Emerging Social Networks in Peer-to-Peer Systems

Yamini Upadrashta
Department of Computer Science
University of Saskatchewan
57 Campus Drive
Saskatoon, SK S7N 5A9
Canada
ysu156@mail.usask.ca
Supervisors: Prof. Julita Vassileva and Prof. Winfried Grassmann

ABSTRACT
The objective of this project is to simulate a Peer-to-Peer type of environment with the JADE multi-agent system platform to investigate the use of social networks to optimize the speed of search and to improve quality of service in the Peer-to-Peer environment. Our project uses the Gnutella protocol as a starting point. The Gnutella protocol broadcasts messages for searching files. This message passing generates much traffic in the network. This degrades the quality of service. We propose a model where each peer has a "friends list", for each category of interest. Once peers generate their "friends list", they use these lists for searching files in the network. The model has been implemented and some initial experiments have been performed.

KEYWORDS
Peer-to-Peer systems, social networks, multi agent systems

1. INTRODUCTION
The research area Peer-to-Peer (P2P) systems are fairly new in the field of distributed computing. P2P systems are typically decentralized, distributed and anonymous systems. One common protocol for P2P computing is Gnutella, which broadcasts messages to all the peers in the path of the query [8]. A querying peer sends the query to all of its peers, who in turn send the query to all of their peers until the query reaches a peer that produces a hit matching the query. This peer sends back a reply containing its address, the size of the file, speed of transfer, etc. The reply traverses the same path but in reverse order back to the querying peer. This passing of messages generates much traffic in the network, often leading to congestion and slow responses. Thus, the quality of service becomes poor, since the responses to queries are delayed.

We propose a model based on social network to alleviate this problem. Just as people have social contacts in different areas and use them accordingly, the model generates a "friends list" for each category of interest to the user. This list is used to send the initial messages when searching files in that category. We attempt to show that this model achieves responses faster than standard Gnutella. This paper is organized as follows. Section 2 gives an overview of the areas of peer-to-peer networks, social networks and the application of social networks in peer-to-peer networks. The conceptual design of our model is explained in section 3 and the conceptual design of the simulation is described in section 4. Details of the experimental and simulation parameters and some preliminary data are given in section 5. Section 6 concludes the paper and gives directions of future work.

2. PREVIOUS WORK
Peer-to-Peer systems are usually defined as distributed systems where peers or entities share computer resources and services by direct interaction among themselves [9]. There is no central server in P2P systems, unlike client-server distributed systems. Client-server systems are defined as a single or small number of servers connected to many clients. Clients issue a request and the server provides the client with the appropriate service. The resources that the peers share in the P2P systems could be files, CPU power, and disk space. Efficient searching of files is the key to achieve success in such networks. The next section explains how a search is performed in several popular P2P systems.

2.1 Peer-To-Peer Networks
P2P systems are characterized as robust, anonymous, flexible and self-organized systems [9]. Some of the popular P2P systems are Napster, Gnutella, FreeNet, KaZaa, and Limewire. Most of these systems are for music sharing and file sharing. P2P systems can also be used to share computing resources as in SETI@HOME. Some of these systems are explained below.

2.1.1 Napster
Napster [10] was a P2P system mainly for music sharing. It has a central server that keeps track of all the peers in the system. When a peer joins the network it advertises to the server the files it is willing to share with other peers. Thus, the server has information about all the peers and their files and maintains a centralized directory of the shared files. Request for files are send to the server. The server looks in its directory and finds the peer who has the file and sends the peer's address to the querying peer. The peers in the system can specify certain peers that they prefer to contact. The server takes the preference into consideration before sending a list of peers address to the querying peer. The querying peer than contacts any or all of the peers from the list. A file transfer takes place between them if both of them are in

agreement. The search is very fast since the central server has all the information about the peers. Fig. 1 shows the centralized model of Napster.

