Necessity of Supernodes
Survey

David Hadaller, Kevin Regan and Tyrel Russell

March 16, 2005

1 Introduction

The advent of the Internet has made the transfer of data between people easier and more efficient. One specific and popular application of this process is the peer-to-peer(P2P) network. P2P networks are an overlay network on top of the Internet's topology and accounts for a large portion of Internet traffic[24]. The basic idea of a P2P network is that individual peers can connect to a set of other peers and query them. These queries are forwarded into the network according to a protocol defined by the designers of the P2P network.

There are two major axes along which the different P2P networks are distributed. The first axis is a spectrum running from centralized to decentralized architectures. At one extreme we have a central index which users can query while at the other extreme decentralized axis distributes the index between all of the servers. An example of a centralized system is Napster[5] and an example of the decentralized system is Gnutella[1]. The second axis stretches from structured to unstructured where the structured networks impose a specific network topology on the overlay network and the unstructured networks allows peers to join at any place in the overlay network. An example of a structured P2P network is Chord[32], which is an implementation of a distributed hash table(DHT) and example of an unstructured network is a Gnutella[1].

The motivation for our project is stems from the fact that unstructured decentralized P2P networks are unscalable and the current solution to this problem is to introduce supernodes or super-peers. Supernodes are nodes which have degree that is significantly higher than that the other nodes in the network. Supernodes have been shown empirically to increase the number of nodes that can be added to the network and still achieve efficient performance. Our project aims to provide a theoretical basis for the improved scalability provided by adding supernodes to a P2P network. The remainder of the survey looks at the different aspects of the P2P systems and the theory that underlies these systems.

2 Peer-to-Peer Systems

2.1 What is a Peer-to-Peer System?

Peer-to-Peer (P2P) systems have become extremely popular, mostly due to their file sharing abilities. Ever since Napster's [5] release in 1999, P2P file sharing systems have ballooned in popularity, now connecting millions of users sharing petabytes of data.

While the architecture of various P2P file sharing systems differs between implementations, the basic function remains the same. Users are able to issue a keyword search into the network and the network returns a list of files matching the search and a list of peers which store these files. In these systems, peers operate as both clients and servers, both issuing searches themselves, and accepting searches from their peers.

Peers must form connections in some self-organized, often decentralized, fashion. P2P systems must also support frequent node joins and failures. These challenges are what separate P2P systems from typical well-studied distributed systems problems. Dealing with these challenges in a scalable and reliable way is the goal of P2P research.

The motivations behind a P2P system are numerous. On the technical side, they allow for massive aggregate throughput, dwarfing the capability of a central server. There is no need for large dedicated hardware infrastructure nor the need for central administration; peers cooperatively self-organize themselves without the need for human administration. P2P systems are not the end-all infrastructure for all applications, as the security, scalability, availability, and reliability cannot be guaranteed for particular data items. However, the system as a whole is quite capable of addressing all of these needs.

This work focuses on the challenge of scalability in P2P systems in the context of distributed file sharing. The motivation behind this focus is that efficient scalable search over distributed files is a difficult problem; discussed further in Section 2.3.

P2P systems can logically be divided into two groups: unstructured and structured. Unstructured networks impose no requirement on the arrangement of data in the system. That is, peers may attach to the network at any point and serve data relative to their point of attachment. Examples of such systems include the two most popular protocols: FastTrack (Kazaa [2], Kazaa Lite) and Gnutella [1] (Morpheus [4], LimeWire [3], etc).

Structured systems, on the other hand, impose a logical ordering of data (key-value pairs) in the system. When a peer joins the system, its neighbours are computed using a predetermined function, most commonly using a distributed hash function (DHT). This is discussed further in Section **??**.

Measurements at the University of Wisconsin[1] done in 2001 indicate P2P traffic accounts for approximately 30% of Internet traffic [24], which exceeded the 19% web traffic at the time of measurement. Similar measurements done at the University of Washington [30] in 2002 show that P2P accounted

[1]http://wwwstats.net.wisc.edu/

for 43% of traffic and web for only 14%.

2.2 How do Unstructured Peer-to-Peer Systems Work?

The most widely used P2P systems are unstructured and are currently serving the file sharing needs of millions of users. One of the biggest challenges in these systems is how to perform an efficient search in the network. Because these systems have no enforced structure, searching amounts to a peers passing queries around in an essentially random fashion. No better can be achieve by the system, as it doesn't have information about which peers are likely to contain the matching files.

