Volume 3, Issue 8, August 2013 ISSN: 2277 128X

International Journal of Advanced Research in Computer Science and Software Engineering

Research Paper

Available online at: www.ijarcsse.com

Peer to Peer Networking and Applications

Adarsh Agarwal *IIIT- Delhi, India*

Nipun Bansal *IIIT- Delhi, India*

Sudeep Gupta *IIIT- Delhi, India*

Abstract— a peer-to-peer computer network refers to any network that does not have fixed clients and servers, but a number of peer nodes that function as both clients and servers to the other nodes on the network. This model of network arrangement is contrasted with the client-server model. Any node is able to initiate or complete any supported transaction. Peer nodes may differ in local configuration, processing speed, network bandwidth, and storage quantity. Through this survey we present brief description of each P2P network protocols along with their advantages and disadvantages. Also we look into some security issues related to structured P2P networks and suggest some improvement.

Keywords— Peer2Peer Network, Distributed, Centralized, Structured, Unstructured, Napster, Direct Connect, Gnutella, DHT, FreeNet, EDonkey, Ares, FastTrack, BitTorrent, Chord, Pastry, Tapestry, Kademlia

I. INTRODUCTION

Computing has passed through many transformations since the birth of the first computing machines. A centralized solution has one component that is shared by users all the time. All resources are accessible, but there is a single point of control as well as a single point of failure. A distributed system is a group of autonomous computers connected to a computer network, which appears to the clients of the system as a single computer. Distributed system software allows computers to man- age their activities and to share the resources of the system, so that clients recognize the system as a single, integrated computing facility. Opportunity to attach components improves availability, reliability, fault tolerance, and performance. In such systems, the methods for minimizing communication and computation cost are significant. The widely used client–server model is an example of a distributed system. In this model, the servers are optimised to offer services to several clients. The clients always communicate with the servers and they do not share any services. If the servers fail, the whole services that are offered from the servers to the clients are terminated. The World Wide Web (WWW) can be viewed as a massive distributed system consisting of millions of clients and servers for accessing associated documents. Servers preserve collections of objects, whereas clients provide users a user-friendly interface for presenting and accessing these objects. The inadequacy of the client–server model is evident in WWW. Being resources are concentrated on one or a small number of nodes and to provide 24/7 access with satisfactory response times, complicated load-balancing and fault- tolerance algorithms have to be employed. The same holds right for network bandwidth, which adds to this tailback situation. These two key problems inspired researchers to come up with schemes for allocating pro- cessing load and network bandwidth among all nodes participating in a distributed information system. P2P networks are a recent addition to the already large number of distributed system models. P2P systems—an alternative to conventional client–server systems— mostly support applications that offer file sharing and content exchange like music, movies, etc. The concept has also been effectively employed for distributing computing and Internet-based telephony. A major benefit of P2P file sharing is that these systems are fully scalable—each additional user brings extra capacity to the system. A node (peer) can act both as a client and a server. The participating nodes mark at least part of their resources as 'shared', allowing other contributing peers to access these resources. Thus, if node A publishes something and node B downloads it, then when node C asks for the same information, it can access it from either node A or node B. As a result, as new users access a particular file, the system's capability to provide that file increases.

There are mainly three different architectures for P2P systems: centralized, decentralized structured and decentralized unstructured. In the centralized model, such as Napster, central index servers are used to maintain a directory of shared files stored on peers with the intention that a peer can search for the location of a desired content from an index server. On the other hand, this design makes a single point failure and its centralized nature of the service creates systems susceptible to denial of service attacks. Decentralized P2P systems have the advantages of eliminating dependence on central servers and providing freedom for participating users to swap information and services directly between each other. In decentralized structured models, such as Chord, Pastry, and CAN, the shared data placement and topology characteristics of the net- work are strongly controlled on the basis of distributed hash functions. In decentralized unstructured P2P systems, such as Gnutella and Kazaa, there is neither a centralized index nor any strict control over the net- work topology or file placement. Nodes joining the network, following some loose rules, form the network. The resulting topology has certain properties, though the placement of objects is not based on any knowledge of the topology. The decentralization makes available the opportunity to utilise unused bandwidth, storage and processing power at the periphery of the network. It diminishes the cost of system ownership and maintenance and perks up the scalability.

II. P2P STRENGTHS AND BENEFITS

We should consider the benefits as well as the limitations of building and deploying an application using the P2P approach versus conventional client/server or Web-based approaches. Naturally, P2P might not be the best choice in many cases.

A P2P overlay is a collection of distributed networked hosts whose resources are available for use by the P2P applications associated with the overlay. These resources include computation, network capacity, and file storage. While their host is connected to the overlay, each end user shares in the cost of operating the overlay. This cost sharing by the participants lowers the barrier of entry to overlay providers. The low barrier of entry means that little hardware or network investment is needed to launch a P2P application.