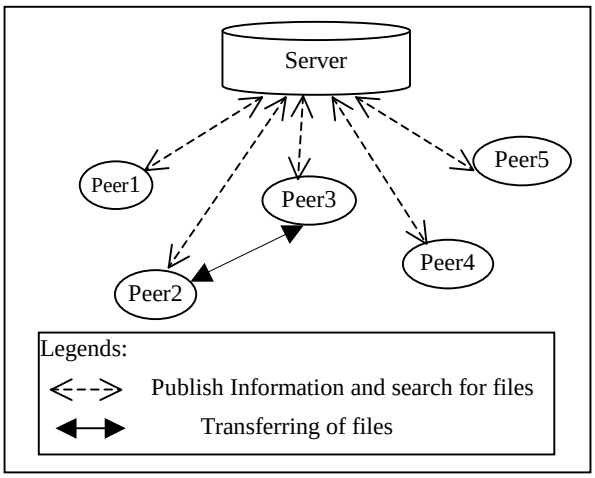

Figure 1: Napster's centralized model

The server thus only sends the addresses of the peer, which have the file. File transfer takes place without the server participating. But problems that plague all central systems, such as bottlenecks at the server, are still possible. Since the actual file transfer does not involve the server, some of the problems happen only when there are too many peers querying the system and the server is looking up its directory. Nevertheless, Napster was very efficient and robust. As the files are stored and transferred between peers, it is a P2P system. The only problem with Napster was that there was no anonymity in the system. The server holds all the peers' addresses and an index of their shared files in its directory. Since music copying is illegal under the copyright law, one could track the peers involved in sharing and downloading music files, by spoofing the server. Napster was shut down.

2.1.2 Gnutella

Gnutella [4] is a completely decentralized P2P protocol. Many systems have implemented this protocol to enable file sharing. Most of those systems like Limewire and KaZaa are music sharing systems. Since Gnutella is decentralized, there is no way that a peer in the system knows if a file exists and which peers have it. For this reason, Gnutella uses a broadcast protocol to search for files in the system. One problem here is that a peer has to know at least one other Gnutella peer to send requests to, but since the system is decentralized, there is no central server or peer that can provide peer addresses. This problem has been solved by publishing the addresses of some designated Gnutella peers on a website. When a peer enters the network, it contacts a designated peer and receives a list of other peers that have recently entered the network. A certain number of these (usually 7) become the neighbourhood of this peer. When the peer needs to send a query, it sends it to its neighbourhood. If those peers do not have the file, they in turn forward the request to their neighbourhoods. Fig. 2 depicts a simplified interaction of peers in the network. The different lines in the figure show the different stages of query propagation. In Fig. 2, Query Peer1, in the left corner, initiates the query and sends it to Peer2. Peer2 forwards the query to Peer7, Peer8 and Peer9. And so on.

Thus, it can be seen that a single query generates an exponential number of messages in the system. To limit the number of messages going through the system, the protocol puts a limit on the depth to which a query can be propagated (or the number of hops the query is forwarded)- a parameter called "Time To Live" (TTL). The TTL of a query is decremented with each forwarding and as soon as the TTL expires the query is no longer forwarded. Replies satisfying the query are sent back to the originator along the same path traveled by the query. This ensures anonymity on both sides since no single peer knows who requested and who responded to a particular query. In this way anonymity is preserved. At any time only queries and replies are circulating in the system.

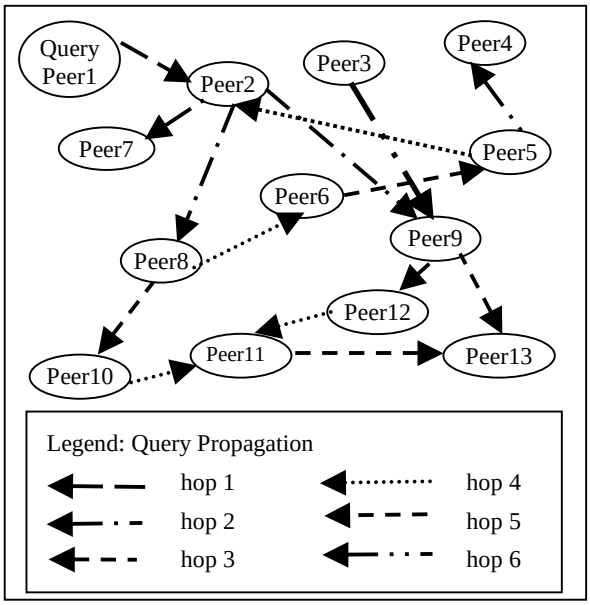

Figure 2: Gnutella's decentralized model

Gnutella is a robust system since the failure of any peer in the network does not affect the system. However, Gnutella leads to a lot of traffic, slow as a consequence, and to not guaranteed responses and poor quality of service. Even if a file exists in the system some peers may not be able to find it, because the TTL restricts the maximum number of hops along each path. Also, there is no way of knowing if a file exists in the system since there is no directory of the shared files in the network.

