Semantic Social Routing in Gnutella

A Thesis Submitted to the College of
Graduate Studies and Research
in Partial Fulfillment of the Requirements
for the Degree of Master of Science
in the Department of Computer Science
University of Saskatchewan
Saskatoon, Saskatchewan

By

Yamini Upadrashta

Keywords: social networks, peer-to-peer, semantic routing

Permission to Use

In presenting this thesis in partial fulfilment of the requirements for a Postgraduate degree from the University of Saskatchewan, I agree that the Libraries of this University may make it freely available for inspection. I further agree that permission for copying of this thesis in any manner, in whole or in part, for scholarly purposes may be granted by the professors who supervised my thesis work or, in their absence, by the Head of the Department or the Dean of the College of Graduate Studies and Research. It is understood that any copying or publication or use of this thesis or parts thereof for financial gain shall not be allowed without my written permission. It is also understood that due recognition shall be given to me and to the University of Saskatchewan in any scholarly use which may be made of any material in my thesis.

Requests for permission to copy or to make other use of material in this thesis in whole or part should be addressed to:

Head of the Department of Computer Science
University of Saskatchewan
Saskatoon, Saskatchewan, Canada
S7N 5C9

ABSTRACT

The objective of this project is to improve the performance of the Gnutella peer-to-peer protocol (version 0.4) by introducing a semantic-social routing model and several categories of interest. The Gnutella protocol requires peers to broadcast messages to their neighbours when they search files. The message passing generates a lot of traffic in the network, which degrades the quality of service. We propose using social networks to optimize the speed of search and to improve the quality of service in a Gnutella based peer-to-peer environment. Each peer creates and updates a "friends list" from its past experience, for each category of interest. Once peers generate their friends lists, they use these lists to semantically route queries in the network. Search messages in a given category are mainly sent to "friends" who have been useful in the past in finding files in the same category. This helps to reduce the search time and to decrease the network traffic by minimizing the number of messages circulating in the system as compared to standard Gnutella. This project will demonstrate by simulating a peer-to-peer type of environment with the JADE multi-agent system platform that by learning other peers' interests, building and exploiting their social networks (friends lists) to route queries semantically, peers can get more relevant resources faster and with less traffic generated, i.e. that the performance of the Gnutella system can be improved.

Acknowledgments

First of all I express my sincere thanks to my supervisors, Dr. Julita Vassileva and Dr. Winfried Grassmann for their support, patience, guidance and financial assistance throughout this project.

I would like to thank the members of my advisory committee: Dr. Ralph Deters, Dr. Gord McCalla, and Dr.Chris Zhang (external) for their valuable advice and suggestions.

I would also like to thank the students, staff and faculty of the Computer Science Department and especially to the students of MADMUC lab for their support. Also thank you to Ms. Jan Thompson for her help and support.

I would like to thank all my friends and family for their understanding and support. I would like to express a very special thank you to my parents and to my husband without whose support and encouragement this work would not have been possible.

I would like to dedicate this thesis to my parents and to my husband.

Table Of Contents

List of Figures

List of Tables

CHAPTER 1

Introduction

The research area of Peer-to-Peer (P2P) systems is fairly new in the field of distributed computing. P2P systems are typically decentralized, distributed and anonymous systems. In contrast to client-server applications, there are no dedicated servers and clients in a P2P system. In a client-server application, clients issue requests and the server provides the client with the appropriate service. In contrast, each node in a P2P system can take the role of both a server and of a client. Nodes in P2P systems are also called servents, a combination from server and client. Each node shares some of its resources with other peers, and it can use the shared resources of its peers. Current P2P systems are used to share resources like storage space, CPU power and data files. The data files shared are in domains such as music, academic / research communities and computation systems. Some examples of P2P systems are Napster, KaZaA, SETI@HOME, FreeNet and MojoNation.

One common protocol for file-sharing P2P applications is Gnutella 0.4 (Krishnamurthy et al., 2001) (Sripanidkulchai, 2001). A querying peer sends a query to all of its neighbour peers, who in turn send the query to all of their neighbour peers, and this spreading broadcast continues until the query reaches a peer that has a file that matches the query, or until a certain predefined maximal number of forwards is reached. If a peer is reached, it sends back a reply containing its address, the size of the file, speed of transfer, etc. The reply traverses the same path as the query but in reverse order back to the querying peer. In this way each query is propagated to up to n^p other peers (some peers may get the same message twice), where n is the number of neighbour peers and p is the maximum number of forwards called “time to live” (or TTL) of the query. 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.
Peers in the P2P system interact with each other to find resources. If they can remember which peers generated good responses for different types of queries, they can use this knowledge to forward future queries of this type to those peers, and not to just a set of random neighbours. This is similar to the way people use their knowledge from previous interactions with other people to form social networks. Research studies in sociology and in the area of social networks have found that social networks help in finding and dissipating information quickly and efficiently.

We propose a query routing approach based on social networks and the semantics of queries to alleviate the performance problems caused by the flooding algorithm in Gnutella systems. This approach introduces the concept of friends lists (lists of peers that have been shown to be useful) in different semantic areas so that queries can be routed semantically to these peers. Using friends lists potentially helps in reducing search time and decreasing traffic by minimizing the number of messages circulating in the system as compared to standard Gnutella.

The goal of the thesis is to demonstrate that by learning the other peers' interests, building and exploiting their social networks (friends lists) to route queries semantically, peers can get more relevant resources faster and with less traffic generated, i.e. the performance of the Gnutella 0.4 system can be improved. The thesis also investigates what conditions bring the greatest improvement. Methods of theoretical analysis and simulation are used. The potential contribution of the thesis is a modified protocol and a node architecture that allows reducing search time and decreasing traffic in the Gnutella 0.4 system.

This thesis is organized as follows. Chapter 2 gives an overview of the areas of peer-to-peer file-sharing applications, focussing on Gnutella-based systems. It discusses other approaches to improving the performance of Gnutella and gives a brief overview of the area of social networks. The conceptual design of our social-semantic approach is

explained in chapter 3 and the design of the experimental model is described in chapter 4. Chapter 5 presents and discusses the results and Chapter 6 gives future directions and conclusion.

CHAPTER 2

Literature Survey

Peer-to-Peer (P2P) systems are usually defined as distributed systems where peers or entities share computer resources and services by direct interaction among themselves (Milojicic et al., 2002). Unlike in client-server distributed systems, there is typically no central server in P2P systems. Client-server systems are defined as a single or small number of servers connected to many clients. Clients issue requests and the server provides the client with the appropriate service. In P2P systems, in contrast, the individual nodes can be both clients and servers, i.e., the node issues requests and also provide services and/or resources. The resources that the peers share in the P2P systems could be files, CPU power, and disk space. In client-server systems, the server needs to be maintained so that the system works efficiently. This involves cost and resources, for example, a system administrator to look after the server. The system can be scaled, in regards to how many nodes can connect to the server, only to a certain limit depending on the memory, CPU power and storage of the server. In P2P systems, since there is no central server, no maintenance is required for the system to work and thus there is no cost for up-keeping. This system has no limit on the number of nodes it can handle. An example of a P2P system is the e-mail system. E-mail is autonomous, robust and distributed. Sharing music and movies files is currently the area were most P2P networks have grown. Efficient file-search is the key to achieve success in such networks. The next section explains how search is performed in several popular P2P systems.

2.1 Review of the Area of File-sharing Peer-To-Peer Networks

P2P systems are robust, anonymous, flexible and self-organized systems (Milojicic et al., 2002). P2P can be classified as resource sharing systems and P2P

computing platforms (Milojicic et al., 2002). Resource sharing focuses on sharing data, hard drives and files. Even though the area of sharing computing cycles is promising, we will not discuss these approaches but will focus on file-sharing P2P systems.

File sharing systems are classified as Centralized P2P, Pure Distributed P2P, Semi-Centralized P2P and document routing systems. Some of the most popular P2P systems in these categories are *Napster*, *Gnutella*, *FreeNet*, *KaZaA*, and *Limewire*. Examples of P2P systems that share computing resources are SETI@HOME and Avaki. There are two main types of P2P file-sharing architectures: centralized and decentralized. In addition, there are hybrid architectures and document-routing architectures. These are presented below along with characteristic example systems.

2.1.1 Centralized P2P System: Napster

Napster (Napster, 2001) is a P2P system used mainly for music file 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 sharing with other peers. Thus, the server has information about all the peers and their files and maintains a centralized directory of the shared files. Requests for files are sent to the server. The server looks up in its directory 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; for example a peer can specify that only those peers with certain bandwidth B be returned. The server takes the preferences into consideration before sending a list of peers' addresses to the querying peer. The querying peer then contacts any or all of the peers from the list. A file transfer takes place between them if both of them are in agreement, for example if the bandwidth is agreeable. The search is very fast since the central server has all the information about the peers. The server thus only sends the addresses of the peers that have the file. File transfer takes place without the server participating. As the files are stored and transferred between peers, it is a P2P system. Fig. 2.1 shows the centralized model of Napster.

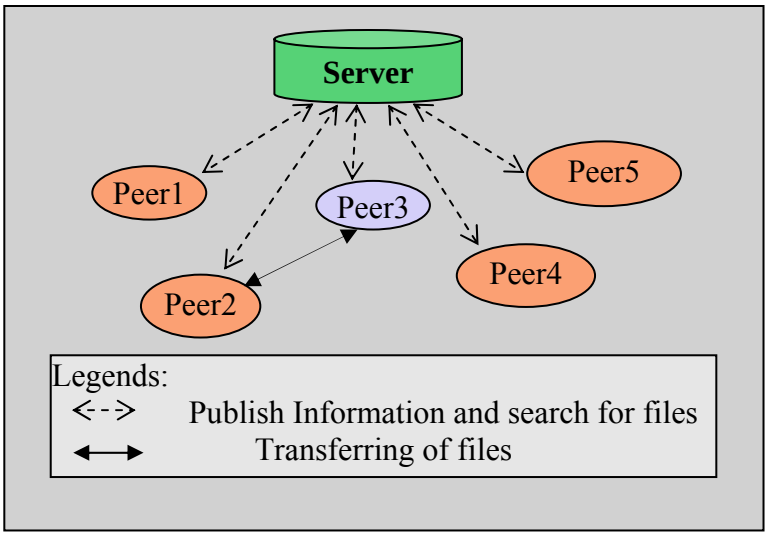

Figure 2.1: Napster's centralized model

Despite their excellent performance all centralized systems are plagued by the "single point of failure" problem. One type of such problem is a bottleneck at the server. Since the actual file transfer does not involve the server, a bottleneck happens only when there are too many peers querying the system and the server is looking up its directory. The second type of "central point of failure" problem is that there is no anonymity in the system. The server holds all the peers' ids (IP address) 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. Therefore, it was possible to sue Napster for providing the service through the server; as a result of the lawsuit, Napster was forced to shut down in 2001.

2.1.2 Fully Decentralized P2P System: Gnutella

Gnutella (version 0.4) (Gnutella, 2001) is a protocol for completely decentralized P2P systems. Most of the file-sharing systems adopting Gnutella, e.g. Limewire and KaZaa, are also music sharing systems. Gnutella uses a broadcast protocol to search for files in the system. A peer has to know at least one other Gnutella peer to send requests to, there is no central server or peer that can provide an index with peer addresses. When a peer enters the network, it contacts a designated peer (the addresses of some designated Gnutella peers are published on a website) and receives a list of other peers that have

recently entered the network. It is important to note that a designated peer does not maintain an index of all peers and / or their files, but only a list of some recently activated peers within a certain time-framework. A certain number of these peers (usually 7) become the neighbourhood of the new peer entering the network. 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. 2.2, Query peer P1, in the top, initiates the query and sends it to P2, P3, P4 and P6. P2 forwards the query to P7, P8 and P9. And so on.

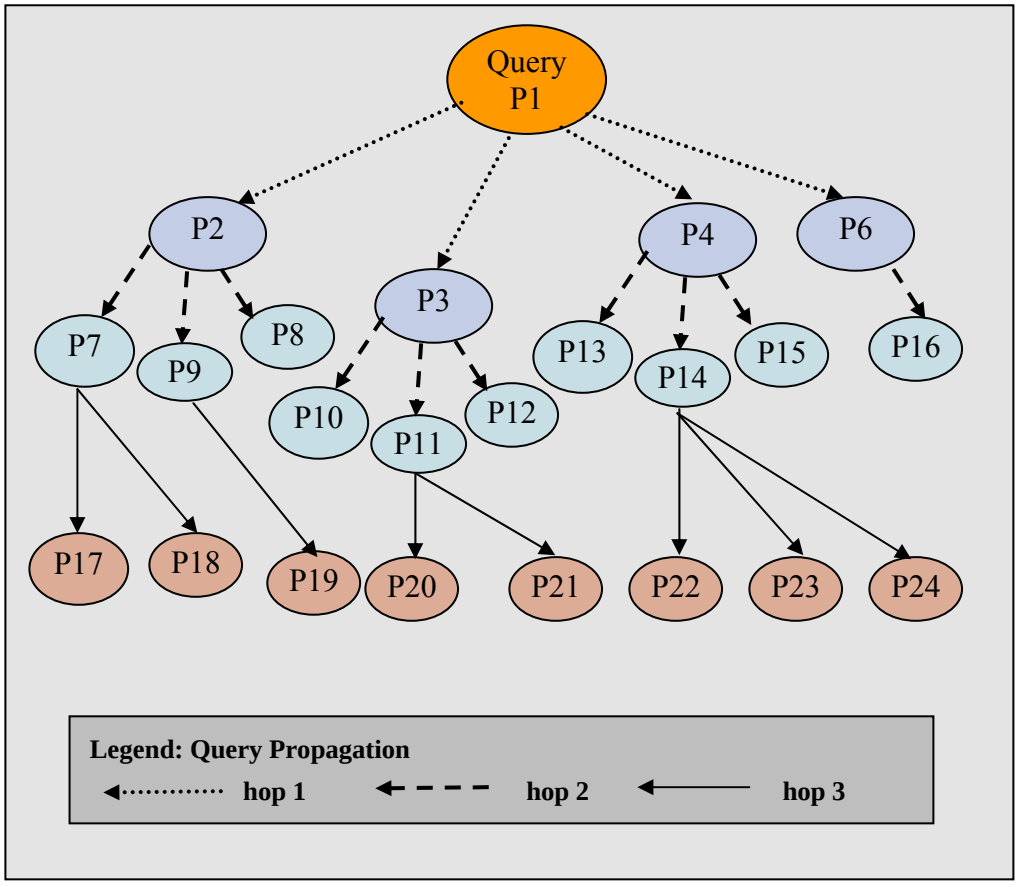

Figure 2.2: Gnutella's decentralized model

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 that was traveled by the query. This ensures anonymity since no single peer knows who requested and who responded to a particular query. At any time only queries and replies are circulating in the system.

Gnutella is a robust protocol since the failure of any peer in the network does not affect the system. However, Gnutella leads to a lot of traffic. As a consequence the responses are slow and not guaranteed, which leads sometimes to 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 Semi-Centralized P2P: KaZaA

KaZaA (KaZaA, 2002) is a Gnutella-based file-sharing application, which is currently one of the most popular file-swapping P2P applications. KaZaA uses a newer version of Gnutella using the notion of “Super” Peers, known as SuperNodes (Yang et al., 2003). Peers self-define as SuperNodes and maintain an index of the files shared by the peers connected to them. When a peer connects to a SuperNode, it sends a list of all the files it shares. Querying peers send requests for files to the SuperNodes. The SuperNode checks its index for the file requested and sends the addresses of peers found to be sharing that file. If no match is found it forwards the request to other SuperNodes in the system. Thus traffic and latency is reduced. But there is still the problem of bottlenecks and servers failures. Also SuperNodes compromise to a certain extent the anonymity of the Gnutella protocol, since a malicious peer may volunteer as a SuperNode so that it can retrieve the IP address and shared files of the peers that are connected to it.

2.1.4 Document Routing: FreeNet

FreeNet (FreeNet, 2001) is an example of a document routing system. 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. In this way the peers do not know what files they are storing and no peer knows where exactly their shared files are stored; therefore this system ensures full anonymity. 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 can 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 sent 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 file 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 files not frequently requested will be dropped from the system; therefore if all peers aren't ready to contribute a lot of storage space, the system will have to reduce the diversity of files and users have no control and are not informed about what is being deleted. Also if a peer is not online then the files stored in the disk space of that peer disappear, i.e., they are not available for search by other peers. Another constraint is that in order to search,

the user has to know the exact file name or the keywords that were used when the file-id was generated.

Chord (Stoica et al., 2001), CAN (Ratnasamy et al., 2001) and Pastry (Rowstron et al., 2001) are some other research systems implemented for document routing (Aberer et al., 2002). These systems differ only in the way documents are inserted and in the information peers maintain in the system so that search can be done efficiently. Pastry is one of the more popular systems, and it is presented below.

Pastry (Rowstron et al., 2001) is an object routing and location overlay network. In this network each node maintains a 'routing table', 'neighbourhood set' and 'leaf set' that are used to store the addresses of neighbouring nodes. Each node in the Pastry network is assigned a unique 128-bit random node-Id. A 128-bit key accompanies each message. This key is used to route the messages through the network. When a query arrives at a node, that node sends the message to the node whose node-Id seems closer to the key. Thus files are inserted at the node with the closest node-Id and also replicated in the nodes with closer node-Id. Similarly files are also retrieved by routing messages to the node with the closest node-Id as of the key. Thus if a node fails, the files are not lost unless all the neighboring nodes, i.e., nodes with closer node-Id, fail simultaneously. Thus the system is more robust in comparison with FreeNet. However, as in FreeNet, to locate a file the exact key must be known and an arbitrary string is not sufficient.

2.1.5 Summary

P2P systems like Napster, Gnutella and FreeNet have been successful but there are some common problems with all three systems. Napster provides good speed for searching and retrieving music files but it is based on a centralized model and peers could be tracked easily. Due to the centralization approach Napster eventually got shut down. 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 a routing-based algorithm, as in FreeNet, imposes limitations on users; they have to know the exact file-ids of the files they are searching

for. Exact keywords (Tang et al., 2002) or identifiers' (FreeNet, 2001) are required to search in such systems.

Gnutella is a dynamic and self-configuring system (Jovanovic, 2001). Of all discussed architectures, standard Gnutella imposes the least number of constraints. For example, the peers that make up the system can differ in their operating systems, disk space for shared files, memory allocated to the servent and in their connection bandwidth. Standard Gnutella does not impose any specific topology for the network, for example star, tree or graph topology. In contrast, document routing models, have a hierarchical topology and need a uniform hashing function for both queries and documents. Also, Gnutella requires no special information, like the naming keys or identifiers, to be assigned to files or used for search by peers. Usually the users do not have complete knowledge of the document or file they are searching for and hence the methods described in CAN, Chord etc. are not suitable for searching new documents, while in contrast Gnutella allows for searching unknown files with names that match only partially the user query.

For all the above mentioned reasons, standard Gnutella was chosen for this thesis as a powerful and least restrictive P2P file-sharing architecture. The rest of the thesis focuses on improving the performance of Gnutella.

2.2 Optimizing the Performance of Gnutella Networks

A brief description of the protocol messages used in the Gnutella system is listed below (Gnutella, 2003).

1) Ping: This message is used to actively discover nodes on the network. A node that receives this message generally responds with a Pong message.

2) Pong: This message is sent in response to a Ping message from a node. The message generally includes the address and the port number of the node.

Additionally, information about the amount of data it shares is also sent.

3) Query: This is the main message used for searching the network. Generally, a node that receives this message responds with a QueryHit message if a match is found in its local resources.

4) QueryHit: This message is sent in response to Query message if a match is found in its local resources. It sends information on how to obtain the requested file.

5) Push: This mechanism allows a node behind a firewall to contribute file-based data to the network.

2.2.1 Messages Contributing to Traffic in Gnutella

A study (Vaucher et al., 2002) showed that traffic in the Gnutella system is mainly due to messages for initial connections and queries. From the table below it can be seen that the pong messages and query messages were the majority of messages circulating in the Gnutella system.

Table 2.1. Message Type Distribution (Vaucher et al., 2002)

	Experiment C (Nov. 28th, 2001)	*Experiment E (Jan. 4th, 2002)*	*Experiment F (Dec. 18th, 2001)*
Messages received	36,797,372	278,871	5,965,273
Pings	9%	11%	9%
Pongs	57%	26%	72%
Queries	30%	57%	15%
Replies	4%	5%	4%

Ripeanu (Ripeanu, 2001) reported that, after some changes to the Gnutella system, the traffic generated in Gnutella now consists of 92% Query messages, 8% Ping messages and the rest other messages.

Markatos (2002) in another study reports data from three clients, in three different locations, on the average number of queries seen. Average query requests messages measured per second by clients in Rochester, Crete and Norway are 45.9, 47.9 and 52.3 respectively. And the average query response messages measured per second are 32.2, 44.2 and 26.9 respectively. They report observing bursty traffic in both cases.

From the above two studies we note that queries still make up a significant proportion of messages. It was found nodes with more connectivity, in a high traffic network, get overloaded with queries and responses are delayed. Thus reducing query messages would help in reducing traffic and not overload high-connectivity peers. Studies have also found that Gnutella systems suffer from bandwidth, congestion and latency problems (Jovanovic, 2001) (Markatos, 2002) (Portmann et al., 2001) (Saroiu et al., 2002).