As discussed earlier, the P2P architecture is inherently self-scalable, since each new peer adds additional capacity to the system. However, the developers of early P2P applications soon discovered that not all peers have equal capacity to contribute. For example, the host might be relatively limited in terms of CPU speed and memory capacity. Or the host might be behind a firewall, making it difficult for that peer to participate in the routing algorithm of the overlay. Or the host might be used for other applications that consume much of the available capacity. Consequently, some designs have organized peers into different categories depending on their capacity and reliability. The more capable or super peers might perform all the overlay operations, whereas the less capable peers play a more limited role. The self-scaling property by itself doesn't necessarily translate into good performance under heavy loads since the load might not be uniformly distributed across the overlay. To illustrate, consider the well-known phenomenon of flash crowds that occurs when a very popular item is first available at a Website. As word spreads about the availability of this new item, large numbers of users simultaneously try to retrieve it using their Web browsers. This creates a sudden and excessive load on the Web servers that provide the object. Examples of objects that cause flash crowds include major news stories or new music or video releases by popular artists.

III. CENTRALIZED NETWORK

Napster and other similar systems have a constantly- updated directory hosted at central locations (e.g., the Napster web site). Nodes in the P2P network issue queries to the central directory server to find which other nodes hold the desired files. Such centralized approaches do not scale well and have single points of failure.

A. Napster

It was the first P2P system that emerged and which gained mass popularity. It was widely used for sharing of mp3 files on university network by the students. It was proposed as Two tier architecture ,consisting of peers and a central indexing server. The server consisted of a central index where all the available files and their locations were kept. The search request returned all such results and the peers then negotiated a transfer amongst themself. The problem with this architecture was of centralisation- a single point of failure. Hence the system was not robust. Moreover, the privacy was com- promised as the server indexed all the files. Hence, there was an inherent need for some decentralization, and robustness in the architecture. The centralized server failed to provide both.

After its launch in 1999, Napster achieved huge notoriety, first as the earliest and as an immensely popular file-sharing system and, subsequently, in a relative short period of time, as a legal test case for personal use of shared media. After losing on the legal front, Napster was shut down. In reaction to the legal issues Napster faced, subsequent file-sharing systems used a full P2P architecture for both the file directory and file transfer functions. The majority of these designs used an unstructured overlay mechanism. Although Napster is not a full P2P system, it popularized the P2P concept in the mass media and influenced subsequent file-sharing systems.

IV. UNSTRUCTURED OVERLAYS

Most deployed P2P applications have used unstructured topologies. Here we look at this important class of over- lay in detail, starting with the basic routing mechanisms. Influential designs such as Gnutella, FreeNet, FastTrack, and Gia are then discussed. Ideas from social networks, especially the small-world phenomenon, are related to unstructured topologies, and an overview of social overlays follows.

A. BASIC ROUTING IN UNSTRUCTURED OVERLAYS

I. *Flooding and Expanding Ring:* Let's assume that each peer keeps a list of other peers that it knows about. We can call these peers the neighbors. If neighbor relations are transitive, we have connectivity graphs such as the one shown in Figure. In this particular graph, peers have between two and five neighbors in the overlay. The number of neighbors a peer has is called the degree of the peer. Increasing the degree of the peers reduces the longest path from one peer to another (the diameter of the overlay) but requires more storage at each peer.

Once a peer is connected to the overlay, it can exchange messages with other peers in its neighbor list. An important type of message is a query for specific information.

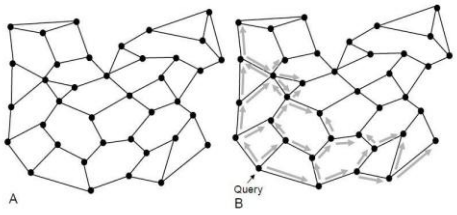

Figure 1. (A) Unstructured topology showing connections between peers and **(B)** query flooding to four hops.

II. *TRADE OFF BETWEEN ROUTING AND PATH DISTANCE:* If each peer had complete information about all other peers in the overlay, each P2P message would take at most one hop. However, all peers would need to maintain an *O*(N) routing table size, and each join and leave event would need to propagate to all other peers in the overlay, creating a large maintenance load. This maintenance load would grow with both the size of the overlay, N, and the churn rate. On the other hand, if a node knows only the link to its successor on a ring, routing state and maintenance load would be nominal but routing performance would be *O*(N).

