Journal of Multidisciplinary Engineering Science and Technology (JMEST)
ISSN: 3159-0040
Vol. 1 Issue 5, December - 2014

Information System for Designing a Robust and Structured P2P File Sharing with NAT Support

Md. Tareq Hasan,
Institute of Information Technology (IIT),
Jahangirnagar University, Dhaka, Bangladesh.

M. Mesbahuddin Sarker,
Faculty, Institute of Information Technology (IIT),
Jahangirnagar University, Dhaka, Bangladesh.

***Abstract*—With the pervasive deployment of computers, P2P is increasingly receiving attention in research, product development and investment circles. This interest ranges from enthusiasm, through hype, to disbelief in its potential. Some of the benefits of a P2P approach include: improving scalability by avoiding dependency on centralized points; eliminating the need for costly infrastructure by enabling direct communication among clients; and enabling resource aggregation. This paper describes the basic requirements for peer-to-peer file sharing and presents Chord, a distributed lookup protocol that focuses that it is scalable, with communication cost.**

Keywords: P2P Network, Chord Model, NAT Support, Internet Migration.

I. Introduction

Peer-to-peer (P2P) computing is the computer resources and services sharing within computers by direct exchange. These resources and services are the exchange of information, processing cycles, cache storage, and disk storage for different types of files. P2P computing subsumes concepts used in communication technologies, distributed system models, applications, platforms, etc. It is not a particular initiative, nor architecture or specific technology; it rather describes a set of concepts and mechanisms for decentralized distributed computing and direct peer-to-peer information and data interchange [1,2]. The P2P systems hinges upon the concept of decentralization – that is the systems are distributed computing models that enable decentralized collaboration by integrating computers (and/or any devices that are capable of communicating among themselves) in to networks in which each can consume and offer services. According to Clay Shirky [3], “P2P is a class of applications that takes advantage of resources – storage, cycles, content, human presence – available at the edges of the Internet”. Simply put any networking model that shifts processing and/or storage responsibility from the server to the clients and encourages the clients (i.e. peers) to communicate among them possibly without any central control is P2P. It has many benefits like efficient use of resources and unused bandwidth, storage, processing power at the edge of the network. Reliability is another benefit here includes replicas, geographic distribution and no single point of failure. Ease of administration is big advantages in peer to peer network where nodes are self organize, no need to deploy servers to satisfy demand, built-in fault tolerance, replication, and load balancing. P2P systems are two different classes:

i. Structured P2P systems, and

ii. Unstructured P2P systems.

In structured P2P systems, fixed connections exit among peers in the network, and information maintained by the peers about the resources (e.g., shared content) that possessed by their neighbor peers. Hence, the data queries can be efficiently directed to the neighbor peers that have the desired data, even if the data is extremely difficult to find. Structured P2P systems impose constraints both on node (peer) graph and on data placement to enable efficient discovery of data. The most common indexing that is used to structure P2P systems is the Distributed Hash Tables (DHTs) indexing. Similar to a hash table, a DHT provides a lookup service with (key, value) pairs that are stored in the DHT. In a centralized unstructured P2P system, a central entity is used for indexing and bootstrapping the entire system. In contrast to the structured approach, the connection between peers in the centralized unstructured approach is not determined by the central entity. A “BitTorrent” network is an example of a centralized unstructured P2P network [4]. BitTorrent uses tit-for-tat policy. BitTorrent is a popular peer-to-peer file sharing protocol that was created by Cohen Bram [5]. BitTorrent has been shown to scale well with large number of participating end hosts. Ipoque (http://www.ipoque.com/) measurements for years 2008-2009 show that BitTorrent is the dominant protocol in the Internet, and that it accounted for approximately 20-57% of all Internet traffic depending on the geographical location.

II. Terms & Definition

A. P2P Network

P2P networks typically used for file sharing applications, which allow the peers to share digitized data like general documents, audio data, video data, electronic books etc. Here each peer is both a client and a server. Now it has many advanced applications like real-time conferences, online gaming, and media streaming have also been deployed over peer-to-peer networks.

B. Client Server Network

Normally, interaction held with applications over a network using client- server architecture. Web sites are a great example of this. When browsing a web site we normally send a request over the Internet to a targeted web server, which then returns the information to us that we require. If we need to download a file from the server, we do it directly from the web server. Similarly, desktop applications that include local or wide area network connectivity will typically connect to a single server, for example, a database server or a server that hosts other services.

C. NAT Support

Network Address Translation (NAT) technique was developed to allow the use of a single IP address for a whole network of computers. NAT sits in between the public Internet and the network it serves, and works by rewriting IP addresses and port numbers in IP headers on the fly so the packets all appear to be coming from (or going to) the single public IP address of the NAT device instead of the actual source or destination. NAT is now commonly employed in small home-office routers and in software used by consumers to connect several personal computers to a single cable modem.

