REPLICATION WITH INCENTIVES IN CENTRALIZED PEER TO PEER NETWORKS

A Paper
Submitted to the Graduate Faculty
Of the
North Dakota State University
Of Agriculture and Applied Science

By

Akshay Mudgal

In Partial fulfillment
For the Degree of
MASTER OF SCIENCE

Major Department:
Computer Science

April 2013

Fargo, North Dakota

North Dakota State University
Graduate School

Title

REPLICATION WITH INCENTIVES IN CENTRALIZED PEER TO PEER NETWORKS

By

Akshay Mudgal

The Supervisory Committee certifies that this ***disquisition*** complies with North Dakota State University's regulations and meets the accepted standards for the degree of

MASTER OF SCIENCE

SUPERVISORY COMMITTEE:

Dr. Juan Li
Chair

Dr. Simone Ludwig

Dr. Jing Shi

Approved:

July 1st, 2013	Dr. Brian Slator
Date	Department Chair

ABSTRACT

We are living in a digital age in which data production has become so ubiquitous that the demand for sharing will increase as more aspects of life become digital. There are several ways we share data: via websites and special networks like peer to peer. Although peer to peer networks such as Bit Torrent allow users to upload and download files but do not work well when user is offline. In this paper, I propose a file replication technique with incentives in a centralized peer to peer network to automatically replicate most downloaded files to other peers on the network. The peer selection algorithm looks at the peer's reputation which is basically an indication of high availability. To gain a high reputation a peer must allocate a part of its resources to the network; thus in return the peer gets to download the file from many peers simultaneously.

ACKNOWLEDGMENTS

The work described in this master paper would not have been possible without the assistance, guidance, advice and support of my advisor, Dr.Juan Li, who helped me towards my academic as well as my personal success. I appreciate her willingness to help students and guide in their research which made me enjoy doing this paper.

I would like to thank my thesis committee, Dr Simone Ludwig and Dr Jing Shi, who spared much time in their tight schedules and provided insightful and constructive comments on this paper. Their insights proved to be quite helpful in extending and deepening my knowledge in this research field.

I would like to thank my friend Paul Loree for giving me ideas during the initial topic selection for my paper.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1. INTRODUCTION

We are living in a digital age that is full of information surrounding us in the forms of computers, smartphones, and tablets. These produce data and thus there arises a need for sending and sharing data among devices. The most common way to share data is through email, but it has size limitation and time delay because of potential stress from all the data transfers on the particular service used. The second popular way is carrying data on removable storage devices, but that requires physical movement of data storage from one device to another. The most recent and most popular technique is real time transfer of data from one device to another either through a third host or direct transfer using some sort of network connection. For example, client- server architecture is a time tested trust worthy implementation that fits most applications but can have a bottleneck on the server side as the network grows, indeed if the server is down then the whole network is down hence no more data sharing. Peer to peer networks are data sharing networks that have grown tremendously in popularity over the past decade. Peer to peer networks (Bit Torrent, Napster, Gnutella, Limewire, and Freenet) have attracted millions of users and are still growing. The typical process of joining a peer to peer network includes, first downloading, then installing, running the client that understands the respective network protocol and finally sharing data. This type of network is a fast way to setup data sharing and requires at least two users. A peer to peer (direct transfer) network is a type of network that is an implementation of distributed system architecture in which the tasks are divided among peers or members of the network. This type of network consists of peers which could be heterogeneous or homogeneous computer nodes with their own computing resources. Peers are equally privileged and equipotent in this network architecture. Peer to peer networks for sharing files depend upon the availability of files useful to the peers on the network. When a user tries to find a file over the network but does not find it or is stuck with a partial download, the user is then disinclined to repeatedly share his own files and this refusal to share leads to the death of the network because the network only exists until the point it has peers to form it. Research has been done separately for file replication and reputation schemes in peer to peer networks. However the existing work in reputation schemes are not used in conjunction with file replication.

A solution to this type of problem would be to implement an incentive system which would incline users to share more files, thus having more files available to the network. In return users receive incentives in the form of faster downloads. The network would benefit from this as peers would try to be more available so as to receive more incentives. This is a win-win situation for both parties.

My approach builds on this potential solution. This research studies a file replication scheme in which incentives are provided to users which allocate resources so that the network can replicate files. By replicating the top most popular files percentile to highly reputed peers on the network, files that are downloaded most often will have a higher availability on the network.

The scheme implements a reputation system based on allocation of download/upload bandwidth, allocation of file space on the peers' hard disk drives and the uptime of the peers' connection to the network. These resources are crucial for replicating files on the network to provide high availability of files for other peers to access. In the scheme, files are given a popularity ranking and only the most popular files are replicated to other peers, this is done in order to keep the overhead low on the network and increase availability of files more likely to be accessed. Peers which allocate resources for replication of popular files are then provided incentives for their contribution to the network based on their reputation.

The motivation for replicating files to peers based on reputation calculated using resources allocated by peers in the network is to provide better availability of highly requested files on the network regardless of the original publisher's availability. The replication process then in turns improves network efficiency for file transfers by allowing users to simultaneously download portions of files requested from either original published sources, replicated file sources or combination of both. This process would reduce the overhead needed by any one peer attempting to share a file.

This fulfills the basic purpose of peers which is to receive the data they are looking for and hence lead to growth of the network. This paper has been divided into 5 sections, Introduction, Background knowledge and Related Work, Design and Architecture, Experiment Results, and Conclusion.

2. BACKGROUND KNOWLEDGE AND RELATED WORK

In this chapter, I briefly present the necessary background knowledge to the research.

2.1. Peer – Peer Networks

A peer to peer network is a type of network that is an implementation of distributed system architecture in which the tasks are divided among peers or members of the network. This type of network consists of peers that can be heterogeneous or homogeneous computer peers with their own computing resources. Peers are equally privileged and equipotent in this network architecture.

Peers provide part of their computing resources, disk space, and network bandwidth to the network or to other peers so as to achieve some task assigned to the network. This sharing of resources does not need to be regulated or governed by a central server or by stable hosts. All peers in this network architecture assume the role of consumers and suppliers of resources, this is very different than a client server architecture in which only the central server is the supplier and all the clients are consumers.

In essence, peer to peer networks allow each peer to communicate with all other peers without the need of a central server to coordinate their communication. Peer to peer networks are classified into two types of network architecture:

1. Structured Peer to Peer networks.
2. Unstructured Peer to Peer networks.

2.1.1. Structured Peer to Peer Networks

In this network architecture, peers are organized following specific criteria and algorithms, which lead to overlays with specific topologies and properties. This system employs a globally consistent protocol to ensure that any peer can efficiently route a search to a peer that has the desired file even it is rarely available. For such a guarantee to exist, a more structured pattern of overlay links needs to be implemented. The most common type is the Distributed Hash Table, in which a variant of consistent hashing is used to assign ownership of each file to a particular peer. Some examples of this system are Chord, BitTorrent's distributed tracker, the Kad Network, the Storm Botnet, YaCy and the Coral Content Distribution Network.