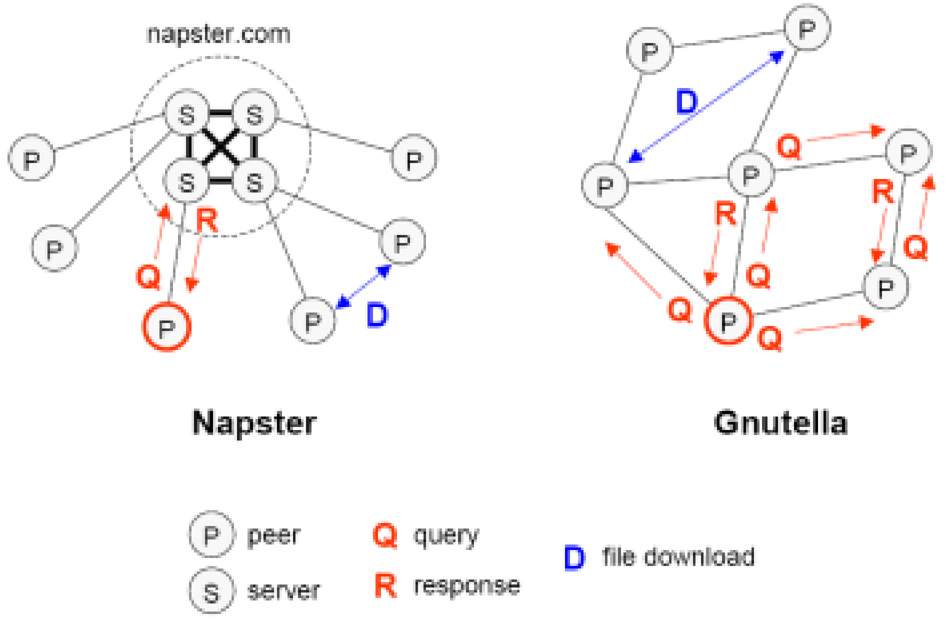

Figure 1: File search and retrieval in Napster and Gnutella, obtained from [29]

Although Napster was the pioneering effort behind P2P file sharing systems, its design, unlike current P2P systems, relied on a set of central servers. As the left half of Figure 1 shows, each peer in the Napster network maintains a connection to one of a set of central servers. These servers maintain a global index of all of the files stored on the peers. To search the network, a user issues a search to one of the central servers, which performs the keyword search and returns the list of peers storing the matching files to the user. The user may then select a file he wishes to retrieve and begin downloading it directly from the peer storing the file.

While Napster was a successful system, it was not able to boast the same potential for robustness and scalability as its well-known decentralized successors Gnutella [1] and Kazaa [2]. The original implementation of the Gnutella protocol was completely decentralized, imposing no hierarchy or differentiation of any kind between peers. In Gnutella, peers are again connected in an unstructured fashion and search proceeds using a flooding-style algorithm. When a user issues a search, the peer submits the search to all of its immediate neighbours in the connectivity graph, who then process the search and forward the search on to their neighbours. The depth of the search is limited using a time-to-live (TTL) value, and duplicate searching is eliminated by attaching an ID to each search. Each peer who receives the search processes it on its local data and returns any matching results to the originating user. This process can be seen in the right half of Figure 1.

The original design of Gnutella had inherent scalability issues due to the massive amount of flooded search traffic that was required to return an acceptable number of results. These issues are discussed in the next section. Motivated by this problem, a new system emerged, most well-known as Kazaa [2]. Kazaa is fundamentally the same as Gnutella in design, except that is has a clear notion of peer heterogeneity. Certain peers are appointed more responsibility in the system and assigned superpeer status. A superpeer has significantly more neighbours than a regular peer and can be thought of as the top level in a two-level hierarchy of peers. **FIND FIGURE**. In this topology, regular peers are connected directly to a superpeer, which indexes the files stored at its neighbours. Search proceeds as in Gnutella, except the search need only be flooded between superpeers and not regular peers, significantly reducing messaging overhead.

The Kazaa network is currently the most popular file sharing P2P network, accounting for **percent** of all Internet traffic.

2.3 Scalability Issues

The original implementation of the Gnutella protocol was completely decentralized with no notion of peer heterogeneity. As mentioned above, this had inherent problems scaling to a large number of users. Ritter[27] points out that as the number of nodes grows, the amount of search traffic required to search the network grows to an unsustainable rate. To reach more nodes, depth (TTL) and degree (number of neighbours contacted at each peer) must be increased, which results in huge traffic requirements to propagate a single query for only slight increases in these values.

Figure 2: Collapse point of simulated Gnutella network, obtained from [10]

Chawathe et al.[10] examines the scalability problems in the original Gnutella. Of particular interest is their validation of the idea that under increasing load, there is a point where the system fails to be useful. Figure 2 demonstrates this through simulation of a 10,000 node network. In this simulation, offered load (queries per second) is increased and the success rate is measured. The success rate is the proportion of queries that returned results before the simulation ended. A query is unsuccessful if it remains stuck in a queue at a peer and no results were returned to the original requestor. As the load on the system increases, a distinct drop in the success rate is noticeable as the system becomes overloaded. This point is referred to as the collapse point of the system.