D. Internet Migration

The target environment for P2P consists of the Internet, intranets, and ad-hoc networks. The most frequent environment is personal home computers connected to the Internet. The early P2P systems in this environment were primarily used for content sharing. Examples include Napster, Gnutella, and Aimster. Distributed computing in a P2P fashion also started on desktop computers on the Internet such as SETI@home, are expected to follow with a wider deployment of handheld computers & wireless network.

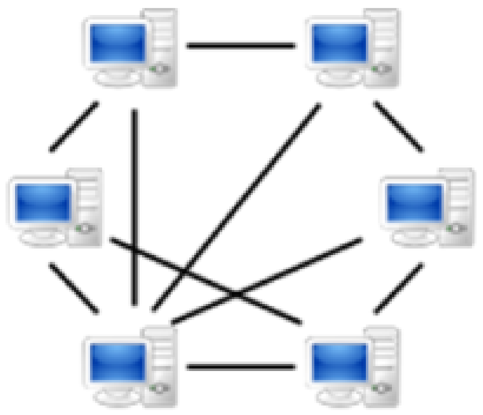

Figure 1: A Peer-to-peer system of nodes without central infrastructure

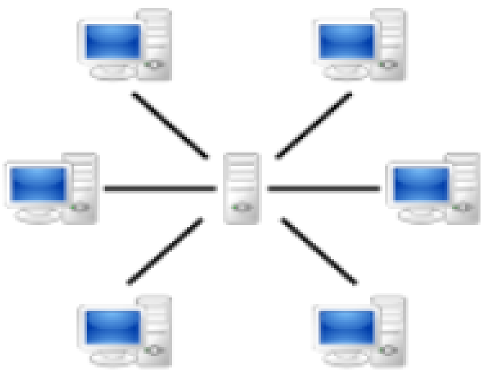

Figure 2: A Peer-to-peer system of nodes with central infrastructure

III. Related Works

There has been previous work in the area of decentralized location systems. Chord is based on consistent hashing; its routing information may be thought of as a one dimensional analogue of the GRID [6] location system. OceanStore uses a distributed data location system described by Plaxton et al. [7], which is more complicated than Chord but offers proximity guarantees. The Chord algorithm is also very similar to the location algorithm in PAST [8]. Many Peer to Peer applications are available which work on the computer and mobile, such as Gnutella, Napster, Bittorent, and SymTorrent. File sharing causes a lot of the traffic on the network, thus some of the technology is used to reduce the traffic and find the files easily. Mobile devices are becoming multifunctional, so why not create a peer-to-peer file sharing system between the mobile devices. Adel Ali Al-zebari [9] explains some existing systems such BitTorrent, Napster, SymTorrent. Besides, recent years have witnessed the advent of large scale real-world peer-to-peer applications such as Kazaa, Morpheus, BitTorrent, and many others. Several distributed hash tables (DHTs) have been introduced which are provably robust to random peer deletions. Different exixting systems are describe below:

A. Bayeux

Another proposal for doing multicast using general-purpose overlay networks is the Bayeux [10] system for Tapestry. Bayeux builds a multicast tree per group differently from the approach examined in this paper. Each request to join a group is to the root, which records the identity of the new member and uses Tapestry to route another message back to the new member. Every Tapestry node along this route records the identity of the new member. Requests to leave the group are handled in a similar way. This introduces two scalability problems when compared to tree based approach used here.

Firstly - It requires nodes to maintain more group membership information.

Secondly - Bayeux generates more traffic when handling group membership changes.

B. BitTorrent

BitTorrent is a peer-to-peer file sharing protocol designed by Bram Cohen it has been one of the most popular file sharing systems during the last few years used for distributing the content of data. As Guo et al. reported, according to last measurement by CacheLogic, BitTorrent traffic on the internet represented 53% in June 2004. BitTorrent shares the same type of files between peers through the internet unlike the Gnutella, eDonkey, and eMule that share different files. BitTorrent is a peer-to-peer distribution network and this kind of network uses the bandwidth of its users.

C. Freenet

The Freenet peer-to-peer storage system [11], like Chord, is decentralized and symmetric and automatically adapts when hosts leave and join. Freenet does not assign responsibility for documents to specific servers; instead, its lookups take the form of searches for cached copies. This allows Freenet to provide a degree of anonymity, but prevents it from guaranteeing retrieval of existing documents or from providing low bounds on retrieval costs. Chord does not provide anonymity, but its lookup operation runs in predictable time and always results in success or definitive failure.