2.2.2 Improving Search in P2P Systems with Simple Approaches

When the TTL value was reduced, a "short-circuiting" problem was reported (Jovanovic, 2001). The number of messages decreased but it was found that queries could not get responses since the request did not go far enough in the system. So decreasing the value of TTL by itself is certainly not the way to reduce messages and limit the flooding problem.

The following approaches limit the broadcast space of the standard Gnutella. One approach (Kalogeraki et al. 2002) broadcasts queries to only a subset of its neighbouring peers and not all of its neighbours in order to reduce the number of messages circulating in the system. The goal of finding the document may be achieved but the chance is smaller than with standard Gnutella, since the generated search space is smaller. The

number of messages is reduced as compared to the standard Gnutella since the number of peers querying are lower. The peers are not learning from their experiences.

(Lv et al. 2002) introduces two approaches. The first approach is an "expanding ring" where the TTL value is increased gradually to find the resource. Initially the peer assigns a small TTL value and starts the search. If no result is found then the peer restarts the search with an increased TTL value. But the messages go through the same peers once again. There is duplication of messages to the same peers and peers do not learn from past experience to bypass previously forwarded peers. This approach still floods the network with messages. The other approach is a "random walk" where the peer sends a query to only one peer at a time at each hop and sequentially searches through the system to get the results. Once the results are found the query terminates. Thus, fewer messages are generated but the search time increases, as the search is sequential rather than parallel. To limit the delay, the authors experimented with different numbers of peers receiving the initial messages. The messages then traverse similarly thereafter but each time the peer checks with the querying peer to see if the result has been found and then the message is forwarded. There is still time delay, as each peer that gets the request has to check if the request has already been fulfilled. Here too the peers do not learn about resourceful peers and subsequent searches are done again in the same sequential way.

A super-peer network as discussed in section 2.1.3 (Yang et al., 2003) improves significantly the performance of standard Gnutella. A super-peer network is a hybrid (semi-centralized) system containing few servers and many peers. Traffic and latency is reduced, as broadcasting is limited to super-peers only. But there is still the problem of bottleneck and failures when the super-peers fail.

2.2.3 Improving Search in P2P Systems with Semantic Routing Approaches

This section lists several approaches to optimizing the search for files based on the semantics of the queries. Different authors and papers use different terms for the semantics of a query. The terms are as listed: query type (Wang et al., 2002), search

criterion (Ramanathan et al., 2001), a category (Crespo et al., 2002.), semantic category (Triantafillou et al., 2002) and category of data (Ng et al., 2002). Henceforth all the terms are referred as *semantic category* in this section.

One approach called "Directed Breadth First Search" (DBFS) (Yang et al. 2002) selectively forwards queries to peers who are seen to have returned good results for previous queries. For this kind of intelligent routing, the peer has to keep information on the neighbouring peers about how many hops that peer had to forward the query to obtain results, the number of results returned by that peer, and the length of the message queue of that peer. All this information has to be stored and processed by a peer when it wants to request a file. Since the neighbouring peers do not change, the query message travels the entire path until it finds the peer that has the document. This approach still generates a lot of messages. The authors do not explain how a peer will know what constitutes a good result. If the number of results is taken to be an indication of good results, then a peer that returns irrelevant but many results would be considered a good candidate.

In the "Buddy Web" approach (Wang et al., 2002), routing is done based on similarity of interest. The main reason for building a "Buddy Web" 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 query and has already received results for it. By caching previous search results and searching first through them, the university can reduce cost and utilize the network better. The similarity between the user's interests and the files' semantics that is used for routing inside the network is calculated in several ways. One way is based on the meta-data of the file, e.g. the title tags downloaded by the user. Another way is by storing words that the user selected while browsing through its contents in a vector. By summing up points for the words, which have already been assigned some weight by some weighting scheme, a calculation expressing the value of interest is performed. A few servers maintain the interest value along with the peer's address. A peer calculates similarity with other peers in the network by comparing its own interest value to those

of the others. When a peer is querying, the query is first sent to the underlying network, called BestPeer. The system sends the query to those peers whose interest values are close to the querying peer. Once results are found, they are sent to the querying peer. If the system does not find any results then it sends the request outside the network. An external search takes place in the traditional manner. A peer does not learn if the peers who have been found to be similar are useful or not. The authors do not mention how long a cache is stored. If a peer is searching vigorously, it may overwrite the cache with new documents and older documents will not be found.

BestPeer (Ng, et al., 2002), mentioned above, is a P2P network prototype using the Buddy Web approach, implemented in a university setting. The main aim of this network is code-shipping and data-shipping. It is a 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. The peers that are deemed beneficial are kept in a list. A beneficial peer is defined as a peer that answered a lot of queries or a peer providing the most amount of files. Also distant peers (i.e., when the number of hops to reach a peer is large), who are found beneficial, are kept in the list. Each peer has its own limit of the number of peers that it can connect directly to. The authors found that the response time in this system is shorter than in the Gnutella system. However, since beneficial peers were determined based on how many queries they responded to or files downloaded from it, it doesn't mean that the files were of good quality or were found relevant by the querying peer. If a peer changes its query type, the best peer list keeps changing as new peers deemed beneficial are added and the older beneficial peers, of other query type(s), are lost. The authors experimented on star, tree and line topologies in the network, and their results are based on these topologies.

To provide better service in P2P systems, several research projects investigated how the traffic generated due to the broadcast protocol in a Gnutella system could be minimized while still preserving the quality of search results. One such approach is based on the idea of learning who are the "good peers" in a P2P system (Ramanathan et al., 2001).

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. Peers are deemed as "good peers" of peer *C* when these peers have responded the most to queries of peer *C*, regardless of the semantics of the query. The peer sends subsequent requests to the peers in its list. The authors found that this reduces traffic in the network as peers now send queries selectively to other peers. Also this selective dispatching of queries does not affect the search results at the end and good responses are obtained in a short interval. This works when the user is consistently searching in a single type of semantic query. However, users search often for more than one semantic category at a time (Iamnitchi et al., 2002). So when the user changes its query semantics, then these "good peers" will not be able to provide the responses. The peer will find new "good peers" for this type of query again, but the peers of the old query type will be dropped.

Some research explores the interest-based behaviour of peers. (Sripanidkulchai et al., 2003**)** proposed a solution where peers loosely organize themselves into an interest-based structure. Initially, the algorithm for querying is flooding or broadcast as in Gnutella. When replies are returned, the peer chooses a peer randomly from all the peers who replied and adds it to the "shortcut list". Subsequent queries are sent to the peers in the "shortcut list". If there are no responses obtained by using this technique then the peer uses the flooding algorithm again. Each peer assigns space for storage of this list. The results show that the "shortcut list" is effective in finding both popular and unpopular content (files), and that 45-90% of files are found quickly. But a drawback to this approach is that it will create a lot of traffic and time delay as the search may keep reverting back to the flooding algorithm when the document is not found through that single peer chosen from the subset of peers responding to a similar query. The shortcut list contains peers who have returned response regardless of the semantic category of queries. A modification to the model that the authors suggest is to add to the list all the peers who have replied to a query. But the benefit of using a short-cut list may be lost as peers just keep getting added and deleted when the peer changes its interest for searching since there is only a limited space in the list.

Another approach (Kalogeraki et al., 2002) proposes that the peer maintains a profile of all other peers who answered its requests in a local repository by keeping a list consisting of (query, peer) pairs. A query is defined by the set of keywords used for the search. All peers that respond to a query are added, together with the query, to the pair-list. When the peer initiates a new query q, it matches it with the previous ones stored in its pair-list and picks out the peers that have answered similar queries. This is achieved in the following way: In this model the neighbouring peers connected to the peer remain the same from the start and only the profile is updated. Each query in the list is assigned a similarity value relative to the present query q. It then adds up all the similarity values for a peer and forwards the requests to only the subset of peers from its neighbourhood, that are found to have the greatest value for similarity with respect to the present query. A random peer, a peer not found to have answered similar queries, is also chosen for forwarding to explore the network. The authors conclude that the traffic is reduced. The amount of storage is limited and a node uses a Least Recently Used (LRU) policy to decide which (query, peer) will remain in the local repository. Thus if a peer is searching consistently in a query type then, older pairs of a different query type will be deleted from the repository. If now the peer wants to search in a different query type, the peer has lost the best peers for that query.

Some authors (Crespo et al., 2002) suggest creating groupings of peers, which advertise together the resources available to each member of the group. The idea is similar to creating super-peers like in KaZaA, but the index of files is not maintained by a super-peer but by each member of the group. Peers use indices to keep track of the number of documents that can be accessed through each of their connected peers in a semantic category and communicate to neighbours the total number of documents that they can access in a category. Thus each peer has information about which neighbouring peer to choose when a query arrives, based on the total number of resources accessible through the peer.

The approaches described above explore the semantics of queries and implicitly model the interests of peers in different semantic areas. These approaches all selectively forward queries to peers based on the knowledge obtained and stored. Some research studies have used semantic routing of queries by classifying queries based on keywords.

Neurogrid (Joseph, 2002) uses keywords, derived from the file meta-data, for searching and then matching a query. Peers are associated with keywords based on the contents they store. When a peer issues a query, it looks for remote peers characterised by keywords similar to the query and forwards the request to those peers. The peer also changes its neighbourhood by adding remote peers that have answered its queries. Thus peers learn about other peers, along with all the queries they have answered, and route queries according to this knowledge. However, storing all this information is expensive since the storage is bounded and there is a limit to how much information each node can maintain.

In another approach (Triantafillou et al., 2002) documents are accompanied by keywords, which semantically represent the contents of the documents, and these keywords are classified according to semantic categories. The system is assumed to be logically organized so that clusters of peers are formed based on semantic categories. However, the researchers do not specify how this is achieved. Each category could have one or more clusters associated with it. Peers have knowledge about all the categories and the entire associated set or subset of peers in the system. When a search request comes in, a peer forwards queries to the cluster associated with the semantic category of the search keyword. If the document is not found, then the peer in the cluster sends the queries to other peers in the same cluster or another cluster associated with the same category. In this model each peer in the system has to know all the categories existing in the system and clusters associated with them, which is hard to achieve without a central server maintaining a list of categories and clusters and peers.

Semantic routing has also been used by (Ng et al., 2002) in a P2P system for sharing images. The authors propose a search strategy based on clusters of peers with similar

properties. Each peer according to its interests calculates its signature value using a certain function. When a peer joins a network it learns about a few peers, called "random links", through the Gnutella ping-pong messages and adds them to its list representing a peer-cluster. Queries by the peer are forwarded through these "random links". A peer stores another peer in its peer-cluster list if the data they share are similar; such peers are called "attractive links". Peers thus get clustered according to similar signature value. Thus, when a query is sent by a peer it gets checked against its signature value and if no close match is found, the query is forwarded randomly to peers. Once the query reaches the appropriate peer in a cluster, the query is broadcast using the peer's "attractive link" lists inside the cluster.

2.3 Social Networks

Concepts of social networks can be applied in conjunction with approaches based on learning lists of peers, who have been useful in the past and seem to have interest in a given area, for optimizing search in P2P systems. Social networks are groups of people, be it in a social setting or an organization connected by relationships (Wellman, 1997). According to (Newman, 2000), Stanley Milgram, in the late 1960's, did one of the first experiments to investigate social networks. In his experiment (Milgram, 1967), he addressed letters to a particular stockbroker in New York and gave them to people randomly picked at various locations in the United States, far away from the final receiver. They had to send the letter so that it reached the addressee, but under the condition that one could post the letter only to people one 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 was postulated. This characteristic, that nodes or people are connected to each other by only a few links is known as the "small world networks" (Scharnhorst, 2003). These social network approaches are not focussed on P2P networks.

Social science has shown that social networks benefit people in their everyday lives. Social networks 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 the web (Kautz et al., 1997). The system was built to get expert advice in some fields. Users who registered in the system had to fill out a form about their publications. The system would also search and find information about co-authors in these publications. In this way the system gained knowledge about social networks based on co-authorship of 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. These social ties form the basis for information retrieval depending upon the type of relation existing between the two people.

Definition 2.3.1 *Social Relationships*

Social relationships are defined as the interactions among a set or group of people. They are characterized by how frequent the interactions happen and what type of interactions occur (Waloszek, 2002).

Definition 2.3.2 *Strong Ties*

Strong ties are relationships that connect people within the same community and involve frequent interactions. Due to the frequent interaction, people involved in strong ties usually share the same body of knowledge and can find information quickly through these strong ties. Waloszek (Waloszek, 2002) points out that strong ties let people access information in a timely manner and information is given freely since these are people who interact with each other very often. This behaviour can be used for cooperating in the network. Other studies (White et al., 2003) on strength of ties also found that information can be found quickly through strong ties as these ties have been found to be connected to knowledgeable people in the community.

Definition 2.3.3 *Weak Ties*

Weak ties are relationships among people that are not in the same community or coalitions. They are characterized with relatively infrequent interactions.

People with a large number of weak ties are like "Super-peers". While they don't interact frequently with any individual peer, these peers provide referrals and expertise in the form of forwarding requests to other peers capable of responding correctly (Venkataraman et al., 2000). Weak ties give access to new information. These are generally from remote sources.

The benefit of knowing people with weak ties is that they provide information about experts or knowledgeable people that are not in the community (Granovetter, 1973). For example, to get information about real estate the first step would be to find a person who has knowledge in that area or someone who knows about a person in that area, so that the query reaches that community.

Social networks are also used to study the interaction patterns between groups of people in a certain context. They help in understanding to what degree the behaviour of an individual is influenced by constraints in their environment and how individuals use their social network for their benefit (Webster et al., 2001). It has been found that, if given a possibility, people tend to manipulate circumstances so that they benefit in socializing with their choice of people (Newcomb et al., 1965). A social network, once established, can be used for ones' own benefit.

Social networks can be studied through the evolution of societies of artificial agents simulating real people. Simulations allow studying patterns, diversity and behavioural changes in groups of people due to changes in the environment (Pearson et al., 2001). Social networks can be visualized using graphs with nodes representing people and arcs of different colours and thickness representing relationships with different strengths (Freeman, 2000).

Studies of the qualities of social networks that allow faster transfer of information show that social networks with varying tie strength are better than random networks, as random networks have been found to lack the ability of finding a target or specific person quickly (Waloszek, 2002). Thus interactions occurring between peers can be

exploited to find social networks or communities and use them to retrieve information in the form of data or documents. Since each individual maintains her own personal network that may contain different people specialized in the same field, this distributes the load of dissipating information by these specialized people, as the same person may not be contacted for every bit of information. Strong ties also indicate close relationship and so it is easier to get close to the knowledge base once a person reaches a group with strong ties, and this can be either through a weak tie or a strong tie. Weak ties are credited with diffusion of knowledge and ideas (Granovetter, 1983).

2.4 Summary

Standard Gnutella-based P2P systems suffer from scalability problems like high traffic and poor quality of service due to the flooding algorithm. Various approaches like "random walk" and forwarding to only a subset of neighbouring peers try to limit message traffic. But in those approaches the peers do not learn from their experiences. In other approaches peers use their experiences to learn about peers to route queries taking into consideration their similarity of interest. A drawback of these is that keywords do not always adequately reflect the semantics of the data, and the quality of results returned, is not used to learn if a peer is really similar or not, but only the number of returned results is used as a measure of a peer's similarity.

Other approaches use keywords to classify documents to categories and then classify queries into categories based on words used. One or more clusters of peers are formed, which belong to a semantic category, so that the peers in one cluster would forward queries to peers of another cluster in that same semantic category. Peers store, compute and maintain all the information of at least a few peers from each and every cluster of every category so that these systems succeed. Typically the amount of information that must be stored and the computation power needed to update information of the cluster and the categories is a drawback of these approaches when peers have limited resources.

Social networks or personal networks also group users with similar interests. It has been found that these social networks help people in finding information quickly. Thus by

maintaining contacts of peer(s) in one's areas of interest, information can be found as these peers would also maintain contacts with other peers in that area of interest. They form clusters of similar peers and information can be found quicker. The relationships inside a cluster, known as strong ties, are important to find information within the cluster and obtain expert advice or help. On the other hand, weak ties help in finding information outside the cluster, which could be other similar clusters in the network or newly formed cluster in the network. Also weak ties help in finding information outside the area of interest by linking peers to other clusters or peers who have expertise in that area of interest.

We combine the concept of social networks and semantic routing to model a system that reduces traffic and obtains good quality of service in the Gnutella system. The following Chapter, Chapter 3, introduces our model and describes it in detail.

CHAPTER 3

Semantic-Social Routing Approach

Problems Addressed

This study focuses on systems based on Gnutella 0.4 protocol. As already noted in Chapter 2, Gnutella–based systems suffer due to the flooding algorithm (Jovanovic, 2001). However, other scalability problems have been observed like latency, bandwidth and congestion in the system (Jovanovic, 2001) (Markatos, 2002) (Sripanidkulchai, 2001) (Ramanathan et al., 2001). Ongoing research is trying to reduce traffic, relieve congestion and improve the search capability of these systems.

(Jovanovic, 2001) mentions that Gnutella systems exhibit "small-world network" phenomena (ref. Pg. 19). This finding can be exploited to improve search, but it has not been taken into consideration yet. On the other hand, semantic-based routing helps find queries faster and without overloading the system with messages (Tang et al., 2003). The model discussed in this thesis combines semantic routing with the concept of social networks. Messages are routed to different contacts depending on the semantics of the query. Each peer learns from experiences, its social networks in different semantic areas.

3.1 Defining Semantics of Queries

There are different ways to define the semantics of a query. It can be expressed by the keywords used in the query. Similarity between the keywords of the present query and previous queries can be used to intelligently route the queries to the peers in the system. The most successful search systems are based on keywords. However, in order to identify the semantically meaningful words in a document, more complicated techniques

are needed like Latent Semantic Analysis (LSA). In our model "semantics" is defined by a finite set of categories.

Definition 3.1.1: *Category of interest*
A category is defined as a semantic area characterized by a set of topics or keywords (Vassileva, 2002). For example, topics like distributed databases, and peer-to-peer systems characterize the area of distributed systems, which is a category in our model.

One can define hierarchical links among categories, resulting in category trees, or subject classification schemes, similar to those used by the ACM or IEEE.

If files are assumed to have associated meta-data containing keywords (Joseph, 2002), they can be categorized dynamically using the keywords from the meta-data of the document. Since a document may contain keywords from different categories, quantitative measures define how exactly a given document will be clustered.

Definition 3.1.2: *Peer's Interests*
A Peer's interests are defined as a list of categories in which the peer initiates queries or shares files.

Users can be interested in one or more areas (corresponding to a set of categories). It is possible to infer that a user may have interest in a more general category (according to the category abstraction hierarchy) if s/he has shown interest in a specific sub-category. However, when users search for documents in a given category or sub-category, they are usually interested in narrowing down their search rather than generalizing it.

Definition 3.1.3: *Friends list*
The "friends list" of a peer contains the names/numbers of other peers that are assumed by the peer to be interested in a given category.

The semantic categorizing of "friend"- peers creates a social network for a particular area of interest which allows peers to benefit from previous experience by using their relationship to other peers in the search for resources. Peers, which have shown interest in a particular category by sharing a file in this category are likely to have and share other files in this category, so future queries sent to these peers are more likely to yield results than those sent to peers who haven't shown interest in the category. Peers who are in the friends list of a particular peer for a given category share files that have matched the queries of the peer in the past, have proven useful and probably share similar tastes or criteria for considering a paper interesting. Therefore the friends list helps the peer bias its search according to its interests and tastes.

3.2 Learning the interests of other peers

There are two kinds of evidence about being interested in a given category: initiating queries in this category and sharing files in this category. Peers can learn about another peer's interests by keeping track of all the peers who responded to a query in a given category. The response obtained is taken as an indication of that peer's interest because the peer is keeping and sharing documents pertaining to the category being queried. A peer can also learn about the interests of other peers by analyzing who initiated a query in a category. Thus peers can be classified according to their interests and the queries can be forwarded to peers in the same interest groups. A peer maintains separate lists of friends for all of its interest categories and adds peers to the lists based on evidence about their interests. When a peer queries and gets responses, it keeps track of all those peers who returned the responses; they are assumed to be interested in that category. For example, if peers, *A*, *B* and *C* responded to peer *G's* query in category *X*, then *G* adds *A*, *B* and *C* to its friends list related to category *X*. On the other hand, all the responding peers *A*, *B*, *C* will know that *G* is interested in category *X*, since it generated a query in this category, so they will add *G* to their lists of friends in category *X*.

This idea of keeping lists of friends for different categories is similar to social networks. People maintain social contacts with individuals in different interest groups (McCarty, 2002). The reason behind the need to maintain these contacts is that these people are

good candidates for recommendation or advice in the interest area, if one ever needs them. A person in a "friends" group about a given subject has more knowledge about that subject than a randomly chosen person. He/she also knows other people with similar interests, so if it can't answer, it will be able to give a good referral. As seen in (Kautz et al., 1997) people who were co-authors in a paper were seen as good referrals for advice or contacts. Studies have shown that social networks are beneficial in contacting and getting help (Granovetter, 1973).

In a similar manner, when we are seeking information we tend to first seek out somebody who we know either has knowledge in the area or knows somebody who has knowledge. This mechanism can be easily implemented in the Gnutella protocol.

For example, Peer *A* in Fig. 3.1 has created different friends lists for different categories: *X*, *Y*, *Z*. When a query is initiated in a particular category (say *X*), peer *A* chooses to forward the query to the first few (three in this case) peers with highest relationship strength.

