

An eavesdropper is an adversary who can monitor traffic to or from peers in a P2P network. Normally an eavesdropper will only be able to intercept communications over a small fraction of a network. Most P2P systems use secure communication channels (e.g., SSL or HTTPS) to prevent an eavesdropper from determining message contents.

## **1.8 RSA Public-Key CryptoSystem:**

RSA is a popular public-key cryptosystem. In RSA, data is encrypted with a public key and decrypted using a private key. The private key is kept secret and public key is generally available to the public. The key pair has a mathematical relationship so that if data is encrypted with the public key then it can be decrypted with private key and vice versa. Because RSA requires 100-1000 times more computation than symmetric cryptosystems, it is not suitable for bulk encryption and decryption. Instead, RSA is widely used to distribute keys for symmetric cryptosystems.

## **1.9 AES Symmetric-Key Encryption:**

Advanced Encryption Standard is the most widely used Symmetric-Key Encryption Scheme. It is used in TLS(Transport Layer Security) protocol. It is a block-cipher based symmetric-key encryption scheme. The key must be securely exchanged between the peers. The key-size can be either 128, 192 or 256-bits. In our design, this encryption scheme is used extensively. Initially, RSA is used to exchange the AES-keys between the nodes and after that every communication is encrypted using AES.

## **CHAPTER-2**

# **PROBLEM STATEMENT AND SOLUTION**

## 2. PROBLEM STATEMENT AND SOLUTION:

### 2.1 Problem Statement:

It is required to build a P2P network that works without a centralized server in order to provide uninterrupted service. This project introduces a way to construct such networks using the method of *Overlay network*.

While sharing the files, Anonymity is another important issue to deal with. Most users want to conceal their identities (for safety issues) while sharing files on a network. This project gives importance to protect the anonymity of a user by using the concept of *Relay nodes*.

### 2.2 How do Decentralized Networks work?

The networks operate as a web or mesh of neighboring node connections. One node connects to a few other nodes in the network, and those nodes connect to a few other nodes, which in turn connect to a few other nodes, and so on. A portion of one of these networks might look something like this:

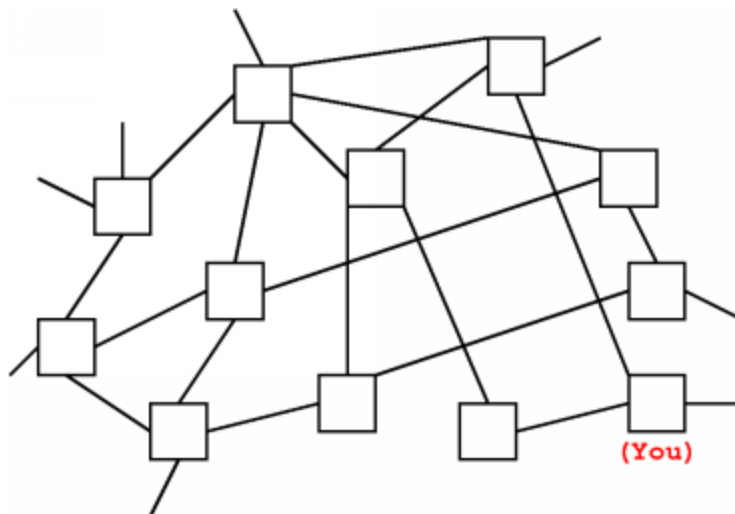


fig 2.1 A Decentralized network

When a file is being searched for in the network by a node, a search request is sent to its neighbours, they send the request on to their neighbours, and so on. Eventually, the request reaches



many nodes in the network. For example, a node starts out a search for "Cse doc". Lots of nodes receive this request, but only a few of them are actually sharing any cse doc. Those that do have matches will send their results back to the node. These results look something like this:

My Address: <b>170.30.0.10</b>	My File: <b>CSE_4-1</b>
My Address: <b>170.30.0.10</b>	My File: <b>CSE_e-books</b>

The address listed is the Internet address of the computer that has the file. IP Addresses are like Phone numbers. Computers use these addresses to make connections to each other over the Internet, and the node which requested the search can use this address to make a connection to the node that is sharing the files being searched for. Also like phone numbers, Internet addresses can be traced to find out who owns them. Suppose that the A node is the node at 172.30.0.10 that returned the results.