D. Gnutella

The Globe system [12] is one of the most popular peer-to-peer file sharing systems to date used to exchange and share data between the peers. It has a wide-area location service to map object identifiers to the locations of moving objects. Globe arranges the Internet as a hierarchy of geographical, topological, or administrative domains, effectively constructing a static world-wide search tree, much like DNS. Information about an object is stored in a particular leaf domain, and pointer caches provide search short cuts. The Globe system handles high load on the logical root by partitioning objects among multiple physical root servers using hash-like techniques. Chord performs this hash function well enough that it can achieve scalability without also involving any hierarchy, though Chord does not exploit network locality as well as Globe.

E. Napster

In 1999, Shawn Fanning created a technology to share music files over the internet that allowed users to share music. The architecture of Napster was based on a central server in the network. Therefore, a peer first joins a network by connecting to a central point known as a broker. A peer passes all the music files that it makes available to the server. The server stores the received information about the music files in the database and the information of all the peers that currently logged into the server is stored in the database. A peer sends a query to the server for a particular file then the server sends back a list of files and a list of peers that have the same file. The advantage of the Napster network is that the query is efficient and finding the files guaranteed. On the other hand, all the participating peers depend on the central server and if this server goes down the network become useless [13].

F. Ohaha

The Ohaha system uses a consistent hashing-like algorithm for mapping documents to nodes, and Freenet-style query routing. As a result, it shares some of the weaknesses of Freenet. Archival Inter memory uses an off-line computed tree to map logical addresses to machines that store the data [14]. The Ohaha system uses a consistent hashing-like algorithm for ID mapping, and a Freenet-style method of document retrieval; it shares some of the weaknesses of Freenet [15].

G. Overcast

Another scalable overlay multicast system is Overcast [16]. Like Bayeux, Overcast requires that joining nodes coordinate with a central root node. A significant amount of work has also gone into overlay networks and application-level multicast systems not designed to scale, such as Resilient Overlay Networks (RONs) [17], End System Multicast [18], and ISIS/Horus-style Virtual Synchrony [19], but which provide other benefits. Of course, all the work for constructing multicast distribution trees builds upon the techniques originally developed for IP Multicast [20].

H. SymTorrent

SymTorrent is the only BitTorrent client for mobile Symbian OS. The target platform for SymTorrent is S60 3rd and 5th edition and it is freely available under the terms of GNU General Public License. The UI in SymTorrent is built independently as part of an application in a separate DLL. This means that the porting to different devices is easier. Recently, SymTorrent has been downloaded more than twenty thousand times and the users used for downloading files between each other through GPRS and WLAN.

IV. Client-Server Vs P2P Systems

Peer-to-peer networking has recently emerged as a new paradigm for building distributed networked applications. The peer-peer approach differs from the traditional client/server approach towards building networked applications in several crucial ways. Perhaps most importantly, a peer is both a producer and a consumer of the implemented service. In a peer-peer file-sharing application, for example, a peer both requests file from its peers, and stores and serves files to its peers. A second important difference is that a peer’s lifetime in the system is transitory — a peer may be active in the system for some time (both generating requests and serving the requests of others) and then go off-line, removing itself from the system. Considerable research has been devoted to developing a fundamental understanding of the performance of traditional client-server applications [21].

There is no clear border between a client-server and a P2P model. Both models can be built on a spectrum of levels of characteristics (e.g. manageability, configurability), functionality (e.g. look-up versus discovery), organizations (e.g. hierarchy versus mesh), components (e.g., DNS), and protocols (e.g. IP), etc. Furthermore, one model can be built on top of the other or parts of the components can be realized in one or the other model. Finally, both models can execute on different types of platforms (Internet, intranet, etc.) and both can serve as an underlying base for traditional and new applications. Downloading mechanism between Client-Server model and P2P modeling may perform in following steps:

Client-Server Model	P2P Model
i. Requester contacts a server; ii. Server then downloads from the appropriate client; and iii. Requester then downloads it from the server.	i. A peer requests another peer for a specified file; ii. It then forwards the request to next peer; iii. When the specified file is found then search result is send to the requestor peer; and iv. Then it can download the requested file.

Mathematical Relation Between Client-Server & P2P System are shown below: If we consider that every request takes T_{req}, the time for searching is T_{search} and download time between two nodes is T_D, hence time taken for overall process is,

$$^{client-server}T_{total} = 2*T_{req} + T_{search} + 2*T_D$$

On the other hand, assume that in a P2P system (flooding approach is not applicable); to download an item, request has to travel through N numbers of nodes. Then total time taken is,

$$^{P2P}T_{total} = N*T_{req} + \sum_{i=1}^{N} T_i + T_D \text{ for worst case.}$$

$$^{P2P}T_{total} = T_{req} + T_1 + T_D \text{ for best case.}$$

Where, T_i : Search time for ith node;

T_D : Download time between two nodes; and

T_{req} : Request time between two nodes.