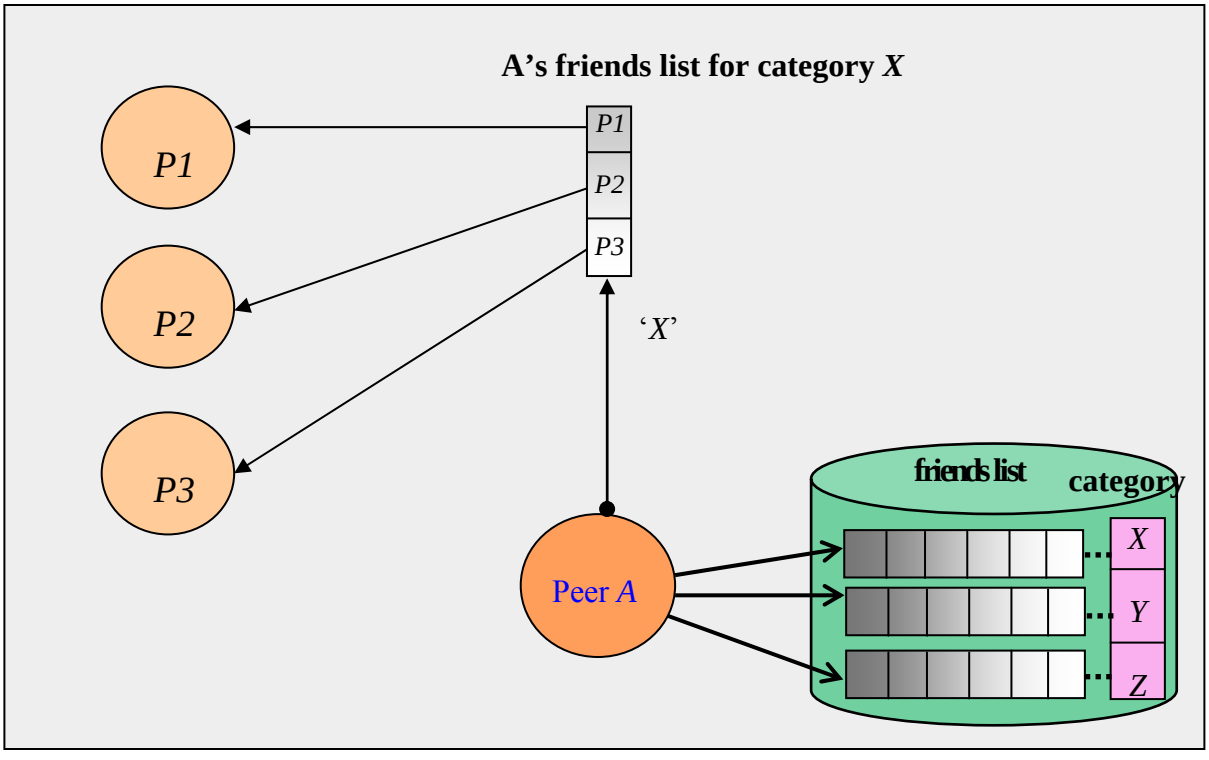

Figure 3.1: Creating friends list for a category.

If any of the "friend-peers" does not generate the response, then the friend-peers will use its own friends list for the category of the query to propagate it further. Thus a combined semantic-social network routing generates a chain of forwarding, eventually finding the answer to the query with a higher likelihood. Figure 3.2 shows the forwarding to friends list where peer *A* generates the query.

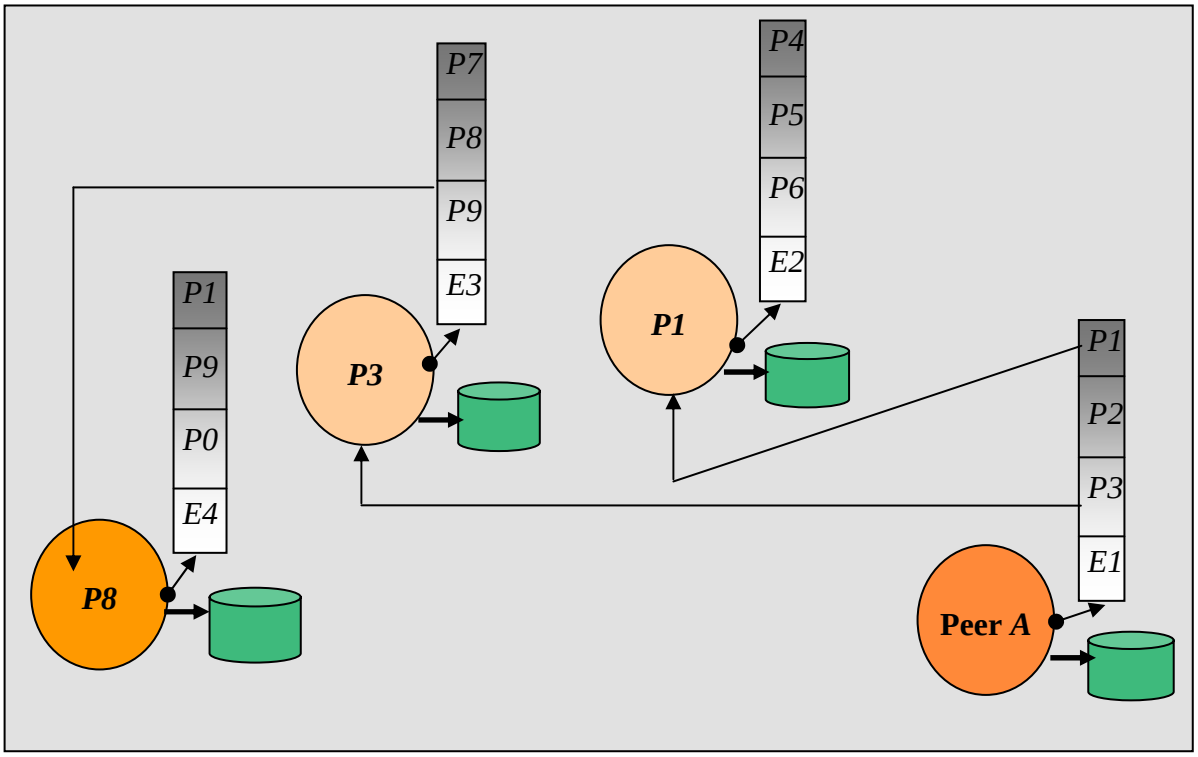

Figure 3.2: Query Routing

If the query is sent to random peers who have no interest in that category the query will most likely not succeed. The peer will forward it to other peer(s), which most likely won't have interest in the category. If they don't have the file, they will have to forward it further. The query may get aborted due to exhausting its time to live. Also, replies will be delayed as a result of this forwarding.

However, if a peer has the file, it will generate a response and it won't forward the query further, so there will be no more traffic generated further from this peer in the network. Since the likelihood of a peer on a semantic routing chain to have the queried file is higher (since the peer has interest in the category of the query and is more likely to keep files in this category) responses will come with a higher likelihood, faster and will generate less traffic. Thus, the benefit of this "semantic-social" routing approach is that queries will travel less and will be more likely to achieve success with a smaller number of hops.

Even though in this way, a peer will only send a query to other peers it knows, it will never "discover" new friends. Thus, if none of the peers in the friends lists have the file, then results are not obtained. Therefore, there is need for at least one "wildcard" or random peer, to allow for discovery in the system. This peer could be any one of the peers from other categories friends lists or an unknown peer.

3.3 Centralized versus Distributed Classification of Peers

3.3.1 Performance and Costs

Should we model a social network or a yellow page directory? Instead of having each peer maintain its own friends list in each category it is interested in, we can have a global central directory listing all peers and their interests. Thus, the central server will be like a yellow page directory and have knowledge about all the interests of all peers and this makes it easier for the peers in the system to retrieve information about peers interested in a category. Such a solution has some advantages. To search for a document the peer has to contact the server and get the list of peers interested in that category. Afterwards, the peer can retrieve from the proxy server a list with those peers who are currently on-line. Then it can query exclusively those peers that are strongly interested in the category and are currently on-line. The server will update its lists using input from many peers who have discovered peers interested in an area and it will be more reliable (in terms of strength of evidence about the interests of peers). There are also not that many messages circulating in the system and thus the system is not congested. A centralized approach like this is similar to the Napster architecture. But the drawback of

such a centralized approach is that the entire system will stop working if the central server fails because peers will be unable to get a list of peers from the server. Another issue with using a central server is that it becomes the bottleneck in the system; responses are delayed when an increased number of peers contacts the server for lists of peers each time they are searching / forwarding a query for a document. A centralized server needs some maintenance as well to ensure that it is always available, which means monetary costs.

3.3.2 Privacy, Anonymity, Vulnerability

In the case of Napster (Napster, 2001), the centralized directory was the main reason for its shutdown since it was easy for the server to be tracked down and charged with copyright violation. It also gives malicious peers ways to bring down the system by denial of service attacks.

A central server also stores information about the peer, its interest and the contact IP address. Thus it becomes easy for any peer in the system to get an IP address and use it for any malicious purpose, like spamming. Privacy cannot be maintained since the central server interest directory is open for any peer to look up any other peer. Peers in general may not be interested in revealing publicly their areas of interest.

In our model every peer maintains its own friends lists. If each peer maintains lists locally then there will be no need for a central server to keep information about other peers in the system for forwarding the query. The redundancy ensured in this way helps the system to keep running in the event of any peer's failure. Since each peer is responsible for maintaining its list for the interest categories, its failure does not affect other peers in their capability. Also when each peer maintains its own list of peers for a category based on its own interactions, that peer keeps private the identity of all those peers who interacted with it and no peer in the system will be able to keep track of all other peers' interactions. Hence the interaction is private with no third party involved. However, complete anonymity, a desirable feature of Peer-to-Peer systems, is not maintained in this system. The originator of the query in our model knows about the

responder of the query at the time of document download. A malicious peer can pretend to be interested in a given category and quickly find a list of peers interested in this category and the sources of the documents. However it will not be a complete list, but only a local one based on the experience of the peer. So the social network approach is less vulnerable than a centralized approach.

3.3.3 Personalization / Subjectivity

Another advantage of the "social network" approach is that it allows for subjectivity. Each peer can choose the criteria according to which it adds or removes peers from its friends list. When a peer maintains his/her friends lists individually, it can choose to add or delete any peer from its list. The lists of friends reflect similarity of tastes in a category between the peers. For example, a number of peers may be interested in a particular category, but a peer may be interested in some specific topics in that category; therefore, its friends lists will contain peers with interest in these topics only, while another peer may be interested in another set of topics and will have a different list of friends. Just like humans have their own criteria for quality and choose things accordingly, each peer also has a certain threshold by which it judges the quality of documents or the interestingness of resources. These are subjective criteria and are hard to measure and sum up objectively; imposing any "objective" measure of similarity is meaningless. In a centralized approach each peer sends the number of documents it shares in each category to the central server when the peer joins the system. The server keeps a list of all the peers it communicated with along with the number of documents it shares. In this centralized approach, the usefulness of a peer would have to be an objective measure, for example, the number of documents shared by a peer in a category. Peers can also report their satisfaction with regards to any peer to the central server, allowing it to compute a "reputation" value for each peer, which can be looked up by other peers. The disadvantage of the centralized approach is that such "average" objective measure cannot capture the specific needs of all peers. A peer that is highly specialized won't score very high on an objective measure since only a few peers would use it. However, it may be invaluable for a specific peer who shares exactly these special interests. As the heterogeneity of the user population grows, the meaning of the

"reputation" value will decrease and become a subject of emergent power-law effects "the rich get richer" or "the popular become even more popular", effects that reduces the overall utility in the peer community. In a centralized approach, the list containing peers is complete. In our model, where each peer maintains its own list, the list is not complete as only those peers who responded in a category are kept. There is more work to be done and less information about peers in a decentralized approach.

In summary, considering the mentioned above factors, it seems that the approach where each peer maintains its own list is better suited than the centralized approach where all the peers and their interests are listed in a central server.

3.4 Strength of Relationships

Definition 3.4.1 *Strength of Relationship*

We define the strength of relationship S of peer A with peer B as a numeric value $S \in [0,1]$, computed by A, which represents the previous experience peer A has from interacting with peer B.

From this definition it follows that if peer A and peer B are involved in a relationship, each of them keeps a subjective value of the strength of the relationship, based on its own experience with the other peer. Depending on the criteria for judging experience, the values of the strengths they assign to the relationship can be different, which parallels the asymmetry of relationships that exists in the real world, e.g., 'A likes B, but B dislikes A'.

Peer A computes the strength of relationship with another peer B in the system based on evidence from interactions, e.g., the reliability and usefulness that the peer B showed in its interactions with A. Reliability in the context of P2P systems means being frequently online, having good resources and having a good connection. The strength of relationship is updated after interactions with the other peer. The evidence taken into consideration when updating the strength of relationship includes the success rate the peer had with queries sent to the other peer, the reliability of the other peer while

downloading documents, (e.g., does it stay active or disconnects when document is being downloaded), the quality and the usefulness of the resource.

A peer attaches a strength value to each relationship with a peer from its friends list for each category. The strength of the relationship with a peer from a friends list related to one category is independent of the strength of the relationship with the same peer if it happens to be in a list for a different category. If x is the strength that peer P assigns to its relationship with peer Q for category X and y is the strength that peer P assigns to the relationship with peer Q in another category Y, y is independent of x. For example peer P may have downloaded documents from Q in two different categories X and Y: many times in X and only a couple of times in Y. Therefore x will be higher than y.

3.5 Updating the Strength of Relationship

To create friends lists in different areas of interests and use these lists for query routing, peers need to learn about other peers' interests. New friend-peers need to be discovered in the interest categories of the peers. Adding new peers is important since these peers may have some other resources and may also be connected to other unknown peers interested in the same category; therefore discovering new friends will allow the peer to access more resources in the system. Thus maintaining a long list of friends for each category of interest is desirable, since in case of failure or inactivity of one of the friends, the peer will still be able to route its queries to the remaining friends.

A peer can learn the interests of peers in the system in a variety of ways. One approach is by asking its neighbours explicitly about their interests and storing them for future use. This is of limited use since a peer will know only about its immediate neighbours' interests and not that of other peers in the system. Also each peer has to be sure that the relayed information is true (i.e., it must trust the source).

Another approach is by using "gossiping" as proposed in (Yu et al., 2000). Peers gossip or send messages to each other about the interests of peers they have interacted with. For this technique to work, each peer has to trust other peers in the system, so that from the

information obtained about the interests of other peers in the system it can compile a list of peers that it can use for searching in that interest category. But peers have different interests and quality of service criteria and so all the peers in the system cannot be trusted equally. This requires extra knowledge about which peers are trustworthy. A method for managing and propagating trust values has been proposed by (Wang et al. 2003). If the trust value of peers is also to be gossiped about then each peer has to also decide which peer's "gossip" weighs more and the trust problem shifts to a higher-level.

Therefore, we propose that each peer should learn from its own experiences. The peer's own experiences are the most trusted source since each peer can judge by its own criteria how to update the strength of relationship with another peer after an interaction and which peers to keep in its friends lists. This technique does not require relying on other peers for information (recommendations). When peer *A* gets a response and downloads the document, the responding peer *B*, without giving explicit information about its interests, is seen as interested in the category of the query. *A* will update the strength of its relationship depending on its criteria for quality of service. If for some reason a peer is not seen to be a good peer (e.g., file is corrupted, transfer is interrupted etc.), the strength of the relationship will be decreased after an unsuccessful interaction, again depending on peer *A*'s own criteria (judgement) for successfulness of interaction. Some peers may have a higher tolerance while others may be more demanding.

Reinforcement learning is an appropriate technique for learning from experiences, and can be used to update the strength of the relationships and thus learn about the interests of other peers. The strength of a relationship can be calculated in two different ways, depending on whether there is a need to take into account how recent experience with the peer was (e.g., decay of old experiences). The first way can be used when it is assumed that the interest and cooperativeness of the peers does not change over time. The second way takes into consideration changes over time and allows the building of more explorative vs. conservative (or reverse) attitude of the peer that computes the strength.

The first way to calculate the strength of relationship computes the percentage of responses in a given category produced by a peer, i.e.,

$$S_{AX}^{C} = \frac{R_{AX}^{C}}{Q_{A}^{C}} \qquad (3.1)$$

where, S_{AX}^{C} is the relationship strength between peers A and X, R_{AX}^{C} is number of responses by peer X to queries in category C issued by peer A and Q_{A}^{C} is total number of queries issued in that category by peer A.

In this way all responses received from are treated equally and are taken into consideration in evaluating that peer's performance, i.e., old experiences count as much as new experiences. This is reasonable if peers are assumed to behave consistently in time. In reality, however, peers change their interests and behaviour, so it may be more reasonable to give more attention to recent experiences.

Another way of computing the strength of relationships allows taking into account changes in the interests of peers and gives different weights to past and recent experiences according to the following reinforcement learning formula

$$S_{AX}^{t} = \alpha * S_{AX}^{t-1} + (1- \alpha) * e_{AX}^{t} \qquad (3.2)$$

where, S_{AX}^{t} is the relationship strength value between *A* and *X* at the present time *t*, S_{AX}^{t-1} is the relationship strength value at the previous time *t-1* and e_{AX}^{t} is the evaluation of the current experience and α is a value between 0 and 1. The experience is evaluated as a function of the quality of the service and/or resource obtained, e.g., bad format/good quality, problems while obtaining the document, the speed of network connection at the

time of obtaining the document. Each peer has its own function for computing the value of *experience* depending on its preferences.

The formula above takes the relationship strength at time *t-1* and the current experience to calculate the relationship strength at the present time *t*. Depending on the value of α, either

1) $\alpha > 0.5$

   the previous relationship strength is given more importance when calculating the present relationship strength then current experience, i.e., the peer is conservative, (appropriate if system changes slowly.)

   or

2) $\alpha < 0.5$

   the previous relationship strength gets rapidly discounted and current experience is given more importance when calculating the present relationship strength, i.e., the peer is more explorative, which is appropriate if system changes rapidly.

Thus, if $\alpha < 0.5$, *e.g.*,$\alpha = 0.2$ new peers can have a chance to get involved in strong relationships sooner and will be more likely to be chosen for forwarding queries.

It may be good to have smaller values for α in the beginning when a peer builds up new relationships and increase it later, i.e., the peer will become more conservative in changing already established relationships.

3.6 Semantic Routing

As argued in the previous sections, it would be good to use the knowledge of a peer, which it had accumulated by experience to guide its future actions. Just like in social networks where people request from other people services or information based on knowing these people's strengths or knowledge, a peer can request files or documents in a given area from peers who it believes are more interested in the area. Therefore, we propose to route queries semantically, i.e., to peers in the friends list in the appropriate category.

When a query is initiated or is received by a peer, the query is classified into a category and the request is sent to the peers from the friends list associated with that particular category. Thus, the queries get forwarded to peers in similar interest category groups. This is beneficial since the query is now circulating among peers who have shown interest in that category and who may have the file with a greater probability, therefore the chance of the document being found is higher and the likelihood of finding it faster is higher than if the peer was searching among random peers.

A peer can send a query to all peers in its friends list for the category of the query. Depending on the length of the friends list a lot of traffic can be generated since it is in fact again broadcasting, just on a smaller scale. Therefore, it is a better solution to forward queries only to a small number of "best friends", based on the "strength of relationship".

Peers with high relationship strength are chosen for forwarding queries. The high relationship strength indicates those peers that have been most helpful, reliable and have shown a greater success rate in the past.

Thus peers chosen from friends-list to route queries are like strong ties in social networks, where the query in this social network finds the response, as queries now circulate among the peers in the category. Random peers used to forward queries, in the semantic model, are representative of weak ties found in social networks, as these peers help in finding peers interested in the category of interest.

3.7 Discovering New Friends

A P2P system is highly dynamic: peers come and go and change their interests. Consequently there is always a need by peers to discover new friends in the system. In our model, new friends are discovered by sending queries to a few peers that do not belong to the friends lists of the interest group of the querying category. That means that the neighbourhood of a peer, i.e., the peer to which a new query is sent, is formed by *m*

unknown peers and *n-m* friends. *n* can have different values. The standard Gnutella neighbourhood has size *n* = 7. For example in Fig. 3.3, Peer1 sends the request to seven peers in total out of which five are from the friends list and two are random peers. Through the random peer the peer explores the network: potentially new peers interested in the same semantic category may be found and added to the friends list.