2.1.3 FreeNet

FreeNet [3] is based on the model of document routing. It is a decentralized file sharing system in which the anonymity of the peers is maintained. If the files exist in the system, they are located faster than in Gnutella. All peers in the network have peer-ids and have to contribute some disk space to the network so that files can be stored there. When a peer wants to store and share a file in the network, the file is given an id computed from the name of the file and its description using a hash function. The file, along with the file-id is propagated through the system. The file is

routed in the network and stored in the space contributed by the peer whose peer-id is the closest to the id of the file being stored. Thus, files with similar or close ids are clustered together in the network. Likewise when a query for a file is issued, the request is forwarded to the peer whose id is closest to the file-id being searched. Just like in the Gnutella protocol, at least one peer in the network has to be known to a new peer so that a query could be forwarded in the network. As in Gnutella, a few persistent peers in the network are published on the FreeNet website. From the system design, it can be seen that searches are faster since the peers send their request directly to the peer whose id matches closely the querying file-id. When a result of the query is found, the file is send through the query path and the file-id is stored in a local routing table. Subsequent queries are first checked in the routing table and the query is routed to the peer closest to that of the file-id. One problem in this system is that files storage is not persistent. Peers contribute storage space from their resources. If a peer reaches its full capacity and a file has to be stored at that location then the system uses the Least Recently Used (LRU) method to delete files and store the new one. There is a high probability that not frequently requested files will be dropped from the system. Another constraint is that one has to know the file name or exactly the same keywords that were used when the file-id was generated.

P2P systems like Napster, Gnutella and Freenet have been successful but there are some common problems with all the three systems. Napster provides a good speed for searching and retrieving music files but it was based on a centralized model and peers could be tracked easily which led to its shutdown. The Gnutella protocol is completely decentralized, but due to its flooding search algorithm, systems using Gnutella have performance problems, like huge network traffic, slower response and congestion. Using routing-based algorithm, as in Freenet, imposes limitations on users, for instance they have to know the semantic or content of the files they need to start searching, since exact file-ids are needed. None of these systems exploit the fact that people with similar interests are likely to store files that would be useful for all other people sharing those interests. People with similar interests form communities that allow them to exchange resources more efficiently.

2.2 Improving Search in P2P Systems

P2P systems are currently being studied to find efficient ways in searching for files or resources in the network. Some work done on optimizing the search for files is described below. A system called “Buddy Web” [19] was built in which routing is done based on similarity of interest. The main reason for building “the Buddy Web” system was to reduce the amount of traffic in and out of the university network, which costs the university money. For a given query, there is always the possibility that someone else on campus has already done the same type of query and has already received results for it. By caching these results and searching through them, the university can reduce cost and utilize the network better. Similarity, used for routing inside the network, is calculated by searching through the meta-data of file, e.g. the title tags downloaded by the user. Searching through the file for words selected by the user while browsing through the file contents can also be taken as indication of the important keywords for that file. The key words are stored in a vector. By summing up points for the words, which have already been assigned some weight by some weighing schemes, a calculation of interest is performed. A peer calculates similarity with other peers by comparing its own interest value to those of others. When a peer is querying, the query is first sent to the underlying network, called BestPeer [19], so that it can be routed first within the network to get results. The system sends the query to those peers whose similarity values are found to be close to this peer. Once results are found they are send to the querying peer. If the system does not find any results then it sends the request outside the network. This model works only for internal routing and not for external routing. An external search takes place in the traditional manner.