Modifications to the Gnutella protocol are also discussed in [10]. They were able to achieve three to five orders of magnitude improvement in system capacity through the combined use of: topology adaption, flow control, one-hop replication, and random-walk based search. Their topology adaption techniques assign more neighbours to higher capacity nodes, effectively creating superpeers.

After Napster was shut down by the US Legal system, Gnutella saw a huge influx of users, which made clear the inherent scalability problems [12]. As a result, Gnutella introduced the notion of ultrapeers into their system [6], which are functionally the same as superpeers in Kazaa. Zegura et al.[13] found that each ultrapeer maintains many (between 10 and 100) connections with regular peers and a small (less than 10) number of connections with other ultrapeers.

2.4 Host and Network Characteristics

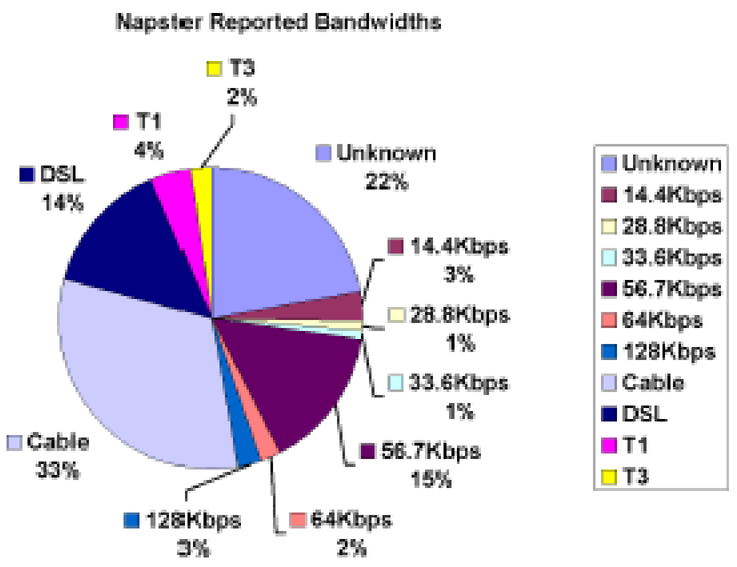

Figure 3: Reported bandwidths for Napster peers, obtained from [29]

A key observation made, which naturally fits the idea of superpeers, is that certain peers exhibit more server-like characteristics than others. While most peers have low-bandwidth and short connection time, Sariou et al.[29] found that 6% of Napster users and 7% of Gnutella users reported bandwidths of T1 or greater (see Figure 3), and that about 10% of sessions last more than 5 hours each. These peers are natural candidates for superpeers.

Although the Gnutella network topology is unstructured, Jovanovic et al.[18] found that the topology exhibits strong small-world properties. That is, peers tend to be in small well-connected clusters with relatives few links between clusters.

As well, Jovanovic et al.[18] found that node degrees in the Gnutella were power-law distributed. This means that peers with few neighbours are extremely common, where as peers with many neighbours are extremely rare.

These observations are important for understanding the context of our problem in currently deployed networks.

3 Graph Theoretic Approaches

We can gain insight into the limits of flood searching in P2P networks by focusing on the theoretical underpinnings of the problem using the language of graph theory. To reformulate our search problem into the terms of graph theory, we begin by thinking of our network as a an undirected graph $\mathcal{G}$ with a set of vertices $\mathcal{V}$ consisting of each node in the P2P network. There is an edge between any two vertices u, v in our graph if u can sends search queries directly to v. Before proceeding to discuss a graph representation for a P2P flood search, we first review some standard graph theory definitions that will be used throughout the section.

3.1 Definitions

Maximum Degree We denote Δ as the maximum degree over all vertices of a graph.

Minimum (u,v)-path We denote the minimum path between two vertices $u, v, u \neq v$, as $d(u, v)$.

Diameter We define diameter $\mathcal{D}$ as max $\{d(u,v) : \forall\, u, v \in \mathcal{V}\}$

Girth We define the girth to be the length of the smallest cycle in a graph.

Regular Graph An r-regular graph is a graph in which every vertex has degree r.

Complete Graph A graph in which every pair of vertices is connected by an edge.

3.2 Flood Search Representation

To investigate a lower bound on the efficiency of search in the P2P network without supernodes, we assume that given the bound on the degree of vertices, the graph will be constructed in the best way possible and the document will be placed in the worst position possible. We can transform this interpretation of the problem into the following graph theoretic question.