Since neither of these two operating points is generally practical, there has been a great deal of interest in exploring the state versus overhead trade-offs

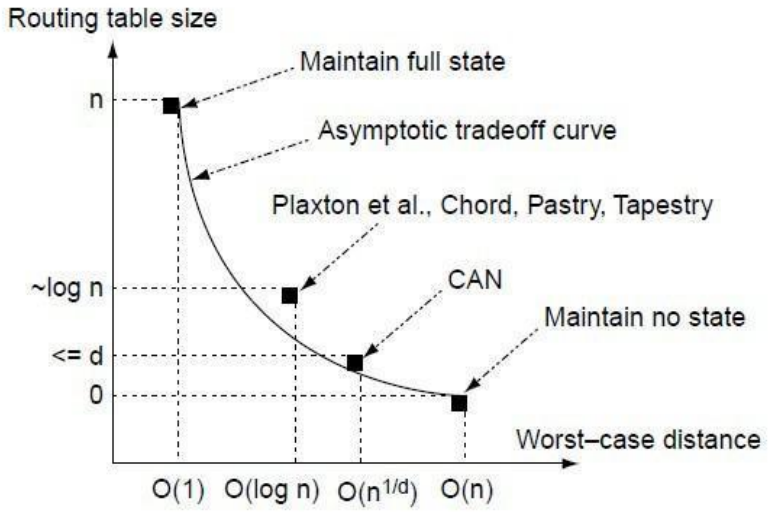

Figure 2. Asymptotic trade-off between peer-routing table size and overlay diameter) of various algorithms.

B. DECENTRALISED/UNSTRUCTERED (EARLY) SYSTEMS

The original DC architecture consists of hubs and peers. The hubs act as distributed servers, which are in charge of bookkeeping of clients and helped in negotiating query and transfer requests. Each hub was self-regulated, and client could connect to multiple hubs that it was a part of. Problem: 1) The security of the whole net- work is compromised if the hub starts acting maliciously. 2) The DC network is susceptible to DDoS attacks. 3) The decentralisation is still not achieved as hubs act as partial servers. Conclusion: There was felt a need for greater decentralisation, and security within the P2P network. Though the DC networks were a significant improvement on Napster, they were still not robust.

1. *GNUTELLA:* It was the first fully decentralised P2P system. Each client would bootstrap to at least one client to find other client. It was like a collection of nodes in an unsupervised network with each node joining and leaving the network at any time. Introduced layering in the net- work by the concept of peer, super peer, and ultra-peers. In Gnutella, peers are referred to as servants, which is a combination of the words server and client. Later versions of Gnutella introduced ultra-peers, which are high-capacity and stable peers. Each ultra-peer maintains connections to a set of other ultra-peers. The Gnutella protocol consists of a set of basic messages and an optional set of extensions.

Problems: 1) The search time was exponential to the number of users. 2) The slow nodes choked the network. 3) The network was unstable and search results would be dropped as nodes were free to join/leave.

Conclusion: The general notion of complete decentralisation was not found to be desirable as file searching took exponential time and was often not complete.

2. *FreeNet:* FreeNet was proposed by Ian Clarke in 1999 as a distributed peer-to-peer file sharing mechanism featuring security, anonymity, and deniability. The FreeNet design discussed here is described in a paper published in 2000. Both objects and peers have identifiers. Identifiers are created using the SHA-1 one-way hash function. Peer identifiers are called routing keys. Each peer has a fixed- size routing table that stores links to other peers. Each entry contains the routing key of the peer. FreeNet uses key-based routing for inserting and retrieving objects in the mesh. Requests are forwarded to peers with the closest matching routing key. If a request along one hop fails, the peer will try the next closest routing key in its routing table. The routing algorithm is steepest-ascent hill climbing with backtracking until the request TTL is exceeded. Consequently, depending on the organization of the links and the availability of peers, it is possible that requests could fail. FreeNet counteracts this by caching objects along the return path both on lookup and insert requests. An object is stored at a peer until space is no longer available and it is the least recently used (LRU) object at that peer.

Problems: 1) The FreeNet tends to forget the less retrieved data, so this greatly reduces dependability. 2) It's easy to snoop on known FreeNet nodes. 3) The additional encryption of data requires the publisher node to maintain the original unencrypted copy to access the file. 4) Direct peer to peer transfer was not available which increased the transfer time.

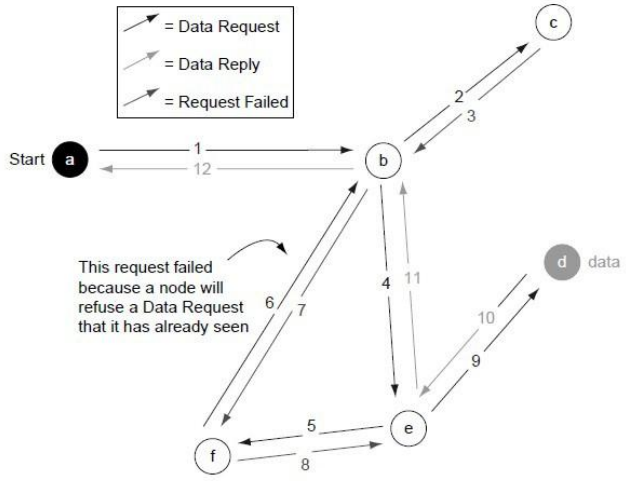

Figure 3. FreeNet distributed key-based routing

3. *FastTrack:* FastTrack is another unstructured P2P overlay that appeared around the same time as Gnutella and was used by a number of file-sharing clients, including KaZaA, Grokster, and Imesh. It is a proprietary system that uses an encrypted protocol.