To download a file from this node, the sender node makes a direct connection to it using the address 172.30.0.10. After the node connects to the A node, the A node knows your computer's Internet address as well, say 175.18.92.15. The A node sends the file over this connection. This connection, which is separate from the other neighbor connections in the network, would look like this:

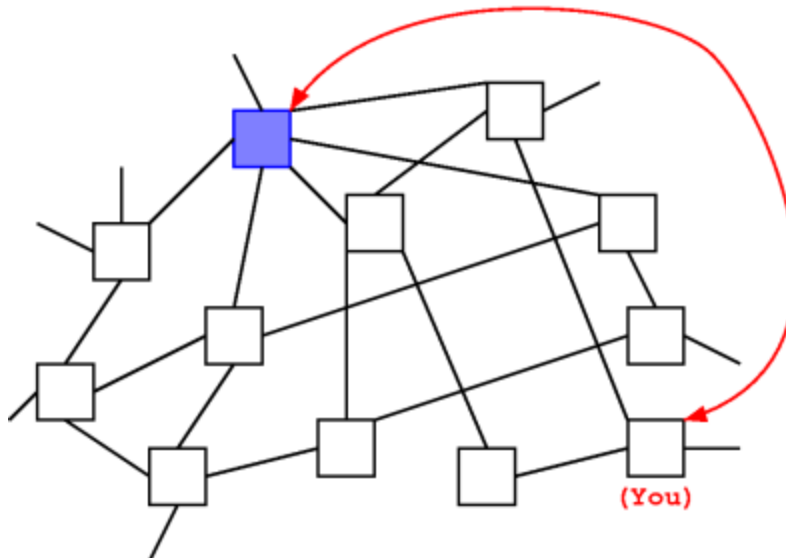


fig 2.2: sending file over direct connection in Decentralized Network

Just by performing a search, we got the Internet address of someone who is sharing Metallica. When we downloaded a file from that person, they got our Internet address as well.

## 2.3 How the RIAA Finds People to Sue

The Internet Service Provider, or ISP, provides everyone with their Internet address, much in the same way that your phone company provides customers with their phone number. And, like a phone company, an ISP knows who is using each Internet address that it gives out. In general, the ISP will keep the identity private. ISP doesn't share the identity with others unless being asked by the RIAA.

Suppose that a node is sharing a large collection of music files, and assume that this collection contains more than 1000 songs. Also, suppose that most of the songs in this collection are "owned" by record labels that are represented by the RIAA. When someone searches for "mp3" in the file sharing network, that node returns a lot of results. Now suppose that one of the nodes in the network happens to be owned by the RIAA:

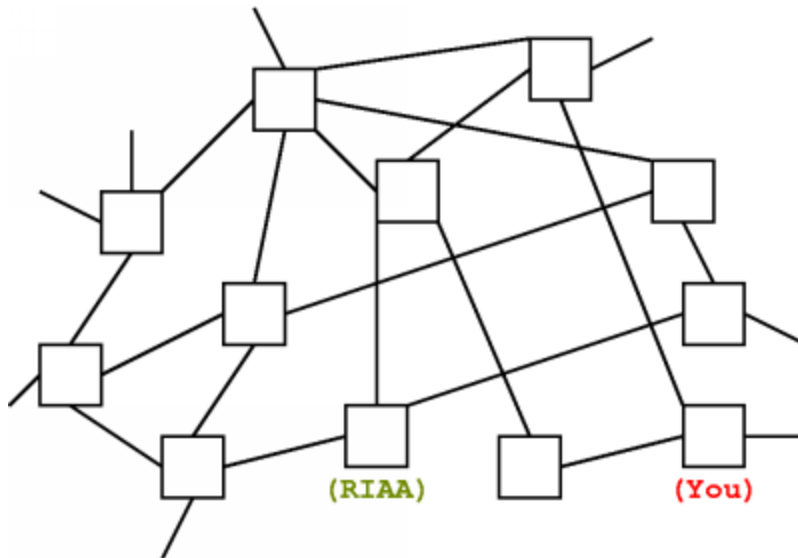


fig:2.3 RIAA in the Network

The RIAA performs a search in the network for songs that it cares about. Since RIAA record labels "own" the vast majority of music that is published in throughout the world, we can simplify things by assuming that the RIAA cares about most songs. Thus, the RIAA performs a search for "mp3", and that node returns over 1000 results

With the list of 1000+ infringing songs in hand, it files a subpoena against the ISP ("Who is using 113.18.92.15?") and demands that the ISP hand over the personal information. Once the RIAA has your personal information, it is ready to file a lawsuit against you for copyright infringement.

## 2.4 The Key Privacy Weakness

With standard file sharing networks, the key weakness is that Internet addresses for everyone who is sharing files are readily available. By sharing copyrighted files without permission, you may be breaking the law as it stands (though the legality of copyrighted file sharing is still up for debate). Returning to the phone analogy, using standard file sharing networks is much like making prank phone calls to someone who has caller ID---it is risky and stupid. With the wide use of caller ID, anyone who makes prank phone calls these days knows to dial "\*67" before each call to hide his or her identity from the party being called (a feature commonly called "caller ID blocking").

The reason Internet addresses are available in standard file sharing networks is because they have to be: there is no way to make a direct connection to a node for a download without knowing that node's Internet address. Likewise, there is no way for a node to accept your download connection without also being able to determine your Internet address. Data transmission on the Internet simply works this way, and there is nothing like "caller ID blocking" built into the Internet. The only way to protect identities is to build something on top of the Internet to avoid direct connections between downloaders and uploaders, and thus avoid the necessity of sharing Internet addresses.