Given a graph $\mathcal{G}$ with n vertices and maximum degree Δ, can we bound the diameter of the graph $\mathcal{D}$ in terms of $n = |\mathcal{V}|$ and Δ? Conversely, given Δ and $\mathcal{D}$ how large can n grow?

3.3 Moore Bound

To answer the second question a natural starting point is a well known bound established by E. F. Moore. He is credited with the following theorem giving an upper bound on the number of vertices n in a graph with a given diameter $\mathcal{D}$ and maximum degree Δ.

Theorem 1 *If* $\Delta \geq 3$ *then*

$$\begin{aligned} n &\leq 1 + \Delta\left\{1 + (\Delta - 1) + \cdots + (\Delta - 1)^{\mathcal{D}}\right\} &(1)\\ &\leq \frac{\Delta(\Delta-1)^{\mathcal{D}} - 2}{\Delta - 2} = n_0(\mathcal{D}, \Delta) &(2)\end{aligned}$$

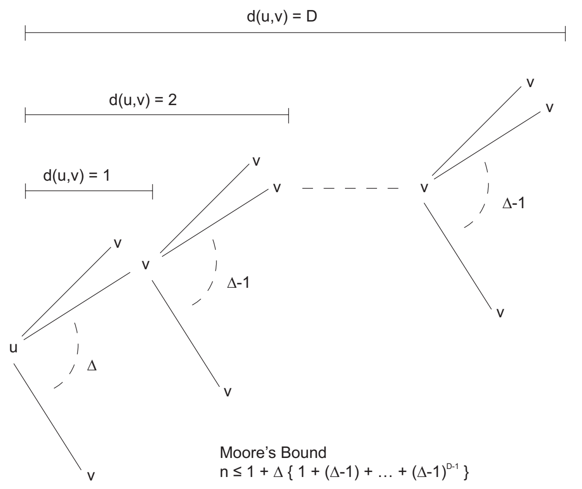

Figure 4: Illustration of the Moore Bound

An intuition as to how this bound is obtained can be developed by thinking about how many vertices can be distance $\mathcal{D}$ from an arbitrary node u. An illustration of such a construction can be seen in Figure 4. To count the total number of vertices of distance $\mathcal{D}$ from u we start with the Δ vertices that are distance 1 from u. We then we add $\Delta - 1$ for each of these Δ vertices to count the number of vertices that are distance 2 from u and we proceed until we have counted the number of vertices that are distance $\mathcal{D}$ from u.

The Moore bound can be given in terms of n and Δ to provide the following lower bound on diameter

$$\mathcal{D} \geq \frac{log\left[\frac{n(\Delta-2)+2}{\Delta}\right]}{log(\Delta-1)} \tag{3}$$

As the number of vertices in a graph grows the diameter will grow by roughly $O\left(log\ n\right)$, since Δ is fixed. It is noted by Bollobás that it is difficult to improve substantially on the Moore bound for general graphs [9], however for some special classes of graphs tighter lower and upper bounds have been established.

3.3.1 Moore Graphs

There is a class of graphs for which $n = n_0(\mathcal{D}, \Delta)$ which are accordingly called Moore graphs and are denoted by $(\mathcal{D}, \Delta)$. Such graphs happen to have the additional properties that they are Δ-regular and have girth $2\mathcal{D} + 1$. While Moore graphs achieve the smallest diameter that can be hoped for, few Moore graphs actually exist. When $\mathcal{D} = 1$ or $\Delta = 2$ we have Moore graphs that are complete graphs and odd cycles resepectively. For $\mathcal{D} = 2$ it was shown that Moore graphs can only exist when Δ takes on the values 3, 7 or 57 [17] (though a Moore graph with $\Delta = 57$ has yet to be constructed) and it was later shown that a Moore graph with $\mathcal{D} \geq 3$ and $\Delta \geq 3$ cannot exist [11, 7]. The Moore graphs (2,3) and (2,7) are shown in Figures 5 and 6.