Another work on efficient search mechanism is the BestPeer system. BestPeer is a P2P network prototype, implemented in a university setting [13]. The main aim of this network is code-shipping and data-shipping. It is a completely decentralized network with many peers and a few servers. These servers essentially work like naming servers giving each registered peer in its network a unique name so that the rest of the peers can identify it. We are mainly interested in the BestPeer file-sharing model. In their model peers query for some file and get back some results. The peers, which returned the maximum number of results, are kept in a list. If after some queries the querying peer finds that there are some peers who return many answers it retains them in a list. Peers are retained in a list if they return many results or if the number of hops to a peer is large. The authors speak of quality of results when deciding about retaining peers but give no indication of how the peer can distinguish between poor and good results without user’s input. Resources at each peer are different. Each time a peer changes its query type there is no guarantee that the same peers will answer for the rest of the queries. The best peer list keeps changing as it sees new peers giving results each time. The model takes into consideration some kind of similar interest among peers, but it is not defined explicitly in the model as to what interests are and how they are formed. Their assumption of the topology of the network, for testing their model and reporting results, is a tree-topology. But in P2P systems peers connect, disconnect and reconnect randomly. A tree topology is very unlikely to be formed.

To provide better service in P2P systems research is being done to investigate how the traffic generated due to the broadcast protocol in Gnutella system can be minimized while still preserving the quality of search results. One such system is finding “good peers” in a P2P system [15]. Peers that have sent a “good” response to a peer’s request are entered in a special list by the peer, following the assumption that these peers may also have good resources for subsequent queries in this area. And so the peer now sends subsequent requests to the peers in its list. This reduces traffic in the network as peers now send queries selectively to other peers. The authors show that this selective dispatching of queries does not affect the search results at the end and one gets good responses in short time. However, a list of “good peers” is related to one search criterion only. When the peer switches its search to another criterion, broadcasting to all peers will occur again. Building of a “good peers” list for a given search criterion can happen only after a certain delay, which is needed so that the peer can register new useful peers in that search cycle. This works if the user is consistently searching for a single criterion several times in one session. However, users search typically for more

than one criterion at a time. Therefore the model will keep generating huge traffic in the network as the user switches search criteria.

2.3 Social Networks

People have already started considering and applying social networks concepts for optimizing search in P2P systems. Social Networks are groups of people, be it in a social setting or an organization connected by relationships [21]. Stanley Milgram, in the late 1960's, did one of the first experiments to investigate social networks [11]. In his experiment he addressed letters to a particular stockbroker in New York and gave them to people randomly picked at locations in the United States far away from that of the final receiver. The condition for passing the letter, so that it reaches the addressee, was that one could post it only to people they knew personally by first name. Eventually most of the letters reached the destination, and the average number of hops was six. Thus the "six degrees of separation" phenomenon came into being.

Studies have shown that social networks benefit people in everyday lives. They are useful in propagating information and also in finding information. This fact was exploited to develop an expert system where experts in a subject are located on web [7]. The system was build to get expert advice in some fields. Users who registered in the system had to fill a form about their publications. In this way the system gained knowledge about social networks based on co-authorship papers. This allowed the system, even when some experts themselves would not have the time or may not like to register in the system, to be referred to as experts by the system since they were found by co-authorship in other expert's publication.

Studies have shown that "weak ties" are more beneficial in a network than "strong ties". Weak ties are among those people that are not in the same community or coalitions. Strong ties on the other hand are those that connect people in the same community. People involved in strong ties usually interact frequently and share equal knowledge. The benefit of people with weak ties is that they provide information about experts or knowledgeable people that are not in the community, therefore spreading knowledge across communities [5].

Social networks are also used to study the interaction pattern between groups of people in a certain context. They help in understanding to what degree the behavior of an individual is influenced by constraints in their environment and how individuals use their social network for their benefit [20]. It has been found that, if possible, people do manipulate circumstances such that they benefit in socializing with their choice of people [12]. From this we can observe how social network once established can be used for ones' own benefit.

Social networks can be studied by computer simulation to investigate the evolution of societies of artificial agents simulating real people. Simulations allow to study patterns, diversity and behavioral changes in groups of people due to changes in its environment [14]. Social networks can also be studied using visualization techniques to show how strong are the relations between people in a group and between groups of people [2]. The visualization techniques involve different colors depicting the strength of the relations between people. They also have techniques allowing people in relations to be differentiated, for example, different shape of nodes for distinguishing genders.