## 2.5 Solution: MUTE Protocol

The main way that this protocol protects the privacy is by avoiding direct connections between downloaders and uploaders. By using the network to route search requests, these networks deliver a particular request to many nodes without making direct connections to any of them.

AP2PFS routes all messages, including search requests, search results, and file transfers, through the network of neighbor connections. Thus, though you know the Internet addresses of your neighbors, you do not know the Internet address of the node you are downloading from.

Now the reply looks like this:

My Address: <b>7213D...2DCA5</b>	My File: <b>CSE_4-1</b>
My Address: <b>7213D...2DCA5</b>	My File: <b>CSE_e-books</b>

Now the "My Address" portion of these responses no longer contains an Internet address. The address shown, which is abbreviated with "..." to fit in the table, is

7213D29781593840CF00CDD1E9A7A425AE16DCA5. This is a "virtual" address. Each node in the network has a virtual address that it generates randomly each time it starts up.

Though the transfer is routed through a node owned by the RIAA, all the node sees are the virtual addresses of you and your file sharing partner. The RIAA can send out a search for "mp3", and it will still get back 1000 results from you, but these results would look like this:

My Address: <b>D1E9A...29781</b>	My File: <b>Madonna__Holiday.mp3</b>
My Address: <b>D1E9A...29781</b>	My File: <b>Fleetwood_Mac__Dreams.mp3</b>
My Address: <b>D1E9A...29781</b>	My File: <b>Journey__Faithfully.mp3</b>
.	
.	
.	
My Address: <b>D1E9A...29781</b>	My File: <b>Bonnie_Raitt__Something_To_Talk_About.mp3</b>
My Address: <b>D1E9A...29781</b>	My File: <b>Poison__Unskinny_Bop.mp3</b>

The RIAA can subpoena your ISP using your virtual address, but your ISP does not know who is using this address. Thus, the RIAA's standard tactic is useless in the MUTE network.

## 2.6 Why maintain anonymity?

P2P users who desire anonymity usually do so as they do not wish to be identified as a publisher (sender), or reader (receiver), of information. Some other reasons include personal privacy preferences such as preventing tracking or data mining activities.

If the material or its distribution is considered illegal or incriminating by possible eavesdroppers or

Material is legal but socially deplored, embarrassing or problematic in the individual's social world then it is preferred to distribute it anonymously.

Anonymous P2P systems may be used for the protection of unpopular speech, but they may also be used to protect illegal activities, such as fraud the exchange of illegal data such as pornography, the unauthorized copying of copyrighted works, or the planning of criminal activities.

Anonymous P2P systems give the general public the freedom to use secure communication channels. When it comes to expressing views on a particular matter, people find it more comfortable speaking anonymously. If anonymity isn't maintained there are chances that the person gets threatened for speaking on controversial subjects. Throughout the history of literature, since the creation of bound texts in the forms of books and codices, various works have been published and written anonymously, often due to their political or controversial nature, or merely for the purposes of the privacy of their authors, among other reasons.

Most countries ban or censor the publication of certain books and movies, and certain types of content. Other material is legal to possess but not to distribute; for example, copyright and software patent laws may forbid its distribution. These laws are difficult or impossible to enforce in anonymous P2P networks.

#### Major Drawback:

Someone can pretend to be someone else by assuming that person's identity, usually as a method to gain access to resources or obtain credit and other benefits in that person's name. The victim of identity theft (here meaning the person whose identity has been assumed by the identity thief) can suffer adverse consequences if they are held responsible for the perpetrator's actions. Identity theft occurs when someone uses another's personally identifying information, like their name, identifying number, without their permission, to commit fraud or other crimes.

# **CHAPTER -3**

## **PROTOCOL DESIGN**

### 3. PROTOCOL DESIGN:

#### 3.1 Virtual Topology Formation:

Each node that enters the network initially sends a service discovery broadcast to all the other nodes. The nodes which are already running in the network reply with their IP Address and Current Degree.

The new node randomly chooses the top two nodes with Least Degree and sends a link establish request. Choosing the nodes with least Degree ensures that all the nodes have almost the same degree by distributing the load equally amongst them.