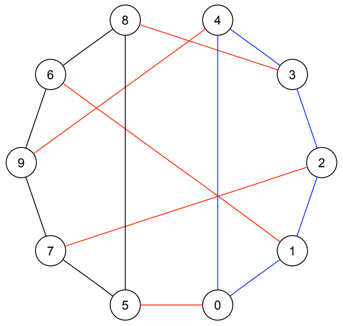

Figure 5: Petersen Graph, $\mathcal{D} = 2, \Delta = 3$

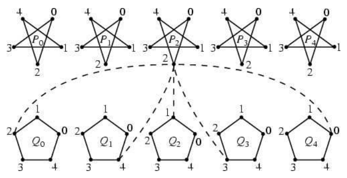

Figure 6: Hoffman-Singleton Graph Construction, $\mathcal{D} = 2, \Delta = 7$

3.3.2 de Bruijn Graphs

Given $h, k \geq 2$ and the set $W = \{1, 2, \ldots, h\}^k$, we define the de Bruijn graph $B(h, k)$ as a graph with the vertex set W and edge set composed of edges between every vertex $a = \{a_1, a_2, \ldots, a_k\}$ and the vertices $(*, a_1, a_2, ..a_{k-1})$ and $(a_2, a_3..a_{k-1}, *)$ which represent a forward and backward shift ($*$ denotes an arbitrary element of $\{1, 2, \ldots, h\}$). An example de Bruijn graph with $W = \{0, 1\}^3$ is shown in Figure 7.

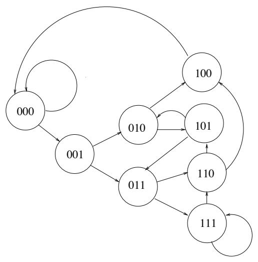

Figure 7: A Bruijn Graph [19], $W = \{0, 1\}^3$

The de Bruijn graph has $n = h^k$, $\Delta = 2h$ and $\mathcal{D} \leq k$. The low upper bound on the diameter of de Bruijn graphs has made them a desirable structure on which to model network topologies. Section

4.1.5 discusses Koorde [19] a Distributed Hash Table which uses the de Bruijn graph arrangement to obtain a low guaranteed search time.

3.3.3 Leland-Solomon Graphs

Given $k \geq 2$ and a vertex set of $V = \{0,1\}^k$, the Leland-Solomon graph G_k is a k-dimensional cube that can be constructed by joining a vertex $(a_1, a_2, \ldots, a_k)$ to the vertices $(a_2, a_3, \ldots, a_k, \bar{a_1})$, $(\bar{a_k}, a_1, \ldots, a_{k-2}, a_{k-1})$ and $(a_1, a_2, \ldots, \bar{a_{k-1}}, \bar{a_k})$.

The Leland-Solomon graph was constructed so that it is 3-regular, with $n = 2^k$ and $\mathcal{D} \leq 3k/2$.

For specially constructed graphs such as the Leland Solomon and de Bruijn graphs a good upper bound on diameter can be obtained, but for general graphs there remains a fair distance between these upper bounds and lower bound given in Equation 3. We will now proceed to an even tighter lower bound than the Moore bound by using the tools of random graph analysis.

3.4 Random Graphs

The theory of random graphs is a useful tool developed by Erdös and Rényi [14, 15, 16] to indirectly reason about the properties of very large graphs. Instead of showing that every graph must have a property $\mathcal{Q}$, the theory of random graphs essentially proceeds by examining all graphs on n vertices and enumerating the number graphs with property $\mathcal{Q}$. This is then compared to the number of graphs lacking property $\mathcal{Q}$. If the proportion of graphs with the property $\mathcal{Q}$ approaches one as $n \to \infty$ then we say that *almost every* graph has property $\mathcal{Q}$.

Some examples of properties that can be shown using random graphs are that

- *almost every* graph has diameter 2
- *almost every* graph is k-connected for a fixed $k \geq 3$
- *almost every* graph has no complete subgraph H_k where $k > 2log_2n$

Using random graphs Bollobás and de la Vega [8] were able to show that for a fixed $r \geq 3$ *almost every* Δ-regular graph has diameter

$$\mathcal{D} \geq \lfloor log_{\Delta-1}n \rfloor + \left\lfloor log_{\Delta-1}logn - log_{\Delta-1}\left(\frac{6\Delta}{\Delta-2}\right)\right\rfloor + 1 \tag{4}$$

This inequality gives a tightened lower bound on the diameter for *almost every* graph with large n, and is reasonable since P2P networks without supernodes will in general be regular.

This inequality can be used to show a lower bound on the number of vertices n given a diameter $\mathcal{D}$ and maximum degree Δ [9], so that we can bound n from above and below as follows.

For a fixed $\epsilon > 0$ and $\mathcal{D}$ sufficiently large

$$\frac{(\Delta - 1)^{\mathcal{D}}}{(2 + \epsilon)\mathcal{D}log(\Delta - 1)} < n \leq \frac{\Delta(\Delta - 1)^{\mathcal{D}} - 2}{\Delta - 2} = n_0(\mathcal{D}, \Delta) \qquad (5)$$

Consequently, in the domain of random graphs, diameter will also be tightly bounded. Bollobás notes that in regular graphs with large Δ, for many values of n a graph can only have one of three possible diameters [9]. However, for general graphs this diameter cannot easily be determined.