P2P systems have also been studied in the area of Multi Agent Systems (MAS). Here each peer is assumed to be an agent that makes autonomous decisions and communicates with other peers / agents. The communication follows patterns of weak ties and strong ties. Peers involved in weak ties were found to be more valuable as they have more contacts with other peers. These peers provide referrals and expertise in the form of forwarding requests to other peers capable of responding correctly [18].

3. CONCEPTUAL DESIGN

In this study we propose a model in which peers keep a list of other peers who they see as being similar to them in some criterion. Each peer can have very many different criteria and they can have a list of peers associated with each criterion. The system has a number of peers, and each peer has some files. Peers share these files with other peers in the network and these files are representative of the peers' interests. The files are broken down according to categories. Peers can show interest in different categories. A category is defined as an area characterized by a set of topics or keywords [17]. For example, topics like distributed databases, and peer-to-peer systems characterize the area of distributed systems, which is a category in our model. There are about n categories in the system and each peer has interests in a few of these categories m, i.e., $m < n$. In real life both categories and files would have descriptive titles, say strings. Here, we represent both categories and files by numbers. For instance, file 25 may belong to category 2.

From time to time, a peer wants to search and have access to files of other peers. In order to do this efficiently, each peer keeps a list of friends, for each category. A peer randomly, with a higher probability, generates a query in one of its interest categories and with a lower probability in other categories. A peer is not allowed to request a file it already has in its resources. This restriction is based on the assumption that one would not request that one already has. There is, however, no restriction on how many times a peer can request a file. This request results in a hit, if the recipient has the requested file, or it is passed on to other peers, if its TTL has not expired yet. The peer originating the query waits for responses to its file request from other peers in the network.

All hits are counted. The peer who requested the file updates its "friends list" by adding the responding peer. The idea for keeping the responding peer in a list is that if a peer has files pertaining to a given category, it probably is interested in that category. Therefore it may have additional resources in the same category. So it is quite possible that this peer would be responding again to another query in that category.

All peers in the system create "friends list" in this way. Hence the queries are more likely to be forwarded to peers with similar interest. Thus, the chances of getting replies back faster are higher. By keeping "friends list" for each category, a peer querying in a given category sends messages to mostly those peers

interested in the same category and avoid broadcasting in the network. This reduces traffic in the network.

The "friends list" is updated by the following mechanism. After the interaction or file transfer takes place, based on the corresponding interaction value (success or failure), the peer calculates the strength of the relationship for that peer. The formula to calculate the strength of the relationship is given in Fig. 3 [17].

$$relationship\_new = \alpha * relationship\_old + (1- \alpha) * experience$$

Figure 3: Formula to calculate strength of a relationship

where α is a value between 0 and 1, we have used a conservative value of 0.8 and experience is a variable with two pre-defined values denoting success and failure of the interaction respectively.

Another way to calculate and learn about relationships might also be to keep a sum of all experiences, as defined above, and to compute the average experience with a peer by dividing the sum of experiences by the number of experiences.

We have chosen the first formula to calculate strength of relationship. The strength of the relationship is maintained between 0 to +1, where 1 denotes a strong relationship. If the strength is higher than 0.2 then it is stored in the "friends list" of the peer for this category, otherwise it is deleted from the list.

4. EXPERIMENTAL DESIGN

Our experimental model has some modifications as compared to the standard Gnutella in order to simplify the protocol for the simulation. The first modification is that peers who have the requested files send directly responses back to the peer who originated the query and not through the queried path. This reduces the overall time for a reply to come back to the peer originating the query as compared to Gnutella and it leads to non-anonymity in the system, as the owner of the file and the downloading peer are known. Assuming that the files shared in this system are publicly available, not copyrighted and not obscene documents, anonymity that is generally valued highly in P2P systems can be parted with to achieve better quality of service. The second modification is that the file is sent back by the responding peer to the peer who originated the query as the response, i.e. there is no "query hit" response sent back, followed by a transaction for downloading the file initiated by the peer who originated the query, but the whole interaction happens at once. The interaction between peers can result in success or failure. The failure can be due to a connection failure, poor quality of the resource or irrelevant response to the query. We model success of interaction as a boolean variable that is generated randomly. This modification is for the entire system and only for the purpose of our simulation. We compare results of the version with "friends list" with the version of without "friends list" on the same modified system simulation. Hence our comparison results are not affected by this modification.