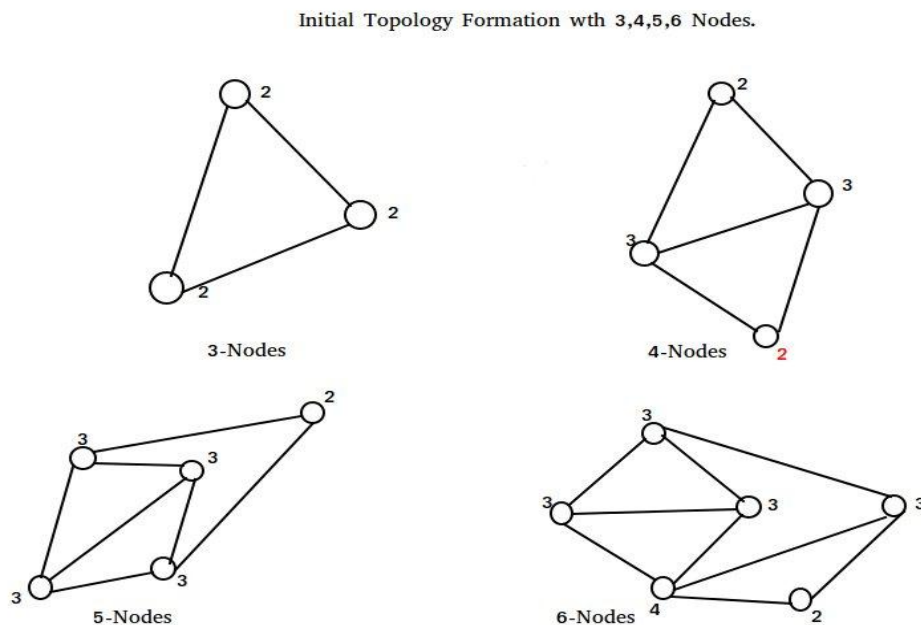


fig:3.1 The above figure shows topology for 3,4,5,6 nodes. Observe that there are no articulation points.



### 3.2 Why to select two nodes?

As already discussed, it is important that each node select top two nodes with the least Degree. This is done to ensure that the graph doesn't contain any *Articulation points*. It means that if any node leaves or goes down, by connecting the nodes in this way, we could make sure that the graph is still connected.

### 3.3 Link Check:

Each node has some adjacent nodes and each node sends a node alive request to ensure that its adjacent nodes are alive. This operation is done after every few seconds which can be configured. The time gap must be as less as possible to ensure the connectivity and reliability of the network.

### 3.4 Hop Count Query:

Another special query sent by a node is the Hop count query which is used to find the number of hops between the source node (which generated the query) and the corresponding destination node. It should not be confused with the normal hops. This hop count is the number of nodes in the virtual topology. The query propagation is as follows:

The source node sends a hop count request to all its adjacent nodes with an initial count of zero. These adjacent nodes forward the same query to their adjacent nodes after incrementing the count by 1. This procedure happens in parallel breadth first search fashion in the whole network. Now there could be two scenarios happening. Case when node is found. The numbers represent the current hop count at each node.

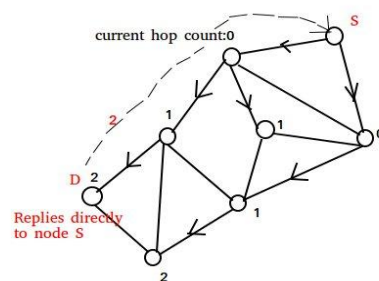


fig 3.2 : Hop count

Node is Found:

If the node is found, it directly replies to the source node which generated the Query with the current count.

Node is not Found:

Sometimes the graph may get disconnected and the actual node may not reply to the source node. The source node waits for a particular time-out and assumes that the destination node is unreachable and hence at infinite distance.

### 3.5 Avoiding Broadcast Storm:

Since the hop count query is purely broadcast, it may create a broadcast storm in which the same node tries to forward a packet which it has already forwarded. To prevent this, each node adds an entry into a table for a source-destination pair, after forwarding a hop count query and removes it after a particular time out.

Hence, when a node receives a hop count query, it checks its table for the entry containing the source-destination pair. If an entry is already there, it assumes that it has already forwarded the query and simply discards the current packet.

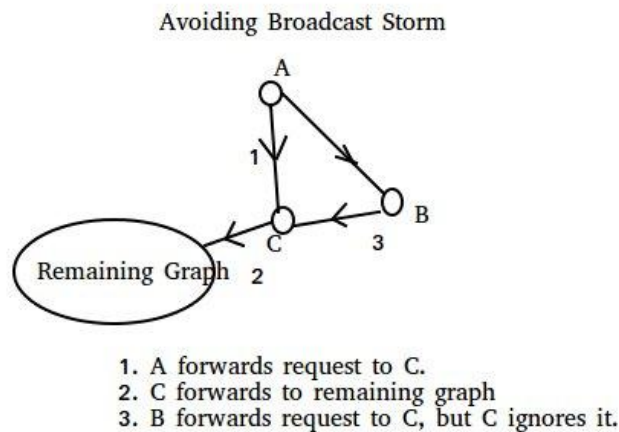


fig 3.3:Avoiding Broadcast Storm

### 3.6 Nodes leaving the network:

Even though there are no articulation points in the network, the graph may still get disconnected when two or more nodes simultaneously get disconnected. Hence, altering the graph accordingly when one or more nodes get disconnected is important.

There are two ways in which a node can get disconnected, namely:

#### 1) Voluntary Disconnection:

The disconnecting node signals all its adjacent nodes about its status and the nodes act correspondingly.

#### 2) Involuntary Disconnection:

The node gets disconnected without actually informing its adjacent nodes. This may be mainly due to power failure, link failure etc.