Examining the efficiency of flood search represented as the diameter of a graph shows that depending on the structure of a network there is a theoretical limit to the efficiency of search. Another approach to efficient search is to abandon the idea of flooding by imposing further structure on the network. In the next section we discuss a set of such approaches which are collectively known as Distributed Hash Tables.

4 Distributed Hash Tables

Distributed Hash Tables (DHTs) have emerged as a subtopic in P2P networks due to the observation that traditional P2P networks like Gnutella and Napster do not scale[26][27][25]. This has led to several different DHT designs for P2P networks [26][28][33][32] in 2001 that all implemented a scalable solution to the peer-to-peer problem by introducing some form of structure and a routing algorithm that did not flood the network. Each of these DHT solutions has the same basic structure. Each node joins some sort of structured overlay network using a predefined protocol and is assigned some sort of ID. It then keeps track of some set of other nodes and contains some data which it is sharing. Each piece of data in the network is mapped with a key/value pair. The nodes in the network forward any request to the node nearest to the location of the actual document which it knows about.

The rest of this section describes several of the DHT implementations. The focus of this will be to analyze these networks for their respective degrees and diameters.

4.1 DHT Implementations

4.1.1 Tapestry

Tapestry[33] was developed for the OceanStore[20] project at the University of California, Berkley. It is designed to be a global routing protocol that is scalable and fault tolerant. It uses the notion of a Plaxton location system[23] to develop a Plaxton mesh for routing in the network. The algorithm constructs neighbour maps, which are used to route messages in the network. The map contains the suffixes of node IDs and forwards request to the node that has the longest matching suffix to the current request. The algorithm achieves $log_b(n)$ search time where n is the number of nodes which can be addressed in the network and b is the base of the IDs. In order to obtain the search time, the node must store maps for each different digit in the ID ($log_b(n)$) and at each digit they must store one pointer for each seperate value of the digit (b). So, the complete neighbour map must be $b \times log_b(n)$ in size. In graph theory terms, this means we have a network that has a degree and diameter of $O(log_b(n))$.

4.1.2 Content Addressable Networks

Content Addressable Networks[26] (CANs) were introduced as a method of overcoming the scalability problem in peer-to-peer networks. CAN generates an overlay network which is logically a torus in d-dimensions. The torus is divided n zones using a d dimensional Cartesian coordinate system. Each key/value pair is mapped to a point on the torus using a hash function known to all of the nodes in the system. The key/value pair is stored at that point in the coordinate system and requests are routed to that node using the coordinate system of the torus. Each node keeps track of all of its neighbours which share coordinates in all but one dimension and off by one in the other. In a 2d Cartesian system, this corresponds to the manhattan neighbours of any node. This network gives us a degree of $2d$ and a diameter of $O(n^{1/d})$.

4.1.3 Pastry

Pastry[28] is another DHT routing scheme which was developed as a replacement for flooding P2P networks. It is a simple structure that organizes nodes in the network into a logical ring which is used to route the messages. Each node maintains a table that contains a row for each prefix of the a node's ID. Each row contains an entry for each possible digit that could be appended to the prefix. The system is set up so that if there are n nodes in the system and each digit is 2^b bits then the table has a total size of $(2^b - 1)(log_{2^b}(n))$. Using this table, it is relatively straight forward to route messages in the network in $log_{2^b}(n)$ hops. This procedure is identical to the process used in Tapestry[33].

4.1.4 Chord

Chord[32] again uses the ring structure which is common among many of the DHT implementations. This approach uses the idea of keeping $O(log(n))$ entries in the table where each entry i in the table is the node in the ring that is closest to the node which has and ID of $current + 2^{i-1}$. If the ID is greater than the number of nodes then a modulo of the number of nodes is taken. Chord also has some variations in the way that the nodes join the network.

4.1.5 Koorde

Koorde[19] is an extension of the Chord protocol that replaces the plain ring with the Bruijn graph using a vertex set of $\{0, 1\}^k$ which implies that each vertex has two edges. One of these edges goes to the node with an ID made by left shifting in a 0 bit into the current ID and the other is made by shifting in a 1 bit into the current ID. Unfortunately, this technique must be modified to allow for sparse graphs where the node at the end of the link may not exist. In the case of Koorde, they create two edges where the first is the original ring link in Chord to the next successor and the other goes to the first node that precedes the node which is twice the value of the current ID called a simulated de Bruijn edge. The node traverses the simultated de Bruijn edge every time that the node is a predecessor of the current shifted string. If not, they take the successor link around the ring until the predecessor is found. This allows Koorde to achieve $O(log(n))$ diameter with only a degree of 2. By increasing the number of links kept to $O(log(n))$, the diameter of the Koorde graph will be $O(log(n)/loglog(n))$.