2.1.2. Unstructured Peer to Peer Networks

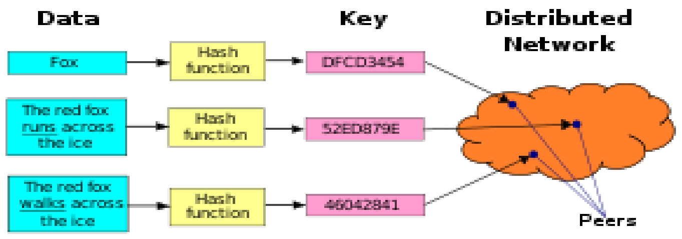


Figure 1. Unstructured Peer to Peer Networks [21].

These types of networks are formed when overlay links are established arbitrarily. Such networks can be easily formed as any new peers that wants to join the network can copy existing links of another peer and then form their own links over time. In contrast to structured peer to peer networks, if some piece of data needs to be found by a peer then the search query has to be flooded throughout the network to find as many peers as possible that are sharing that data.

In such network architecture, searching for files cannot always be successful. A popular file can be shared by several peers, hence anyone searching for that file can find it, however in the case of a search for a very rare file, shared by very few peers, and then it is possible that the search result cannot be successful because the search query broadcast may die before reaching a peer having that file. There are no correlation between peers and the content provided by the peers. In addition flooding can cause high network traffic and thus result in a slower and less efficient search.

These systems are classified into three different sub types of architecture:

1. Pure Peer to Peer networks
2. Hybrid Peer to Peer networks
3. Centralized Peer to Peer network.

2.1.2.1. Pure Peer to Peer Networks

The entire network consists of peers that are equipotent and either directly connected to other peers or through other peers in the network. There is no central server or super peers to control the communication on the network.

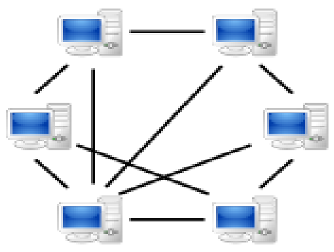


Figure 2. A Pure Peer to Peer Network [19].

2.1.2.2. Hybrid Peer to Peer Networks

This network has some super peers which act as infrastructure peers. All the nodes are divided into client nodes and overlay nodes. Typically in such networks all the client nodes act according to the momentary need of the network and can also become part of the overlay nodes to coordinate the network structure. These networks came into existence to address the scaling problem of early pure peer to peer networks. Examples of this fairly new technologies are Gnutella and Gnutella2. The proprietary software such as Skype also employ a very similar architecture to establish communication between the peers.

2.1.2.3. Centralized Peer to Peer Networks

This architecture is a type of hybrid peer to peer networks and employs a central server which holds the information about the network like its peers and the files available over the network and who has the files. It may also contains some configuration options for the connected peers like their bandwidth numbers and time available to the network. It is basically a mixture of client server architecture and peer to peer architecture.

If a peer wants to search for a file it asks the central server whether or not that file is available on the network, and if it is, then where to find it. Once the server looks up the information in its internal database, it will return a list of peers and their connection options to the peer asking for the file. Then this peer will directly connect to the peers who have the files and start downloading the file. Once complete, the server updates its database to reflect the current state of that file. So the role of the central server is to reply to the queries generated by the peers and, in some cases, also regulate the connection among peers based on rules specified by an algorithm.

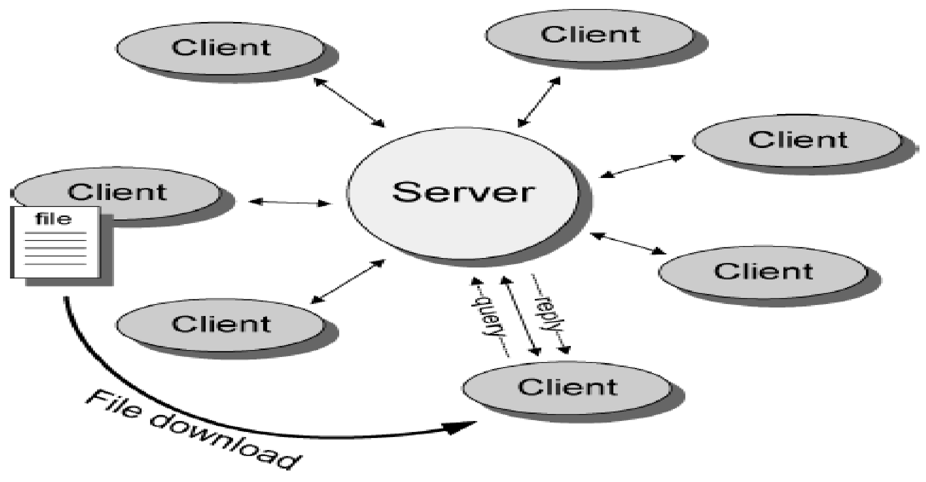


Figure 3. A Centralized Hybrid Peer to Peer System

2.2. Related Work

Androutsellis-Theotokis and Spinellis conducted a study in which they describe various peer to peer network systems and infrastructures that have been developed and analyze those systems for various characteristics [5]. They characterize systems as either peer-to-peer exchange systems or as peer to peer content publishing and storage systems both of which are designed to run on various infrastructures. Infrastructures were also categorized into routing and location, anonymity, or replication management schemes.

In [4], M. Feldman, K. Lai, I. Stoica, and J. Chuang proposed an incentive scheme for unstructured peer-to-peer networks which is based on the generalized Prisoner's Dilemma problem in game theory. Peers in the network use game theory based on local and shared historical information about authors' strategy performances for a period of time to maximize the peers' benefit on the network. Mortazavi and Kesidis proposes a reputation framework in [1] to allow a peer to maximize the number of files a peer is able to receive from others based on peer's reputation. Their framework's reputation is calculated by the number of successful file transactions the peer has with other peers on the network and is designed for long-term cooperation in mind. Each peer was able to establish “Trust groups” for polling the reputation of one node from other peers in the network.

Zhao and others give a mathematical framework for the designing and analyzing incentive policies in peer to peer networks in [2]. They apply their framework on analyzing the image incentive policy when a server peer provides service with the same probability as the peer provides a service to other peers on the network. They also apply their framework on the proportional incentive policy in which a server peer provides services to the client peer dependent on the client peer’s contribution ratio to the network.

In [3], C. Ye, D.M. Chiu propose a file replication scheme which considers file importance as to whether or not to make that file available to the network. They provide a weight to each file which is used to determine if the file is important enough to be replicated onto other peers’ in the network. The authors discuss two algorithms as follows: (1) Distributed replication where the file is replicated to peers based on the file’s weight, (2) Random replication where each file on the network has an equal chance with other files to be replicated onto peers in the network.

In [18], Pradipta Mitra proposes a reputation calculation and management system to identify malicious peers using self-assessment and other peers in the network. She proposes two reputation calculation methods: (1) A Local Reputation calculation which is based on the number of files authenticated and verified by other peers and (2) The voting method that is used to calculate reputation but the voters vote is weighed based on the client peer’s reputation. The author talks about how to select voters using neighbor voting and friend voting.

In [15], Mao Yang, Qinyuan Feng, Yafei Dai, Zheng Zhang propose a pair wise reputation scheme based on the incentive and trust between pairs. They basically say that the reputation is a function of the Incentive and trust other peers have for a particular peer. The authors intend to keep the user on a network by providing them incentives for their services like good upload speed and having a trust system so that the user has confidence in the files. Their scheme also provides for higher incentives when trust is high. They provide various methods to increase the peer's trust on their peers like honest voting of files, honest ranking of others, retaining good files and removing bad files quickly.