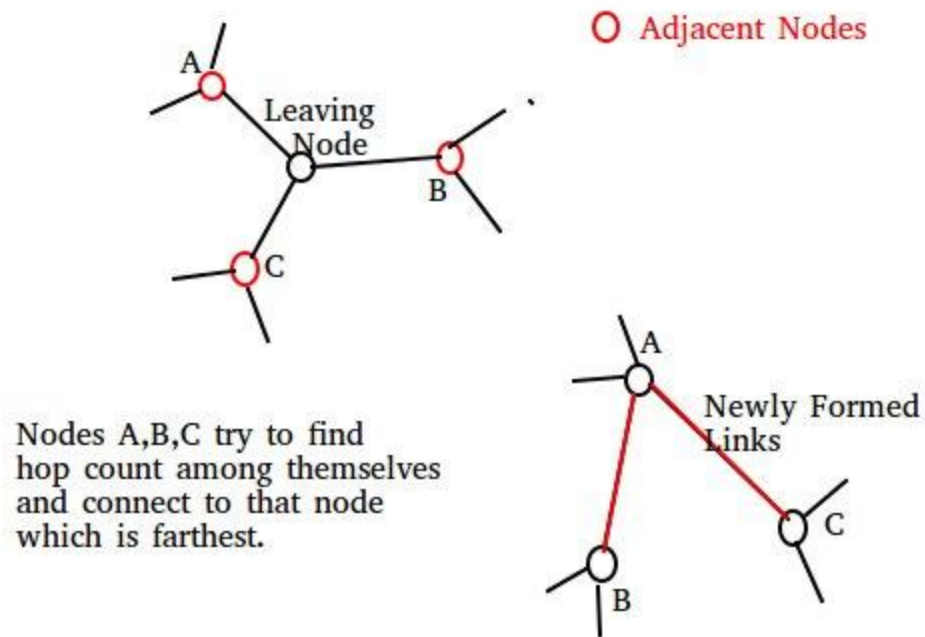
Although the second scenario happens rarely, both of them are handled effectively as follows:

All the adjacent nodes of the node leaving the network, try to find the hop count between each of them and connect to that node which is at farthest distance. Finding hop count between two nodes has already been described.

Note that for this to happen, each node stores additionally the 2-hop neighbors along with the adjacent ones. Each 2-hop neighbor also has a via node that indicates the node between the current one and the given one.

In case of voluntary disconnection, all the nodes are notified about the event and then the query is initiated. On the other hand in the involuntary disconnection, the nodes come to know about the disconnection after performing a link check.

fig 3.4 : Disconnection and New link formation



### 3.7 Optimization:

Since nodes try to find hop counts among themselves, if we find the hop count between the nodes A & B, say from Node A to B, then we need not find the hop count from Node B to A. Hence, all the nodes instead of trying to find the paths immediately wait for a random time and then start the procedure. This prevents unnecessary queries for same path to some extent.

# **CHAPTER-4**

## **FILE-SHARING PROTOCOL**

## 4. FILE-SHARING PROTOCOL:

### 4.1 Search Query Execution:

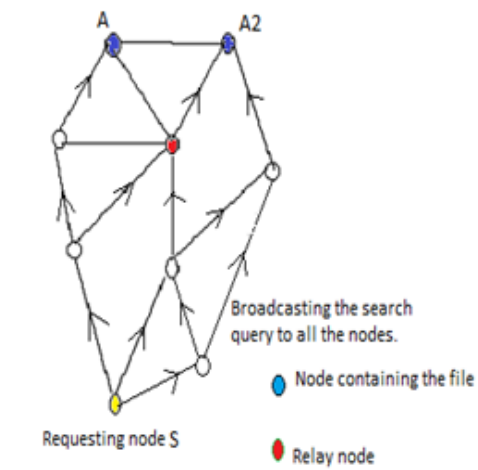
**Query:** Used to look up files in the network. A node receiving a Query message will respond with a QueryResult message in case a match is found against its local data store.

**QueryResult:** The response to a Query message. This message provides the recipient with information needed to get the data matching the corresponding Query. It includes the file list and the search key of the node which actually has the file.

**Search key:** It is used to encrypt the search results. It is not maintained secretly. That is, some of the nodes in the network may know the search key of other nodes. It is similar to the public key in public-key encryption. The key is generated using RSA and has size of 2048-bit. But the public keys of all the nodes are known to everyone in the network prior to the execution where as search keys of only a few nodes are known only during the execution.

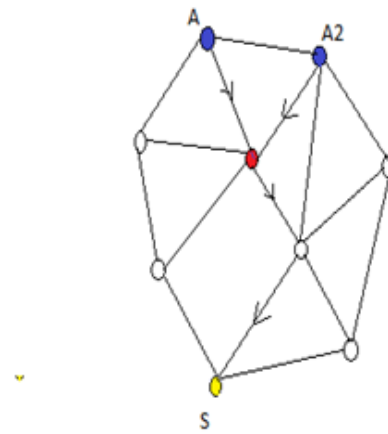
With all the nodes working, when a node, say node S, wants to search for a file, it sends a search query request to all its adjacent nodes which in turn forward these requests further until the whole graph is covered. Also, node S will send its search key along with the filename of the file being searched for.