To maintain the same initial configuration of the system, we assume that peers do not replicate the file they have requested. The file distribution in the system therefore remains unchanged during the simulation. For the purpose of this project the system assumes that all the peers created at the start of the system are active during the entire simulation. This assumption makes the simulation less complex. Following the same assumption, the number of peers in the system does not change during the simulation but can be changed for different simulation runs.

Peers generate queries that are random natural numbers, $F = 1,2,3\ldots$ representing files. The files (i.e., the numbers representing them) are classified into categories, also represented by numbers $C = 1,2,3$. The file classification in categories is predetermined. The number of files per category can be changed. To keep the model simple, we have an equal distribution of files into categories.

Other values chosen for the simulation are: three categories of interest in total in the system, from these three categories in the system, each peer is assigned two categories of interest randomly. Each peer stores ten files in their categories of interests. The values were chosen in such a manner so that there would be enough interactions between the peers to generate, maintain and use their "friends list". If, for instance, the number of categories were six, then, if each category has ten files there would be sixty files. If there were ten peers in the system, seven queries from each peer would be needed to generate two queries in the same category. For ten peers the minimum number of queries needed to generate a "friends list" and to use this list would be at least seventy queries. With our settings the minimum number of interactions in the same category for a peer if there are three categories would be three, for the same number of queries, i.e., seventy in the system. From this calculation we would have a good data set by having three categories in the system.

When the system starts there are no relationships among the peers in the system. This is because there have been no interactions between them as yet. Interactions happen as queries are generated and responses come. Queries are generated by the system by randomly choosing peers from the list of all peers in a fixed interval of time. The peer randomly selects a category from its two categories of interest with 90% probability and 10% probability in other categories. Once the category is chosen, the file number to be queried is randomly generated. The peer then requests the file from other peers. Those peers search through their resources and return a "hit" message to the queried peer directly if it finds it. A "hit" message is equivalent to downloading the file. No more messages take place between the responding peer and peer originating the query with regards to that query. Each query is identified by a unique id, used to keep track of which query is being processed, and the originator of the query so that files can be send back to that peer. The unique id also helps to discard the query if the query gets forwarded more than once to any peer. This way the peers do not have to process the same query once again. This reduces the number of messages circulating in the system.

The peer originating the query keeps track of which peers responded to each of its queries and also keeps track of the outcome of the interaction (generated randomly). It then

calculates strength of the relationship with that peer. After that it stores the peer in a relation vector containing peer id, the strength and the category of the query. If the strength of the relation of a peer in a category goes below 0.2 that peer will be dropped from the "friends list" in this category. If the peer is a member of other 'friends list" of the requesting peer, in different categories of interests, these lists will not be affected.

Our hypothesis that the "friends list" allows reduced search time and reduces traffic is tested via simulation. During the simulation we collect data to find the average time taken for replies. We hope to show that the "friends list" reduces the time for searches.

The model allows various possible initial configurations along two main dimensions. The first dimension is the number of categories in a peer:

1) Each peer has equal interest in all the categories assigned to that peer.
2) Each peer has different degree of interest in the different categories of interest assigned to that peer.

The second dimension is the file distribution among peers:

1) Equal number of files is allotted to all peers in the system.
2) The files are distributed so that some peers get a larger number of files. These become large peers. Other peers get fewer files and so they become small peers.

The model was tested with one configuration: equal interest in both the categories in the peer and equal number of resources in each peer.

5. PRELIMINARY RESULTS

The simulation has been implemented in Java on JADE [6], a Multi-Agent platform. There are other multi agent platforms that can be used, for example, FIPA-OS and the Agent Development Kit (ADK), but JADE was chosen because of its ease of use. JADE is FIPA-Compliant, multi agent platform specification using Java. JADE [1] has the abstract notion of behaviours associated with each action. The communication between the agents in Jade is through ACL (Agent Communication Language). Each agent in Jade is a thread. When an agent receives an ACL message, the agent retrieves the relevant section of the message interprets the message and carries out appropriate methods or tasks according to the interpretation of the message.