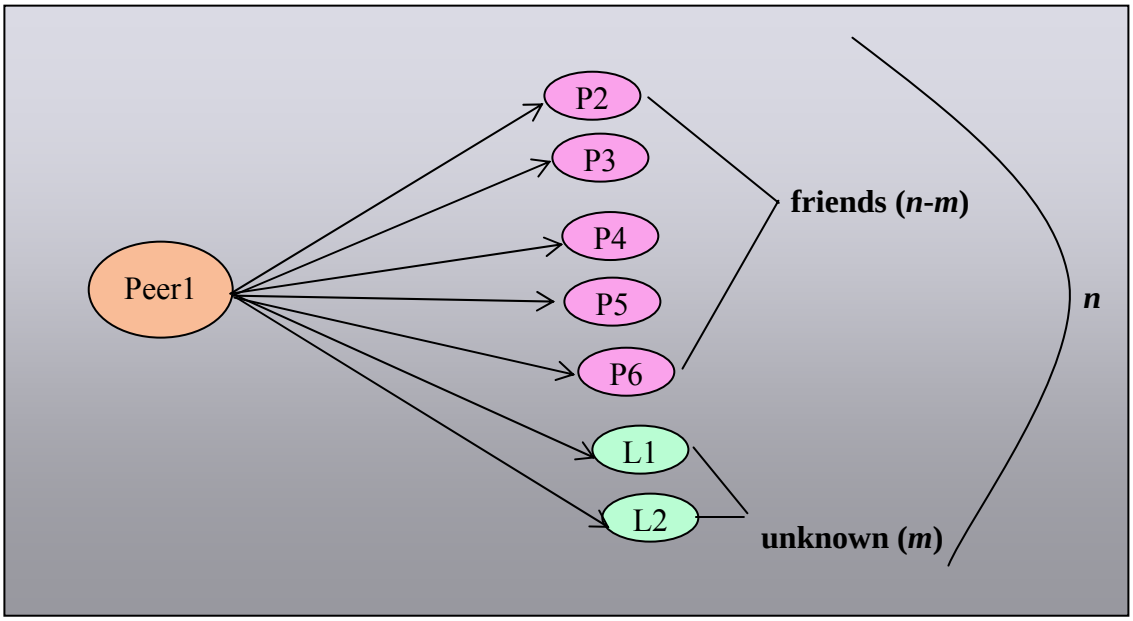

Figure 3.3: Discovering New Friends

An unknown peer once chosen will send the query to its friends (if it has any) in the querying category. The peers chosen by this independent peer for forwarding the query further may be new or not related to the querying peer. Thus, a peer chosen outside the friends list group lets the query go to new or unrelated peers who may have either entered the system recently or acquired document(s) of the interested category recently. This enables discovering new friends with the trade-off that more messages will be generated. The worst case will be that all the peers are chosen from outside the interest group of the querying category since the peer doesn't have friends yet and they have completely different interests and have no friends lists in the category of the query. This will result in a standard Gnutella routing based on the neighbours that are currently on line. The result is no new friends are discovered but messages will be generated until the query expires.

The above scenario indicates that it is necessary to strike a balance between choosing friends versus unknown peers for forwarding a query, i.e., choosing a good value for *m*. This is necessary to derive benefits from past experience and still be able to explore the network. Different policies can be implemented in the model for various situations. For example: starting with $m = n$, i.e., queries will be sent initially to unknown peers until enough interactions occur and the peer creates a friends list. Then the peer will forward queries to a few peers from the querying category's friends list and to a few unknown peers. Afterwards it will gradually forward to friends, keeping only 1 or 2 unknown peers in its neighbourhood to allow for exploration. If a query is unsuccessful then the forwarding is reverted to the original policy i.e., sending to all unknown peers or to peers outside the interest groups.

3.8 Summary and Discussion

A method for building friends lists and using these lists to semantically route queries for an individual peer has been introduced in this chapter. This method specifies how friends are discovered in the system, how the strength of relationship is calculated and updated, how the query is routed through the system.

Similar to the BestPeer system (Ng, et al., 2002), our method also reconfigures the network by connecting to different peers for different queries, but peers chosen in our model are from the friends list of that category. In our method beneficial peers are separated into different lists so that they are not lost if the query is from different category. We have used the concept of semantic category that other approaches have also used (Triantafillou et al., 2002) (Joseph, 2002).

As the number of peers in the system increase, the friends list will also grow in size. This in turn translates into costs of maintaining friends lists, e.g., storage for keeping friends list and computation and delay for calculating the strength of relationships of friends. However, the benefits of peers keeping friends lists – the decreased network traffic, increased speed of responses, success rate of queries, and amount of relevant content – typically outweigh the costs, as computer memory and storage is getting

cheaper and computers are getting faster. In case peers have limited computational resources, they can limit the size of their friends lists or implement specific policies for updating the strength of relationships (e.g. after every second interaction with a given peer rather than after every interaction) and still yield some benefits, depending on the dynamics of the system.

To test our model we simulate a Gnutella-based system with peers using our method of building friends lists and we compare it with a system where peers do not build and exploit friends lists. The next chapters describe the experimental set-ups, hypotheses and the results.

CHAPTER 4

Experimental Model

4.1 Hypothesis

In the previous chapter we have introduced the semantic-routing model and its properties. To evaluate the benefits of this model we need to compare its performance with a standard Gnutella system. For this purpose we simulate a peer-to-peer file-sharing environment.

The aim of the experiments is to show that a friends list used to semantically route queries reduces the search time for documents and decreases traffic by reducing the number of messages circulating in the system.

Our hypotheses are that:

1) Peers who share a lot of files in the system will become top peers in the friends' lists of every peer.
2) A system with a friends' lists performs better in terms of time for search and number of messages generated than a system with no friends lists whatever combinations of factors mentioned below is used.

We want to investigate the quantitative impact of searching semantically on the traffic in the network of the following factors:

1. Size of community;
2. Number of papers/files/documents shared in the system;
3. Distribution of files/queries in the system;
4. Number of neighbour peers the query is forwarded to.

We carry out experiments in two environments, static and dynamic. In the static environment, all peers are active throughout the entire simulation whereas in the dynamic environment the peers in the system can switch between online and offline state which is characteristic for real systems. In either environment we do not deal or take into account any network problems or delays in the system.

In the static system we compare the results of a system where peers use no friends with a system where peers send requests to three friends from their friends list and with a system where peers send requests to four friends. The experiments will be repeated for different numbers of peers in the system and different sizes of the friends list. The underlying peer system would be the same so that our comparisons is not affected by any other factors. For this purpose, our experimental model contains some modifications to the standard Gnutella 0.4 protocol in order to simplify the model and remove external factors. The modifications are described in the next section.

In the dynamic system we will compare results between two systems where one has a uniform distribution of queries and files kept by peers and the other has a more realistic Zipf-distribution of queries and files. (Wang et al. 2004) (Tsoumakos et al. 2003) (Lv et al. 2002).

4.2 System Model

Several modifications were made to the standard Gnutella 0.4 protocol, in order to allow a comparison between the systems without the influence of random factors. These are described in the next section. The system model design is described in section 4.2.2.

4.2.1 Protocol Modifications

The first modification to the standard Gnutella 0.4 protocol is that peers who have the requested files send responses directly back to the peer who originated the query and not through the queried path. The reason for this modification is that it allows us to focus on the time for search rather than considering the entire transaction, which can be influenced by a number of external factors, such as general network delays.

Following the same reasoning, a second modification has been introduced where the file is sent back by the responding peer to the query initiating peer as the response instead of the two process transaction in Gnutella: the responding peer sending a "query hit" response first, followed by a request for downloading the file by the querying peer and finally the transfer of the document. We assume here that the querying peer does not replicate the file, so a peer can search for a file repeatedly. Also, we ignore the file size as a parameter, i.e., we assume that all files take a constant amount of time to download.

A third modification is that once a peer sends a "hit", the responding peer no longer forwards the query. In this way we can focus on the traffic generated until the first responses start coming and ignore the echo effects generated when this peer still forwards the query in the system. Also if a peer has already forwarded a query, it does not forward the message further if it gets the same query again.

The model of the system used for simulation is based on the assumption that the system is a document file sharing system that contains a relatively small number of peers and categories. Also, too many categories will lead to an increase in the calculation time within each peer. A large number of peers lead to a very large runtime of the simulation. Therefore we were able to experiment only with 100 peers.

4.2.2 System Model Design

The documents in the system are pre-classified into categories, represented as numbers. This numeric representation does not affect the routing since both file-names represented as strings and file-names represented as numbers have to be matched exactly. Numbers are easier to match in the simulation. The files shared by peers are also represented by numbers. The same rationale applies here: numbers are easier to match in a simulation. The neighbours for any peer in the system are selected randomly; the number of neighbouring peers is fixed. This applies to both systems: with and without friends lists. The file numbers and their distribution are set at the start of the simulation and remain unchanged during the course of the simulation.

We start with a baseline model of the system with the objective to determine how quickly peers learn:

1) about resourceful peers in the system, and
2) about peers with similar tastes.

In the baseline model, we can further assume without loss of generality that there is only one category of interest in the system. This assumption can be justified as follows. Recall from section 3.5 that the friends lists for all the various categories are kept separate by every peer. The peer's relationship strength in a list is not influenced by the strength of the relationship with same peer in any other lists it features in. Since the peers in the friends lists are used to send queries only for the particular category for which the peer has generated the query and there is no interaction between queries, the speed of learning of new friend peers in a given category will not influence the speed of learning of friends in a different category. Thus the system can be broken down into several independent subsystems, each of which can be simulated independently. From this we can assume that in the system there are some peers that are interested in a category and the rest of peers in the system are not interested in that category. So, we can consider without loss of generality a system that has only one category.

For exploring the system and learning about potential new friend peers in a category a peer needs to send queries also to peers outside of the peer's friends list of the category. As explained in section 3.7, these peers can be new peers and/or peers that are not yet known, or who share files in another category. We can assume also that there are peers in the system that are not interested in any category and do not share any files (empty peers) along with peers interested and sharing files in the category. The empty peers just facilitate the infrastructure of the system, i.e. they forward queries.

In the beginning there are no relationships among the peers in the system, since there have been no interactions among them yet. Interactions happen as queries are generated and responses arrive. Queries are generated by randomly chosen peers in a fixed interval

of time. The file number to be queried is generated randomly. The query is sent to the peer's neighbourhood consisting of *n-m* friends and *m* random new peers. These peers search through their resources and if they have the file, they return a "hit" message. Each query is identified by a unique id and the id of the originator of the query so that hits can be sent back to that peer. The unique id also helps to discard the query if the query gets forwarded more than once to any peer. In 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 all peers that respond to each of its queries. It then calculates the strength of the relationships with these peers and stores them in its relationship list as tuples containing peer id and the relationship strength. In the simple model all peers in the list are retained in the friends list and none are dropped even if the strength of the relationship is very small. This helps to see the dynamics of the relationship strength with the peers who once interacted with the initial querying peer.

The simulation system contains a varying number of peers with and without files and a single category of interest. The set-up of the system is such that the file distribution varies among the peers sharing files. For example: 57% peers that share ten files, 29% peers sharing twenty files and 14% peers sharing forty files. This file distribution remains consistent, though the actual number of peers sharing files in the system changes. The relationship strengths for all peers are calculated after a set time and sorted so that the list is updated. Thus a peer can use the most recently updated value of relationship strength to pick peers to forward queries.

We experimented with different proportions of peers sharing files (i.e., interested) in the category: 8%, 10%, 15%, 30%, 50%, 70%, 90% and 100% of the total number of peers simulated in the system. In each of these cases, the remaining peers are considered to be not interested in the category and thus they do not share files in that category. They just facilitate the circulating of messages in the system.

The number *n* of neighbour-peers to which a peer sends its requests is set to five (5). Two sets of experiments were performed with different values of *m*, i.e., the number of neighbours chosen from the friends list. In the first experiment n = 5, m = 4, i.e., a peer has a neighbourhood of five out of which four peers are chosen from its friends list (the top four with strongest relationship) and one peer is chosen randomly from the rest of the peers in the system. This simulates a peer using 4 strong ties and 1 weak tie in its search. In the other experiment n = 5, m = 3, i.e., a peer uses more weak ties (2) in its search. The TTL (time to live) of the requests is 4 in both cases.

For the purpose of comparison, a baseline system with the same number of peers, all interested in the same category, was implemented. The file distribution among peers remains consistent with that described above. Here each peer at set-up picks five other peers randomly and makes them its neighbours. The neighbourhood is fixed throughout the length of the simulation, as in the standard Gnutella 0.4.

4.2.3 Static System

In the static system all the peers created at the start of the system are assumed to be active during the entire simulation. The number of peers in the system does not change during one simulation run but can be changed for different simulation runs. This assumption makes the simulation less complex. Since we are interested to find the performance at "snapshots" of time when queries are generated we can assume that the system is static (i.e., no new peers or leaving peers).

The first set of experiments, in Table 4.1, finds the impact of the total number of peers in the system on the average number of messages and hops.

Table 4.1: Details of the number of peers and files in the first set of experiments

With friends list		**Without friends list**		**Figure Numbers**
Num. of peers	**Num. of files**	**Num. of peers**	**Num. of files**	
50	200	50	200	Fig. 5.1 and 5.2
100	200	100	200	Fig. 5.3 and 5.4

The second set of experiments, in Table 4.2, investigates the impact on the performance of the total number of files shared in the system. Here the system that sends queries to peers from its friends list is compared with the system that does not have friends list. Each set was repeated thrice and the average is used to get the final graphs.

Table 4.2: Details of the number of peers and files in the second set of experiments

With friends list		Without friends list		Figure Numbers
Num. of peers	**Num. of files**	**Num. of peers**	**Num. of files**	
100	200	100	200	Fig. 5.3 and 5.4
100	100	100	100	Fig. 5.5 and 5.6
100	150	100	150	Fig. 5.7 and 5.8

4.2.4 Dynamic System

In the dynamic system two experiments were performed. In the first one files shared by the peers follow a uniform distribution and also the queries follow a uniform distribution.

In the second experiment, the files shared by the peers are Zipf distributed, which implies that a few popular files are shared by a large number of peers and some less popular files are shared by very few peers. Also the queries were generated according to the Zipf distribution, which means that peers generate more queries for popular files and fewer queries for files that are not very popular. Files are ranked from 1 to i in order of popularity, 1 being the most popular file and so on. The Zipf law states, “the occurrence of an event is inversely proportional to its rank” (Adamic et al. 2002). The distribution that follows this law is the Zipf-distribution. Popular files are copied often and shared by a lot of peers in a Gnutella system (Wang et al. 2004) (Schlosser et al. 2003).

According to Zipf’s-law, the probability of file P being queried or chosen to share is given as:

$$P(i) = 1/ i^{a} \qquad (4.1)$$

where i is the rank of that file and a an exponent. In our simulation we use a = 0.82 (Wang et al. 2004) (Tsoumakos et al. 2003) (Lv et al. 2002).

The modifications described in section 4.2.1 to the standard Gnutella protocol hold also for the dynamic system. Also the system model as described in section 4.2.2 applies here. The query routing and the strength of relationship of each peer is calculated and updated periodically. The only differences from the static system are that peers can go offline and then re-enter the system and that the files and queries follow particular distributions (uniform and Zipf). The data used for the times a peer is online were obtained from real usage data of the COMTELLA system (Vassileva, 2002) and is given in Table 4.1 and 4.2. The times when a peer is online and goes offline are exponentially distributed. The data, from which one calculates the frequency of logging in and the time that the peer is online, are given in Table 4.3 and Table 4.4. For calculating the overall time a peer is online, we generate a random number and then find its upper and lower bound from the graph.

Table 4.3: Data for the total percentage of time peers are online

Percentage interval of total login time	Number of peers in the interval	Cumulative total
0-10	25	25
10-20	1	26
20-30	1	27
30-50	4	31
50-70	2	33
70-100	2	35

Table 4.4: Data for the number of times (frequency) a peer logs in to the system

Frequency interval of number of login	Number of peers in the interval	Cumulative total
0-5	10	10
6-10	13	23
11-15	4	27
16-20	4	31

The exponential graphs obtained from this data given by Fig. 4.1 and 4.2 are used to obtain the total time and frequency a peer is online at any time.

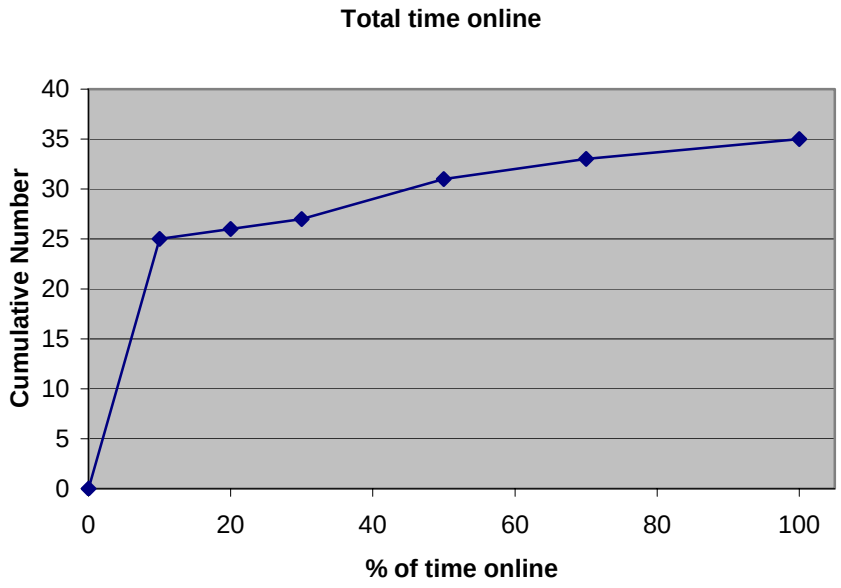

Figure 4.1: Graph obtained from the data for the total percentage of time peers are online in the system.

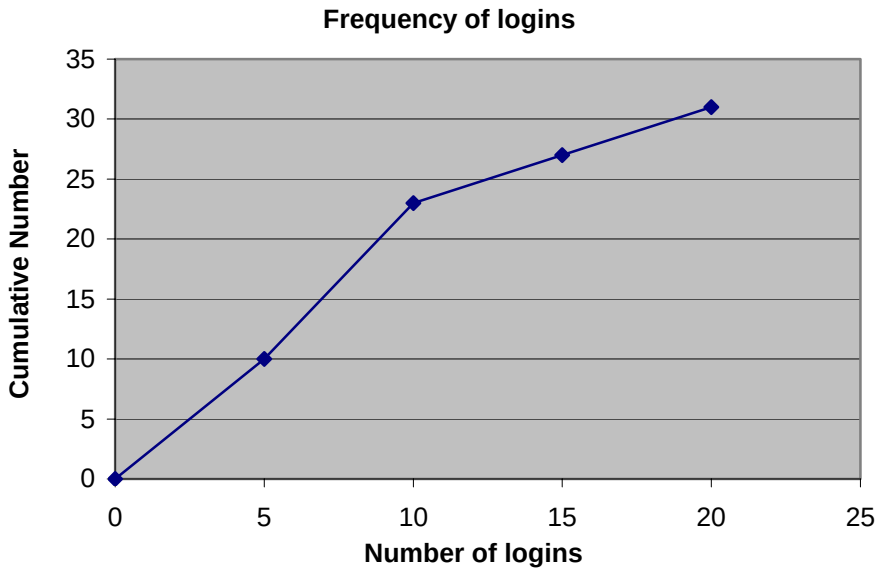

Figure 4.2: Graph obtained from the data for the total number of time peers are login into the system.

To obtain the actual time a peer is offline and online we assume that both times, i.e., online and offline, are exponential. The inverse-transform formula requires a uniformly distributed random variable r, $0 \leq r \leq 1$.

We use the inverse-transform formula (Law, et al. 2000):

$$t = -E(x)\ln(r)$$

Following from the above formula we calculate the time peer I is online at a single login time as:

$$t = (-((ts * (dI/100)) / nI) * \ln(c)) \qquad (4.2)$$

and the time the peer I is offline before logging in is:

$$t = (-((ts * (1 - (dI/100))) / nI) * \ln(d)) \qquad (4.3)$$

where nI is the frequency a peer logs in, dI the percentage of time the peer is online, *c* and *d* are random numbers between the interval [0-1] and *ts* is the total time of the simulation.

We need the frequency, i.e., how many times a peer logs in, in a given time period and the percentage of time it is online for the simulation. This is chosen randomly using the empirical frequency and percentage of time online from the data obtained by COMTELLA. The formulas used to calculate nI and dI are given below.

The data in Tables 4 and 5 were used to calculate dI and nI respectively. First we do a table lookup and find the interval in the cumulative distribution where the uniform random variable, uniformly distributed between 0 and the number of observations, fits in. From the empirical distribution for uniform distribution formula (Law, et al., 2000), we calculate the total time a peer I is online as:

$$\mathrm{dI} = x1 + (((x2 - x1)/(y2 - y1)) * (a - y1)) \qquad (4.4)$$

where dI is the time a peer is online overall, *a* is the uniform random value between 0 and the number of observations, *x1* and x2 are the class limits for *a*, and y1 and *y2* are the cumulative totals corresponding to *x1* and *x2*.

The number of times a peers logs into the system, nI, is also calculated in a similar way as given in equation 6.

The number of files shared by peers is according to the Zipf-distribution, since in reality many peers share few files and few peers share a lot of files. It has been noted that in Gnutella relatively few peers, e.g., 20% share most of the files, e.g., 90% (Adar et al. 2000) (Saroiu et al. 2002). For this reason, we have, as mentioned above, set up peers such that 57% of peers share 10 files, 29 % share 20 files and 14 % share 40 files. We define the distribution similarly to the distribution in the static system. Good peers, i.e.,

those peers that share the most and qualitatively good files, stay online about 45% of the time or more (Wang et al. 2004) (Saroiu et al. 2002). This also has been taken note of and modeled in the system. Thus, in the simulation peers that share 40 files are online at least 50% or more of the time.

The reinforcement learning formula (2) to update the peer's relationships strength used here is from chapter 3. The value of α used is 0.1. This means peers give more weight to current experiences than past experiences since the system is fairly dynamic. Both experiments were repeated three times and the average of these runs was used to obtain the final data and graphs.

4.2.5 Performance Measurement

We define performance of a given system as a measure consisting of: the average time for a query to receive response (hit) (measured in hops) and the average number of messages generated by a query. The average time is measured as the number of hops and not the actual time in say, seconds. This is because there could be a delay in processing the query and its responses due to long message queues at the peer in the simulation environment. The time thus calculated in seconds or milliseconds is not indicative of the actual time for the response to come back to the querying peer. Therefore, the number of hops, which is not affected by delay times, is used to calculate speed.