In [14], Razvan Andonie, Luke Magill, James Schwing, Ben Sisson, and Brandon Wysocki propose a replication scheme consisting of a mobile load balancing peer to keep replications in check. In their scheme a file is replicated to peers if they access it, and the authors always try to keep some minimum number of replications. They employ a mobile load balancing peer which goes through the network to find and eliminate file replicas that have lower priorities. They use internal states of the accessed object to make decisions to eliminate or keep a replica. Their system is event driven and a query or update from a node triggers event which update the internal state of the accessed object. They also proposed a search technique in which a node broadcasts the query to its 4 neighbors and they in turn do the same.

In [16], Roslan Ismail, Colin Boyd, Audun Josang and Selwyn Russell propose an improvement to the Fahrenholtz and Lamersdorf reputation scheme. In FL scheme an entity called Portal is used to monitor transactions conducted between peers. The Portal also records the number of transactions conducted, as well as the number of feedbacks obtained by each peer. This is achieved via the use of ticket and nonce. The proposed improvement is the introduction of Privacy as a major concern.

3. REPLICATION SCHEME: DESIGN & ARCHITECTURE

People have been using peer to peer networks like Bit-Torrent and Limewire over the years to find content that might be available elsewhere on the internet. Although the transfers succeed sometimes without any problems but many a times the peer(s) that host the file go offline and the download is stuck until they come back online. The basic idea behind a peer to peer network is that everyone can equally share and download content. So building upon this framework I propose that peers be asked to share a portion of their storage that can be used by the network to increase file availability by replicating the most popular files on the network. In return peers would get an incentive to download the files from more peers simultaneously hence leading to faster download.

This scheme would be effective because we humans are not very patient and if given a chance to speed up things at a cost we would certainly choose to do it. The cost here is the disk space which is very inexpensive so it is very possible that the network might see a lot of shared disk space available for replication. The design of the network is such that the peers can be motivated to get faster downloads and so as people join, the shared disk space will increase, thus allowing the system to replicate more files, which in the end would lead to faster and successful downloads for peers.

3.1. Design

3.1.1. Reputation Computation

Existing schemes build reputations generally based on how much a client is willing to share on the network. This process of reputation building is based on the number of files a client has successfully shared and/or the total volume of data a client has shared. However, these types of reputation schemes have a drawback because the reputation does not help with file replication and can be skewed by clients adding files to the network which other users may download but are not the actual file the downloading peer believed it to be. The reputation scheme introduced in this paper is a motivation for the peers to allocate more resources and be available to the network in order to receive the incentives. The ability to download a file in parts simultaneously from several peers on the network. Each peer that connects to the network puts in his download speed, upload speed, and the disk space allocated to the network for replication purposes. After this information is sent to the server, a periodic reputation computation algorithm runs on the database and computes and updates the reputation for all the peers.

The server uses the following formula to calculate each peer's reputation:

$$R = \frac{\frac{D_{cID}}{max(D_{ID})} + \frac{U_{cID}}{max(U_{ID})} + \frac{stor_{cID}}{max(stor_{ID})} + \frac{UP_{cID}}{UP_S}}{4}$$

Equation 1

In this formula reputation is calculated by determining the client's allocated download bandwidth in ratio to the highest recorded allocated download bandwidth of all peers in the network as shown in the first term of the equation. In the equation D_{cID} represents the download bandwidth allocated by a particular peer, $max(D_{ID})$ is the maximum allocated download bandwidth for all peers on the network. U_{cID} is the upload bandwidth allocated by the peer, $max(U_{ID})$ is the maximum allocated upload bandwidth for all peers on the network. $stor_{cID}$ is the storage allocated by a peer. $max(stor_{ID})$ is the maximum allocated storage of all the peers on the network. UP_{cID} is a client's uptime. UP_S is the server's uptime. Taking the ratio between the client's uptime to the server uptime can be penalizing for a client which joins the network at the latter time so the server uptime for a client is calculated from the date/time the client first joined the network. These terms are averaged together to determine the client's reputation on the network.

The client's reputation is periodically reevaluated by the server over specified time. This is done periodically in order to keep the peer's reputation current and to remain consistent with the peer's settings for allocation of resources.

When a client requests a file on the network, this reputation is used to determine the total number of peers with the file it may connect to as an incentive to the user allocating these resources to the network for file replication. The amount of incentives a client receives is determined by calculating the rank of the client in comparison to other users by dividing the client's into specified percentiles by the server. Once the client's rank has been determined, the percentile in which the client lies determines the number of connections to other peers which either own or have a replicated copy of the file on the network.

For example, suppose a client requesting a file which has been replicated to ten peers and is owned by an additional ten peers which have downloaded and republished the requested file. If the requesting client falls into the third quartile of all peer reputations then the incentive policy could allow the client to connect to all of the peers with the files. However, if the client falls between the 50th and 90th percentile the incentive policy could allow this particular client to connect to only a subset of the peers which contain the file, such as half of the users with the file. This allows users with a higher reputation to receive the file much faster as an incentive since the user would be allowed to simultaneously download much smaller parts of the file from more users than larger parts from fewer. Additionally this lessens the burden placed on any given node in the network from supplying the file to the requesting client.

3.1.2. The Replication Process

This subsection describes the replication procedure based on a file's popularity. Each time a file has been successfully downloaded on the network the file's download count is increased by the server and stored in the database. The file popularity is then computed by taking the ratio of an individual file's download count to the sum of all files' download counts on the network as shown in equation 2:

$$P = \frac{N_{fID}}{\sum_{i=0}^{n} N_i}$$

Equation 2

In the equation N_{fID} is the number of times a file has been downloaded. $\sum_{i=0}^{n} N_i$ is the total number of downloads of all the files. The file popularity is periodically calculated by the server and stored in the SQL database. After a specified time period, the server automatically ranks the files by popularity into percentiles, similar to the reputation system, in order to determine which files should be replicated on the network and where to store the replicated files. Files are replicated first to highest ranking peers, divided into percentiles based on the reputation of the peers without the files. This allows the files to be replicated onto peers without the files which have the highest probability to keep the files available on the network and the resources needed for replicating the files on the network.

For example, suppose a file has been ranked based on its popularity into the 90th percentile. The server would then query the database and determine ranks of the peers which do not have the file. Once the peers have been ranked, the server selects a certain percentile of these clients to replicate the file to, for instance the peers that rank in the third quartile may replicate files in the 90th percentile ranks. Peers that are in the third quartile of rankings would then be checked to determine if they have enough free space available on their hard disk for the file to be replicated to it.

All the replicated files are stored in ‘C: \Replications’ of the client computers. After each file is replicated, the database is updated to reflect the remaining free space on the clients.