This semi-distributed architecture allows data to be decentralized without requiring excessive overhead at every node, thereby increasing scalability and reliability. Also unlike Napster, it enables users to download from multiple sources simultaneously, thereby not restricting the user to the bandwidth of one source only however, the increased workload of supernodes generally requires additional network bandwidth and CPU time.

Conclusion: There is a need for better hashing algorithms for creating file hashes. The existing algorithms are tedious when they calculate the same. To reduce load from one node which is sharing a file, different pieces could be downloaded from different nodes which lead to efficient utilisation of bandwidth,

4. *BitTorrent:* BitTorrent is a protocol designed for distributing large files in pieces using mutual distribution of the pieces between a set of peers called a swarm. If the swarm is sufficiently large, this offloads the Web server that is the primary source of the content. Generally peers join a swarm to download a specific file and leave the swarm shortly after the file download completes.

The file to be downloaded is available at a Web server called the seed, which also provides a torrent file corresponding to the content file. The torrent file identifies the individual fixed-size pieces of the large content file and specifies a host that monitors the peers, called the tracker. Peers that want to access the pieces of the content file contact the tracker to determine other peers that have already joined the swarm. Thereafter, peers directly communicate with one another using the peer list provided by the tracker, without requiring the tracker to participate further. In some designs, the tracking is done using a DHT. A trackerless DHT design using the Kademlia algorithm.

A peer seeking to download a file first obtains the tor- rent file and a URL to the tracker. The tracker returns a list of peers randomly selected from the swarm that are currently downloading the file. Each peer requests pieces to download in random order from its neighbor peers. Each time a peer has successfully retrieved a piece of the file, the downloading peer announces the download to the other peers in the swarm to which it is connected. The goal of each BitTorrent peer is to maximize its piece download rate in a reciprocal manner. Several policies are implemented in the protocol to achieve fairness, provide incentives for mutual ex- change, avoid overloading, and find better peers with which to exchange pieces. Each peer should avoid being overloaded by requests for piece transfer. For this reason and for good TCP performance, each peer limits the number of simultaneous active connections, typically four. These active connections are placed by the peer in the unchoked state, while the other neighbor connections are placed in the choked state. Frequent changes in neighbors' choked state, known as fibrillation, are limited to specific time intervals, currently 10-second intervals. Exchanges with neighbors should be fair, that is, each peer should reciprocate by supplying pieces to peers that provide downloads to it. Finally, each peer should try other peers periodically to see if the download rate is better than the current active set.

*C. **SUMMARY***

Unstructured overlays have been used in several widely used file-sharing systems, despite their inefficiencies. In the research community there has been much effort to study the properties of these overlays using crawlers to measure the overlay network, peer, and content proper- ties. In addition, many improvements have been suggested to increase their performance and reduce over- head. Since structured and unstructured overlays have somewhat complementary characteristics, there are proposals to create hybrid overlays that combine both types of routing algorithm. Patterning the overlay organization on power law and social networks has also drawn a great deal of interest.

V. STRUCTURED OVERLAYS

The earliest peer-to-peer systems used unstructured over- lays that were easy to implement but had inefficient routing and an inability to locate rare objects. These problems spawned many designs for overlays with routing mechanisms that are deterministic and that can pro- vide guarantees on the ability to locate any object stored in the overlay. The large majority of these designs used overlays with a specific routing geometry and are called structured overlays. At the same time, many unstructured overlays have incorporated some degree of routing structure, such as clustering, near/far links, and semantic links to improve search efficiency. In addition to the structured and unstructured overlay categories, there are hierarchical models.

A. *DISTRIBUTED HASH TABLES*

DHT is a class of a decentralized distributed system that provides a lookup service which are similar to a hash table, in which key, value pairs are stored in a DHT, and any participating node can efficiently retrieve the value associated with a given key A P2P protocol using the concept of DHT must provide facilities for:

- Mapping content keys to network nodes while observing load balancing considerations.
- Each node must be able to forward a given content key to a node whose ID hash is closer to the content key.
- The nodes must be able to build routing tables adaptively as new nodes join and existing nodes leave.
- Routing tables are used to speed up the search for the node that is closest to a given content key and to facilitate recovery from node failures. Concept of DHT raises questions like how do we let new nodes join the network and existing nodes leave it at their own pleasure? A related question would be how do we make sure that our distributed database can handle node failures? The question regarding new nodes joining in and old nodes leaving has to be examined from the perspective of the extent to which the content keys must be reassigned to the various nodes. The notion of consistent hashing addresses this issue.