4.1.6 HyperCuP

HyperCuP[31] is a proposed DHT algorithm which is based on the well studied hypercube structure. A hypercube of l dimensions has $n = b^l$ nodes where b is the number of nodes in a dimension. Given this structure, each node in the hypercube has a degree of $(b-1)l$ and the diameter is $log_b(n)$. The main contribution of this paper is to introduce a distributed algorithm for generating the hypercube. They show that they can add or remove a node from the hypercube with only $log_b(n)$ messages. This is done by only increasing the dimension when a node no longer has any links that can be filled. There is a problem with this algorithm in that it does not guarantee that it will build a balanced hypercube. The authors feel that the random nature of joins will sort out the balancing of the hypercube.

4.2 Comparison of Implementations

With each of these implementations, the algorithms obtain scalability by using some sort of structure which is constructed to have a logarithmic diameter and by having a routing algorithm, which does not flood. Table 1 shows the different search times of DHT algorithms with respect to the number of nodes. Koorde has the best performance of all the algorithms with a constant degree and logarithmic diameter. They showed this to be the optimal in the following lemma whose proof can be found in [19].

Lemma 1 *An n-node system with maximum degree d requires at least $log_d n - 1$ routing hops in the worst case and $log_d n - O(1)$ on average.*

All of the bounds presented in these papers assume that their algorithms are adding and removing nodes in manner that leaves the structure stable and relatively balanced. Liben-Nowell et al.[21] show that Chord can efficiently execute search in its overlay network with minimal housekeeping to keep the network balanced. Loguinov et al.[22] also looked at de Bruijn graphs in DHTs and came to the same conclusions as the Kaashoek and Karger[19], who worked on Koorde.

DHT	**Number of Nodes**	**Degree**	**Diameter**
Tapestry	n	$log_b(n)$	$log_b(n)$
CAN	n	$2d$	$O(n^{1/d})$
Pastry	n	$log_{2^b}(n)$	$log_{2^b}(n)$
Chord	n	$O(log(n))$	$O(log(n))$
Koorde	n	2	$O(log(n))$
HyperCuP	$n = b^l$	$(b-1)l$	$log_b(n)$

Table 1: Comparison of the different DHT implementations with respect to degree and diameter.

4.3 Comparison of DHTs and Flooding P2P Networks

We have examined the efficiency of the flood search used in P2P networks and observed that as the network grows, the time it takes to search the network also grows by roughly $O(logn)$. For large networks this leads to search times that are unacceptably long and as the Time To Live of a search grows, P2P networks run into scalability problems. DHTs have addressed some of these issues by using a directed search. However, the general bounds arrived at using graph theory are consistent with the results of many DHT architectures in which search (while directed) still grows by roughly $O(logn)$. While the DHT discussed surmount the scalability issues faced by P2P networks, they add a great deal of complexity to manage their specialized network topologies and have yet to see wide spread implementation.

5 Conclusion

From the analysis using graph theory, it can be shown that a peer-to-peer network with a fixed search time is not achievable for large numbers of nodes unless the degree is increased. This is because degree is inversely proportional to diameter and the number of nodes is directly proportional to the diameter. While both unstructured and structured networks have a logarithmic bound on the diameter, it can clearly seen that the structured networks scale better with respect to the number of nodes. The intuition behind this is not in the diameter of the network but rather the routing protocol. Unstructured network floods the network with messages, where the number of which grows exponentially with respect to the diameter, and structured networks uses a directed search that forwards the message only along a single path, which is linear with respect to diameter. Some possible reasons for the lack of acceptance for the structured networks are the increased complexity of implementation, the possbile loss of anonimity, and the cost of network maintenance messages.

6 Research Direction

o finish the analysis on P2P networks, we want to look at tightening the Moore Bound for graphs with special properties. One specific example is the graph which represents the Gnutella[18] network. That is a network with small connected clusters with relatively few links in between clusters.

We will also look at the theoretical impact of introducing supernodes. One goal of this project will be to determine how many super nodes should be added or what degree these nodes should have. We will investigate these results in terms of worst and average case improvements.

References

[1] Gnutella. http://www.gnutelliums.com/.

[2] Kazaa. http://www.kazaa.com.

[3] Limewire. http://www.limewire.com.

[4] Morpheus. http://www.morpheus.com/.

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

[6] C. Rohrs A. Singla. Ultrapeers: Another step towards gnutella scalability. http://www.limewire.com/developer/ultrapeers.html, 2002.

[7] E. Bannai and T. Ito. On finite moore graphs. *J. Fac. Sci. Univ. Tokyo, Sect. IA. Math*, 20:191–208, 1973.

[8] B. Bollobas and W. F. de la Vega. The diameter of random graphs. *Combinatorica*, 2:125–134, 1983.