The peers in the simulation are represented as Jade Agents. Simulation and testing was done on a windows PC machine with 523 KB RAM and Win 2000. Ten peers were created in the system and these peers interacted among themselves by querying and responding. There were three categories in the system in total, and ten files in each category (a total of thirty files in the system). Each peer was interested in two fixed categories. The simulation initially needed approximately 30 seconds to completely set up the peers and its resources. Hence, queries were generated at a 30 seconds interval. A random generator picks the peer that would initiate the query. The query is generated randomly in one of the two categories of interest of the peer. The peer originating the query records the time at which the query was sent. When a reply comes, the peer notes the time at that moment and then calculates the round trip time (RTT) defined as the total time elapsed from sending the query until the reply came through. This data is recorded in a file.

As described previously, when the system starts there are no "friends lists" for any peer. A peer generates a query and if "hit" messages are received, the querying peer builds a "friends list" for that category of interest. However, it may not be the case that all the peers in the system generated queries in the simulation or gotten "hits" for their queries. As a result those peers could not build "friends lists". Some peers may have built "friends lists" for a category but did not generate queries again in that category to use their "friends lists". Thus, in each simulation we have some peers who built "friends lists" and used these lists, some peers who built "friends lists" but could not use them and some peers who could not build "friends lists" for any category. The simulation results show RTT recorded by two peers, who have been found to build "friends lists" and used them for searching in subsequent queries.

The simulation was run three times to ensure that the random generation of queries by peers does not affect our results. The configuration was same for all the three simulations, i.e., the peers had equal interest in the two categories assigned to it and each peer had the same number of files to share. The data obtained from this experiment is presented in Table 1.

Table 1: Values of the Round Trip Time (RTT)

Run	without list RTT (ms)	with list RTT (ms)
1	315.50	175.33
2	431.00	289.17
3	378.50	152.75

Table 1 shows the average values for the RTT for queries issued by peers with and without a "friends list" for the three simulation runs. The data in second ("without") column of Table 1 is the average round trip time for 2 queries when the peer did not have the friend's list. The data in the third column of Table 1 is the average round trip time for 2 queries when the peers have list of friends' for that category. The queries for both columns were in the same categories.

It can be seen that the time for a query to get results back when the peers keep no "friends list" is higher as compared to the time taken when the peers keep "friends list" and send queries directly to peers in the friends' list. This is because peers who keep "friends' list" in a given category have a greater likelihood of accessing relevant resources in that category by querying first its friends rather than sending their requests to random peers. Since all peers are homogeneous, the friend-peers that receive the query will forward it further to their friends who they know are interested in that category. In this way peers who share similar interests are queried first (of course, if they are available) and the likelihood of one of them having the file is higher than that of

random peers. Therefore, the round trip time of the queries is reduced.

6. CONCLUSIONS AND FUTURE WORK

The objective of the project was to investigate the use of social networks to optimize search and quality of service in the Peer-to-Peer environment. We simulate a Peer-to-Peer type of environment with JADE Multi-agent system platform. In our model each peer builds a "friends list", for each category of interest. Once peers generate these "friends lists", they use them for searching files in the network. From the results obtained we see that creation of "friends list" helps in reducing search time for queries.

Future work would be to see how the system behaves when peers are programmed to learn from other peers' queries. At present each peer discovers on its own about other peers by sending a query and building "friends list" for that category. However, a peer can also learn by observing the traffic in the system, i.e., by keeping track of queries passing through it and the peers' that initiated these queries, and adding those peers to its "friends list" in that category.

Another area for future work would be to see the impact on system's behaviour when peers make decisions to maximize their utility. Peers can make decisions depending on the availability of resources (storage, bandwidth, CPU) to "specialize" either as a "middleman" peer, keeping a large list of friends, that have the documents, or to specialize as a "server" peer, that stores the actual documents. All peers in the system may not have large storage space to keep files and enough bandwidth for giving the files to other peers. Each peer can maximize its own utility based on storage space and available bandwidth and decides whether to keep friends list for categories or to store documents. Storing documents would utilize more disk space as compared to keeping lists of peers to contact. As a result it is possible that most peers would prefer to keep friends lists. But when some peers utilize certain documents frequently, it would be an incentive for them to store and share these documents rather than having to download them each time these documents are needed. We can create a system with utility maximizing peers and explore how networks of specialized interest groups are formed and track peers' specialization within various lists.