1. *CHORD PROTOCOL:* Chord organizes peers on a logical ring, and peers maintain neighbor pointers spaced at logarithmic intervals around the ring. In addition, each peer has a link to its predecessor and successor peers on the ring. The Chord routing table is called a finger table. Chord's routing function uses its successor ring in the last hop and uses the finger table to maximize the size of the step toward the destination.

   Chord Comparison with Napster, Gnutella, FreeNet

   - Compared to Napster and its centralized servers, Chord avoids single points of control or failure by a decentralized technology
   - Compared to Gnutella and its widespread use of broad- casts, Chord avoids the lack of scalability through a small number of important information for routing.
   - Compared to FreeNet and its allows to have anonymity and prevents guaranteed retrieval of existing documents, Chord does not provide anonymity but its lookup operation runs in predictable time and always results in success or definitive failure

   Chord Advantages

   • Anonymous Nodes: allows nodes to be anonymous, so all nodes do not need to have ids. In chord, however, all nodes need to have ids.
   • Malicious nodes: malicious nodes are less likely to bring down the whole network.

   Chord Disadvantages

   • Data Lookup: both applications perform data lookups using user supplied keywords, which presents difficulties. User supplied keywords can be vague and incomplete, leading to failures and excessive delays in data lookups.
   • Single Point of failure: napster uses a central index server which represents a single point of failure. If this server goes down, users of napster can no longer perform any lookups.
   • Network Congestion: Gnutella floods queries over the entire system, leading to high processing costs and network congestion

B. *NODE PROXIMITY ISSUES IN ROUTING WITH DHT'S*

So when an application program seeks the overlay node that is responsible for a given content key, in all likelihood that query will make multiple hops around the globe even when the overlay node of interest is sitting right next to the computer running the application pro- gram. This problem arises because the basic Chord protocol does not take into account any proximity between the nodes in deciding how to route the queries. By proximity between two nodes in the overlay we could mean the number of hops between the nodes in the network that underlies the overlay. The next P2P protocol Pastry, is more aware of proximity between the nodes.

1. *PASTRY PROTOCOL:* Pastry, like Chord, creates a self-organizing overlay net- work of nodes. As in Chord, each participating node is assigned a nodeID by possibly hashing its IP address and port number. When deciding at which node to store a message, Pastry uses the same basic rule as Chord: A message is delivered to the node whose nodeID is closest to message key. But Pastry gets to that final node in a manner that is different from Chord.

Features**:** Pastry is completely decentralized, scalable, fault-resilient, and reliably routes a message to the live node with a nodeId numerically closest to a key. Pastry can be used as a building block in the construction of a variety of peer-to-peer Internet applications like global file sharing, file storage, group communication and naming systems.

Benefit**:** It's redundant and decentralized nature, there is no single point of failure and any single node can leave the network at any time without warning and with little or no chance of data loss.

Pastry vs. Chord: Advantages:

• Knows Network: like tapestry, pastry takes into account network locality; it seeks to minimize the distance messages travel, according to a to scalar proximity met- ric like the number of IP routing hops.
• Performance: like Chord, the look up time for tapestry protocol is lg(n)
• Distributed: does not require a centralized server
• Scalability: handles node arrivals and departures well, unlike tapestry

Disadvantages**:**

• Complexity: like tapestry, pastry has a more complicated join protocol. A new node's routing table will be populated with information from nodes along the path taken by the join message. This leads to latency.

2. *TAPESTRY PROTOCOL:* California at Berkeley by Zhao et al. Tapestry forms an overlay network that sits at the application layer (on top of an Operating System). If tapestry is installed on different network nodes it will allow any one node to route messages to any other node running tapestry, given a location and a network independent name. Also nodes in a Tapestry network can advertise location information about data it possesses in a specific format understood by other nodes running tapestry. This special format allows the other nodes to find and access this data easily and efficiently, given that they know the data name. So, we can see that tapestry allows nodes the ability to share data, thereby creating their own p2p system.

Chord vs. Tapestry

Advantages:

• Knows Network: tapestry has an advantage over Chord protocol because the algorithm knows the network topology, so queries never travel more than the network distance required to reach them
• Performance: like Chord, the look up time for tapestry protocol is log (n)
• Distributed: does not require a centralized server

Disadvantages**:**

• Scalability: does not handle node joins and failures as well as chord as it is more complicated. So, chord is better for p2p systems with many nodes arriving and departing at random intervals.