[9] Bela Bollobas. *Random Graphs.* Academic Press, 1985.

[10] Yatin Chawathe, Sylvia Ratnasamy, Lee Breslau, Nick Lanham, and Scott Shenker. Making gnutella-like p2p systems scalable. In *SIGCOMM '03: Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 407–418. ACM Press, 2003.

[11] R. M. Damerell. On moore graphs. *Math. Proc. Camb. Phil. Soc.*, 74:227–236, 1973.

[12] Clip2 DSS. Gnutella: To the bandwidth barrier and beyond. http://lambda.cs.yale.edu/cs425/doc/gnutella.html, 2000.

[13] R Liston E Zegura. Determining characteristics of the gnutella network. http://www.cc.gatech.edu/ amogh/project1final.ps.gz, 2002.

[14] P. Erdos and A. Renyi. On random graphs. *Publ. Math. Derecen*, 6:290–297, 1959.

[15] P. Erdos and A. Renyi. On the evolution of random graphs. *Publ. Math. Inst. Hungar. Acad. Sci.*, 5:17–61, 1960.

[16] P. Erdos and A. Renyi. On the evolution of random graphs. *Bull. Inst Int. Statist. Tokyo*, 38:343–347, 1961.

[17] A. J. Hoffman and R. R. Singleton. On moore graphs with diameters 2 and 3. *IBM J. Res. Dev*, 4:497 – 504, 1960.

[18] M. Jovanovic, F. S. Annexstein, and K. A. Berman. Scalability issues in large peer-to-peer networks - a case study of gnutella. Technical report, University of Cincinnati, 2001.

[19] F Kaashoek and D.R. Karger. Koorde: A simple degree-optimal hash table. In *IPTPS*, 2003.

[20] John Kubiatowicz, David Bindel, Yan Chen, Steven Czerwinski, Patrick Eaton, Dennis Geels, Ramakrishna Gummadi, Sean Rhea, Hakim Weatherspoon, Chris Wells, and Ben Zhao. Oceanstore: an architecture for global-scale persistent storage. *SIGARCH Comput. Archit. News*, 28(5):190–201, 2000.

[21] David Liben-Nowell, Hari Balakrishnan, and David Karger. Analysis of the evolution of peer-to-peer systems. In *PODC '02: Proceedings of the twenty-first annual symposium on Principles of distributed computing*, pages 233–242. ACM Press, 2002.

[22] Dmitri Loguinov, Anuj Kumar, Vivek Rai, and Sai Ganesh. Graph-theoretic analysis of structured peer-to-peer systems: routing distances and fault resilience. In *SIGCOMM '03: Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 395–406. ACM Press, 2003.

[23] C. Greg Plaxton, Rajmohan Rajaraman, and Andr&#233;a W. Richa. Accessing nearby copies of replicated objects in a distributed environment. In *SPAA '97: Proceedings of the ninth annual ACM symposium on Parallel algorithms and architectures*, pages 311–320. ACM Press, 1997.

[24] D. Plonka. An analysis of napster and other ip flow sizes. http://moat.nlanr.net/natimes/april2001.pdf, April 2001.

[25] S. Ratnasamy, S. Schenker, and I. Stoica. Routing algorithms for dhts: Some open questions. In *IPTPS*, 2002.

[26] Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp, and Scott Schenker. A scalable content-addressable network. In *SIGCOMM '01: Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 161–172. ACM Press, 2001.

[27] Jordan Ritter. Why Gnutella can't scale. No, really., 2001.

[28] Antony I. T. Rowstron and Peter Druschel. Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. In *Middleware 2001: Proceedings of the IFIP/ACM International Conference on Distributed Systems Platforms Heidelberg*, pages 329–350. Springer-Verlag, 2001.

[29] S. Saroiu, P. Gummadi, and S. Gribble. A measurement study of peer-to-peer file sharing systems, 2002.

[30] Stefan Saroiu, Krishna P. Gummadi, Richard J. Dunn, Steven D. Gribble, and Henry M. Levy. An analysis of internet content delivery systems. *SIGOPS Oper. Syst. Rev.*, 36(SI):315–327, 2002.

[31] M. Schlosser, M. Sintek, S. Decker, and W. Nejdl. Hypercup – hypercubes, ontologies and efficient search on p2p networks. In *Workshop on Agents and P2P Computing*, 2002.

[32] Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek, and Hari Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *SIGCOMM '01: Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 149–160. ACM Press, 2001.

[33] Ben Y. Zhao, John D. Kubiatowicz, and Anthony D. Joseph. Tapestry: An infrastructure for fault-tolerant wide-area location and. Technical report, University of California at Berkeley, 2001.