For the first case, if we consider T_{req} and T_{search} are negligible comparing with $2*T_D$ and the server is extremely powerful, we can rewrite the equation for client-server system to,

$$^{client-server}T_{total} = 2*T_D$$

For the second case, if we consider the search time in every peer is equal to T_s then we can rewrite the equation for P2P system to,

$$^{P2P}T_{total} = N*(T_{req} + T_s) + T_D$$

From above equations, it is clear that total time is constant for client-server model. In P2P system total time can be reduced if traveling nodes N can be reduced. The condition involved to perform better in P2P system over client-server model is,

$$N*(T_{req} + T_s) \langle T_D$$

So, by using efficient algorithm, N can be reduced and thus the performance of P2P system is improved considerably.

V. Model Choosing

It is observed that the CIA (Centralized Indexing Architecture) [22] outperforms DIFA (Distributed Indexing with Flooding Architecture) and is slightly better than DIHA (Distributed Indexing with Hashing Architecture) when the population size is small. This is primarily due to the higher capacity of the centralized query lookup. This trend persists until the central server in CIA becomes the bottleneck. At higher population sizes, DIFA and DIHA perform better than CIA, since the serving capacity in these systems scales with the increase in the population. DIFA however suffers from another drawback: the probability of query failure increases with the population size. This occurs since queries can only reach a bounded number of peers, which is independent of the population size, which implies that queries only search over a bounded subset of the index space. Thus, DIFA cannot capitalize on the potential capacity of peer-peer systems, as queries that could have succeeded fail. This limits the effective throughput of DIFA architecture, resulting in a lower performance than DIHA in terms of system throughput.

The choosing of the model depends on how the model would behave on the level of required outcomes coming from the network architecture. P2P network is designed for various purposes. One of them is distributed file sharing. In order to implement file sharing, network it can be designed in a structured way (document routing model). That is, various operations like node joining, leaving and locating or placing identifier key of files etc. are executed in a systematic order. Various structured algorithm exists nowadays. There are many implementations of this interface such as Chord, Pastry, Tapestry, Content Addressable Network (CAN), SkipNet, Kademlia, Viceroy, P-Grid, Freenet. A comparison among Chord with models are shown below:

Table 1: Comparison of Chord with other Models

Models	Functions
Chord & Free Net:	- Chord & Free net, both are decentralized and symmetric. - Both automatically adapt when hosts are leaving and joining
Chord & Pastry:	- Pastry (Location algorithm used by PAST system): Similar to Chord, but uses prefix-based routing protocol. - Chord: Each node and each key's id is hashed to a unique value. The process of look up tries to find the immediate successor to a key's id. The routing table at each node contains O (log (n)) entries. Inserting and deleting nodes requires O (log (n) 2) messages. More robust than Pastry, and more elegant. -
Chord & OceanStore:	- Ocean Store is very close to Chord protocol, stronger guaranties than chord. But it is so complicated than chord. Chord is less complicated. -
Chord & Napster:	- Compared to Napster and its centralized servers, Chord avoids single points of control or failure by a decentralized technology. Napster is centralized (contain single point of failure).
Chord & Gnutella	- Compared to Gnutella, Chord avoids the lack of scalability through a small number of important information for routing. - Gnutella is Inefficient (searching by flooding) where chord is efficient.

Model	Characteristics
Chord	- Solves problem of locating a data item in a collection of distributed nodes, considering frequent node arrivals and departures. - Core operation in most P2P systems is efficient location of data items. - Supports just one operation: given a key, it maps the key onto a *node.* - Simplicity, *provable* correctness, and *provable* performance. - Each Chord node needs routing information about only a few other nodes. - Resolves lookups via messages to other nodes (iteratively or recursively). - Maintains routing information as nodes join and leave the system. - Traditional name and location services provide a direct mapping between keys and values. - Chord can easily implement a mapping onto values by storing each key/value pair at node to which that key maps. - Does not provide anonymity. - But its lookup operation runs in predictable time and always results in success or definitive failure.
Free Net	- Provide anonymity. - Prevents guaranteed retrieval of existing documents

VI. Chord Mapping Process

A peer-to-peer lookup system. Given a key (data item), it maps the key onto a node (peer).

i. Uses consistent hashing to assign keys to nodes,

ii. Solves problem of locating key in a collection of distributed nodes, and

iii. Maintains routing information as nodes join and leave the system.

Some addressed problems, solved by chord:

i. **Load balance**: Distributed hash function, spreading keys evenly over nodes,

ii. **Decentralization**: Chord is fully distributed, no node more important than other, improves robustness,

iii. **Scalability**: Logarithmic growth of lookup costs with number of nodes in network, even very large systems are feasible,

iv. **Availability**: Chord automatically adjusts its internal tables to ensure that the node responsible for a key can always be found, and

v. **Flexible naming**: No constraints on the structure of the keys – key-space is flat, flexibility in how to map names to Chord keys.