3. *CONTENT ADDRESSABLE NETWORK:* Content Addressable Network or CAN is a constant- degree structured multihop DHT that organizes peers in a d-dimensional Cartesian coordinate system. Like the other systems we discuss, CAN peers and objects have identifiers from the same virtual address space. Each peer's position in the d-dimensional space and the bound- aries it shares with other peers determine the extent of the zone of the space for which the peer is responsible. To join the overlay, a new peer performs three steps. First, it locates some peer already in the CAN. Second, it randomly selects a peer whose zone will be split to accommodate the new peer, and it sends a join request to it via the first peer. Third, split the existing zone and notify the neighbors of the split zone so that the routing decisions include the zone changes. When a zone is split, the original peer retains those key-value pairs that fall within its new subdivided zone. The remaining key value peers go to the joining peer. Similarly, the joining peer inherits the neighbors of the original peer that abut the edges of the zone for which the new peer is responsible. Both the new peer and the original peer become neighbors, and the original peer prunes its neighbor list and notifies its neighbors accordingly. Neighboring peers exchange heartbeat messages to verify that they are still connected to the overlay. If missing heartbeat messages indicate that the peer is no longer available, the neigh- bors of the leaving peer need to coordinate to determine which peer will assume ownership of the zone held by that peer. A message exchange is conducted so that the neighbour with the smallest zone assumes ownership of the orphaned zone.

4. *KADMELIA PROTOCOL:* Kademlia is important because its DHT is employed by the very popular BitTorrent protocol (for downloading music and movies) when it is used in a tracker less mode. Kademlia uses the same identifier space as Chord. It specifies the structure of the network and the exchange of information through node lookups. Kademlia nodes communicate among themselves using UDP. A virtual or overlay network is formed by the participant nodes. Each node is identified by a number or node ID. The node ID serves not only as identification, but the Kademlia algorithm uses the node ID to locate values (usually file hashes or keywords). In fact, the node ID provides a direct map to file hashes and that node stores information on where to obtain the file or resource. Further advantages are found particularly in the decentralized structure, which increases the resistance against a denial of service attack. Even if a whole set of nodes is flooded, this will have limited effect on network availability, since the network will recover itself by knitting the network around these "holes".

5. *COMPARISON AND EVALUATION:* At this stage a reasonable question to ask is, How do all these overlay designs compare, and which ones per- form best? The answer to this question depends on the types of application the overlay

is to be used for. Also, when designs have similar performance characteristics, other less tangible attributes may be considered, such as relative complexity and available implementations. Considering performance, we can examine several different dimensions of overlay operation. We divide these into two categories: algorithm correctness and operational metrics. Correctness of performance includes two aspects:

• Convergence. Is the routing algorithm guaranteed to reach the destination in a practically bounded hop limit?

• Stability. Is the overlay provably stable under churn? At what churn rate does the overlay destabilize? Performance metrics have to do with the amount of state, bandwidth, and other resources consumed under a given workload.

C. ***SUMMARY***

In the design of structured overlays, the geometry of the overlay is a key design decision. To be effective for P2P use, the geometry must meet several criteria, including:

• The overlay must be fully connected for resiliency to peer failure.

• Peers throughout the peer population, must have uni- form degree to avoid load imbalance.

• There must be support for at least one type of dis- tributed routing function that converges.

• Efficient and easy construction and maintenance of the overlay routing state and topology using a distributed algorithm are required.

To organize the large space of structured overlays, we can use the following criteria: multihop versus O(1)- hop, logarithmic degree versus constant degree, and routing using prefix, Euclidean distance, and XOR metrics. Some overlay designs might fit more than one criterion. Performance analysis is analytic, through simulation, or through implementations, which are not in scope with this survey report.

VI. SECURITY ISSUES IN P2P PROTOCOLS

Security is an essential requirement and an important component of any communication and computing sys- tem. This is certainly true for a peer-to-peer system. In fact, security in P2P is an issue of particularly concern to many. With Napster's debut in 1999, P2P file sharing became immensely popular. The public's concern with information security has also increased tremendously in the past eight years. Web searches on keywords such as: P2P security news, P2P security concerns, and P2P security story; all return long lists of results with many news headlines.

We believe that P2P file-sharing networks represent a significant and poorly understood threat to business, government, and individuals. Given the nature of the threat, we would argue that many individuals may be experiencing identity theft and fraud without ever knowing the source of their misfortune. Furthermore, we see many of the current P2P trends increasing the problem. We urge both corporate executives and government officials to educate themselves and their constituencies to the risks these networks represent The basic protocols for open DHT-based overlay networks are founded on the assumption that every node joining the overlay can be trusted to provide its own nodeID that can be assumed to come from a uniform probability distribution over the entire node identity space. When no constraints are placed on who can join a P2P overlay, security problems can be created by any or all of the following possibilities:

1. A new node supplying a legitimate nodeID but falsify- ing information in its own routing table.
2. A new node supplying a fake nodeID that is meant to cause harm to the operation of the overlay.
3. The same new node joining an overlay repeatedly with different nodeIDs.
4. A set of nodes conspiring together with fake values for nodeID to disrupt the operation of the overlay.