3.2. Architecture

This paper also discusses about developing a Centralized peer to peer network. The two main components of this network are:

1. Client
2. Multithreaded Server

The client has been written using Microsoft .Net Framework 3.5 C# and it uses .NET’s TCPCLIENT class to connect to the central server and other peers on the network. Each client is made up of two threads:

1. User Interface Thread
2. Listener Thread

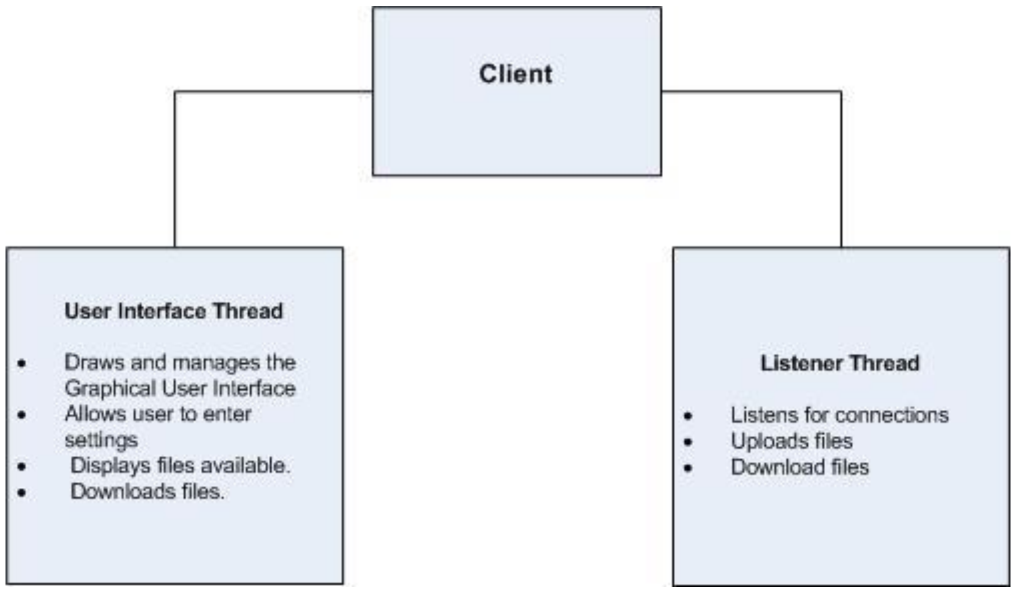

Client
User Interface Thread
• Draws and manages the Graphical User Interface
• Allows user to enter settings
• Displays files available.
• Downloads files.
Listener Thread
• Listens for connections
• Uploads files
• Download files

Figure 4. Client Architecture

The User Interface thread draws the User Interface and deals with all the interactions that the user does with the system and it is also handles the file download process. User interface provides the user a mouse and keyboard based screen area through which he/she can look at files available on the network and allows the selection and download of a file. User Interface also allows the user to set his connection Upload/Download bandwidth and the amount of disk space he/she wants to allot to the network for replication purposes.

Listener thread is a background thread that is always running as long as the client is in execution. This thread requires no user interaction and is primarily responsible for accepting connection requests from the central server and other peers on the network. This thread accepts all the replication requests from the server and initiates the download process in background without any user intervention. It also starts the upload process whenever a file download request is sent by any other peer on the network. So basically this thread handles all the tasks that are crucial to network functioning without asking for any user intervention.

The server has been written using Microsoft .Net Framework 3.5 C# and uses ADO.NET to do all data transaction with the SQL Server. It uses TCPLISTENER class and TCPSERVER class to listen and answer connection requests from peers on the network. The multi-threaded server is the backbone of this network architecture and is responsible for keeping an account of all the files available, all the peers connected to the network, and running algorithms which calculate each peer's reputation and the replication process. The server stores all the information related to the network in a database that is running on SQL Server 2008.

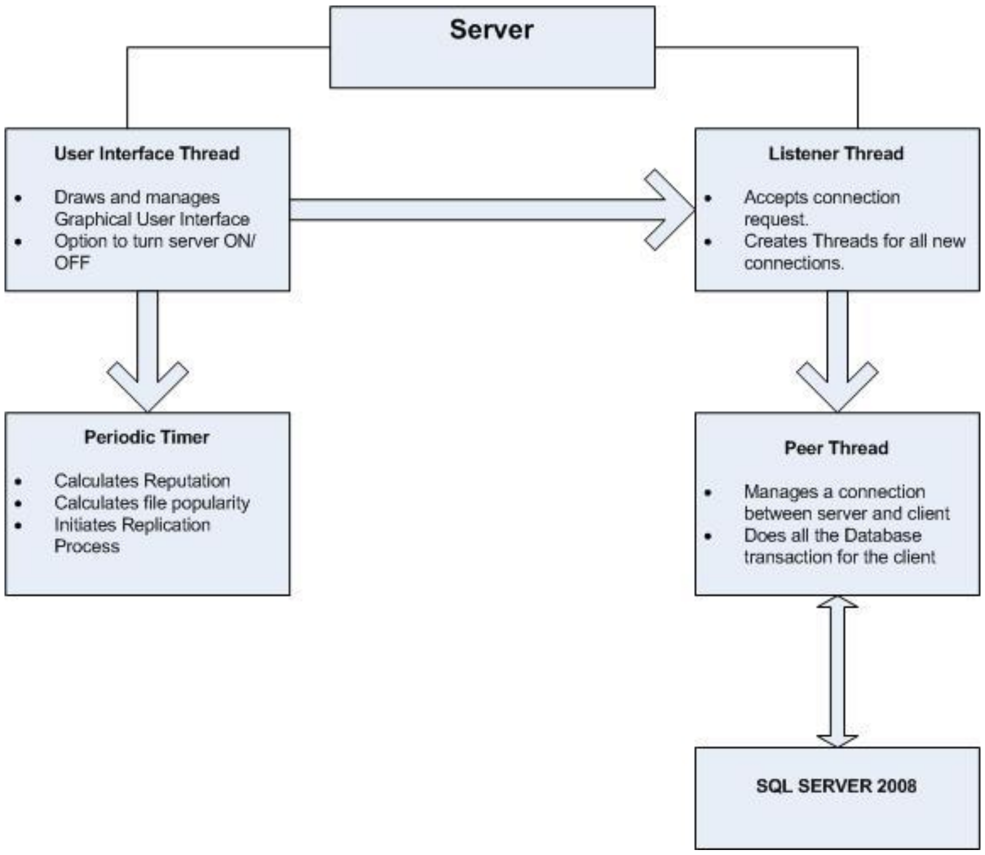

Server
User Interface Thread
Draws and manages Graphical User Interface
Option to turn server ON/OFF
Listener Thread
Accepts connection request.
Creates Threads for all new connections.
Periodic Timer
Calculates Reputation
Calculates file popularity
Initiates Replication Process
Peer Thread
Manages a connection between server and client
Does all the Database transaction for the client
SQL SERVER 2008

Figure 5. Server Architecture

3.2.1. The Connection Process

The process of downloading a file starts with a peer joining the network. Whenever a peer starts the client program and if it's the first time the peer is running the program then the client program registers the peer to the central server by sending a list of all media access control (MAC) addresses. Each peer is uniquely identified by its MAC address and this step makes sure that the server has the list of all MAC addresses for a peer. Since a peer can connect using any of the available network interfaces and thus reflecting a different MAC address, in this case the server won't know about peer's current reputation and thus affecting the incentives. So to address this problem, every time the client connects to the server it sends all the MAC addresses available on the peer's machine and the server updates it's database of alternate MAC addresses for that peer. This way the reputation algorithm becomes independent of the way a peer might connect to the network and thus his reputation is calculated properly and he gets all the earned incentives.

If the peer is connecting for the first time then it is asked for its download/upload and the amount of disk space it will allocate to the network for replication. After the settings are entered and Save is clicked, these settings are uploaded to the server and saved by the server. *Figure 6* illustrates it.