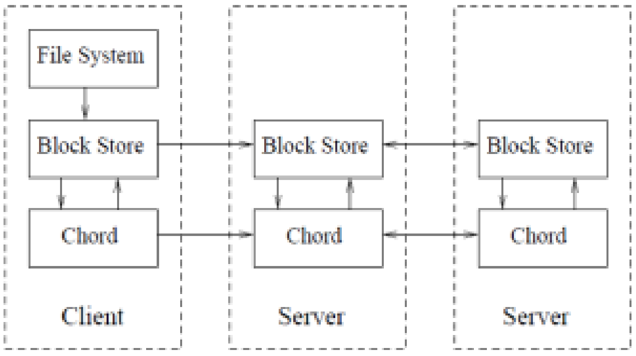

Figure 2: Structure of Chord-based distributed storage system.

The Chord protocol takes as input an L-bit identifier (derived by hashing a higher-level application-specific key), and returns the node that stores the value corresponding to the key [23]. Each Chord node is identified by an L–bit identifier and each node stores the key identifiers in the system closest to the node's identifier. Each node maintains an L–entry routing table that allows it to look up keys efficiently. Results from theoretical analysis, simulations, and experiments show that Chord is incrementally scalable, with insertion and lookup costs scaling logarithmically with the number of Chord nodes. Let L be the number of bits in the binary representation of key/node identifiers. Each node, n, maintains a routing table with L entries, called the *finger table*. The ith entry in the table at node n contains the identity of the *first* node, s, that succeeds n by at least 2^i on the identifier circle, i.e., s is the successor of $(n+2^i)$, where$1 \le i \le L$ (and all arithmetic is modulo 2^L). We call node s the ith *finger* of node n. There are two observations of this scheme:

i. **First :** Each node stores information about only a small number of other nodes, and the amount of information maintained about other nodes falls off exponentially with the distance in key-space between the two nodes; and

ii. **Second :** The finger table of a node may not contain enough information to determine the successor of an arbitrary key k.

What happens when a node n does not know the successor of a key k? To resolve this, node n asks another node in the network to try and find k's successor. Node n, aims to find a node closer to k than n, as that node will have more "local information" about the nodes on the circle near k. To accomplish this task, node n searches its finger table for the closest finger preceding k, and forwards the query to that node. As a result the query moves quickly to the target identifier.

VII. Node Joining and Leaving/Failure

A. Node Joining

In a dynamic network, nodes can join and leave at any time. The main challenge of implementing these operations is preserving the ability to locate every key in the network. To achieve this goal, we need to preserve two invariants:

i. Each node's finger table is correctly filled.

ii. Each key k is stored at its corresponding successor node.

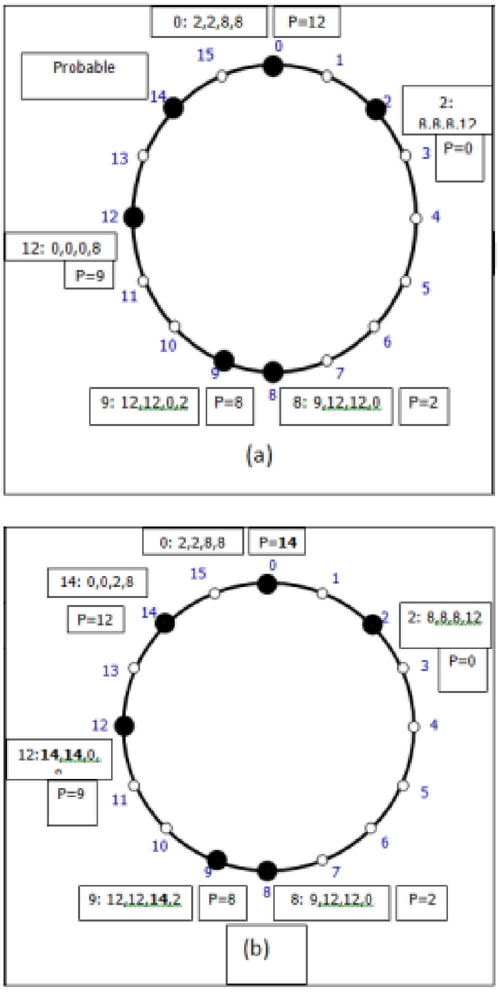

Figure 3:

(a) The finger tables associated to each node before node 14 joins the network.

(b) The finger tables associated to each node after node 14 joins the network.