The nodes, which contain the matching file, do not reply directly to the source node along with their IP Address. The nodes having the file will reply to the initial node, node S here, via an adjacent intermediate node( say node B) called as the Relay Node. While sending the Query Result from the actual node with the file(say node A) to the relay node(node B), the node A ensures to alter the source IP Address in the packet. These Query Results are encrypted using the search key which was sent by the initial node, node S. The encryption ensures that no other nodes, except for the node which is sending the reply, can read the reply being sent. From the Relay node these encrypted packets are forwarded to the initial node, node S. The initial node now accumulates all these results and selects one node(node A) to download the file.



→ sending the search request+ search key of the requesting node S

Sending the search query by S.



→ The query replies are sent by A and A2. It contains the file list and download key of respective sender. Whole packet is encrypted using the search key of S. S selects one out of A and A2

The query replies sent by A and A2.

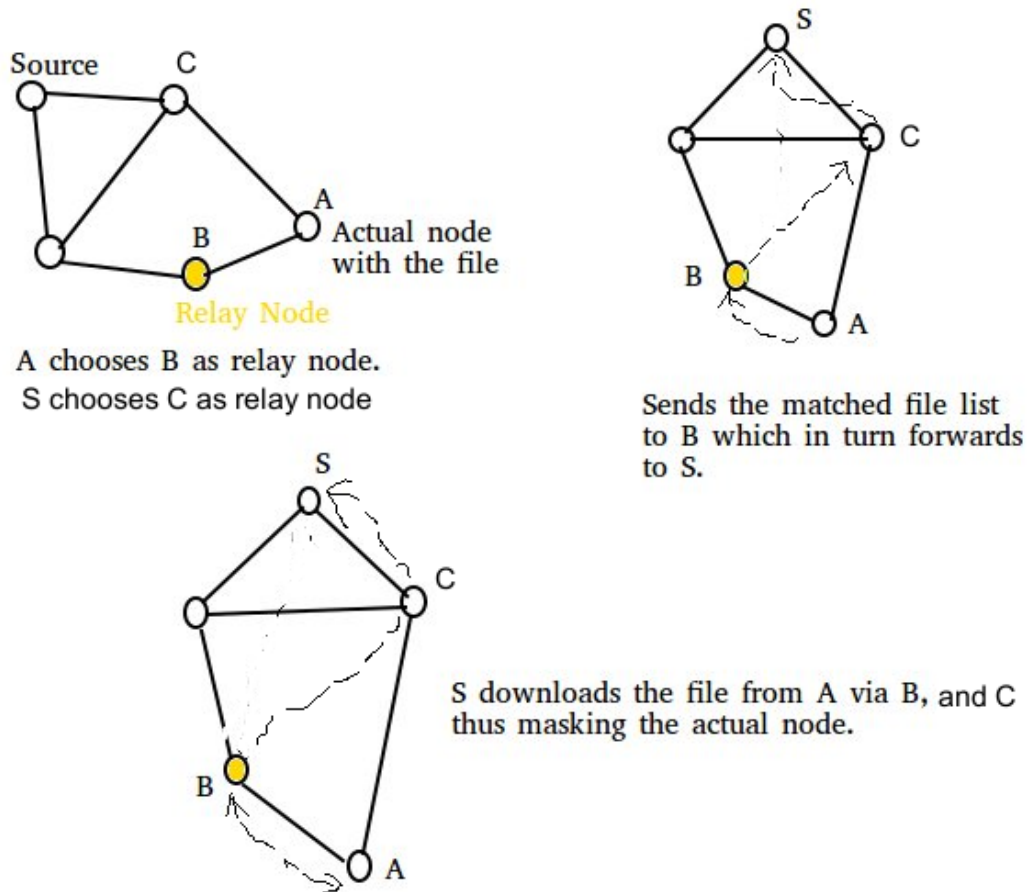
fig 4.1:search query execution

## 4.2 How relay node works:

Whenever a node wants to download a file from another node, it chooses a random relay node, from one of its neighbours. This is also done on the sending side. These two relay-nodes in turn connect with each other and the data transfer starts.

With this setup, the relay nodes have no idea of what data is being transmitted, since its an encrypted communication using AES. The relay node on the destination side, doesn't know the source node, even if it is some how able to find the name of the file. On the other hand, the source node's relay node, has no idea to where the data is being transmitted even if it caches all the search queries in the network.