The client downloads all the files to a default folder 'C: \Downloads' and always sends a list of all files that are in this folder to the server. The list consists of the files that are available to the network whenever a peer is online. The screen changes to a navigation screen once the settings are saved. All the files available on the network are represented in a tree-like structure. *Figure 7* illustrates it.

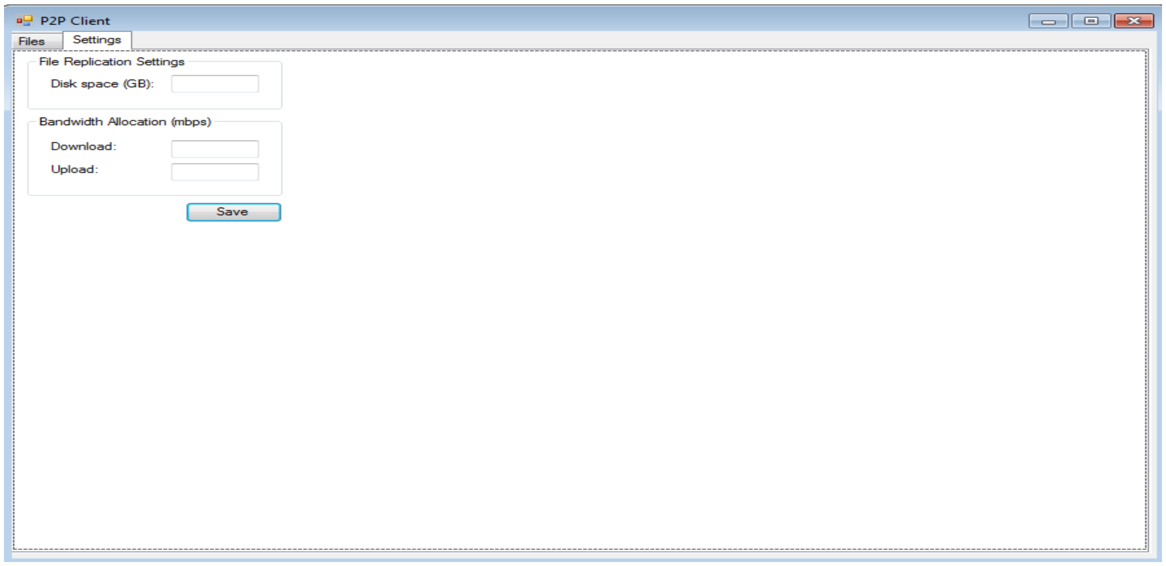


Figure 6. Client Settings Screen

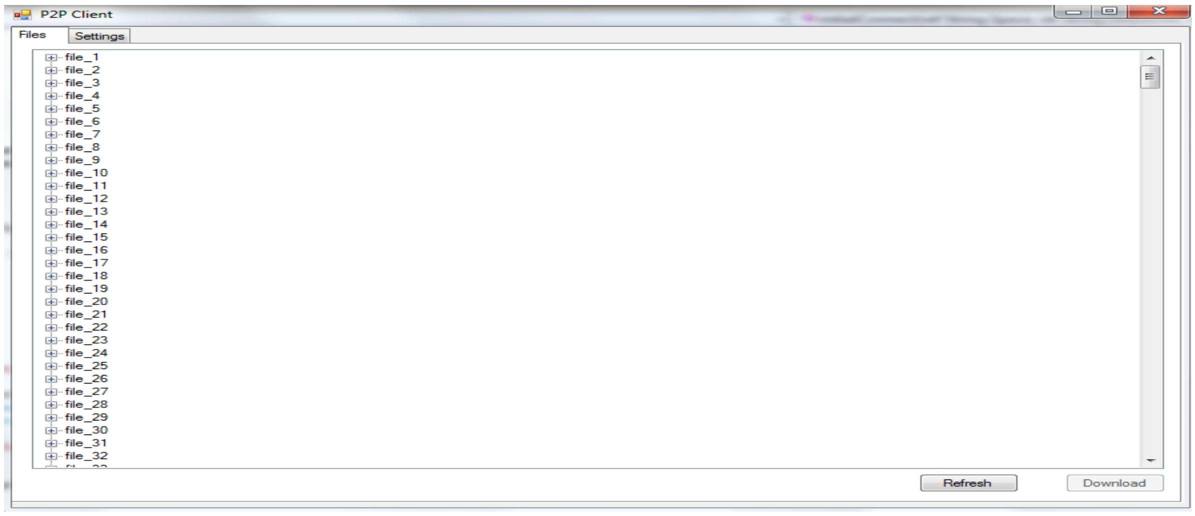
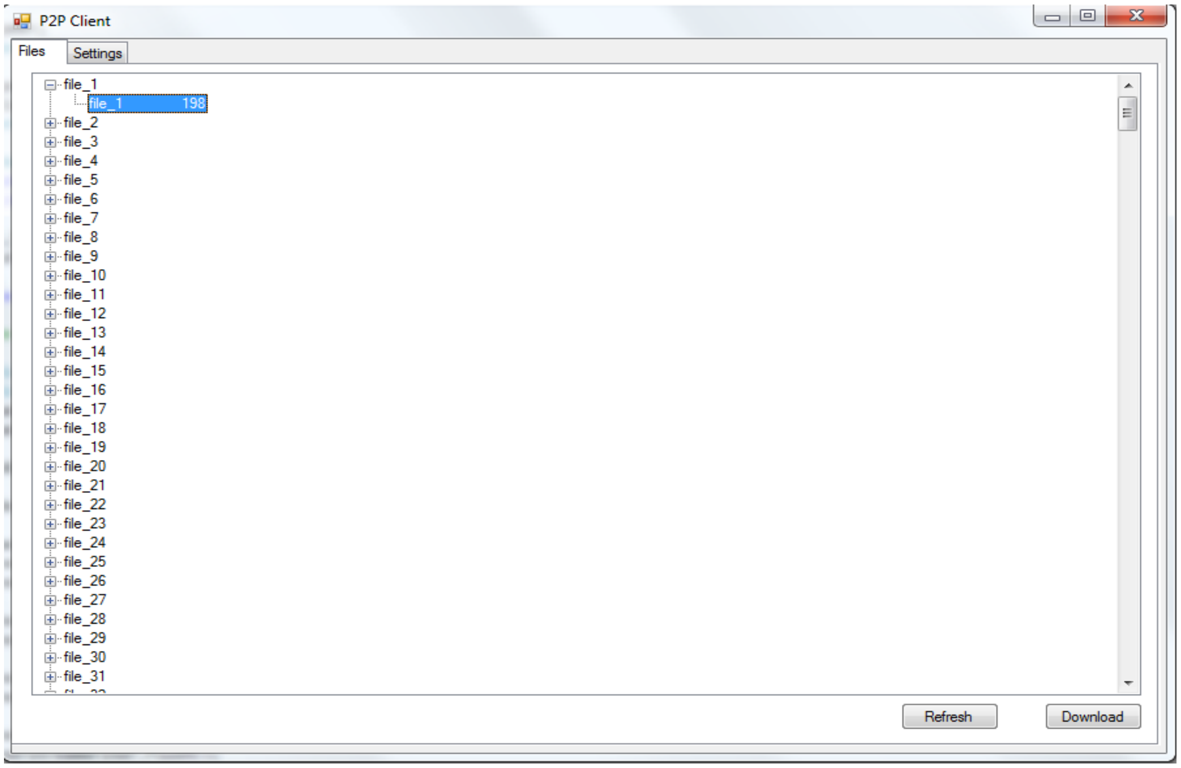