The changes in the finger tables stored by each node as a result of a node joining are shown in dark; the unchanged entries are shown in light. Suppose the network is in stabilized position. And new node 14 tries to enter the network. In this case the following steps will be occurred:

i. New node 14 will get finger table from its predecessor (12);

ii. It will construct its finger table by successive appropriate queries by using 12's finger table;

iii. Then node 14 informs it's predecessor (12) that it successfully constructed its finger table;

iv. Node 12 will inform its immediate successor (0) that it (0) should change its predecessor to 14; and

v. The built-in stabilization process altars necessary changes to the other nodes if needed; as node 9 changes one of its entries to 14.

B. Node Leaving/Failure

Node Leaving can be categorized to two kinds; one in which a node willingly leaves the network, and the other where a node is not available due to failure in the communication path or system itself. Node leaving s automatically maintained by the system with the help of Tracker system.

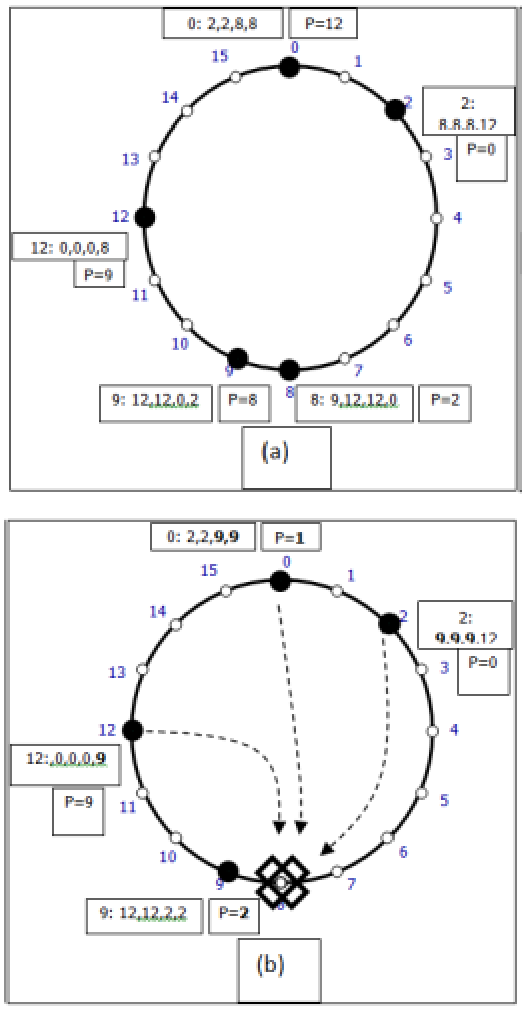

Figure 4:

(a) The finger tables associated to each node before node 8 leaves the network.

(b) The finger tables associated to each node after node 8 leaves the network.

The changes in the finger tables stored by each node as a result of a node joining are shown in dark; the unchanged entries are shown in light. The built-in stabilization process runs in every node of the network. This is like every node tries to knock every entry of its finger table requesting periodically for that entry's immediate successor and predecessor. If any entry cannot be reached; it assumes the node fails. So it alters that corresponding entry to the immediate successor of the failure node which had already been saved in the previous attempt. Here, node 8 contains 12, 12, 0 & 2 nodes as its successor. Similarly 9 contain 12, 2, 0, and 2; 12 contain 0, 0, 0 & 8. And 0 contain 2, 2, 8 & 8. Suppose node 8 fails .As every node knocks to its successors periodically so they will unable to connect to node 8. So these nodes will alter the necessary changes to their corresponding successor lists.

VIII. File Sharing

File sharing is the practice of distributing or providing access to digital media, such as computer programs, multimedia (audio, images and video), documents or electronic books. File sharing may be achieved in a number of ways. Common methods of storage, transmission and dispersion include manual sharing utilizing removable media, centralized servers on computer networks, World Wide Web-based hyper linked documents, and the use of distributed peer-to-peer networking. Following are key characteristics of file sharing:

A. File Key Generation

File Key or simply the key is the hash code generated by the same hashing mechanism used for deriving a peer ID. We may consider the file name, criteria, or content for the hashing purpose. SHA-1 function can be an efficient method for both key and peer ID generation [24].

B. File Key Distribution

Each time a file is given share, or any new peer joins or leaves, the key is redistributed and updated in the corresponding peers. The keys are available to only that node in the network whose Peer ID is immediately greater than or equal to the key. For this purpose, the keys are forwarded by the same mechanism as searching and using the Finger Table until such criterion is met. Besides finger table each has to maintain a table containing the keys. Each key corresponds to the original location of the file.

C. File Searching

Efficient file searching can only be done after ensuring the key distribution properly. Whenever a search request is on, the key for the corresponding file is generated and forwarded using the Finger Table. At each peer, it is checked whether the key is less than the Peer ID. If not, the file cannot be in that peer, and the request is forwarded to the peer having the closest ID less than the key. Whenever a node is found whose ID is greater than the key, the forwarding is stopped, and if the file is in the network, the key for it must belong to that peer.