Since each query can get more than one hit, the average number of hops for each query is calculated first and then the average number of hops for all queries is calculated. Similarly, the number of messages for each query is summed up and then the average number of messages for all queries is calculated.

4.3 Simulation

The simulation has been implemented in Java on JADE (JADE 2.61), 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 a FIPA-Compliant, multi agent platform specification using Java. The communication between the agents in Jade is through ACL (Agent Communication Language). JADE (Bellifemine et al., 1999) has the abstract notion of behaviours associated with each action. Each agent in Jade is a thread and the actions of the agents are executed linearly. When an agent receives an ACL message, the agent retrieving the relevant section of the message interprets the message and carries out appropriate methods or tasks according to the interpretation of the message. Besides behaviours, JADE also has timer tasks defined for its platform. A timer task is used to implement the firing of a query. A timer event is also used to update the friends lists periodically so that the peers in the simulation use these updated friends lists to semantically forward the queries.

The peers in the simulation are represented by Jade Agents. Peers were created in the system, and these peers interacted among themselves by querying and responding. After the set up, queries are fired at a fixed time interval. The main class in the simulation chooses the peer, picked by a random generator that would initiate the query. The peer then randomly generates a file number to query, with the constraint that it is not one of its own resources. A peer can repeat queries since the peer does not replicate the file it queried before. When a hit reply comes, the peer originating the query notes the number of hops taken to get the result of that query. Also the number of messages circulating in the system for each query in the system is tracked. These data are recorded in files.

The simulations runs were done on a MS Windows machine (Windows Server 2003 with intel pentium III Xeon 700 MHz) and on a Solaris machine. The total time for the simulation includes:

1) an initial fixed time of 20 seconds for creating the peers and their resources
2) a variable time needed for the simulation to complete. The time to create a single query is 20 seconds.

Since the percentages of peers in the category in each simulation differ, to obtain comparable results, each peer on average is assumed to query files twenty (20) times.

That means, for example, that if the percentage of peers interested in a category is 10%, the total queries in the system would be 200 i.e., 10 * 20 A query is fired approximately every 20 seconds in the system, so the time for the 200 queries is 4000 seconds, and the total time for the simulation in this case is: 20 + 4000 = 4020 seconds. Thus the time taken for the simulation varies according to the percentage of peers in the category as the total number of queries differs in each simulation. Each set of experiment was repeated three times and the average of these three runs were used to obtain the final data and graphs.

In addition to the main questions asked in the beginning of this chapter (section 4.1) related to the evaluation of the semantic routing approach, the simulation allows also the investigation of the following interesting questions:

1) Does the total number of peers in the system influence the average number of hops and average number of messages generated?
2) Does the different percentage of peers interested in a category affect the performance of the system? If so, what percentage of peers interested in a category does the proposed semantic routing approach improve the performance most?

CHAPTER 5

Results and Discussion

5.1 Results of the static system

Our goal here is to compare the systems: the system where peers send queries to friends with the system where peers do not have friends. The TTL of the query is 4 and the total number of neighbours that a peer forwards queries to is 5. In both systems, in the first set, the number of active peers was 100 and they shared 200 files.

The graphs in Fig. 5.1 and Fig. 5.2 show the data gathered from this experiment.

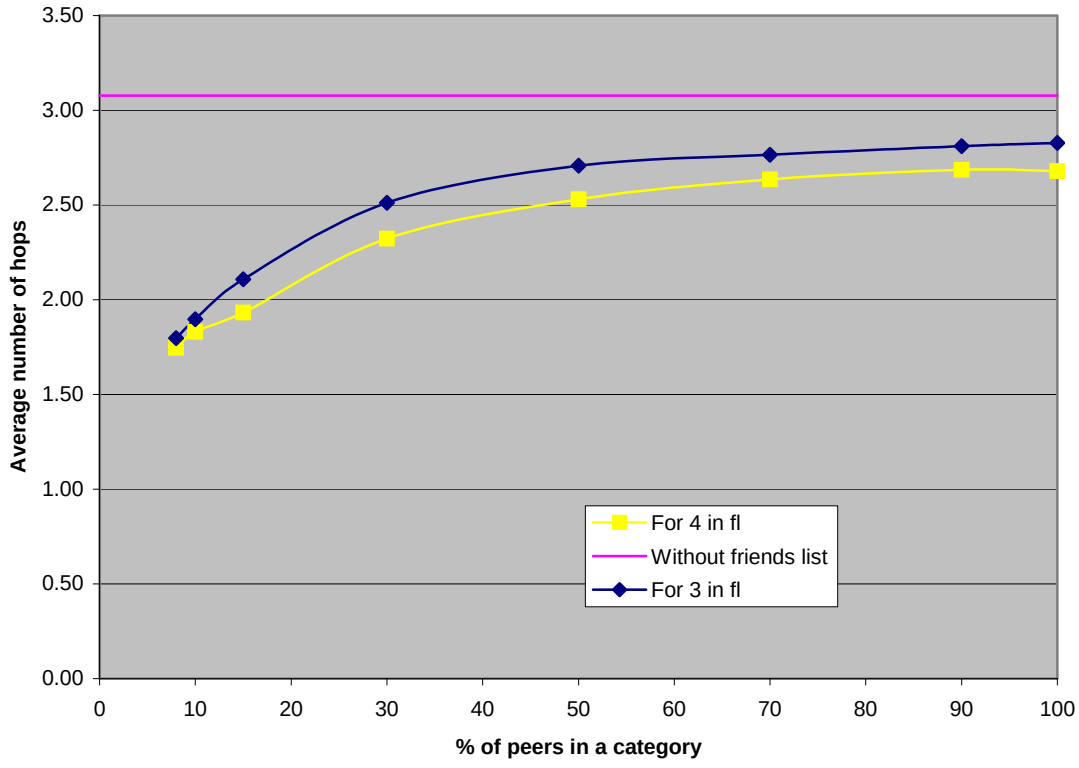

Figure 5.1: Average hops vs. percentage of peers in category for 100 peers and 200 files

In Fig. 5.1,

- the straight line indicates the average number of hops in the baseline system where peers do not maintain friends lists,
- the lighter line with squares indicates the average number of hops in the systems where the peers send queries to four peers from the friends list and one random peer;
- the darker line with diamonds indicates the average number of hops in the systems where peers send queries to three peers from their friends list and to two random ones.

From Figure 5.1, we see that when four neighbours are chosen from the friends list, the average number of hops increases from 1.75 (when 8% of the peers in the system are interested in the category) to 2.68 (when 100% of peers are interested in the category). When three neighbours are chosen from the friends list, there is a slight increase in the number of messages and the average number of hops increases from 1.80 (when 8% of the peers are in the category) to 2.83 (when 90% of the peers are in the category). Therefore, a system where peers send queries to 4 friends performs better (by 0.05 – 0.15 hops) than a system with 3 friends. With respect to hops the system where peers send queries to friends performs better than the system where peers do not have friends.

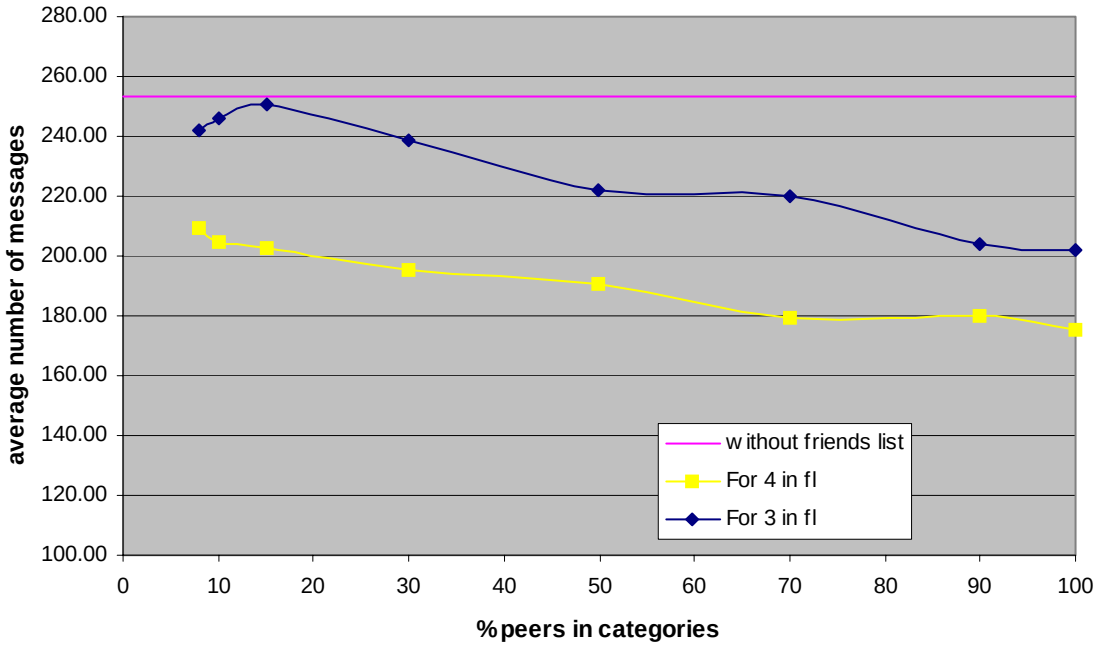

Figure 5.2: Average messages vs. percentage of peers in category for 100 peers and 200 files

In Fig. 5.2, the straight line indicates the average number of messages generated when no friends list is maintained, the lighter line with squares indicates the average number of messages when four peers from the friends list are used and the darker line with diamonds indicates the average number of messages when three peers from the friends list are used. Figure 5.2 indicates that when four peers are from the friends list the average number of messages decreases from 209.66 (when 8% of the peers are interested in the category) to 175.01 (when 100% of the peers are interested in the category), with a slight increase in between to 180.11 (when 90% of the peers are interested). Similarly when three peers are from the friends list the average number of messages decreases from 241.83 at 8% in category to 201.92 for 100% in category. The system with 4 peers from the friends list performs better (32.17 - 26.91 messages difference) than the system with 3 peers from the friends list.

Additional graphs showing the percentage of benefits we obtain by using a friends list are given in Appendix A. The standard deviation values, for the three runs done to obtain the average number of messages, for all the static experiments are given in Appendix B.

Our next goal is to investigate the influence of the number of peers. We compare here systems with different size of peer community but the same number of shared files. So in this system the number of peers active were 50 and the files shared were 200.

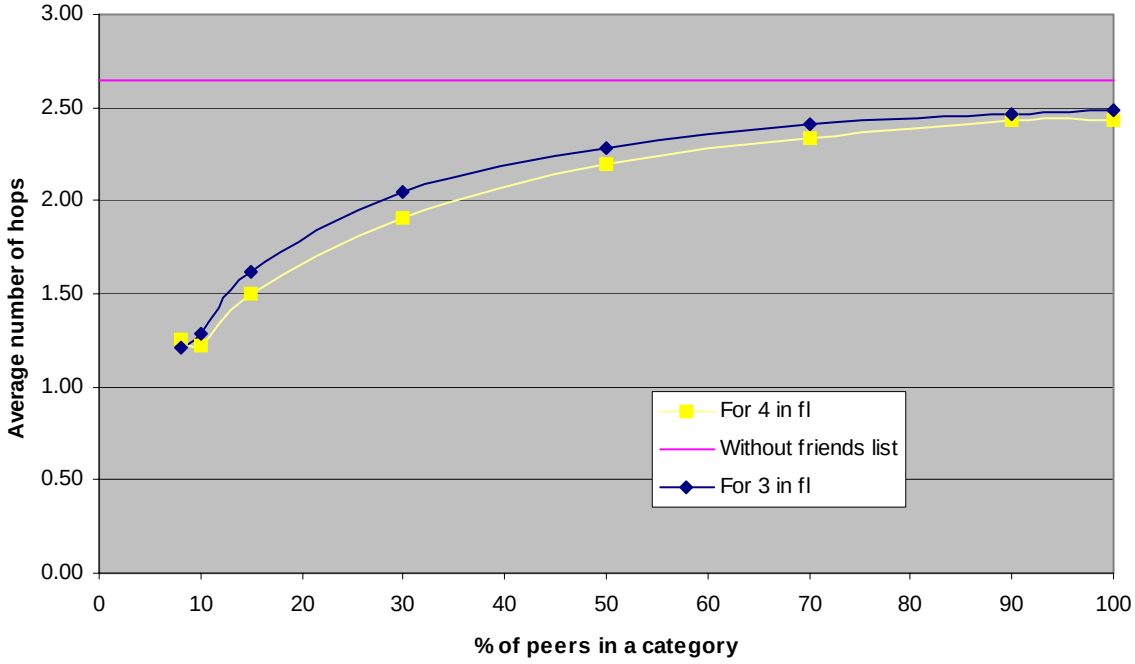

Figure 5.3: Average hops vs. percentage of peers in category for 50 peers and 200 files

In Fig. 5.3 we see that when four neighbours are chosen from the friends list, the average number of hops increases from 1.25 (when 8% of the peers in the system are interested in the category) to 2.44 (when 100% of peers are interested in the category). When three neighbours are chosen from the friends list, there is a slight increase in the

average number of hops from 1.22 (when 8% of the peers are in the category) to 2.48 (when 100% of the peers are in the category).

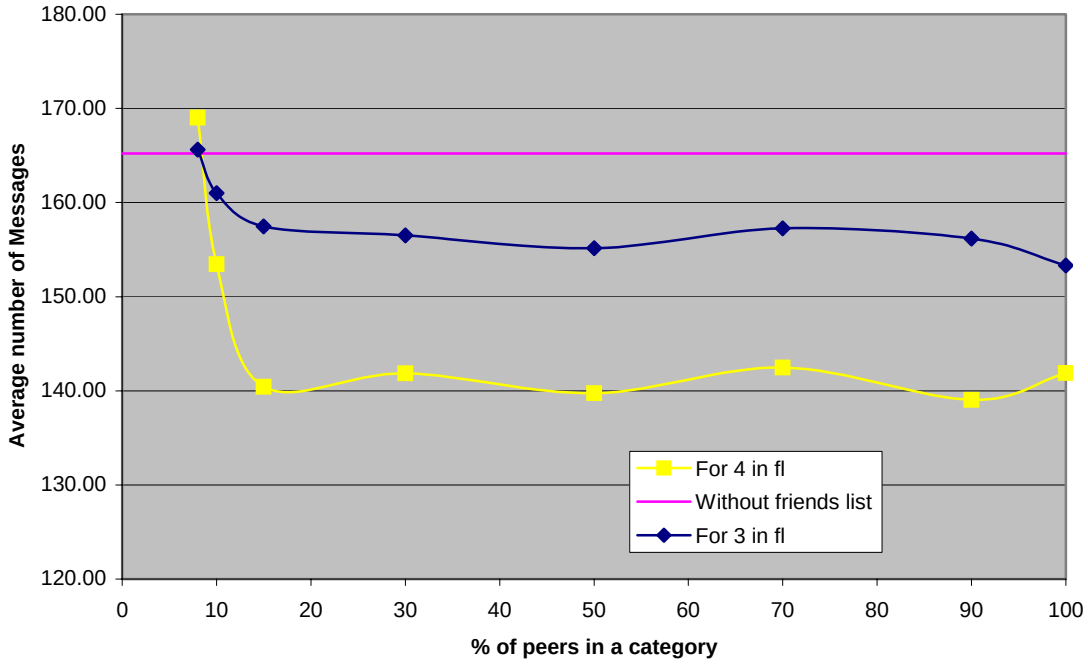

Figure 5.4: Average messages vs. percentage of peers in category for 50 peers and 200 files

In Fig. 5.4, it can be seen that when four peers from the friends list are used the average number of messages varies from 169.01 (when 8% of the peers are interested in the category) to 141.89 (when 100% of the peers are interested in the category), with a slight increase in between to 142.48 (when 70% of the peers are interested). Similarly when three peers from the friends list are used the average number of messages varies from 165.62 at 8% in category to 153.52 for 100% in category, with a slight increase in between to 157.25 for 70% in category.

When we compare the systems differing only in the number of peers, i.e., one system with 50 peers and the other with 100 peers (results in Fig. 5.1 and 5.2) with the same systems without friends list, we find that the hops follow a similar pattern. There is a

benefit of about 40% to 50% when the percentage of peers interested in a category is small. The greatest benefit with respect to speed in obtaining a response is when the peers choose four neighbours from their friends lists. Similarly the average number of messages circulating in the system is less when peers choose four neighbours from their friends list. From these observations we can say that when the system uses a friends list to send queries the traffic (i.e., the number of messages generated) is smaller and the speed of responses is increased compared to the system without friends list.

Our next goal is to investigate the influence of the number of files in the system. The next experiment compares the same type of systems with 100 active peers in the system and different number of files shared by the peers. There are three data sets for this experiment. The first data set in this experiment is explained in Fig 5.1 and Fig. 5.2 where the peers share 200 files. The second data set, used for Fig 5.5 and Fig 5.6, in here is with 100 peers and share 100 files.

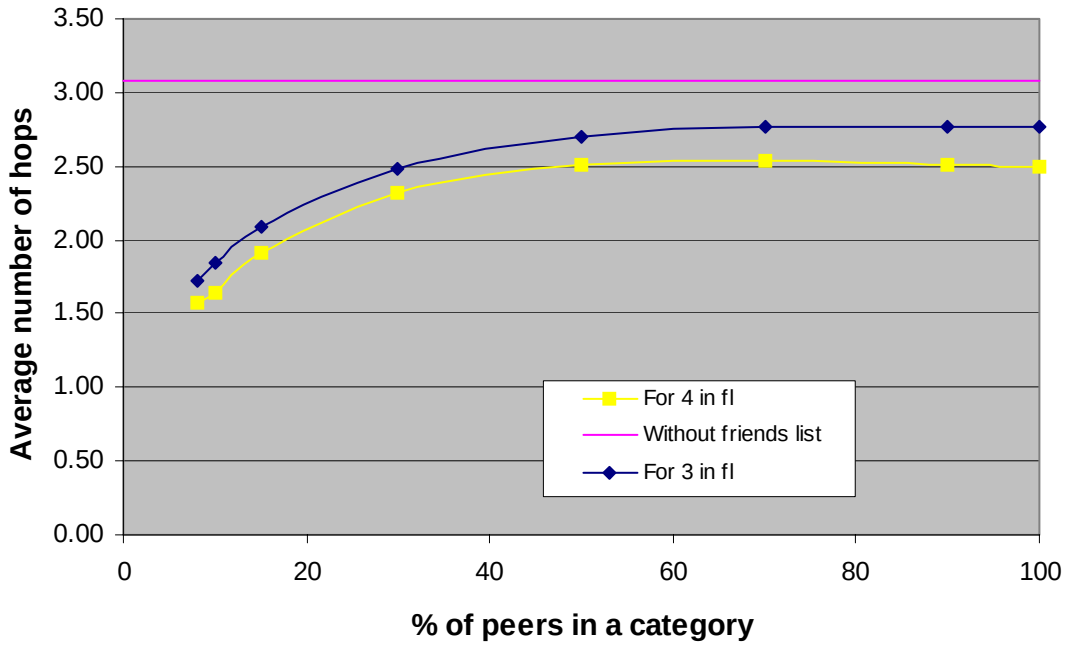

Figure 5.5: Average hops vs. percentage of peers in category for 100 peers and 100 files

In Fig. 5.5 we see that when four neighbours are chosen from the friends list, the average number of hops increases from 1.58 (when 8% of the peers in the system are interested in the category) to 2.50 (when 100% of peers are interested in the category). When three neighbours are chosen from the friends list, there is a slight increase in the average number of hops from 1.72 (when 8% of the peers are in the category) to 2.76 (when 100% of the peers are in the category).

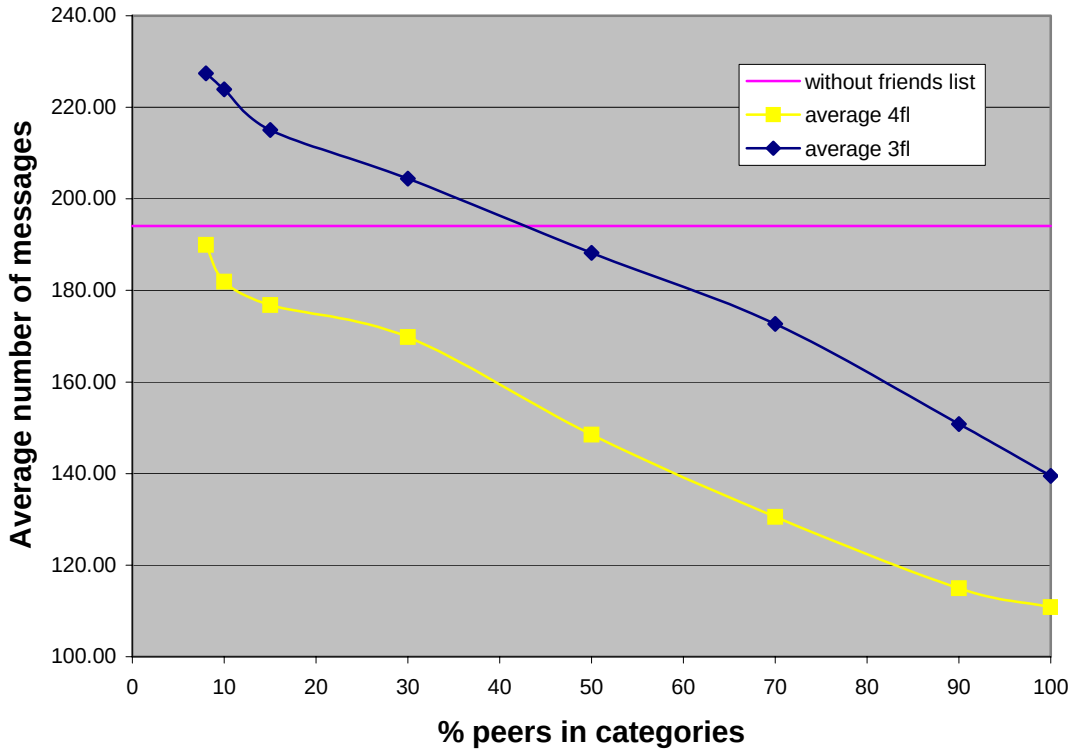

Figure 5.6: Average messages vs. percentage of peers in category for 100 peers and 100 files

In Fig. 5.6 we see that when four neighbours are chosen from the friends list the average number of messages decreases from 189.93 (when 8% of the peers are interested in the category) to 110.81 (when 100% of the peers are interested in the category). Similarly when three neighbours are from the friends list the average number of messages decreases from 227.40 at 8% in category to 139.51 for 100% in category.