Figure 7. Main Navigator Screen

3.2.2. The Download Process

Every time a client is started and it is already registered on the network then it shows a file navigator which lists all the files available for download in a tree-like structure. Once a file is selected, the "Download" button becomes intractable and user can click on it. The download process starts by the peer requesting for a list of peers having the file from the server. The server based on the peer's reputation returns the list of users in accordance with the incentive policy. The peer contacts all the peers received from the server and asks for a piece of a file then after simultaneously getting each part, joins the part by writing them into a file in 'C: \Downloads'. Finally a list of all files in 'C: \ Downloads' is sent to the server.



file_1
file_1 198
file_2
file_3
file_4
file_5
file_6
file_7
file_8
file_9
file_10
file_11
file_12
file_13
file_14
file_15
file_16
file_17
file_18
file_19
file_20
file_21
file_22
file_23
file_24
file_25
file_26
file_27
file_28
file_29
file_30
file_31

Figure 8. File Download Screen

4. EXPERIMENTS

4.1. Experimental Setup

This section describes the generation of results with a simulated run of the network for 5 days with 500 peers connected.

Since the network is a peer to peer network and the design of the network requires that the peers be independent computers I had to simulate 500 peers connecting to the network having their number of files shared, allocated space, download bandwidth, upload bandwidth randomly generated using a set of random number generators. This was done so that I get a set of peers whose values are randomly selected and, thus, closer to a real world scenario where we have all kinds of peers.

4.2. Experiments and Results

Figure 9 depicts the distribution of peer reputation across all the percentage slabs which are quite similar to a real world scenario consisting of different kinds of peers. The higher the reputation, the greater incentive it is for the peer. From the chart below we can see that 19% peers are in top 90th percentile, 10% are in 80-89th percentile, 19% are in 70-79th percentile, 15% are in 60-69th percentile, 16% are in 50-59th percentile, 10% are in 40-49th percentile, 1% are in 30-39th percentile, 10% are in 20-29th percentile. The x-axis is percentile group and the y-axis is the percent of peers.

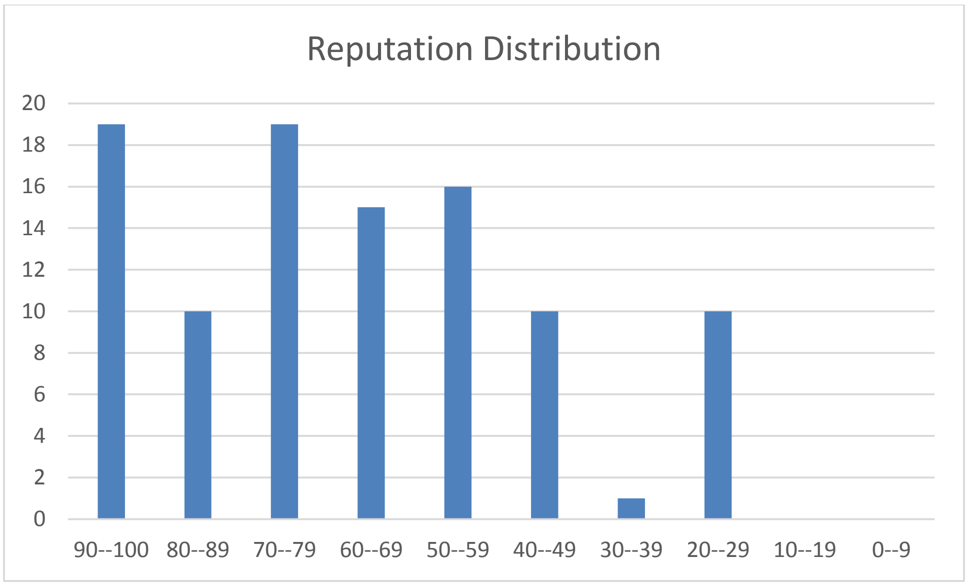
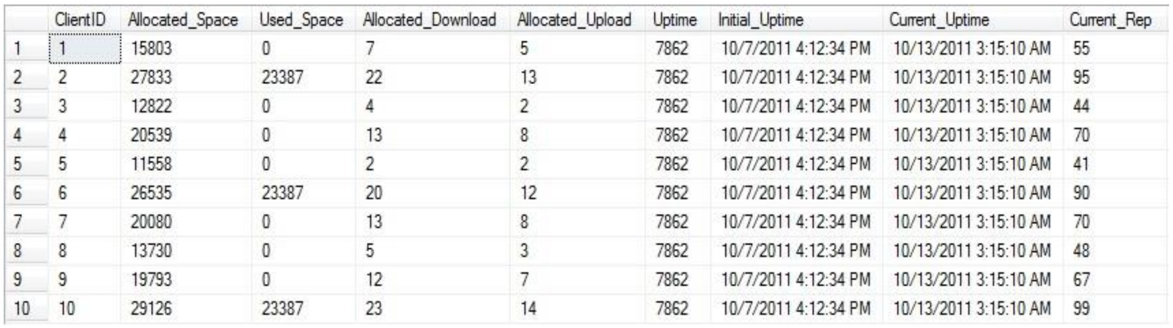


Figure 9. Reputation Distribution

In *Table 1* we can see various information that is stored for a peer on the server. CLIENTID is a unique identifier for each peer connected to the network. This is used for storing all the data related to a peer. UPTIME is the total time in minutes a peer has been online on the network. INITIAL_UPTIME is the date time when the peer first connected to the network. This is also used to calculate server uptime in *Equation 1* for the peer. CURRENT_UPTIME stores the date time last reputation calculation was done. CURRENT_REP is the current reputation of the peer.

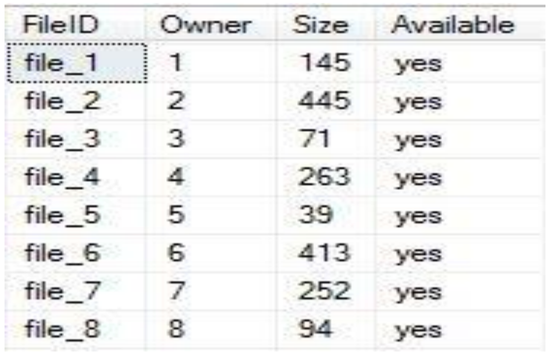
Table 1. Various stored Information about a Peer



	ClientID	Allocated_Space	Used_Space	Allocated_Download	Allocated_Upload	Uptime	Initial_Uptime	Current_Uptime	Current_Rep
1	1	15803	0	7	5	7862	10/7/2011 4:12:34 PM	10/13/2011 3:15:10 AM	55
2	2	27833	23387	22	13	7862	10/7/2011 4:12:34 PM	10/13/2011 3:15:10 AM	95
3	3	12822	0	4	2	7862	10/7/2011 4:12:34 PM	10/13/2011 3:15:10 AM	44
4	4	20539	0	13	8	7862	10/7/2011 4:12:34 PM	10/13/2011 3:15:10 AM	70
5	5	11558	0	2	2	7862	10/7/2011 4:12:34 PM	10/13/2011 3:15:10 AM	41
6	6	26535	23387	20	12	7862	10/7/2011 4:12:34 PM	10/13/2011 3:15:10 AM	90
7	7	20080	0	13	8	7862	10/7/2011 4:12:34 PM	10/13/2011 3:15:10 AM	70
8	8	13730	0	5	3	7862	10/7/2011 4:12:34 PM	10/13/2011 3:15:10 AM	48
9	9	19793	0	12	7	7862	10/7/2011 4:12:34 PM	10/13/2011 3:15:10 AM	67
10	10	29126	23387	23	14	7862	10/7/2011 4:12:34 PM	10/13/2011 3:15:10 AM	99

In *Table 2,* we can see the information stored for a file. FILEID is the name of the file as it is stored on a computer, and this is what is used when a user searches for a file. OWNER is the peer which has the file, SIZE is the file size in kilobyte and AVAILABLE holds whether the file is currently available on the network. By availability, I mean whether the peer who has the file is online or not.