IX. NAT Mishap & Proposed Solution

First requirement of any structured P2P network system is the uniqueness of the nodes which is implemented using associated IP of the node. Since it is desirable that every PC in the internet uses unique IP it should imply that the ID obtained from IP be unique for every node. However, NAT violates the very first requirement of the Internet by allowing more than one PC to connect using only one IP. This is implemented using a NAT router having a global IP. The network behind the NAT router is considered to be a private network having a fixed well-defined range of IPs. The NAT router is responsible for maintaining the connections between a computer behind the NAT and the rest of the Internet.

The whole scenario becomes much problematic for a structured implementation of a P2P network, since we cannot initiate a connection to the computers behind the NAT router from outside. So a different ploy has to be taken in order to allow the computers under the private networks to participate in the P2P networks.

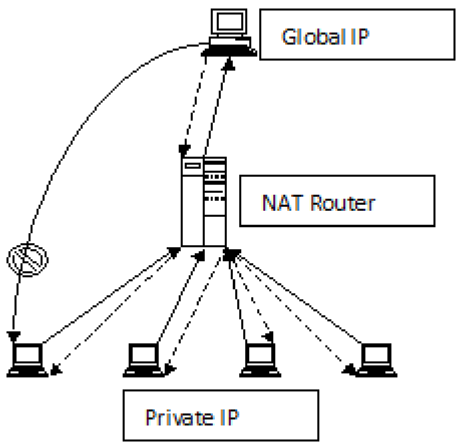

Figure 5: Global IP vs. private IP

An alternate way that can be proposed to solve this problem is to consider the computers behind NAT as irregular nodes. The properties of an irregular node are as follows:

i. An irregular node will have a private IP address within a private network;

ii. Connection initiation is always one way that is from private IP to global IP;

iii. An irregular node will not be considered as a participant in the P2P ring; rather a node that has been knocked by it will supervise it, and hence will be an intermediate node for all the requests destined for any node behind the NAT; and

iv. The supervising node will take special care for an irregular node and maintain the uniqueness within the same supervising node.

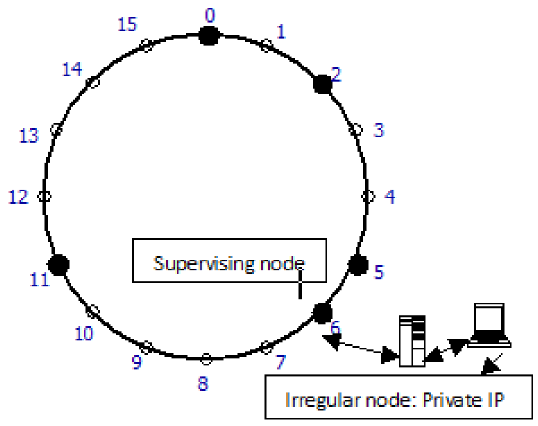

Figure 6: Node behind NAT

X. Conclusion

Peer-to-peer (P2P) networking eliminates the need for central servers, allowing all computers to communicate and share resources as equals. Music file sharing, instant messaging and other popular network applications rely on P2P technology. P2P computing is currently a hectic research area in the field of computer networking. As the problems related to it start to be solved the technology will be adopted more and more by the business and scientific community as well. P2P systems have been said to be “the poor man’s supercomputer”. The performance improvement over typical enterprise servers for appropriate applications can be phenomenal. Whether or not businesses decide to jump on the P2P bandwagon, the current lack of security features in P2P applications must be remedied. The next step is for businesses to realize that the potential of distributed processing and distributed file systems is too promising to ignore. Rather than spend money on expensive server class hardware, businesses could instead rely upon a distributed network of workstations. These workstations would each maintain a small chunk of the corporate data locally, as well as offer spare processing cycles to applications that need it. The ultimate P2P business system would be completely distributed. Backups would be greatly simplified, as the data would be redundant by default (i.e., the same piece of data would sit on multiple computers); such a system increases robustness as well. As future networking technologies provide more and more bandwidth, the power and efficiency of P2P systems will definitely hit a high-resulting more people in the business community adopting it. New techniques of finding resources on a P2P network, routing, data management, and security are being devised. As P2P becomes a highly accepted model of computing more complex scientific and engineering as well as business problems can be solved affordably using a number of ordinary desktops PCs. But before that happens, a standard set of protocols and mechanisms must be defined that takes into account all existing technologies different corporations and organizations employed.

XI. References

[1] D. Milojicic, V. Kalogeraki, R. Lukose, K. Nagaraja, J. Pruyne, B. Richard, S. Rollins, Z. (2002): "Peer-to- Peer Computing," hp Technical Report, 2002.