The third data set, used for Fig. 5.7 and Fig. 5.8, in the second experiment is when the peers shared 150 files.

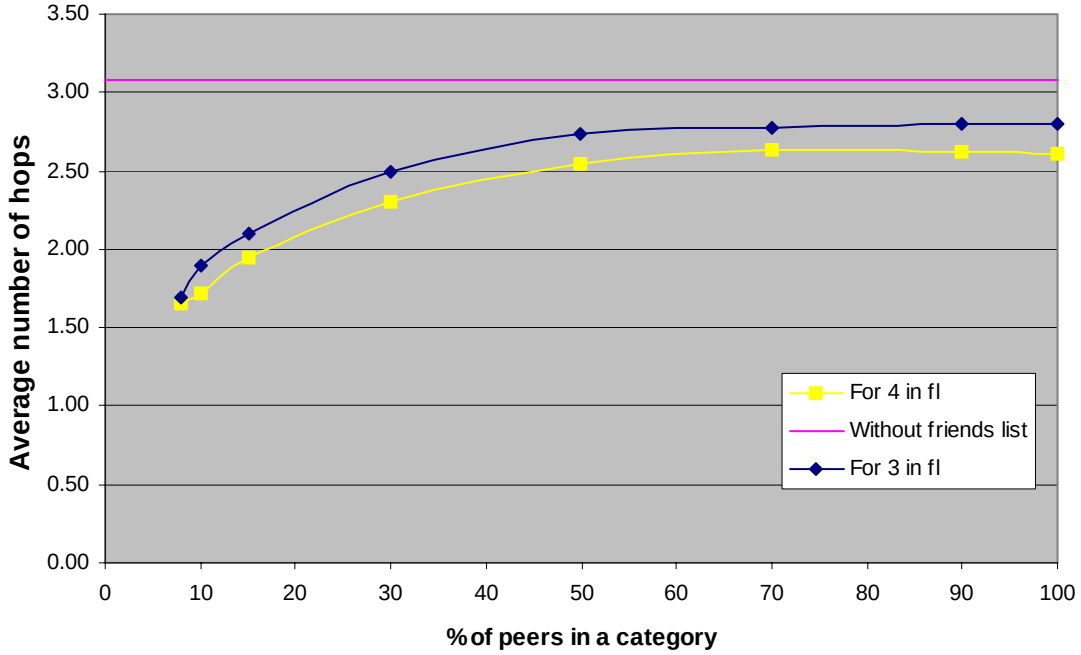

Figure 5.7: Average hops vs. percentage of peers in category for 100 peers and 150 files

In Fig. 5.7 we can see that when four neighbours are chosen from the friends list, the average number of hops increases from 1.66 (when 8% of the peers in the system are interested in the category) to 2.61 (when 100% of peers are interested in the category). When three neighbours are chosen from the friends list, there is a slight increase in the average number of hops from 1.70 (when 8% of the peers are in the category) to 2.81 (when 90% of the peers are in the category), followed by a slight decrease in the number of hops to 2.79, when 100% of the peers are in the category.

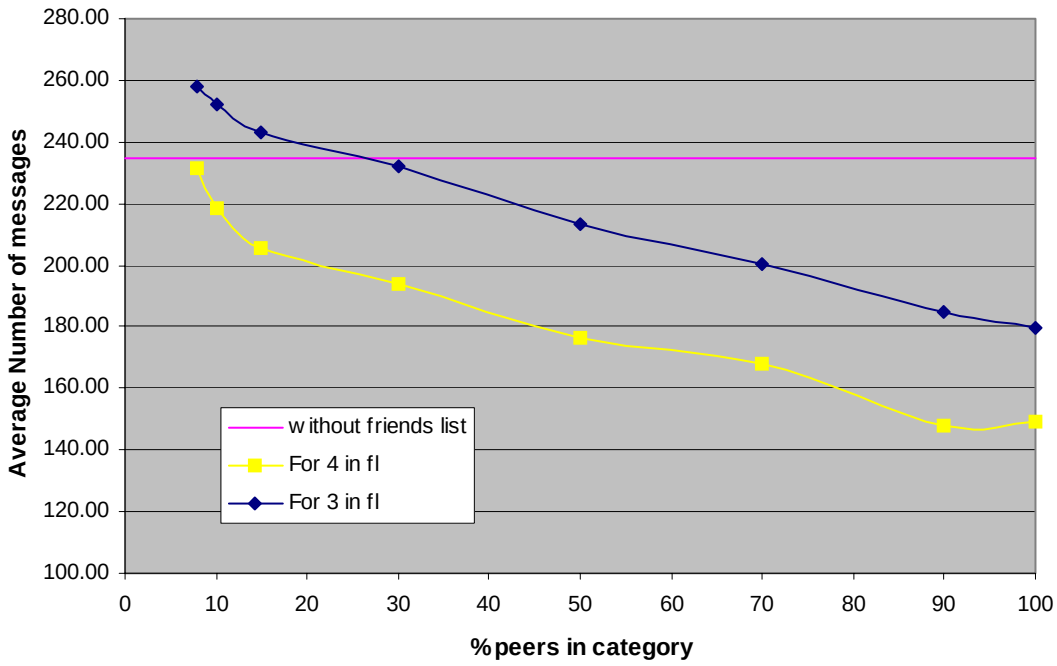

Figure 5.8: Average messages vs. percentage of peers in category for 100 peers and 150 files

In Fig. 5.8 it can be seen that when four neighbours are chosen from the friends list the average number of messages decreases from 231.61 (when 8% of the peers are interested in the category) to 148.92 (when 100% of the peers are interested in the category). Similarly when three neighbours are from the friends list the average number of messages decreases from 258.19 at 8% in category to 179.87 for 100% in category.

When we look at the three data sets differing only in the number of shared files i.e., 100 files shared, 150 files shared and 200 files shared, but 100 active peers in the system, we find that in general using a friends list is beneficial. It is clear, that the percentage of peers interested in a category does influence the performance. The greatest benefit of our model with respect to speed in obtaining response from all these data sets is when the percentage of the peers interested in a category is small, i.e., about 8-15% and when

the peers choose four neighbours from their friends lists. In some cases when the percentage of peers in a category is lower, for example from 8% to 30%, it is seen that the number of messages circulating in the system when three peers are chosen from the friends list is greater than the number of messages without a friends list. This is because the random peers chosen keep finding more random peers thus querying most of the peers in the system.

The reason for number of hops increasing as the percentage of peers in the category increases is because as the growth of percentage of peers increases in the category, the heterogeneity increases and ultimately with 100% it is the same as having no categories at all. So we find that the number of hops in both the systems is similar. At this point the friends list is not useful anymore, i.e., it does not bring any benefit to the system.

The decrease in number of messages as percentage of peers in the category increases can be explained in the following way: When the percentage of peers in the category is small, e.g., 8% peers are interested in the category, the queries fired by a peer obtain hits at most in 2 hops. This is because in our system of 100 peers, 5 neighbours and TTL = 4, the friends lists of all the peers interested in the category will have typically the top peers in their list. Thus, replies are obtained faster. Since only 8% of the peers are in the category, the fifth random peer chosen will likely be outside the category and since these peers do not have friends lists they send messages to random peers, which again are more likely not to be in the category. Thus the probability that random peer will find peers in the category is small. The messages will go on till the TTL of the query expires or till the peer has already seen the query. Thus a lot of messages circulate in the system.

When the percentage of peers in a category increases, i.e., 90% of peers are interested in the category, the random peer, chosen by each peer for forwarding the query would most likely belong in this category, i.e., the probability that the random peer chooses peers interested in the category is large, and will route the message using its friends list. Since the likelihood of each peer to which a message is forwarded to select a random peer that is outside the category is small, the query is propagated through peers that are in the

category and are therefore more likely to have the file, generate a response and not forward the message further. Alternatively, if they don't have the file, it is more likely that those peers have already seen the query and will not forward it further. Therefore the number of messages is smaller.

From both the experiments, one in which the number of peers differs and the number of files shared remains the same and the other in which the number of peers active in the system remains constant but the number of shared files differs, we can conclude that generally, our model is always beneficial over the standard Gnutella in terms of speed and number of messages circulating in the system regardless of what percentage of peers are interested in the category and how many neighbours are chosen from the friends list. Also from both experiments we can see that the system in which peer sends queries to 4 friends performs better than the system in which a peer sends queries to 3 friends.

5.2 Results of the dynamic system

We carried out two experiments: one where the file and query distribution is uniform, and a more realistic one, where file and query distribution is Zipf.

Experiment with Uniform Distribution

For the first experiment of the dynamic system the distribution is uniform. Our goal is to compare the uniform distribution system that maintains friends list to the system that does not have friends list.

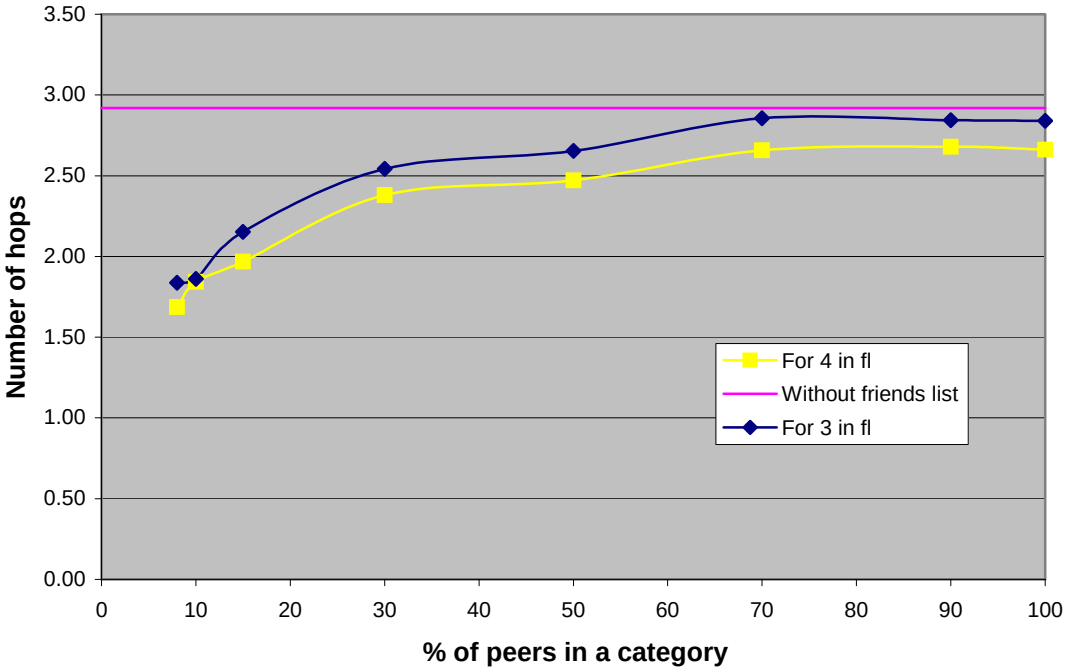

Figure 5.9: Average hops vs. percentage of peers in category for the uniform distribution dynamic system with 100 peers and 200 files

In Fig. 5.9 we can see that whether four neighbours are chosen or three neighbours are chosen from the friends list, the average number of hops is less than in the systems when no friends list is used. There is a slight benefit in the system when four neighbours are chosen over three neighbours. But it can be seen that in both cases the average number

of hops is less than the hops in the system that does not use a friends list to forward queries.

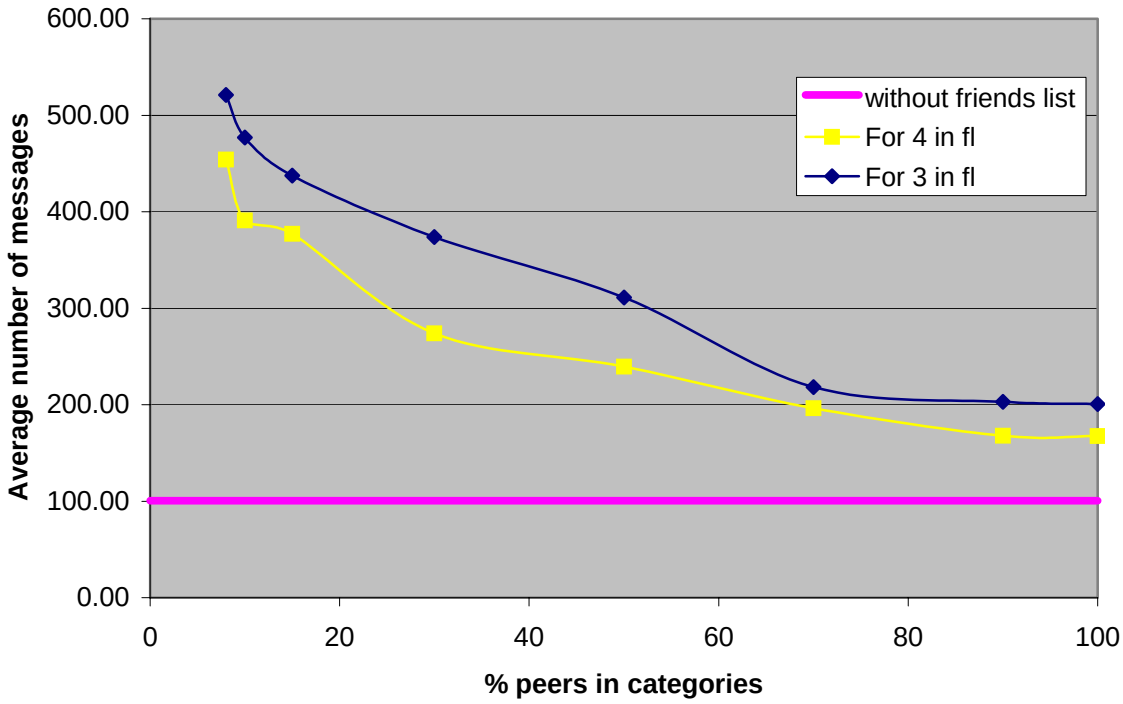

Figure 5.10: Average messages vs. percentage of peers in category for the uniform distribution dynamic system with 100 peers and 200 files

In Fig. 5.10 we note that the system where four neighbours are chosen creates fewer messages than the system where three neighbours are chosen from the friends list. But we also see that the average number of messages in the system without friends is less than in the system where peers send queries to peers from their friends list. It seems that our model performs worse in terms of average messages in the system. However, this conclusion is not correct and this behaviour can be explained as follows. In a dynamic system, peers go offline. When the neighbourhood of a peer is fixed the queries are not forwarded further if the neighbouring peers are offline. Thus, only few messages are passed depending on which neighbour peers are still online and forwarding requests. However, the overall success of queries is less when compared to the system using

friends lists. This can be seen from the data about the success of queries in both systems, presented in Table 5.1.

Table 5.1: Success of queries in the Uniform distribution system

% of peers in the category	***query success in % (where queries are forwarded to 4 friend peers and 1 random peer)***	***query success in % (where queries are forwarded to 3 friend peers and 2 random peers)***	***query success in % (where no friends list is maintained)***
8	47.81	39.13	
10	50.59	53.12	
15	70.58	75.01	
30	88.57	90.93	
50	97.12	97.14	
70	99.58	99.49	
90	99.10	99.63	
100	99.83	99.66	37.11

Experiments with Zipf-distribution

The following graphs show the results from the experiment were queries where fired according to Zipf-distribution with 100 peers in the system and 200 shared files in total.

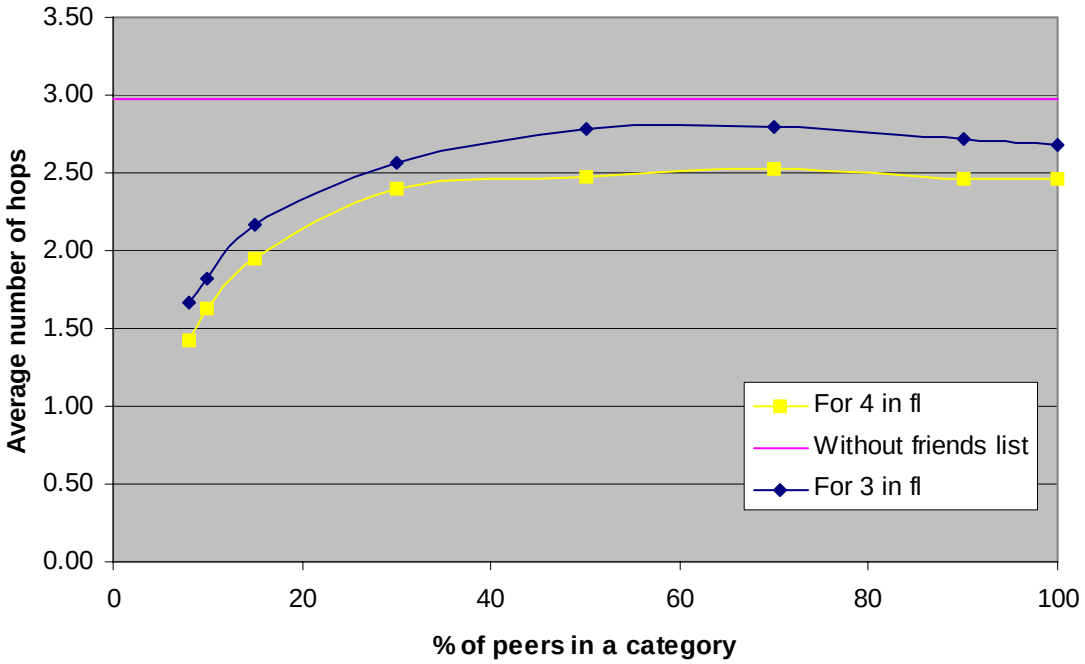

Figure 5.11: Average hops vs. percentage of peers in category for the Zipf-distributed dynamic system with 100 peers and 200 files

Fig. 5.11 shows that whether four neighbours are chosen or three neighbours are chosen from the friends list, the average number of hops is less than when no friends list is used. The performance is better in the system when four neighbours are chosen from the friends list.

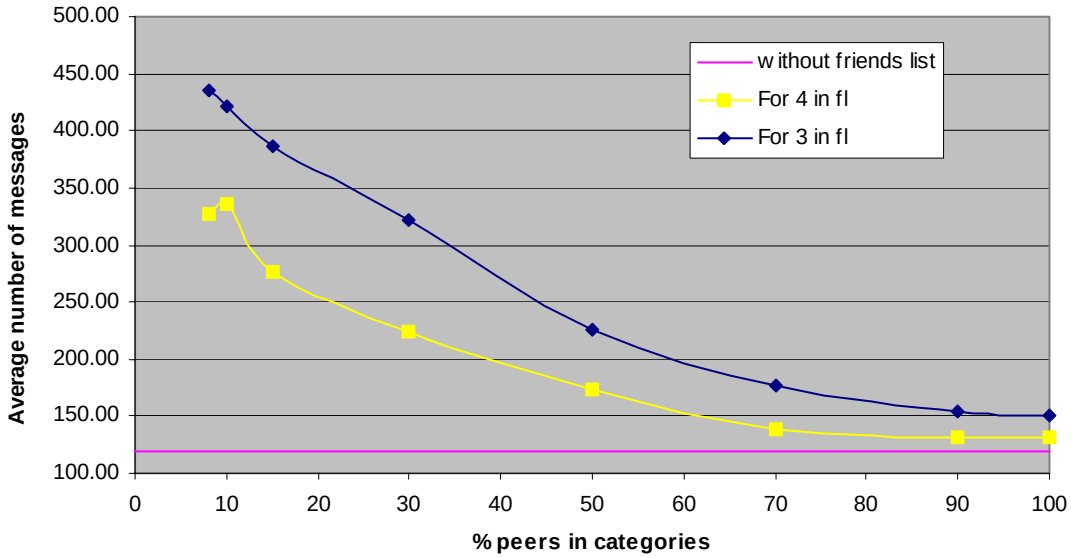

Figure 5.12: Average messages vs. percentage of peers in category for the Zipf-distributed dynamic system with 100 peers and 200 files

Regarding the number of messages, from Fig. 5.12, the system where four neighbours are chosen from the friends list performs also better than the system where three neighbours are chosen from the friends list. But we see that the number of messages in the baseline system with no friends list is lower than in both systems with a friends list. The explanation is similar to that in the case of uniformly distributed files/queries dynamic system. The neighbourhood of a peer is fixed in the system with no friends list. If some of the neighbouring peers go offline the queries are not forwarded further through them. Thus, only few messages are passed depending on which peers are online. While the performance in terms of number of messages of the baseline system is superior, the overall success rate of queries in this system is far smaller compared to the systems where the neighbouring peers are chosen from friends list. This can be seen from the success of queries given below in Table 5.2.