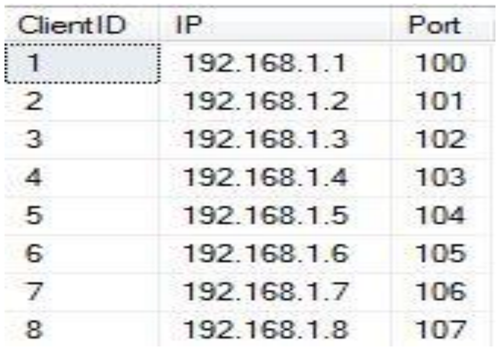
Table 2. File Information stored on Server



FileID	Owner	Size	Available
file_1	1	145	yes
file_2	2	445	yes
file_3	3	71	yes
file_4	4	263	yes
file_5	5	39	yes
file_6	6	413	yes
file_7	7	252	yes
file_8	8	94	yes

In *Table 3* we can see how the information required for connecting to a peer is stored. This basically is all of the information required for connecting to a peer. This information is sent by the server to the clients so that they can connect to peers directly and start the download process. Server uses this information to do replication process in background.

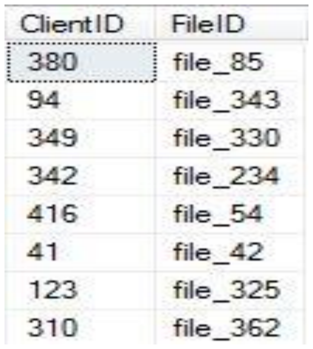
Table 3. The Connected Peers



ClientID	IP	Port
1	192.168.1.1	100
2	192.168.1.2	101
3	192.168.1.3	102
4	192.168.1.4	103
5	192.168.1.5	104
6	192.168.1.6	105
7	192.168.1.7	106
8	192.168.1.8	107

In *Table 4* we can see to whom a file has been replicated. CLIENTID is the unique identifier for a peer. The server uses this table in conjunction to the Files table (*Table 2*) to know which peers have the requested file.

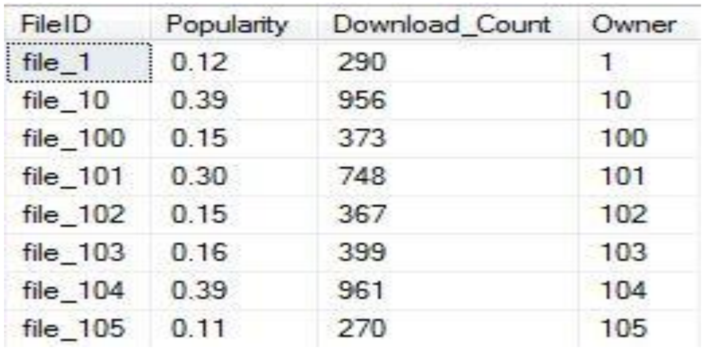
Table 4. The Replication Table



ClientID	FileID
380	file_85
94	file_343
349	file_330
342	file_234
416	file_54
41	file_42
123	file_325
310	file_362

In *Table 5* we can see the information that is stored regarding the popularity of the file. FILEID uniquely identifies a file on the network, POPULARITY is the calculated popularity of the file based on the number of times the file has been downloaded, DOWNLOAD_COUNT is the number of times the file has been downloaded and OWNER is the CLIENTID of the peer storing the file.

Table 5. The File Popularity Table



FileID	Popularity	Download_Count	Owner
file_1	0.12	290	1
file_10	0.39	956	10
file_100	0.15	373	100
file_101	0.30	748	101
file_102	0.15	367	102
file_103	0.16	399	103
file_104	0.39	961	104
file_105	0.11	270	105

Figure 10 depicts the relationship between file popularity and number of downloads. As the download count for a file increases, the popularity also increases as calculated using *Equation 2*. Y axis represents the file popularity percent and the X axis represents the download count.

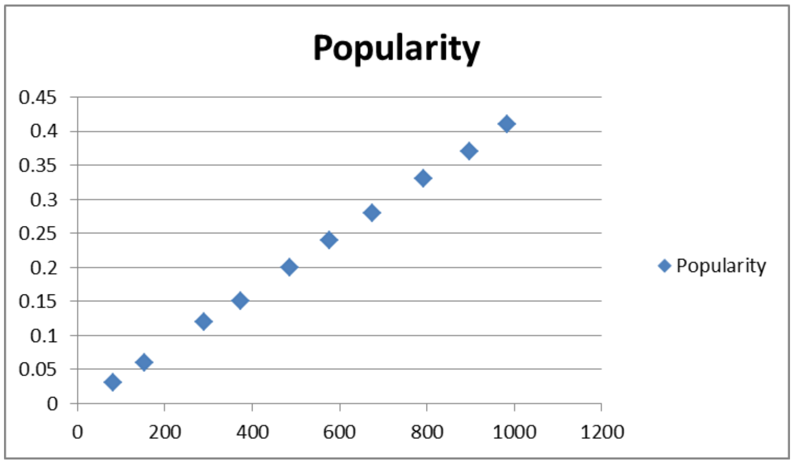


Figure 10. File Popularity with Respect to Number of Downloads

Figure 11 depicts the relationship between file popularity and file replication. Only the most popular files are replicated to increase their availability and fulfill incentives promised to the peers. In *Figure 11* the files that had popularity more than 0.35 are replicated as they fall within the top 90th percentile bucket so we see that they are now present on 200 peers as compared to 1 peer before replication. The y-axis represents File popularity and x-axis represents the number of times a file has been replicated.

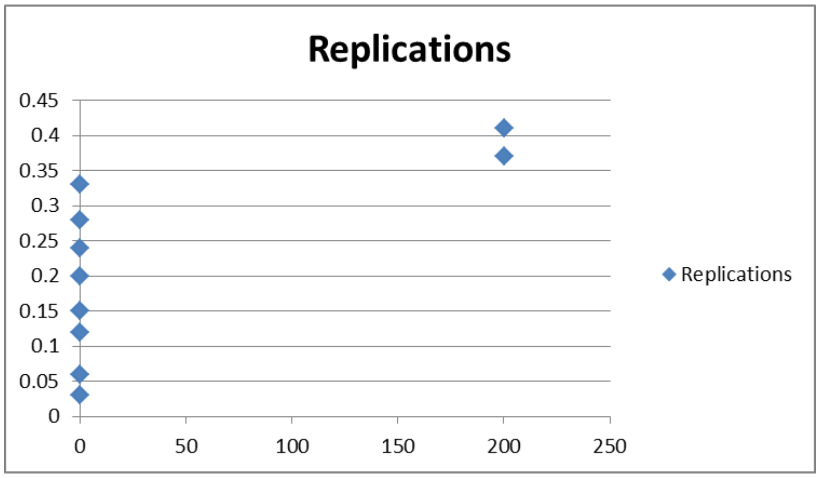


Figure 11. File Replication with Respect to File Popularity

Figure 12 depicts the relationship between popularity and replicated file count. The red line shows files from all popularity ranges were available on individual peers before the replication. The blue line depicts the increase in file count to 201 peers from 1 peer for the most popular files starting at 0.37. Which is the starting of top 90th percentile files.