[2] D. Liben-Nowell, H. Balakrishnan (2002): "Observations on the dynamic Evolution of Peer-to-Peer Networks," Proceedings of the 1st International Workshop on Peer-to-Peer Systems, Cambridge, USA, 2002.

[3] Clay Shirky on P2P: http://davenet.scripting.com/2000/11/15/clayShirkyOn P2P.

[4] Hyunggon Park, Rafit Izhak Ratzin, Mihaela van der Schaar "Peer-to-Peer Networks – Protocols, Cooperation and Competition".

[5] Cohen, Bram (October 2002): "BitTorrent Protocol 1.0". BitTorrent.org. Archived from the original on 8 February 2014. Retrieved 19 April 2013.

[6] Jinyang Li, John Jannotti, Douglas S. J. De Couto, David R. Karger, and Robert Morris (2000): "A scalable location service for geographic ad hoc routing", In Proceedings of the 6th ACM International Conference on Mobile Computing and Networking (MobiCom '00), pages 120–130, Boston, Massachusetts, August 2000.

[7] PLAXTON, C., RAJARAMAN, R., AND RICHA (1997): "A. Accessing nearby copies of replicated objects in a distributed environment", In Proceedings of the ACM SPAA (Newport, Rhode Island, June 1997), pp. 311–320.

[8] Antony Rowstron Peter Druschel (2001): "Past: Persistent and anonymous storage in a peer-to-peer networking environment", In Proceedings of the 8th Conference on Hot Topics in Operating Systems (HotOS 2001), May 2001.

[9] Adel Ali Al-zebari (2014): "**Peer to Peer File Sharing System",** International Journal of Scientific & Engineering Research, Volume 5, Issue 9, September-2014 486 ISSN 2229-5518.

[10] S. Zhuang, B. Zhao, A. Joseph, R. Katz, and J. Kubiatowicz (2001): "Bayeux: An Architecture for Scalable and Fault-tolerant Wide-Area Data Dissemination," in Proc. of NOSSDAV, June 2001.

[11] CLARKE, I., SANDBERG, O., WILEY, B., AND HONG, T. W. (2000): "Freenet: A distributed anonymous information storage and retrieval system", In Proceedings of the ICSI Workshop on Design Issues in Anonymity and Unobservability (Berkeley, California, June 2000).

http://freenet.sourceforge.net.

[12] Gnutella. http://gnutella.wego.com.

[13] Napster. http://www.napster.com.

[14] Ohaha. http://www.ohaha.com/design.html.

[15] Frank Dabek, Emma Brunskill, M. Frans Kaashoek, David Karger Robert Morris, Ion Stoica, Hari Balakrishnan, "Building Peer-to-Peer Systems "With Chord, a Distributed Lookup Service", MIT Laboratory for Computer Science.

[16] J. Jannotti, D. K. Gifford, K. L. Johnson, F. Kaashoek, and J. W. O'Toole (2000): "Overcast: Reliable Multicasting with an Overlay Network," in Proc. of OSDI, October 2000, pp. 197–212.

[17] D. Andersen, H. Balakrishnan, F. Kaashoek, and R. Morris (2001):, "Resilient overlay networks," in 18th ACM SOSP, Oct. 2001.

[18] Y. H. Chu, S. G. Rao, and H. Zhang (2001), "A case for end system multicast," in Proc. of ACM Sigmetrics, June 2000, pp. 1–12.

[19] K. Birman and T. Joseph (1987): "Exploiting virtual synchrony in distributed systems," in Proc of the ACM SOSP, November 1987, pp. 123–138.

[20] S. Deering, D. Estrin, D. Farinacci, V. Jacobson, C. Liu, and L. Wei (1996): "The PIM Architecture for Wide-Area Multicast Routing," IEEE/ACM Transactions on Networking, vol. 4, no. 2, April 1996.

[21] D. A. Menasc´e, V. A. F. Almeida, and L. W. Dowdy (2001): "Capacity Planning for Web Services: metrics, models, and methods," Prentice Hall, 2001.

[22] Zihui Ge, Daniel R. Figueiredo, Sharad Jaiswal, Jim Kurose, Don Towsley "Modeling Peer-Peer File Sharing System."

[23] I. Stoica, R. Morris, D. Karger, M. Kaashoek, and H. Balakrishnan (2001): "Chord: A scalable peer-to-peer lookup service for internet applications," in Proc. of ACM SIGCOMM'01, Aug 2001.

[24] SHA: SHA stands for "secure hash algorithm". The four SHAalgorithms are structured differently and are named SHA-0, SHA-1, SHA-2, and SHA-3. SHA-1 is a function designed by the United States Agency and is a U.S. Standard published by the United States NIST. http://en.wikipedia.org/wiki/SHA-1