fig 4.2 :How relay node works





### 4.3 Download Query Execution:

#### *DQuery:*

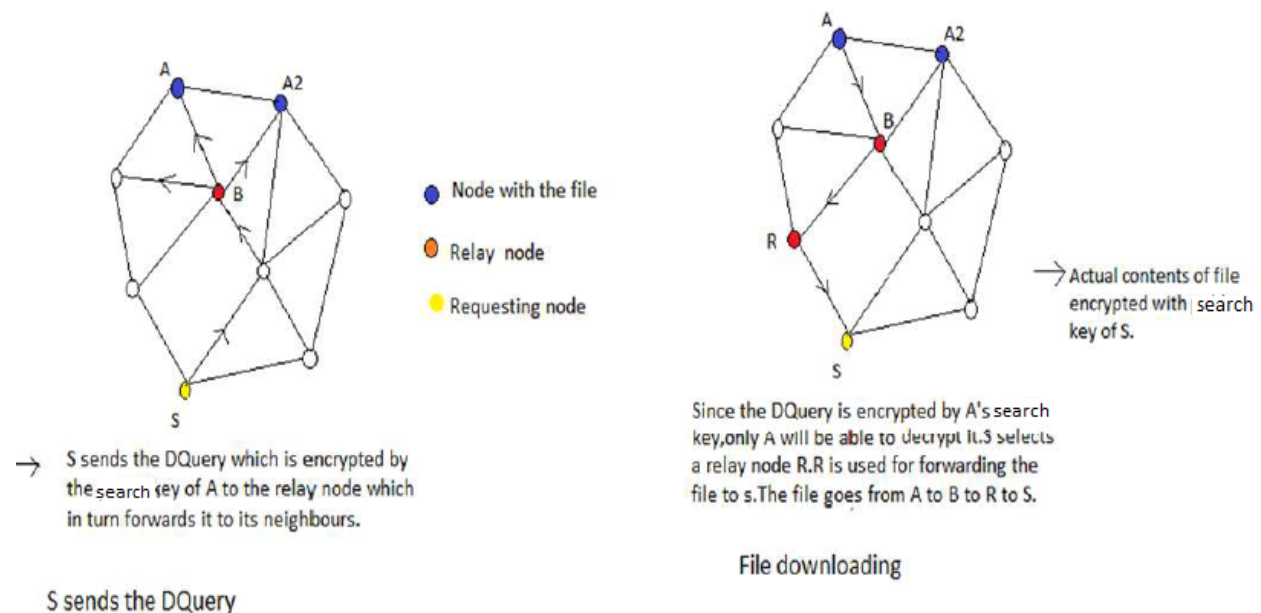
It is used to download the file in the network. The query is encrypted using the search key sent by the node actually having the file. A node receiving this query will reply with a DQueryReply.

#### *DQueryReply:*

It is sent as the reply to the DQuery. This message provides the recipient with the actual file. It is encrypted using the search key of the node which requested the file initially, node S in the case.

A node (node S) which made the file request, after receiving the QueryResult, decrypts the QueryResult using its private key. After verifying that the file is present with some node in the network, it issues Dquery requesting to download the file. The DQuery is sent to the Relay node. Since DQuery is encrypted using the recipient's search key, only the recipient can decrypt it and read the message. The recipient search key is AES key of size 256-bit. Other nodes cannot read the contents. The Relay node forwards it to all its neighboring nodes.

fig 4.3 : download query execution



When the node with the actual file receives the DQuery, it decrypts the DQuery packet using its private key. Now that DQuery was encrypted using the search key of a particular node which actually has the file, node A here, only that particular node can decrypt it and all others will reject the request. Node A will now send a DQueryReply(encrypted using the search key of node S) to the node B(Relay node). Also node S will request for another Relay node,say node R.Now node B forwards DQueryReply to node R and node R forwards it to the requesting node, node S. Node S will decrypt it using its private key and stores it.

#### **4.4 Relay node leaving during data transfer:**

The data transfer is actually done in chunks of different size. The size of each chunk depends on the current link speed. An application layer level acknowledgement will be sent after transferring each chunk of data.

When the node which made the search request, node S, comes to know that the relay node has left, the download will be currently suspended. The requesting node again forms a new relay node and the download resumes from where it was stopped.

#### **4.5 Advantages and Disadvantages:**

##### **Advantages of Peer-to-peer networking over Client –Server networking**

- 1) It is easy to install and so is the configuration of computers on this network.
- 2) All the resources and contents are shared by all the peers, unlike server-client architecture where Server shares all the contents and resources.

- 3) No need for a network operating system P2P is more reliable as central dependency is eliminated. Failure of one peer doesn't affect the functioning of other peers. In case of Client –Server network, if server goes down whole network gets affected. Does not need an expensive server because individual workstations are used to access the file.
- 4) There is no need for full-time System Administrator. Every user is the administrator of his machine. User can control their shared resources.
- 5) The over-all cost of building and maintaining this type of network is comparatively very less.
- 6) No need for specialist staff such as network technicians because each user sets their own permissions as to which files they are willing to share.
- 7) Much easier to set up than a client-server network - does not need specialist knowledge

#### **Disadvantages(drawbacks) of Peer to peer architecture over Client Server**

- 1) In this network, the whole system is decentralized thus it is difficult to administer. That is one person cannot determine the whole accessibility setting of whole network.
- 2) Security in this system is very less viruses, spywares, Trojans , etc malwares can easily transmitted over this P-2-P architecture. Ensuring that viruses are not introduced to the network is the responsibility of each individual user
- 3) Data recovery or backup is very difficult. Each computer should have its own back-up system
- 4) Lot of movies, music and other copyrighted files are transferred using this type of file transfer. P2P is the technology used in torrents.
- 5) Because each computer might be being accessed by others it can slow down the performance for the user.
- 6) Files and folders cannot be centrally backed up.
- 7) Files and resources are not centrally organised into a specific 'shared area'. They are stored on individual computers and might be difficult to locate if the computer's owner doesn't have a logical filing system.
- 8) There is little or no security besides the permissions. Users often don't need to log onto their workstations.