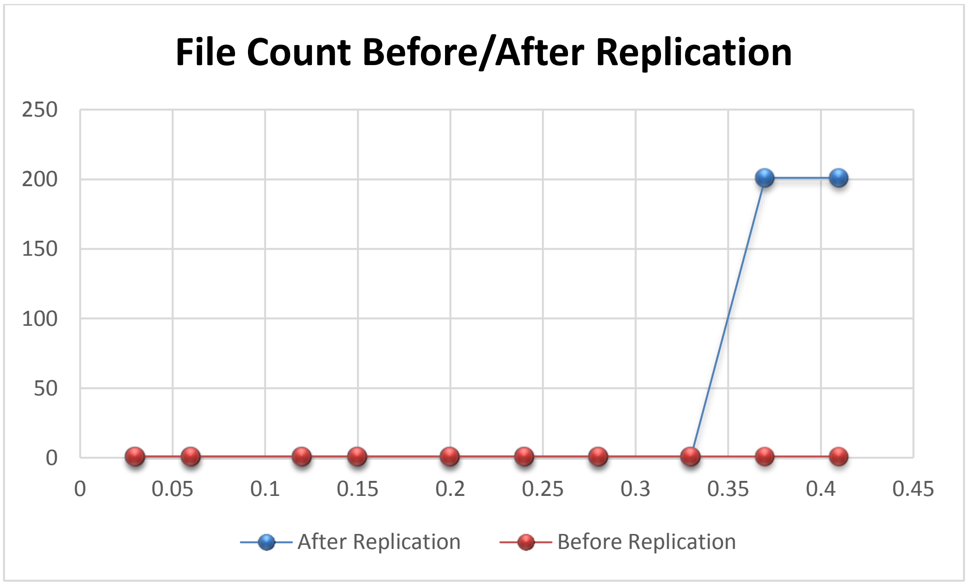


Figure 12. File Availability after Replication

5. CONCLUSION AND FUTURE WORKS

Peer to peer networks allow for sharing of files that are collectively contributed by all the members but the availability depends mainly on the presence of the owner on the network. The replication and incentive scheme discussed here is a technique for increasing file availability on the network. This is achieved in 2 parts: firstly, the incentive of downloading a file fast is achieved by motivating the peer to be online and available to the network longer, thus, also increasing the availability of files hosted by that peer. Secondly, by motivating the user to provide disk space for replication purposes, thus, increasing reputation which leads to faster file downloads and making the most popular files more available. A reputation system has been introduced which calculates the reputation of a peer using the allotted disk space to for replication, download speed, upload speed and the uptime on the network. This reputation is then used to decide from how many peers a file can be downloaded in parallel by a certain peer and to whom the most popular files should be replicated. This scheme of replication did increase the availability of files that were most popular on the network.

In the future, I would like to move this whole system to a DHT based P2P network so that the reputation is calculated by other peers instead of a centralized server and implement some technique to verify from the OS that the disk space allotted for replications is actually available on the drive. The current system, tracks the file popularity by the number of times it was downloaded, and I would like to enhance this tracking by also monitoring how much the file is searched for by the users. This way the system can have an idea of which files might be good candidates for replication. Also I would like to revise the reputation equation to give different weights to each parameter to have better reputation system. I would also make changes to the server code so that it preserves the state during reboots. Since in my experiment, I only had one server which could have been a single point of failure, I would like to invest some time in clustering so that the network has several servers which are in sync with each other and can be used to service many clients.

REFERENCES

[1] B. Mortazavi, G. Kesidis,"Cumulative Reputation Systems for Peer-to-Peer Content Distribution", in Information Sciences and Systems, 2006 40th Annual Conference on

March 22-24, 2006 pages 1546-1552.

[2] B.Q. Zhao, J.C.S. Lui, D. Chiu, "Mathematical Modeling of incentive policies in p2p systems", Applications, Technologies, Architectures, and Protocols for Computer Communication Proceedings of the 3rd international workshop on Economics of networked systems, pages 97-102, 2008.

[3] C. Ye, D.M. Chiu, “Peer-to-peer Replication with Preferences”, Infoscale 2007, Suzhou,China, June 2007.

[4] M. Feldman, K. Lai, I. Stoica, and J. Chuang. “Robust incentive techniques for peer-to-peer networks”, In 5th ACM conference on Electronic commerce, pages 102–111, 2004.

[5] S. Androutsellis-Theotokis and D. Spinellis. “A survey of peer-to-peer content distribution technologies”, ACM Computing Surveys (CSUR), 36(4), December 2004.

[6]. Yingwu Zhu and Yiming Hu. Efficient, Proximity-Aware Load Balancing for Structured P2P Systems.

[7]. Ananth Rao, Kartik Lakshminarayanan, Sonesh Surana, Richard Karp and Ion Storica. Load Balancing in Structured P2P Systems.

[8]. Karl Aberer, Philippe Cudré-Mauroux, Anwitaman Datta, Zoran Despotovic,

Manfred Hauswirth, Magdalena Punceva, Roman Schmidt. P-Grid: A Self-organizing Structured P2P System.

[9]. Zheng Zhang, Shu-Ming Shi and Jing Zhu. SOMO: Self-Organized Metadata Overlay

for Resource Management in P2P DHT.

[10]. Vivek Vishnumurthy and Paul Francis. A Comparison of Structured and Unstructured P2P Approaches to Heterogeneous Random Peer Selection.

[11]. Toshinori Takabatake and Yoshikazu Komano. A P2P File Sharing Technique by Indexed-Priority Metric.

[12]. Sabu M Thampi and Chandra Sekaran K. Survey of Search and Replication Schemes in Unstructured P2P Networks.

[13]. Edith Cohen, Scott Shenker. Replication strategies in unstructured peer-to-peer networks.

[14]. Razvan Andonie, Luke Magill, James Schwing, Ben Sisson, and Brandon Wysocki. An Adaptive Replication Algorithm in P2P File Systems with Unreliable Nodes.

[15]. Mao Yang，Qinyuan Feng，Yafei Dai，Zheng Zhang. A Reputation Scheme Combining Trust and Incentive for P2P File Sharing.

[16]. Roslan Ismail, Colin Boyd, Audun Josang and Selwyn Russell. Private Reputation Schemes for P2P Systems.

[17]. Minaxi Gupta, Paul Judge, Mostafa Ammar. A Reputation System for Peer toper Networks.

[18]. Pradipta Mitra. Reputation Management in P2P Systems.

[19]. Wikipedia, http://en.wikipedia.org/wiki/File:P2P-network.svg.

[20]. Wikipedia, http://en.wikipedia.org/wiki/File:Server-based-network.svg.

[21]. Wikipedia, http://en.wikipedia.org/wiki/File:DHT_en.svg.