One of the easiest ways for a malicious node to cause problems is by falsifying the information in its routing table All DHT-based overlays are vulnerable to false information in the routing tables of the intruder nodes. This may be referred to as a topology attack on a P2P overlay.

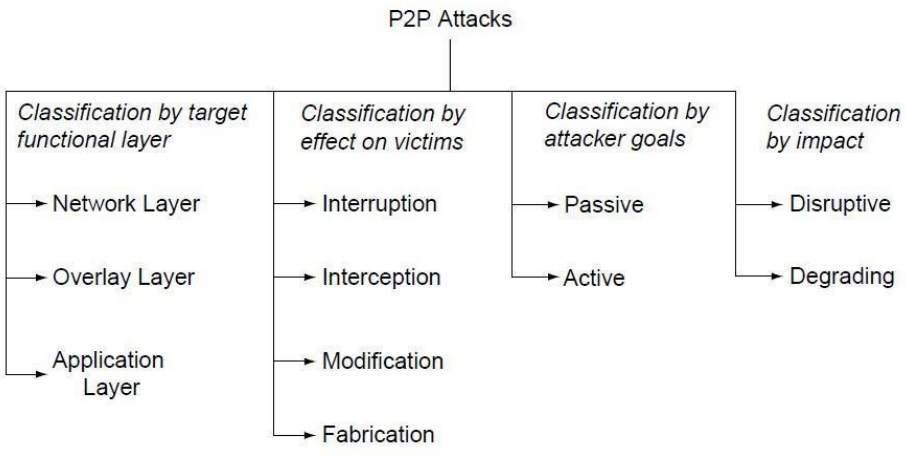

Figure 4. Classfication of attacks on P2P systems

Let's now consider the case when a node supplies a fake nodeID when issuing its request to join the overlay. We will assume that the intruder node is using a legitimate IP address assigned to it. An attack mounted with a fake nodeID, especially if that identity belongs to some other legitimate node, is called the Spartacus Attack. Although not a part of the basic P2P protocols, a node's IP address could be verified by having it acknowledge test messages when it first links up with its neighboring nodes in the P2P network. A node advertising a fake IP address for itself could still receive test messages from other nodes in the network if the fake address belonged to a co-conspirator machine. But, at least for the present, we will assume that such is not the case.

Ordinarily, for the sake of fault tolerance and for dealing with node departures, replicas of the same data object would be stored at a set of nodes in a neighborhood (in the nodeID space) of the node that minimizes the dif- ference between the nodeID and the key. Several nodes conspiring together could hijack a neighborhood around a key and cause disruptions with the delivery of any of the replicas. The same sort of an attack could be mounted by a single node that is able to field multiple nodeID values.When a single malicious node presents multiple nodeIDs to an overlay network, we say the of- fending node is mounting a Sybil attack. Another security problem can occur when several malicious nodes decide to join and leave an overlay in rapid succession. This has the potential of degrading the performance of the overlay network since the routing table updates at all the affected nodes in the overlay may not be able to keep up with the additions and the departures of the offending nodes. As a result, several legitimate nodes may end up with inconsistent routing tables.

VII. Conclusions

Our Survey conducts a comprehensive theoretical survey of various structured and unstructured P2P networks. A brief description of each of the P2P network along with their advantages and disadvantages are summarized. Finally, various security issues related to structured P2P networks is presented.

By far the best P2P network today which has the most number of users, is reliable, robust and available is the BitTorrent network. However, BitTorrent even though a decentralised network uses a part of the Kademlia Protocol. An equilibrium of sorts has been reached between decentralisation and structured network with it. The future however lays in improvement of the centralised part i.e. the DHT protocols. These protocols need to improve upon the various security issues that are there. As of now, people can see who their downloading peers are, their IP address and port number. This comprises the security of the peers. We need to integrate a routing method which makes the peers anonymous. More importantly, we also need a method to circumvent any snooping of the packets, as many organisations tend to censor the torrent traffic.

Another thing which is important is the life of a file in the network. Older files which are less frequently accessed are difficult to find. A policy is required to maintain such files. Maybe they could be shifted and allocated a separate tracker, which would make it easier to search and download. Another policy which may be considered could be that of replication of such content to make available more easily.

The last point to consider is that of spurious data. To- day every P2P network is flooded with spurious data to prevent distribution of copyright material. The existing BitTorrent network consists of voting system, where the bad files are automatically down voted. We need an active policy for moderation, though moderation is not possible for huge and open networks.

REFERENCES

[1] K. Molin, "Measurement and Analysis of the Direct Connect Peer-to-Peer File Sharing Network," Goteborg, Sweden: University of Gothenburg, *June 2009.*

[2] Eytan Adar and Bernardo A. Huberman. Free riding on gnutella. First Monday, 5(10), October 2000.

[3] Maguire, Gerald Q., and Falieris Sotiris. "THE PIRATE BAY: THE DAY AFTER."