## 4.6 Summary of the Protocol:

Suppose, **A** sends a search-query in the network and **B** replies with the search results to one of the intermediate nodes, say **D**, which in turn forwards to **A**. Now, **A** wants to download the file from **B**. Let it choose a relay node say **R1**. **B** also chooses another relay node say **R2**. This entire procedure and the data transferred can be represented as follows:

### Search Query:

**A**  \*  $Search\_Query || IP_A || RSA-2048\ key$

### Search Results:

**B**  **D**  $IP_A || RSA_A(Search\_Results || AES-256_B)$


### Relay-Node Request:

**A**  **R1**  $Relay\_Node\_Request\_For\_Receiving\_Data$

**R1**  **A**  $IP_{R1} || Listen\_Port_1 || Listen\_Port_2$

**A** establishes a TCP Connection with **R1**, with the  $Listen\_Port_1$

### Download Request:

**A**  **D**  $AES-256_B(File\_Name || IP_{R1} || Listen\_Port_2 || AES_A)$

### Relay-Node Request:

**B**  **R2**  $Relay\_Node\_Request\_Sending\_Data || IP_{R1} || Listen\_Port_2$

**R2**  **B**  $IP_{R2} || Listen\_Port_3$

**B** establishes TCP connection with **R2** via  $Listen\_Port_3$ , and **R2** with **R1** via  $Listen\_Port_2$ .

# **CHAPTER-5**

## **RESULTS**

## 5. RESULTS:

The above described protocol was programmed in Python3. PyCrypto module was used for AES and RSA Encryption Schemes. The key-size for RSA and AES was 2048 and 256-bits respectively. The program was tested using NetKit simulator.

It was successful in providing Anonymity. The number of nodes used to test were 10. The IP-Address of the source node was always hidden under various circumstances.

The graph remained connected even after various nodes were leaving the network. The number of links were increasing as the number of nodes leaving the network, thus making the graph strongly connected.

The program was then tested in a physical network using 6 machines in a Wireless Local Area Network. Although, the download speed decreased to some extent, the file was successfully downloaded and the anonymity was maintained as usual.

# **CHAPTER -6**

## **CONCLUSIONS**

## **6.Conclusions:**

This project presents an anonymous P2P file sharing system called APTPFS, which ensures anonymity for both producers and consumers of file. APTPFS effectively prevents network eavesdropping. Because each APTPFS peer uses a virtual address to hide its identity and always avoids direct connections to others, it is highly unlikely that third parties can identify the participants involved in a file sharing session. Additionally, APTPFS anonymous content publishing allows the publisher to plausibly deny that the content originated from him or her .

The future work can include reducing the time taken to reorganize the nodes to make the network connected, when some node leaves the network, Mechanism to transfer the control packets efficiently, Reducing the amount of meta-data each node stores about the network.



## References:

- [1] Yang, Garcia-Molina, “Improving search in peer-to-peer networks”, *22nd International Conference on Distributed Computing Systems*, 2002.
- [2] Xiao Wang , Jinqiao Shi , Binxing Fang and Li Guo, “An Empirical Analysis of Family in the Tor Network” in *Communication and Information Systems Security Symposium, IEEE ICC 2013*.
- [3] Ciglaric, M. ; Vidmar, T., “Ant-inspired Query Routing Performance in Dynamic Peer-to-Peer Networks” in *20th International Conference on Parallel and Distributed Processing Symposium, IPDPS, 2006*.
- [4] Riahla, M.A.,Tamine, K., Gaborit, P. “A protocol for file sharing, anonymous and confidential, adapted to P2P networks” in *6th International Conference on Sciences of Electronics, Technologies of Information and Telecommunications SETIT,2012*.
- [5] Timpanaro J.P.,Chrisment I.,Festor O. “Monitoring anonymous P2P file-sharing systems” in *13th International Conference on Peer-to-Peer Computing(P2P),IEEE 2013*.
- [6] Yong Wang, Xiaochun Yun, Yifei Li, “Analyzing the Characteristics of Gnutella Overlays” in *Fourth International Conference on Information Technology, ITNG 2007*.
- [7] Hongyong Huang, Haiyan Wu, Guozheng Wang, “Study of Distributed P2P Information Sharing System” in *3rd International Symposium on Intelligent Information Technology Application, IITA 2009*.
- [8] Reid, F., Harrigan, M, “An Analysis of Anonymity in the Bitcoin System” in *3rd international conference on social computing, IEEE 2011*