7. REFERENCES

[1] Bellifemine, F., Poggi, A. and Rimassa, G. (Apr. 1999), "JADE – A FIPA- compliant agent framework", Proceedings of PAAM'99, London, 97-108.

[2] Freeman, L. C. (2000), "Visualizing Social Networks", Journal of Social Structure, Vol. 1, No. 1. http://zeeb.library.cmu.edu:7850/JoSS/article.html

[3] FREENET, 2001, www.freenetproject.org.

[4] GNUTELLA, 2001, http://gnutella.wego.com.

[5] Granovetter, M., (1973), "The Strength of Weak Ties", American Journal of Sociology, Vol. 78, 1360-1380.

[6] JADE 2.61: http://sharon.cselt.it/projects/jade/

[7] Kautz, H., Selman, B. and Shah, M. (1997), "Referral Web: Combining Social Networks and Collaborative Filtering", Communications of the ACM, Vol. 40, No. 3, 63-65.

[8] Krishnamurthy, B., Wang, J. and Xie, Y., (2001), "Early Measurements of a Cluster-based Architecture for P2P Systems", Proceedings of the First ACM SIGCOMM Workshop on Internet Measurement Workshop, San Francisco, CA, 105-109.

[9] Milojicic, D. S., Kalogeraki, V., Lukose, R., Nagaraja, K., Pruyne J., and Richard, B. (2002), "Peer-to-Peer Computing". Internal Report Hewlett Packard, March 8.

[10] NAPSTER, 2001, www.napster.com.

[11] Newman, M. E. J. (2000), "Models of the Small World: A Review", Journal of Statistical Physics, 101, 819-841.

[12] Newcomb, T. M., Turner R. H. and Converse P. E., (1965), " Social Psychology: The Study of Human Interaction", Holt, Rinehart and Winston, New York.

[13] Ng, W., Ooi, B. and Tan, K., 2002, "BestPeer: A Self-Configurable Peer-to-Peer System", In Proceedings of the 18th International Conference on Data Engineering, San Jose, CA, 272.

[14] Pearson, D. W. and Boudarel, M. R. (2001), "Pair Interactions: Real and Perceived Attitudes", Journal of Artificial Societies and Social Simulation, Vol. 4, No. 4, http://www.soc.surrey.ac.uk/JASSS/4/4/4.html.

[15] Ramanathan, M. K., Kalogeraki, V. and Pruyne, J. (2001), " Finding Good Peers in Peer to Peer Networks", Hewlett Packard Technical Reports.

[16] Sripanidkulchai, K., (2001), "The popularity of Gnutella queries and its implications on scalability", Featured on O'Reilly's www.openp2p.com website, February 2001.

[17] Vassileva, J. (2002), "Motivating Participation in Peer to Peer Communities", Proceedings of the Workshop on Emergent Societies in the Agent World, ESAW'02, Madrid, Spain. http://www.ai.univie.ac.at/%7Epaolo/conf/esaw02/esaw02accpapers.html.

[18] Venkataraman, M., Yu, B. and Singh, M. P. (2000), "Trust and Reputation Management in a Small World Network", Proceedings of Fourth International Conference on MultiAgent Systems, 449-450.

[19] Wang, X., Ng, W., Ooi, B., Tan, K. and Zhou, A., (2002), "BuddyWeb: A P2P-based Collaborative Web Caching System", a position paper in Peer to Peer Computing Workshop (Networking), Pisa, Italy. http://xena1.ddns.comp.nus.edu.sg/p2p/BuddyWeb.ps.

[20] Webster, C. M., Freeman, L.C. and Aufdemberg, C. (2001), "The Impact of Social Context on Interaction Patterns", Journal of Social Structure, Vol. 2, No. 1. http://moreno.ss.uci.edu/webster.pdf.

[21] Wellman, B., "An Electronic Group is Virtually a Social Network", In Sara Kiesler ed. (1997), Culture of the Internet, Lawrence Erlbaum, Hillsdale, NJ, 179-205.