[4] Rowstron, A., & Druschel, P. (2001, January). Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. In *Middleware 2001* (pp. 329-350). Springer Berlin Heidelberg.

[5] Stoica, Ion, Robert Morris, David Karger, M. Frans Kaashoek, and Hari Balakrishnan. "Chord: A scalable peer-to-peer lookup service for internet applications." In *ACM SIGCOMM Computer Communication Review*, vol. 31, no. 4, pp. 149-160. ACM, 2001.

[6] Zhao, Ben Y., Ling Huang, Jeremy Stribling, Sean C. Rhea, Anthony D. Joseph, and John D. Kubiatowicz. "Tapestry: A resilient global-scale overlay for service deployment." *Selected Areas in Communications, IEEE Journal on* 22, no. 1 (2004): 41-53.

[7] Maymounkov, Petar, and David Mazieres. "Kademlia: A peer-to-peer information system based on the xor metric." *Peer-to-Peer Systems*. Springer Berlin Heidelberg, 2002. 53-65.

[8] Lv, Q., Cao, P., Cohen, E., Li, K., & Shenker, S. (2002, June). Search and replication in unstructured peer-to-peer networks. In *Proceedings of the 16th international conference on Supercomputing* (pp. 84-95). ACM.

[9] Jantunen, Alex, Sami Peltotalo, and Jani Peltotalo. "Peer-to-Peer Analysis." (2006).

[10] Lua, Eng Keong, Jon Crowcroft, Marcelo Pias, Ravi Sharma, and Steven Lim. "A survey and comparison of peer-to-peer overlay network schemes." *IEEE Communications Surveys and Tutorials* 7, no. 2 (2005): 72-93.

[11] Ding, Choon Hoong, Sarana Nutanong, and Rajkumar Buyya. "Peer-to-peer networks for content sharing." *Peerto-Peer Computing: The Evolution of a Disruptive Technology* (2005): 28-65.

[12] Saroiu, Stefan, P. Krishna Gummadi, and Steven D. Gribble. "Measurement study of peer-to-peer file sharing systems." In *Electronic Imaging 2002*, pp. 156-170. International Society for Optics and Photonics, 2001.

[13] Shafaat, Tallat Mahmood. "Dealing with Network Partitions and Mergers in Structured Overlay Networks." PhD diss., KTH, 2009.

[14] Stoica, Ion, Robert Morris, David Karger, M. Frans Kaashoek, and Hari Balakrishnan. "Chord: A scalable peer-to-peer lookup service for internet applications." In *ACM SIGCOMM Computer Communication Review*, vol. 31, no. 4, pp. 149-160. ACM, 2001.

[15] Aberer, Karl, Luc Onana Alima, Ali Ghodsi, Sarunas Girdzijauskas, Seif Haridi, and Manfred Hauswirth. "The essence of P2P: a reference architecture for overlay networks." In *Peer-to-Peer Computing, 2005. P2P 2005. Fifth IEEE International Conference on*, pp. 11-20. IEEE, 2005.

[16] Dhara, Krishna, Yang Guo, Mario Kolberg, and Xiaotao Wu. "Overview of Structured Peer-to-Peer Overlay Algorithms." In *Handbook of Peer-to-Peer Networking*, pp. 223-256. Springer US, 2010.

[17] Buford, John F., and Heather Yu. "Peer-to-peer networking and applications: Synopsis and research directions." *Handbook of Peer-to-Peer Networking*. Springer US, 2010. 3-45.

[18] Ku, Raymond Shih Ray. "The creative destruction of copyright: Napster and the new economics of digital technology." *The University of Chicago Law Review*(2002): 263-324.

[19] Stoica, Ion, Robert Morris, David Liben-Nowell, David R. Karger, M. Frans Kaashoek, Frank Dabek, and Hari Balakrishnan. "Chord: a scalable peer-to-peer lookup protocol for internet applications." *Networking, IEEE/ACM Transactions on* 11, no. 1 (2003): 17-32.

[20] Rowstron, Antony, and Peter Druschel. "Storage management and caching in PAST, a large-scale, persistent peer-to-peer storage utility." In *ACM SIGOPS Operating Systems Review*, vol. 35, no. 5, pp. 188-201. ACM, 2001.

[21] Balakrishnan, Hari, M. Frans Kaashoek, David Karger, Robert Morris, and Ion Stoica. "Looking up data in P2P systems." *Communications of the ACM* 46, no. 2 (2003): 43-48.

[22] Karagiannis, Thomas, Andre Broido, Nevil Brownlee, Kimberly C. Claffy, and Michalis Faloutsos. "Is p2p dying or just hiding?[p2p traffic measurement]." In*Global Telecommunications Conference, 2004. GLOBECOM'04. IEEE*, vol. 3, pp. 1532-1538. IEEE, 2004.