Table 5.2: Success of queries in the Zipf-distribution system

% of peers in the category	*query success in % (where queries are forwarded to 4 friend peers and 1 random peer)*	*query success in % (where queries are forwarded to 3 friend peers and 2 random peers)*	*query success in % (where no friends list is maintained)*
8	53.68	58.24	
10	65.71	64.88	
15	77.70	76.42	
30	85.91	85.86	
50	90.70	92.47	
70	95.36	94.30	
90	97.06	96.37	
100	96.61	97.51	46.01

Comparing the data from Tables 5.1, and 5.2, we find that the success rate for queries is higher when Zipf-distribution is used as compared to the uniform-distribution.

5.3 Summary

A static system and a dynamic system were implemented and studied to find the impact of our approach of friends list and compare the results with that of the standard Gnutella system, i.e., without friends list.

In the static experiment all peers were active throughout the simulation. The objective was to compare the impact on the system when the number of peers in the system is changed and also to see the impact when the number of files in the system was changed. In both cases the system where peers chose four friends from their lists to forward queries performed better than the system where peers chose three friends in terms of number of hops required to get responses and the number of messages circulating in the

system. In general, the system with friends list performed better than the system where the queries were forwarded without the use of friends list.

In the dynamic system peers can go offline and re-enter the system again. We compared systems with different distributions for files and queries in the system: a uniform distribution and a more realistic, Zipf-distribution. Here the system with Zipf-distribution performed better than the system with uniform-distribution in regards with getting responses for queries. Also the rate of success for queries was higher when Zipf-distribution is used as compared to the uniform-distribution.

CHAPTER 6

Conclusions and Future Work

The aim of the thesis is to improve the performance of the Gnutella peer-to-peer protocol (version 0.4) by introducing a semantic-social routing model. In the Gnutella protocol peers broadcast messages to their neighbours when they search for files, i.e., a querying peer sends the query to all of its neighbour peers, who in turn send the query to all of their neighbour peers. This forwarding takes place until the query reaches a peer that has a file matching the query or until a certain predefined maximal number of forwards (hops) is reached. In this way each query is propagated to up to n^p *peers*, where n is the number of neighbour peers and p is the maximum number of hops. This passing of messages generates a lot of traffic in the network, which degrades the quality of service.

To improve the quality of service in a Gnutella based peer-to-peer environment, we proposed a model where peers increase their performance (speed of search) by using social networks. We assume that there are a number of categories of interest. Each peer in our model creates and updates a friends list from its past experience, for each category of interest. Once peers generate their friends lists, they use these lists to semantically route queries in the network. Search messages in a given category are mainly sent to friends, i.e., peers who have been useful in the past in finding files in the same category. This helps to reduce the search time and decreases the network traffic by minimizing the number of messages circulating in the system as compared to standard Gnutella 0.4.

The approach was evaluated in a simulated peer-to-peer environment using the JADE multi-agent system platform. From the results obtained we can say that by learning other

peers' interests, building and exploiting their social networks to route queries semantically, peers get more relevant resources faster. The performance of the Gnutella 0.4 system is improved.

6.1 Conclusions

Two types of static experiments were performed. In the first there are two systems in which the number of peers differed but the number of files shared remained the same. In both systems, the performance was better when peers used their friends lists to forward queries. Moreover, the performance improved when peers had more friends in their neighbourhood.

The other static experiment involved systems in which the number of peers in the system remained constant but the number of files shared by the peers differed. Here, too, the system where queries are forwarded to friends performed better than a system where the queries were forwarded to random peers. Also in this experiment the system performed better when the peers used more friends (e.g. four versus three) as their neighbours to forward queries.

Two types of dynamic experiments were done. The first had a uniform distribution of shared files among peers and for queries generated by peers. In the second, the Zipf distribution was used. In both experiments it was found that the system where four neighbours were chosen from the friends list performed better than when three neighbours were chosen from the friends list. The system with Zipf distribution, which depicts more realistic system, performed better than one with the uniform distribution. While the number of messages generated was lower in a system without friends list, this was at the expense of the success of queries. Systems with friends lists performed significantly better in terms of success of queries.

A general pattern was found in both the static and dynamic system.

1) The number of hops increases with the increasing percentage of peers in a category.

2) The number of messages decreases with the increasing percentage of peers in a category.

The number of hops increases with the increase of the percentage of peers in a category. But there are fewer hops in the system where peers maintain friends lists versus in standard Gnutella. This conclusion applies in general to systems with more than one category of interest. Peers maintain separate friends lists for each category of interest and are unaware of the possible overlaps in these lists (e.g. the same peer can occur in two different friends lists for two different categories). Therefore the search in different categories use different friends lists and different neighbourhoods and do not interfere with each other. The messages related to any query in a system with many categories circulate in the same way as when only one category of interest is present. Therefore, the benefit obtained in decreased number of messages will be also seen in the systems with more than one category of interest. Thus the performance of Gnutella-based systems can be improved.

Thus, the friends list created here forms an abstract cluster of peers for a category. Once a message reaches one of the peers interested in the category then subsequent messages travel mostly to those peers that have interest in that category. This is similar to the information travelling in social networks. The friends lists created by each peer are based on its personal criteria and give the most benefit to that peer. Peers calculate the strengths of relationships of peers in its list and forwards queries to those who are on top of the lists since this helps in obtaining responses fast. We use a learning reinforcement formula to calculate the strength of the relationship, and this helps in considering current experiences against old experiences with other peers, depending on the value of α chosen. Thus one can change the strength of relationship in a system such that the friends list reflects a rapidly changing dynamic system or a slowly changing system. The learning and discovery of new peers, through random peers, helps in finding files in the interest category as either new peers enter the system with files or old peers acquire files. Thus, messages forwarded through random peer(s) not only help in discovering

other peers in the interest group, they also helps in the message not circulating in a set of peers. Thus the approach of social network benefits our model.

The creation and maintenance of friends list can also be used in other types of systems like super-peer networks. Here the super-peer can maintain friends lists based on the interests of its leaf peers.

6.2 Future Work

The model can be studied further. Some of the future works are as follows:

- One interesting issue is finding what value of α in the learning formula for the strength of relationship makes the friends list the most useful, i.e., conservative or explorative, in terms of speed and also in the number of messages. Thus further experiments can be done to obtain the best value for α in the learning formula for
  the strength of relationship.

- Another issue is to speed up the simulation run time and experiment on larger systems, namely increasing the total number of peers.

- The topology of the network can be changed, i.e., the number of neighbours to which a peer sends a query, can be changed. In this topology we can find the ideal number of friends among the neighbours that a query can be forwarded to so that responses to query are faster and the number of messages circulating in the network are reduced.

- Further experiments can also be done such that peers send queries to a varying number of peers from its friends list and not to a fixed number.

- The current approach is designed for the standard Gnutella (version 0.4). It would be interesting to see if this approach would be suitable also for the newer

version of Gnutella, which uses super-peer networks [Gnutella 2.0] and see if the friends list helps reduce the traffic and increase the speed in such a system. In this case, it would be probably more appropriate for the friends list to be maintained by the super-peer(s) rather than by all the peers.

Even though in the thesis the Gnutella 0.4 protocol was used, the proposed semantic social routing approach can be extended to any version of the Gnutella protocol and probably also to other P2P protocols. The model suggests that peers learn from experience the interests of other peers and use this knowledge to search more efficiently. If one substitutes the word “peers” with “users”, the same is shown to be true on a highest level, in human society as shown by Milgram (1967). In fact our approach addresses the application level on top of the P2P network protocol layer, thus it can be applied to a wide class of decentralized applications of peer nature, for example, web services and service oriented computing.

REFERENCES

Aberer, K. and Hauswirth, M. (2002), "An Overview on Peer-to-Peer Information Systems", Workshop on Distributed Data and Structures, Paris, France.

Adamic, L. A. and Huberman, B. A. (2002), "Zipf's law and the Internet", Glottometrics 3, 143-150.

Adar, E. and Huberman, B. A., (2000), "Free riding on gnutella", Technical report, Xerox PARC, 10 Aug., 29 Nov. 2004. <http://citeseer.ist.psu.edu/adar00free.html>

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

Crespo, A. and Garcia-Molina, H. (2002), "Routing Indices For Peer-to-Peer Systems", Proceedings of the International Conference on Distributed Computing Systems, Vienna, Austria.

Freeman, L. C. (2000), "Visualizing Social Networks", Journal of Social Structure, Vol. 1, No. 1, Nov. 29 2004.
<http://www.cmu.edu/joss/content/articles/volume1/Freeman.html >

FreeNet, 2001, Nov. 29 2004. <www.freenetproject.org>

Gnutella, 2001, Jan 13 2004. <http://gnutella.wego.com>

Gnutella Protocol, version 0.4 (2003), Nov. 29 2004.
< http://rfc-gnutella.sourceforge.net/developer/stable/index.html>

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

Granovetter, M. (1983), "The Strength of Weak Ties: A Network Theory Revisited", Sociological Theory, Vol. 1, 203-233.

Iamnitchi, A., Ripeanu, M. and Foster, I. (2002), "Locating Data in (Small-World?) Peer-to-Peer Scientific Collaborations", Proceedings of the First International Workshop on Peer-to-Peer Systems, Massachusetts, USA.

JADE 2.61, 13 Jan 2004. <http://sharon.cselt.it/projects/jade/>

Joseph, S. (2002), "NeuroGrid: Semantically Routing Queries in Peer-to-Peer Networks", Proceedings of the International Workshop on Peer-to-Peer Computing, Pisa, Italy.

Jovanovic, M. (2001), "Modeling large-scale peer-to-peer networks and a case study of gnutella", M.S. thesis, University of Cincinnati.

Kalogeraki, V., Gunopulos, D. and Zeinalipour-Yazti, D. (2002), "A local search mechanism for peer-to-peer networks", Proceedings of the Eleventh International Conference on Information and Knowledge Management, McLean, USA.

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.

KaZaA (2002), Nov. 29 2004 <http://www.kazaa.com/us/index.htm>

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.

Law, A. and Kelton, W., (2000), "Simulation Modeling and Analysis", 3rd Ed., McGraw Hill.

Lv, Q., Cao, P., Cohen, E., Li, K. and Shenker, S. (2002), "Search and replication in unstructured peer-to-peer networks", Proceedings of the 16th international conference on Supercomputing, New York, USA, 84 – 95.

Markatos, E. (2002), "Tracing a large-scale Peer to Peer System: an hour in the life of Gnutella", 2nd IEEE/ACM Symposium on Cluster Computing and the Grid, Berlin, Germany.

McCarty, C. (2002), "Structure in Personal Networks", Journal of Social Structure, Vol. 3, No.1.

Milgram, S (1967), “The Small World Problem”, Psychology Today, 60-67.

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.

Napster, 2001, 29 Nov. 2004 < http://opennap.sourceforge.net/napster.txt >

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

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

Ng, W., Ooi, B. and Tan, K. (2002), “BestPeer: A Self-Configurable Peer-to-Peer System”, Proceedings of the 18th International Conference on Data Engineering, San Jose, CA, 272.

Ng, C. and Sia, K. (2002), "Peer Clustering and Firework Query Model", Poster Proceedings of the 11th World Wide Web Conference, Honolulu, Hawaii.

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, Nov. 29 2004 <http://www.soc.surrey.ac.uk/JASSS/4/4/4.html>

Portmann, M., Sookavatana, P., Ardon, S. and Seneviratne, A. (2001), "The cost of peer discovery and searching in the gnutella peer-to-peer file sharing protocol", Proceedings to the International Conference on Networks, Volume 1, Bangkok, Thailand.

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

Ratnasamy, S., Francis, P., Handley, M., Karp, R. and Shenker S. (2001), "A scalable content addressable network", Proceedings of ACM SIGCOMM, San Diego, USA.

Ripeanu, M. (2001), "Peer-to-Peer Architecture Case Study: Gnutella Network", Proceedings of IEEE 1st International Conference on Peer-to-peer Computing, Linkoping Sweden.

Rowstron, A. and Druschel, P. (2001), "Pastry: Scalable, decentralized object location and routing for large-scale peer-to-peer systems", Proceedings of the 18th IFIP/ACM International Conference on Distributed Systems Platforms, Heidelberg, Germany.

Saroiu, S., Gummadi, P. and Gribble, S. (2002), "A measurement study of peer-to-peer file sharing systems", Proceedings of Multimedia Computing and Networking (MMCN), San Jose, California.

Scharnhorst. A. (2003), "Complex Networks and the Web: Insights from Nonlinear Physics", Journal of Computer-Mediated Communication, Vol. 8 (4).

Schlosser, M. T., Condie, T. E. and Kamvar, S. D. (2003), "Simulating a File-Sharing P2P Network", First Workshop on Semantics in P2P and Grid Computing, May 20, Budapest, Hungary.

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

Sripanidkulchai, K., Maggs, B. and Zhang, H. (2003), "Efficient Content Location Using Interest-Based Locality in Peer-to-Peer Systems", Proceedings IEEE INFOCOM 2003, San Francisco, USA.

Stoica, I., Morris, R., Karger, D., Kaashoek, M. and Balakrishnan, H. (2001), "Chord: A scalable peer-to-peer lookup service for internet applications", Proceedings of ACM SIGCOMM, San Diego, USA.

Tang, C., Xu, Z. and Dwarkadas, S. (2003), "Peer-to-peer information retrieval using self-organizing semantic overlay networks", Proceedings of the 2003 conference on "Applications, technologies, architectures, and protocols for computer communications", Karlsruhe, Germany.

Tang, C., Xu, Z. and Mahalingam, M. (2002), "PeerSearch: Efficient Information Retrieval in Peer-to-Peer Networks", Hewlett Packard Technical Reports.

Triantafillou, P., Xiruhaki, C. and Koubarakis, M. (2002), "Efficient Massive Sharing of Content among Peers", IEEE Workshop on Resource Sharing in Massively Distributed Systems, Vienna, Austria

Tsoumakos, D. and Roussopoulos, N. (2003), "A Comparison of Peer A Comparison of Peer-to to-Peer Peer Search Methods Search Methods", Sixth International Workshop on Web and Databases (WebDB 2003), June 12-13, San Diego, California, USA.

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, Nov. 29 2004.
<http://www.ai.univie.ac.at/%7Epaolo/conf/esaw02/esaw02accpapers.html>

Vaucher, J., Babin, G., Kropf, P. and Jouve, Th. (2002), "Experimenting with Gnutella Communities", Distributed Communities on the Web (DCW 2002), Sydney, Australia. LNCS 2468, Springer Berlin, pp.85-99.

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.

Waloszek, G. 2002, "Personal Networks", SAP Design Guild, Ed. 5: Collaboration, Nov. 29 2004.
<http://www.sapdesignguild.org/editions/edition5/personal_networks.asp>

Wang, H., Lin, T., Chen, C. H. and Shen, Y. (2004), "Dynamic Search and Performance Analysis in Unstructured Peer-to-Peer Networks", Proceedings of the 13th World Wide Web Conference, 17-22 May, New York, USA.

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, Nov. 29 2004.
<http://xena1.ddns.comp.nus.edu.sg/p2p/BuddyWeb.ps>

Wang, Y. and Vassileva, J. (2003), "Bayesian Network-Based Trust Model", Proceedings of IEEE/WIC International Conference on Web Intelligence (WI 2003), Oct13-17, Halifax, Canada.

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, Nov. 29 2004. <http://moreno.ss.uci.edu/webster.pdf>

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.

White, D. and Houseman, M. (2003), "The Navigability of Strong Ties: Small Worlds, Tie Strength and Network Topology", Complexity Vol. 8, No.1

Yang B. and Garcia-Molina H. (2002), "Efficient Search in Peer-to-Peer Networks", The 22nd International Conference on Distributed Computing Systems, Vienna, Austria.

Yang B. and Garcia-Molina H. (2003), “Designing a Super-Peer Network”, IEEE International Conference on Data Engineering, Los Alamitos, CA.

Yu, B. and Singh, M. (2000), "A Social Mechanism of Reputation Management in Electronic Communities", Proceedings of Fourth International Workshop on Cooperative Information Agents, Boston, USA.

Appendix A

Results

The appendix shows the benefits, in percentages, for the average number of hops and average number of messages for the static system.

The graphs in Figs. A.1 and A.2 are when 100 peers are sharing 200 files.
Benefits are calculated as follows:

benefit (for hops) =

$$\frac{\text{average number of hops (when friends list is maintained)}}{\text{average number of hops (when friends list is not maintained)}} * 100$$

benefit (for messages) =

$$\frac{\text{average number of messages (when friends list is maintained)}}{\text{average number of messages (when friends list is not maintained)}} * 100$$

Fig. A.1 shows the benefits with respect to hops, in percentage, gained by the system where peers send queries to friends as compared to the system where peers have no friends.

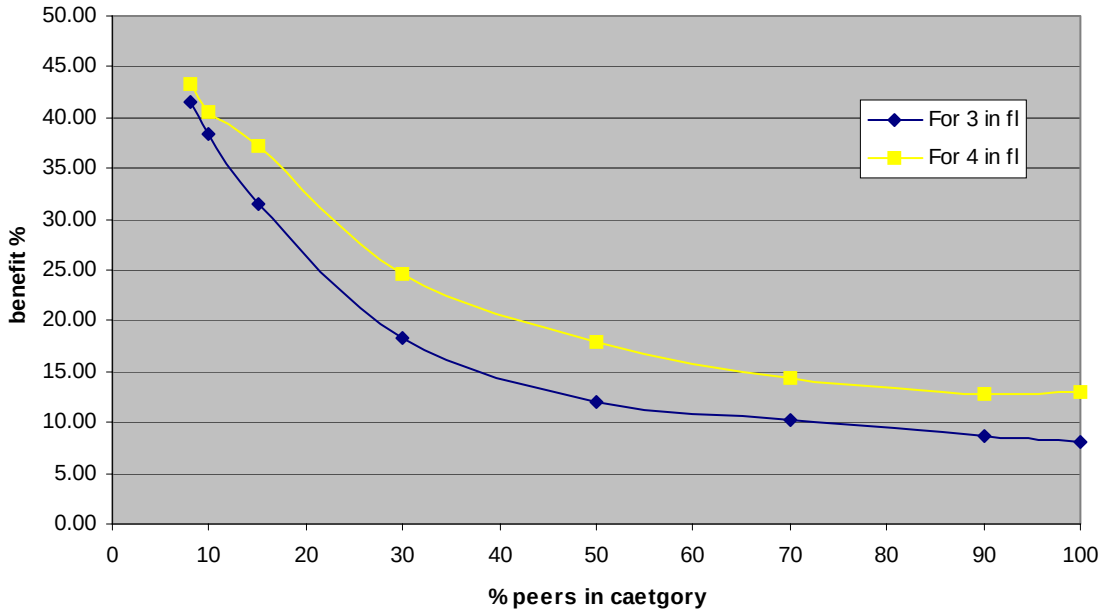

Figure A.1: Benefit with respect to hops in percentage for 100 peers and 200 files

From Fig. A.1 it can be seen that when four peers are chosen from the friends list, illustrated by the lighter line with square, we get a benefit in the average number of hops ranging from 43.25% to 12.97% as the percentage of peers in the category increases from 8% to 100%. Similar benefit is noted when three peers are chosen from the friends list, illustrated by the darker line with diamonds. Here the benefit ranges from 41.62% to 8.12% as the percentage of peers in the category increases from 8% to 100%.

Fig. A.2 shows the benefits with respect to messages, in percentage, gained by the system where peers send queries to friends as compared to the system where peers have no friends.

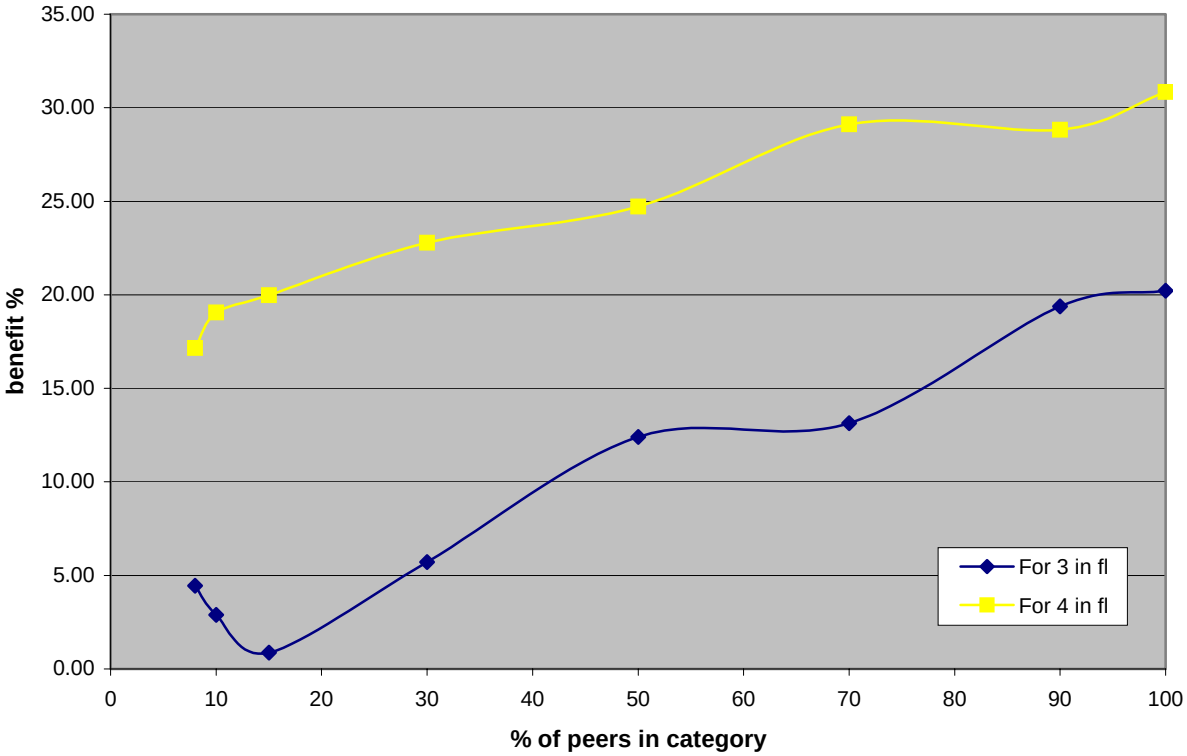

Figure A.2: Benefits with respect to number of messages in percentage for 100 peers and 200 files

From Fig. A.2, we see that the benefit in decrease of average number of messages is between 17.16% and 30.85% when four neighbours are chosen from the friends list (lighter line) and the benefit when three neighbours are from the friends list (darker line) the benefit varies from 4.44% to 20.21% as the percentage of peers interested in the category increases from 8% to 100%. The success of queries for this combination is given in Tables A.1 and A.2. Similar trend is exhibited fro success of queries in all combinations of experimented in the static system, namely, 100 peers sharing 100 files, and 150 files and also for the combination where 50 peers share 200 files.

Table A.1: Success of queries for 100 peers and 200 files when three and four neighbours from friends list are used to forward queries

% of peers in the category	*query success in % (for three from friends list)*	*query success in % (for four from friends list)*
8	47.24	47.24
10	54.77	58.76
15	71.79	68.43
30	90.48	92.19
50	96.50	97.08
70	99.36	98.81
90	100.00	99.37
100	100.00	99.90

Table A.2: Success of queries for 100 peers and 200 files when no friends list is used to forward queries

% of peers in the category	*query success in %*
100	100

The graphs in Fig. A.3 and A.4 show the benefit in percentage for 50 peers sharing 200 files.

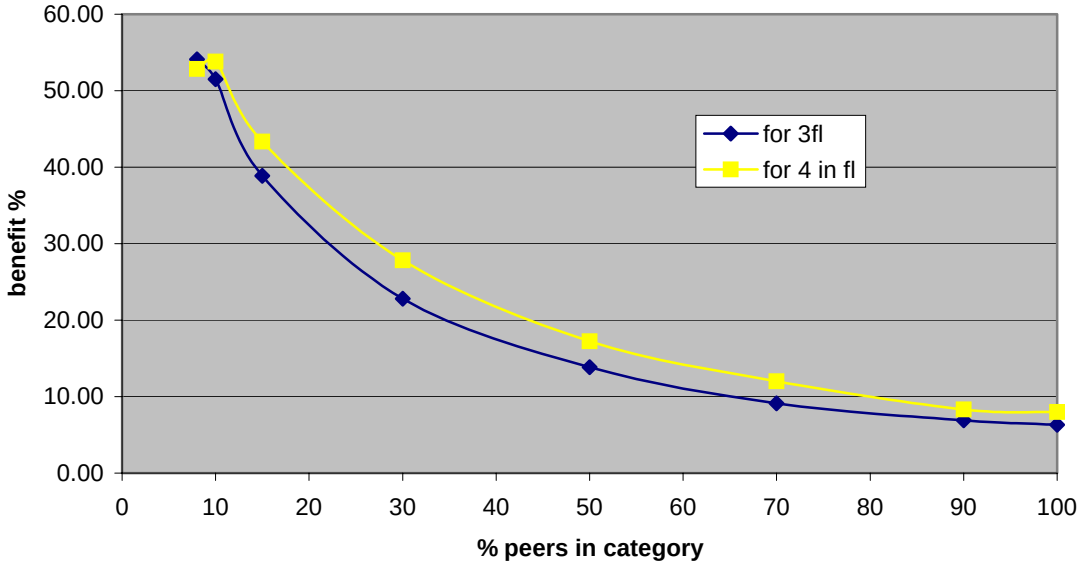

Figure A.3: Benefits with respect to hops in percentage for 50 peers and 200 files in static system

In Fig. A.3 we see a benefit in the average number of hops ranging from 52.83% to 8.00% as the percentage of peers in the category increases from 8% to 100%. Similarly when three peers are chosen from the friends list the benefit ranges from 54.09% to 6.31% as the percentage of peers in the category increases from 8%.

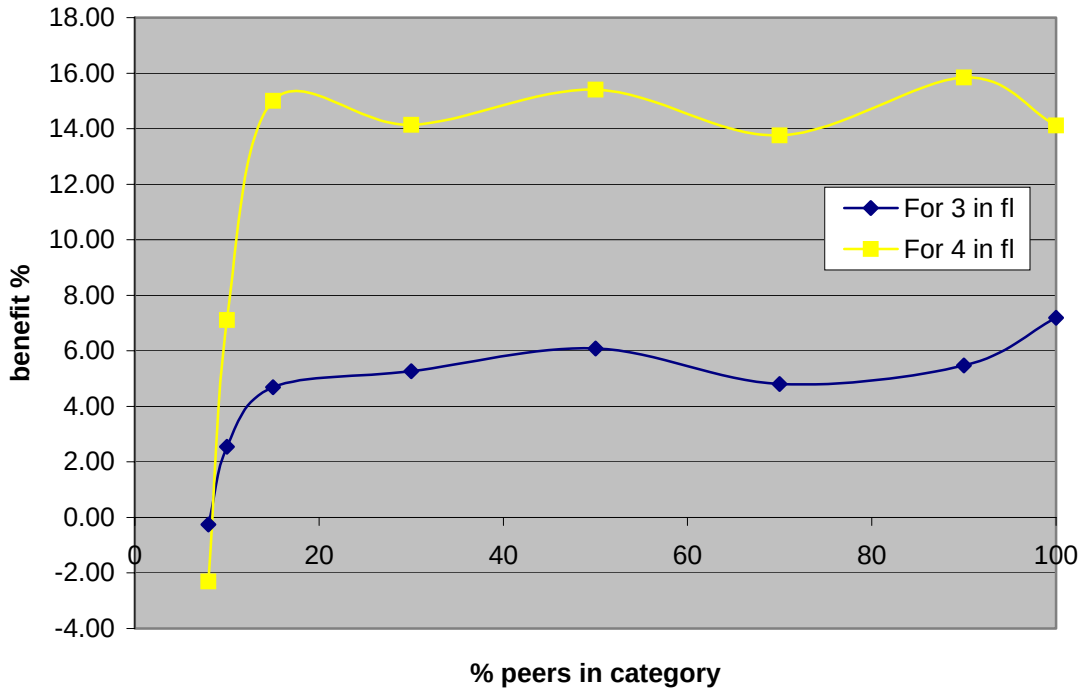

Figure A.4: Benefits with respect to messages in percentage for 50 peers and 200 files in static system

From Fig. A.4, we see that for 8% peers in category the performance of the standard Gnutella like system is better than our friends list model. But after that the benefit increases in decrease of average number of messages between 7% to14% when four neighbours are chosen from the friends list (lighter line) and the benefit when three neighbours are from the friends list (darker line) the benefit varies from 2% to 7% as the percentage of peers interested in the category increases from 8% to 100%.

The graphs in Fig. A.5 and A.6 show the benefit in percentage for 100 peers sharing 100 files.

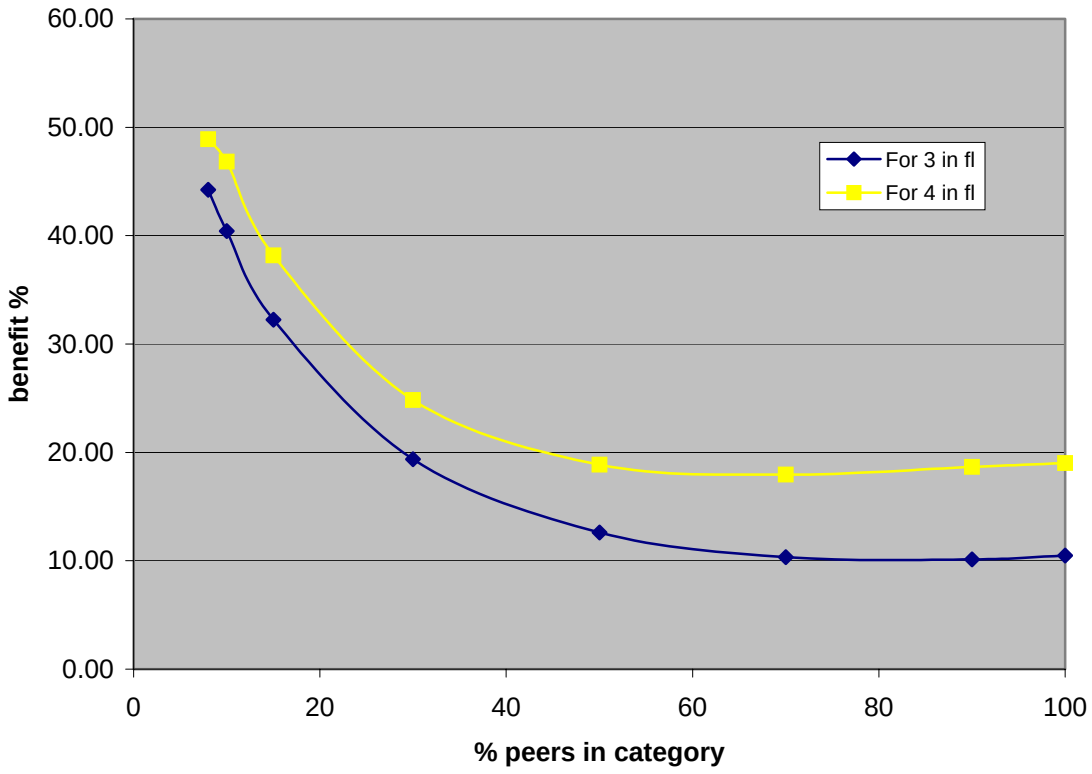

Figure A.5: Benefits with respect to hops in percentage for 100 peers and 100 files in static system

From the Fig. A.5 it can be seen that when four peers are chosen from the friends list we get a benefit in the average number of hops ranging from 50% to 18% as the percentage of peers in the category increases. Similar benefit is noted when three peers are chosen from the friends list ranging from 45% to 10% as the percentage of peers in the category increases.

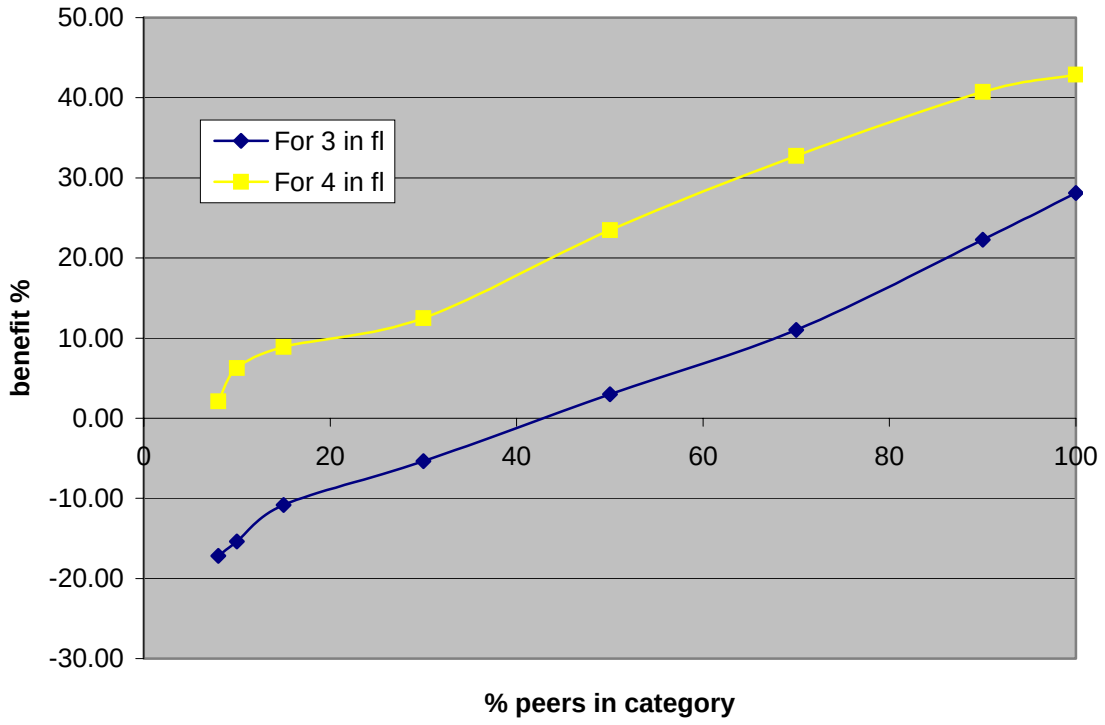

Figure A.6: Benefits with respect to messages in percentage for 100 peers and 100 files in static system

From Fig. A.6, we see that for 8% to 30% peers in category the performance of the standard Gnutella like system is better than our friends list model when three neighbours are chosen from the friends list. But the benefit in the decrease of average number of messages increases from 2% to 42% when four neighbours are chosen from the friends list (lighter line) and the benefit when three neighbours are from the friends list (darker line) the benefit varies from 2% to 28% as the percentage of peers interested in the category increases from 50% to 100%.

The graphs in Fig. A.7 and A.8 show the benefit in percentage for 100 peers sharing 150 files.

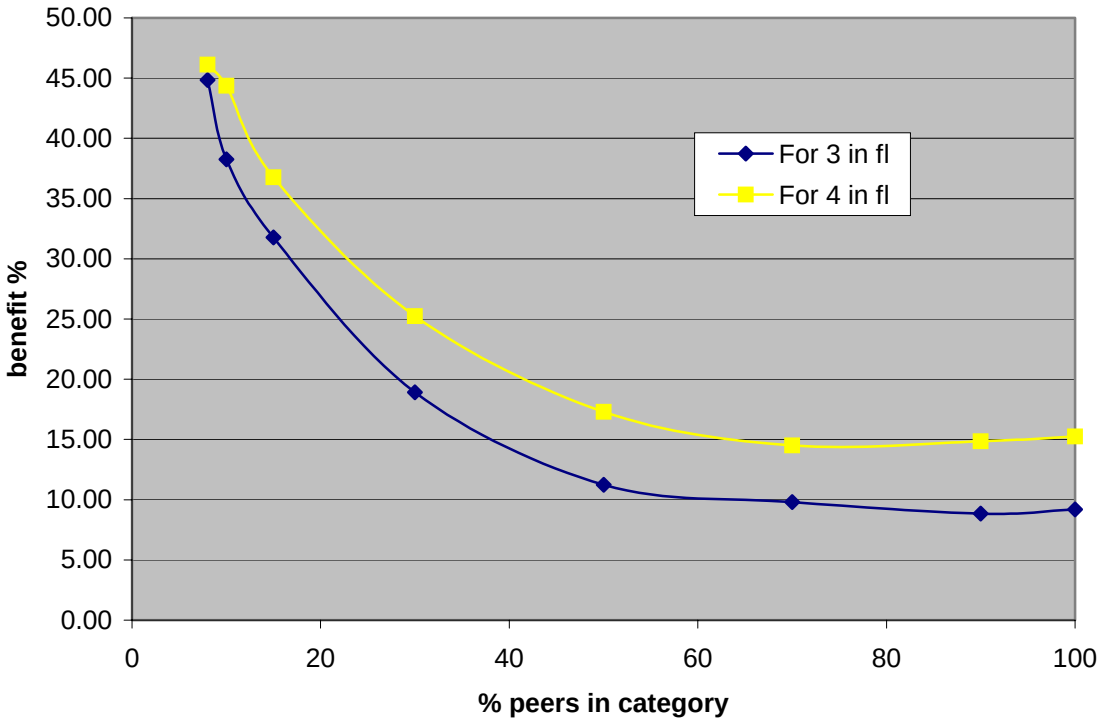

Figure A.7: Benefits with respect to hops in percentage for 100 peers and 150 files in static system

From the Fig. A.7 it can be seen that when four peers are chosen from the friends list we get a benefit in the average number of hops ranging from 46% to 15% as the percentage of peers in the category increases. Similar benefit is noted when three peers are chosen from the friends list ranging from 45% to 9% as the percentage of peers in the category increases.

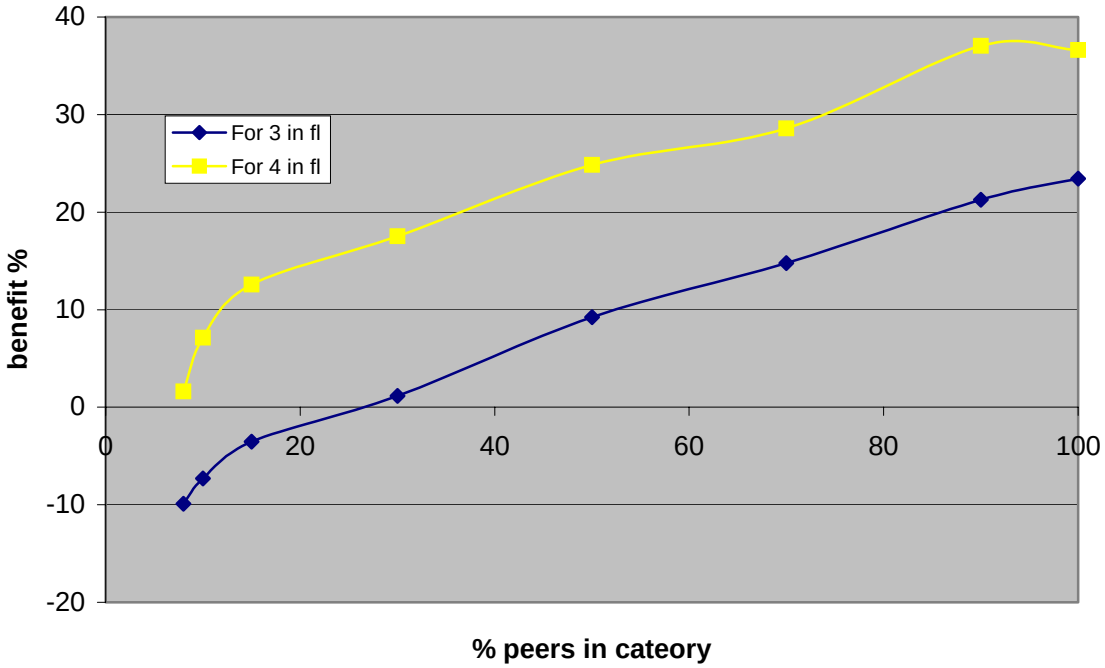

Figure A.8: Benefits with respect to messages in percentage for 100 peers and 150 files in static system

From Fig. A.8, we see that for 8% to 15% peers in category, the performance of the standard Gnutella like system is better than our friends list model when three neighbours are chosen from the friends list. The benefit in the decrease of average number of messages increases from 1% to 38% when four neighbours are chosen from the friends list (lighter line) and the benefit when three neighbours are from the friends list (darker line) the benefit varies from 1% to 23% as the percentage of peers interested in the category increases from 30% to 100%.

Appendix B

The following tables B.1, B.2, B.3 and B.4 show the values of the standard deviation from the three runs for the number of messages in the system, in the different experiments done in the static system.

Table B.1: Standard deviation for the number of messages when there are 100 active peers and 200 files in the system

% of peers in category	Standard Deviation when three peers from the friends list is chosen	Standard Deviation when four peers from the friends list is chosen
8	2.419	6.796
10	3.871	4.729
15	1.578	3.416
30	3.022	2.760
50	0.453	4.709
70	1.665	2.150
90	3.365	2.350
100	8.607	3.184

Table B.2: Standard deviation for the number of messages when there are 50 active peers and 200 files in the system

% of peers in category	Standard Deviation when three peers from the friends list is chosen	Standard Deviation when four peers from the friends list is chosen
8	5.828	4.643
10	6.142	7.516
15	3.988	1.549
30	3.030	4.266
50	2.279	1.082
70	2.106	1.299
90	1.568	3.047
100	0.207	4.204

Table B.3: Standard deviation for the number of messages when there are 100 active peers and 100 files in the system

% of peers in category	Standard Deviation when three peers from the friends list is chosen	Standard Deviation when four peers from the friends list is chosen
8	2.419	6.796
10	3.871	4.729
15	1.578	3.416
30	3.022	2.760
50	0.453	4.709
70	1.665	2.150
90	3.365	2.350
100	8.607	3.184

Table B.4: Standard deviation for the number of messages when there are 100 active peers and 150 files in the system

% of peers in category	Standard Deviation when three peers from the friends list is chosen	Standard Deviation when four peers from the friends list is chosen
8	25.079	10.274
10	19.177	14.384
15	16.257	13.488
30	13.732	9.502
50	7.976	4.744
70	7.032	3.870
90	1.236	6.024
100	3.284	4.244